[MS-CSRA]:
Certificate Services Remote Administration Protocol

Intellectual Property Rights Notice for Open Specifications Documentation

- **Technical Documentation.** Microsoft publishes Open Specifications documentation for protocols, file formats, languages, standards as well as overviews of the interaction among each of these technologies.

- **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 may make copies of it in order to develop implementations of the technologies described in the Open Specifications and may distribute portions of it in your implementations using these technologies or your documentation as necessary to properly document the implementation. You may also distribute in your implementation, with or without modification, any schema, IDL's, or code samples that are included in the documentation. This permission also applies to any documents that are referenced in the Open Specifications.

- **No Trade Secrets.** Microsoft does not claim any trade secret rights in this documentation.

- **Patents.** Microsoft has patents that may cover your implementations of the technologies described in the Open Specifications. Neither this notice nor Microsoft's delivery of the documentation grants any licenses under those or any other Microsoft patents. However, a given Open Specification may be covered by Microsoft Open Specification Promise or the Community Promise. If you would prefer a written license, or if the technologies described in the Open Specifications are not covered by the Open Specifications Promise or Community Promise, as applicable, patent licenses are available by contacting iplg@microsoft.com.

- **Trademarks.** The names of companies and products contained in this documentation may 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 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 specifically described above, whether by implication, estoppel, or otherwise.

Tools. The Open Specifications do 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 are intended for use in conjunction with publicly available standard specifications and network programming art, and assumes that the reader either is familiar with the aforementioned material or has immediate access to it.

Revision Summary

Date	Revision History	Revision Class	Comments
12/18/2006	0.1		MCPP Milestone 2 Initial Availability
03/02/2007	1.0		MCPP Milestone 2
04/03/2007	1.1		Monthly release
05/11/2007	1.2		Monthly release
06/01/2007	2.0	Major	Updated and revised the technical content.
07/03/2007	2.1	Minor	Updates for minor issues.
07/20/2007	2.2	Minor	Updates for minor issues.
08/10/2007	2.2.1	Editorial	Revised and edited the technical content.
09/28/2007	2.3	Minor	Updated the technical content.
10/23/2007	3.0	Major	Updated and revised the technical content.
11/30/2007	4.0	Major	Updated and revised the technical content.
01/25/2008	5.0	Major	Updated and revised the technical content.
03/14/2008	6.0	Major	Updated and revised the technical content.
05/16/2008	7.0	Major	Updated and revised the technical content.
06/20/2008	8.0	Major	Updated and revised the technical content.
07/25/2008	8.0.1	Editorial	Revised and edited the technical content.
08/29/2008	8.1	Minor	Updated the technical content.
10/24/2008	8.2	Minor	Updated the technical content.
12/05/2008	8.3	Minor	Updated the technical content.
01/16/2009	8.4	Minor	Updated the technical content.
02/27/2009	8.5	Minor	Updated the technical content.
04/10/2009	9.0	Major	Updated and revised the technical content.
05/22/2009	10.0	Major	Updated and revised the technical content.
07/02/2009	11.0	Major	Updated and revised the technical content.
08/14/2009	12.0	Major	Updated and revised the technical content.
09/25/2009	13.0	Major	Updated and revised the technical content.

Date	Revision History	Revision Class	Comments
11/06/2009	14.0	Major	Updated and revised the technical content.
12/18/2009	15.0	Major	Updated and revised the technical content.
01/29/2010	16.0	Major	Updated and revised the technical content.
03/12/2010	17.0	Major	Updated and revised the technical content.
04/23/2010	18.0	Major	Updated and revised the technical content.
06/04/2010	19.0	Major	Updated and revised the technical content.
07/16/2010	20.0	Major	Significantly changed the technical content.
08/27/2010	21.0	Major	Significantly changed the technical content.
10/08/2010	22.0	Major	Significantly changed the technical content.
11/19/2010	23.0	Major	Significantly changed the technical content.
01/07/2011	24.0	Major	Significantly changed the technical content.
02/11/2011	24.0	No change	No changes to the meaning, language, or formatting of the technical content.
03/25/2011	25.0	Major	Significantly changed the technical content.
05/06/2011	26.0	Major	Significantly changed the technical content.
06/17/2011	27.0	Major	Significantly changed the technical content.
09/23/2011	27.0	No change	No changes to the meaning, language, or formatting of the technical content.
12/16/2011	28.0	Major	Significantly changed the technical content.
03/30/2012	28.0	No change	No changes to the meaning, language, or formatting of the technical content.
07/12/2012	29.0	Major	Significantly changed the technical content.
10/25/2012	29.0	No change	No changes to the meaning, language, or formatting of the technical content.
01/31/2013	29.0	No change	No changes to the meaning, language, or formatting of the technical content.
08/08/2013	30.0	Major	Significantly changed the technical content.
11/14/2013	31.0	Major	Significantly changed the technical content.

Contents

1   Introduction

The Certificate Services Remote Administration Protocol consists of a set of **Distributed Component Object Model (DCOM)** interfaces, as specified in [MS-DCOM], that allow administrative tools to configure the state and policy of a **certification authority (CA)** on a server.

For a complete understanding of this protocol, familiarity with **public key infrastructure (PKI)** concepts such as asymmetric and symmetric cryptography, asymmetric and **symmetric encryption** techniques, digital **certificate** concepts, and cryptographic key establishment is required. A comprehensive understanding of the X.509 standard, as specified in [X509], is also required.

The Handbook of Applied Cryptography provides an excellent introduction to cryptography and PKI concepts. For more information, see [CRYPTO]. The X.509 standard, as specified in [X509], provides an excellent introduction to PKI and certificate concepts. **certificate revocation** and status checking provides an excellent introduction to **certificate revocation lists (CRLs)** and **revocation** concepts. For more information, see [MSFT-CRL].

Sections 1.8, 2, and 3 of this specification are normative and can contain the terms MAY, SHOULD, MUST, MUST NOT, and SHOULD NOT as defined in RFC 2119. Sections 1.5 and 1.9 are also normative but cannot contain those terms. All other sections and examples in this specification are informative.

1.1   Glossary

The following terms are defined in [MS-GLOS]:

access control entry (ACE)
access control list (ACL)
Active Directory
attestation
attribute
certificate
certification authority (CA)
certificate revocation
certificate revocation list (CRL)
certificate services
certificate template
client
common name (CN)
container
Coordinated Universal Time (UTC)
Cryptographic Application Programming Interface (CAPI) or CryptoAPI
cryptographic service provider (CSP)
Cryptography API: Next Generation (CNG)
discretionary access control list (DACL)
distinguished name (DN)
Distributed Component Object Model (DCOM)
domain
domain controller (DC)
dynamic endpoint
EK public key (EKPub)
encryption

endorsement key (EK)
endpoint
enroll/enrollment
enterprise certificate authority
exchange certificate
execution context
forest
fully qualified domain name (FQDN) (1)
interface identifier (IID)
key
key archival
Key Distribution Center (KDC)
key escrow
key recovery agent (KRA)
key recovery certificate
Lightweight Directory Access Protocol (LDAP)
little-endian
object
object identifier (OID)
offline
object remote procedure call (ORPC)
principal
private key
public key
public key algorithm
public key infrastructure (PKI)
public-private key pair
remote procedure call (RPC)
revocation
role separation
root CA
sanitized name
schema
SHA-1 hash
signing certificates
standalone CA
symmetric algorithm
symmetric encryption
trusted platform module (TPM)
Uniform Resource Locator (URL)
Universal Naming Convention (UNC)
universally unique identifier (UUID)
URI

The following terms are specific to this document:

CA policy algorithm: An algorithm that determines whether to issue a certificate for a specified certificate request and defines how that certificate is constructed.

CA policy module: The Microsoft **CA** implements policy algorithms with policy modules. The policy module can be configured as described in [MSFT-MODULES]. It can also be replaced as described in [MSDN-ICERTPOLICY2].

certificate authority (CA) roles: A list of administrator-defined rights or **access control lists (ACLs)** that define the capability of a particular **principal** on a **certificate authority (CA)**.

CA Roles are specified in [CIMC-PP] section 5.2, and include administrator, operator, officer, and auditor.

Enrollment Agent rights: A list of administrator-defined rights or **ACLs** that define the capability of a particular **principal** to obtain a **certificate**, with subject information pertaining to a different **principal**, from a **CA**. Enrollment Agent is not one of the roles defined in [CIMC-PP].

index: A data structure that is used to quickly locate data in a **table**. For more information, see [GRAY].

issuance: See **certificate**.

log files: The server may<1>keep a log of data value and structure changes in a database. The log is stored in stable storage and is used by the database to restore the last committed values of data items. For more information, see [GRAY].

Officer rights: A list of administrator-defined rights or **access control lists (ACLs)** that define the capability of a specified officer (one of the roles specified in [CIMC-PP]) to approve the certificate requests that are associated with a specific set of principals. **Officer rights**, as specified in [CIMC-PP], are locally configured and stored on a **CA** and enforced by the **CA**.

release from hold: To change the status of a certificate with **Request.Disposition** "certificate revoked" and **Request.Revoked.Reason** "certificateHold" to **Request.Disposition** "certificate issued", using the **RevokeCertificate** method. As detailed in this document in the server processing rules for the **RevokeCertificate** method, only a certificate with **Request.Disposition** set to "certificate revoked" and **Request.Revoked.Reason** set to "certificateHold" can be **released from hold**.

SHA-2 hash: A hashing algorithm specified in [FIPS180-4] that was developed by the National Institute of Standards and Technology (NIST) and the National Security Agency (NSA).

social engineering: The class of attacks in which the attacker uses human-to-human interactions to improperly gain user rights.

subordinate CA: A type of **CA** that is not a **root CA** for a relying party or **client**. A **subordinate CA** is a **CA** whose **certificate** is signed by some other **CA**, as specified in [RFC2510].

table: A set of data elements that is organized into a predefined format of rows and columns. For more information, see [GRAY].

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

1.2   References

References to Microsoft Open Specifications documentation do not include a publishing year because links are to the latest version of the documents, which are updated frequently. References to other documents include a publishing year when one is available.

A reference marked "(Archived)" means that the reference document was either retired and is no longer being maintained or was replaced with a new document that provides current implementation details. We archive our documents online [Windows Protocol].

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. Please check the archive site, http://msdn2.microsoft.com/en-us/library/E4BD6494-06AD-4aed-9823-445E921C9624, as an additional source.

[CIMC-PP] National Security Agency (NSA), "Certificate Issuing and Management Components Family of Protection Profiles", Version 1.0, October 2001, http://www.commoncriteriaportal.org/files/ppfiles/PP_CIMCPP_SL1-4_V1.0.pdf

[FIPS180-4] FIPS PUBS, "Secure Hash Standards (SHS)", March 2012, http://csrc.nist.gov/publications/fips/fips180-4/fips-180-4.pdf

[ITUX690] ITU-T, "ASN.1 Encoding Rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)", Recommendation X.690, July 2002, http://www.itu.int/ITU-T/studygroups/com17/languages/X.690-0207.pdf

[MS-ADA1] Microsoft Corporation, "Active Directory Schema Attributes A-L".

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

[MS-ADSO] Microsoft Corporation, "Active Directory System Overview".

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

[MS-CRTD] Microsoft Corporation, "Certificate Templates Structure".

[MS-DCOM] Microsoft Corporation, "Distributed Component Object Model (DCOM) Remote Protocol".

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

[MS-ERREF] Microsoft Corporation, "Windows Error Codes".

[MS-ICPR] Microsoft Corporation, "ICertPassage Remote Protocol".

[MS-KILE] Microsoft Corporation, "Kerberos Protocol Extensions".

[MS-LSAD] Microsoft Corporation, "Local Security Authority (Domain Policy) Remote Protocol".

[MS-LSAT] Microsoft Corporation, "Local Security Authority (Translation Methods) Remote Protocol".

[MS-NLMP] Microsoft Corporation, "NT LAN Manager (NTLM) Authentication Protocol".

[MS-NRPC] Microsoft Corporation, "Netlogon Remote Protocol".

[MS-OAUT] Microsoft Corporation, "OLE Automation Protocol".

[MS-RPCE] Microsoft Corporation, "Remote Procedure Call Protocol Extensions".

[MS-RRP] Microsoft Corporation, "Windows Remote Registry Protocol".

[MS-WCCE] Microsoft Corporation, "Windows Client Certificate Enrollment Protocol".

[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

[RFC2251] Wahl, M., Howes, T., and Kille, S., "Lightweight Directory Access Protocol (v3)", RFC 2251, December 1997, http://www.ietf.org/rfc/rfc2251.txt

[RFC2315] Kaliski, B., "PKCS #7: Cryptographic Message Syntax Version 1.5", RFC 2315, March 1998, http://www.ietf.org/rfc/rfc2315.txt

[RFC2478] Baize, E., and Pinkas, D., "The Simple and Protected GSS-API Negotiation Mechanism", RFC 2478, December 1998, http://www.ietf.org/rfc/rfc2478.txt

[RFC2510] Adams, C., and Farrell, S., "Internet X.509 Public Key Infrastructure Certificate Management Protocols", RFC 2510, March 1999, http://www.ietf.org/rfc/rfc2510.txt

[RFC2559] Boeyen, S., Howes, T., and Richard, P., "Internet X.509 Public Key Infrastructure Operational Protocols - LDAPv2", RFC 2559, April 1999, http://www.ietf.org/rfc/rfc2559.txt

[RFC2797] Myers, M., Liu, X., Schaad, J., and Weinstein, J., "Certificate Management Messages Over CMS", RFC 2797, April 2000, http://www.ietf.org/rfc/rfc2797.txt

[RFC2986] Nystrom, M., and Kaliski, B., "PKCS#10: Certificate Request Syntax Specification", RFC 2986, November 2000, http://www.ietf.org/rfc/rfc2986.txt

[RFC3280] Housley, R., Polk, W., Ford, W., and Solo, D., "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 3280, April 2002, http://www.ietf.org/rfc/rfc3280.txt

[RFC4120] Neuman, C., Yu, T., Hartman, S., and Raeburn, K., "The Kerberos Network Authentication Service (V5)", RFC 4120, July 2005, http://www.ietf.org/rfc/rfc4120.txt

[RFC4523] Zeilenga, K., "Lightweight Directory Access Protocol (LDAP) Schema Definitions for X.509 Certificates", RFC 4523, June 2006, http://www.ietf.org/rfc/rfc4523.txt

[RFC822] Crocker, D.H., "Standard for ARPA Internet Text Messages", STD 11, RFC 822, August 1982, http://www.ietf.org/rfc/rfc0822.txt

[X509] ITU-T, "Information Technology - Open Systems Interconnection - The Directory: Public-Key and Attribute Certificate Frameworks", Recommendation X.509, August 2005, http://www.itu.int/rec/T-REC-X.509/en

Note There is a charge to download the specification.

[X660] ITU-T, "Information Technology - Open Systems Interconnection - Procedures for the Operation of OSI Registration Authorities: General Procedures and Top Arcs of the ASN.1 Object Identifier Tree", Recommendation X.660, August 2004, http://www.itu.int/rec/T-REC-X.660/en

Note There is a charge to download the specification.

[X680] ITU-T, "Abstract Syntax Notation One (ASN.1): Specification of Basic Notation", Recommendation X.680, July 2002, http://www.itu.int/rec/T-REC-X.680/en

Note There is a charge to download the specification.

[X690] ITU-T, "Information Technology - ASN.1 Encoding Rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)", Recommendation X.690, July 2002, http://www.itu.int/rec/T-REC-X.690/en

Note There is a charge to download the specification.

1.2.2 Informative References

[CRYPTO] Menezes, A., Vanstone, S., and Oorschot, P., "Handbook of Applied Cryptography", 1997, http://www.cacr.math.uwaterloo.ca/hac/

[GRAY] Gray, J. and Reuter, A., "Transaction Processing: Concepts and Techniques", San Mateo, CA: Morgan Kaufmann Publishers, 1993, ISBN: 1558601902.

[MS-FASOD] Microsoft Corporation, "File Access Services Protocols Overview".

[MS-GLOS] Microsoft Corporation, "Windows Protocols Master Glossary".

[MSDN-ICERTPOLICY2] Microsoft Corporation, "ICertPolicy2 interface", http://msdn.microsoft.com/en-us/library/aa385034(VS.85).aspx

[MSFT-ARCHIVE] Microsoft Corporation, "Key Archival and Management in Windows Server 2003", December 2004, http://technet.microsoft.com/en-us/library/cc755395(v=ws.10).aspx

[MSFT-CRL] Microsoft Corporation, "Certificate Revocation and Status Checking", January 2006, http://technet.microsoft.com/en-us/library/bb457027.aspx

[MSFT-MODULES] Microsoft Corporation, "Policy and exit modules" Jan 2005, http://technet.microsoft.com/en-us/library/72e92b2d-80c1-4d61-9625-e00fbacb61db

[MSFT-TEMPLATES] Microsoft Corporation, "Implementing and Administering Certificate Templates in Windows Server 2003", July 2004, http://technet.microsoft.com/en-us/library/c25f57b0-5459-4c17-bb3f-2f657bd23f78

If you have any trouble finding [MSFT-TEMPLATES], please check here.

[PUBKEY] RSA Laboratories, "Crypto FAQ: Chapter 2 Cryptography: 2.1 Cryptographic Tools: 2.1.1 What Is Public-Key Cryptography?", http://www.rsa.com/rsalabs/node.asp?id=2165

[RSAFAQ] RSA Laboratories, "Frequently Asked Questions About Today's Cryptography, Version 4.1", May 2000, http://www.rsa.com/rsalabs/faq/files/rsalabs_faq41.pdf

1.3 Overview

The Certificate Services Remote Administration Protocol consists of a set of DCOM interfaces, as specified in [MS-DCOM], that allow administrative tools to configure the state and policy of a CA on a server. The administrative tools may perform such functions as getting or setting properties on a CA, retrieving data, revoking certificates, or retrieving escrowed **private keys** from a CA.

The following figure reflects only CA administration, not the normal operation of the CA. The protocol for the normal operation of the Microsoft CA is specified in [MS-WCCE].
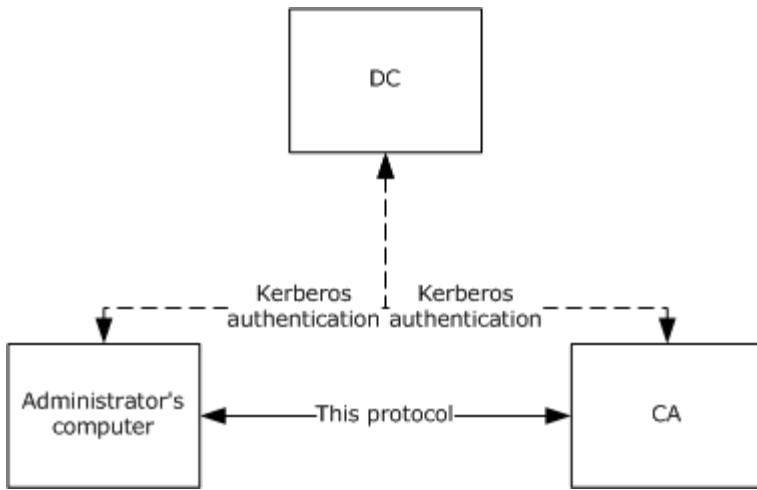
Figure 1: Machines involved in remote administration

In the preceding figure, the principal components are:

- CA: The certification authority (CA) that receives configuration and administration tasks. The remote administration protocol that is defined in this document covers the interactions that are shown as a solid line in this figure.

- Administrator's computer: A **client** to the CA that performs remote configuration or administration tasks.

- DC: An **Active Directory domain controller (DC)** includes a **Key Distribution Center (KDC)** as specified in [MS-KILE]. In most cases, a Kerberos KDC is used to authenticate the parties for authenticated DCOM messages. The protocol that is documented here is built on top of authenticated DCOM messages. Interactions with the DC are shown in the figure as dashed lines. DCOM is documented as specified in [MS-DCOM], which in turn references interactions with the DC.

The protocol uses two DCOM interfaces: **ICertAdminD (section 3.1.4.1)** and **ICertAdminD2 (section 3.1.4.2)**, which offer additional methods. The two interfaces define a total of 46 methods.

The methods of the Certificate Services Remote Administration Protocol fall into the following categories:

- Managing pending certificate requests: A certificate request can be fulfilled immediately or can be held for human administrator approval or other action. When a request is pending human approval, there are **ICertAdminD** methods that allow the human's administrative console to interact with the CA to query and modify pending requests. For additional information on pending requests, see section 3.1.1.1.1 and also [MS-WCCE].

- Configuring or retrieving data from CA databases: For purposes of this protocol, a CA must be built around a logical database, as specified in section 1.3.1.3. A number of methods in this protocol deal with configuration or data retrieval of particular rows or columns of **tables** in the logical database.

- Managing revocation: This protocol includes methods to tell the CA to revoke a certificate, to query the validity of a certificate, and to deal with the mechanics of publication of CRLs.

- Managing audit: This protocol includes methods that allow the administrator to learn and specify which classes of events generate audit trail entries.

- Archived **key** retrieval: This protocol defines one method for retrieving a private key that was archived as part of a certificate request.

- Miscellaneous administrative actions: This protocol includes a number of methods for miscellaneous administrative actions such as determining if the CA is responsive, determining what kinds of rights the caller has, telling the CA to go **offline**, or querying and editing various CA state variables. For details, see the descriptions in sections 3.1.4.1 and 3.1.4.2.

1.3.1 Concepts

The sections that follow define concepts and technologies used by the Certificate Services Remote Administration Protocol.

1.3.1.1 Number Annotation

Numbers expressed in the format 0xXXXX are to be interpreted as hexadecimal. Otherwise, all numbers are to be interpreted as decimal.

1.3.1.2 Object Identifiers

The protocol uses **object identifiers (OIDs)** as unique identifiers for several classes of **objects**, as specified in [X660] and [RFC3280] Appendix A. OIDs are used to uniquely identify **certificate templates** that are available to the CA. Within a certificate, OIDs are used to identify standard extensions, as specified in [RFC3280] section 4.2.1, and some nonstandard extensions.

1.3.1.3 CA Databases

This protocol refers to four databases as tables, each table with rows and columns hosted by the CA. There are two main tables: one for requests and one for CRLs. The Request table has two auxiliary tables: one for a list of **attributes** for a particular request, and one for a list of extensions for a particular request.

The following list contains additional details about the four tables:

- Request table: The Request table holds the history of all requests to the CA, both completed and pending, one row per request.

- Attribute table: The Attribute table holds the attributes, as specified in [RFC2986], that are contained within a specified certificate request.

- Extension table: The Extension table holds the X.509 extensions, as specified in [X509], that are contained within a specified certificate request.

- CRL table: The CRL table holds the revocation data and status for the CA. The CA maintains a CRL database in the form of a table that holds all CRLs (both base and delta, as defined in [RFC3280] section 5) that have been issued.

Methods of this protocol refer to the preceding four tables, which are specified in section 3.1.1.

1.3.1.4 CA Roles and Officer Rights

The Certificate Services Remote Administration Protocol includes methods to get and set **certificate authority (CA) roles** and **Officer rights** (as specified in sections 3.1.4.2.6, 3.1.4.2.7, 3.1.4.2.12,

and 3.1.4.2.13). CA roles are as specified in [CIMC-PP] section 5.2, and include administrator, operator, officer, and auditor. In addition, this protocol contains methods to assign **Enrollment Agent rights** on the CA. While "Enrollment Agent" can be considered a role, it is not one of the CA roles specified in [CIMC-PP].

1.3.1.5   Certificate Templates

An **enterprise certificate authority (CA)** MUST use certificate templates that are configured locally in order to support certificate **enrollment** requests, as specified in [MS-WCCE]. The complete definition of certificate templates, including the list of attributes, flags, and extensions that have been implemented in the Windows Server operating system, is specified in [MS-CRTD] and [MS-WCCE].

1.3.1.6   Sanitizing Common Names

The **common names (CNs)** of Active Directory objects, as specified in [MS-ADTS], that are used by the enrollment protocol are created by sanitizing the names of other objects and shortening the **sanitized name** so that it does not exceed 57 characters, including spaces. Objects are defined as a collection of **Lightweight Directory Access Protocol (LDAP)** attributes. Attributes are defined as LDAP data types, as specified in [RFC2251] and [RFC4523].

The sanitized name MUST NOT exceed 57 characters (bytes) in length. A name is sanitized by replacing the disallowed characters with an exclamation point ("!") that is followed by four hexadecimal digits, together which form one value that represents the 16-bit character being replaced.

In the following example, the opening parenthesis ("(") is replaced with !0028, the number sign ("#") is replaced with !0023, the percent sign ("%") is replaced with !0025, and the caret ("^") is replaced with !005e.

```
Original Name:
    'LongCAName(WithSpeci@#$%^Characters'
Sanitized Name:
    'LongCAName!0028WithSpeci@!0023$!0025!005eCharacters'
```

The algorithm for creating a sanitized name is specified in [MS-WCCE] section 3.1.1.4.1.1.

1.4   Relationship to Other Protocols

The Certificate Services Remote Administration Protocol depends on the Distributed Component Object Model (DCOM) Remote Protocol, as specified in [MS-DCOM]. The DCOM Remote Protocol is built on top of the Remote Procedure Call Protocol Extensions (RPCE), as specified in [MS-RPCE], and this protocol accesses RPCE directly to obtain certain security settings for the client-to-server connections.

This protocol uses the DCOM Remote Protocol to create and use DCOM object references to server objects as described in section 2.1 and [MS-DCOM] section 3.2.4.1. This protocol also uses the DCOM Remote Protocol to select authentication settings. The specific parameters passed from the Certificate Services Remote Administration Protocol to the DCOM Remote Protocol are specified in section 2.1.

Using input from a higher-layer protocol or application, the DCOM Remote Protocol negotiates its authentication method and settings by using the Generic Security Service Application Programming Interface (GSS-API), as specified in [RFC2478]. These settings are in turn passed to the activation

request and ORPC calls made by the DCOM client to the DCOM server, as specified in [MS-DCOM] sections 3.2.4.1.1.2 and 3.2.4.2.

This protocol depends on the Netlogon Remote Protocol Specification, as specified in [MS-NRPC], for locating the domain controller.

No other Windows protocol directly depends on the Certificate Services Remote Administration Protocol. However, this protocol is designed to manage a server that implements the Windows Client Certificate Enrollment Protocol, as specified in [MS-WCCE] as well as the ICertPassage Remote Protocol, as specified in [MS-ICPR]. Certificate Services Remote Administration Protocol shares an ADM with the ICertPassage Remote Protocol and the Windows Client Certificate Enrollment Protocol, as specified in sections 3.1.1.10, 3.1.3, 3.1.4, and 3.1.5. The Certificate Services Remote Administration Protocol, the Windows Client Certificate Enrollment Protocol, and the ICertPassage Remote Protocol use a common list of configuration data elements, defined in sections 3.1.1.6, 3.1.1.7, 3.1.1.8, 3.1.1.9, and 3.1.1.10.

The following diagram illustrates the layering of the protocol in this section with other protocols in its stack.
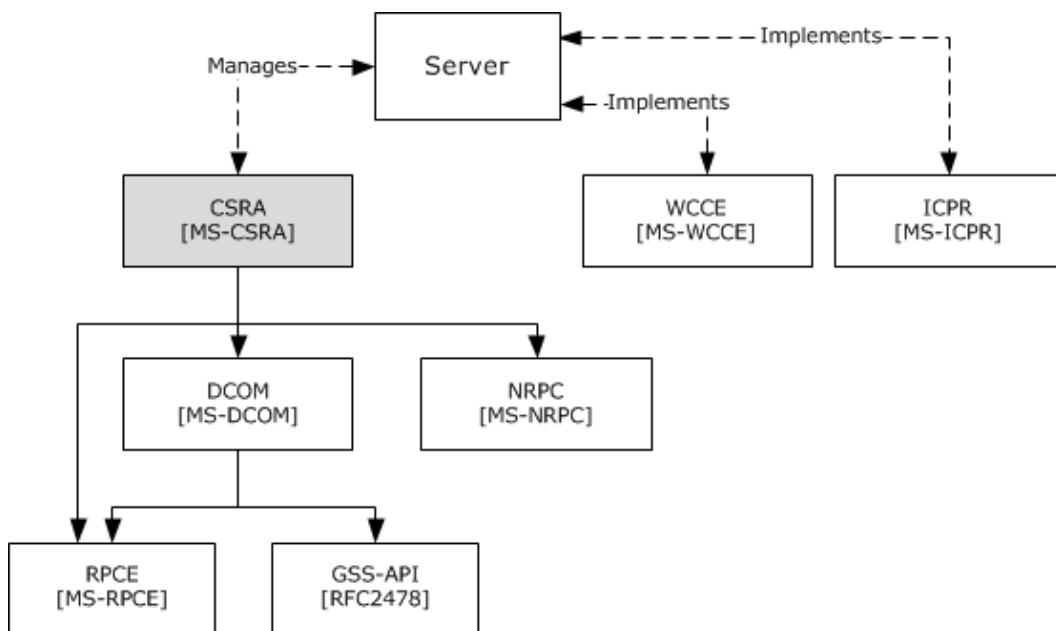
Figure 2: Relationship to other protocols

1.5 Prerequisites/Preconditions

1.5.1 Certificate Template

The Certificate Services Remote Administration Protocol enables the configuration, setting, and retrieval of properties on a CA. A CA can use templates in support of the Windows Certificate Services Enrollment Protocol, as specified in [MS-WCCE]. An enterprise CA requires valid templates that are configured on the CA. Information about certificate templates can be found in [MSFT-TEMPLATES].

1.5.2   CA Name

The Certificate Services Remote Administration Protocol assumes that the client knows the name of the CA server that implements the DCOM interfaces specified in section 3.2.4. Windows-based clients discover Microsoft CAs by reading the certificate enrollment object in Active Directory (as specified by [MS-ADTS]) and by using LDAP (as specified in [RFC2559]).

The enrollment object that defines the names of the CAs is located under the CN=Enrollment Services, CN=Public Key Services, CN=Services, CN=Configuration, DC=ForestRootDomain **container** of Active Directory. Each CA has an entry with a class of **pKIEnrollmentService**, as specified in [MS-ADSC] section 2.217.

The **cn** attribute of **pKIEnrollmentService** is the CA name. The **dNSHostName** attribute ([MS-ADA1] section 2.185) of **pKIEnrollmentService** contains the machine name that hosts the CA service.

1.5.3   Signing Certificate

The CA MUST have access to the entire Signing_Cert Table, including each CA **signing certificate**, defined in [MS-WCCE] section 3.2.1.1.2, and to the private key associated with the CA Exchange Certificate in the Current_CA_Exchange_Cert element, defined in [MS-WCCE] section 3.2.1.1.4.

1.5.4   Database

The tables and fields defined in section 3.1.1 are available.

1.5.5   Configuration

The configuration elements defined in section 3.1.1.10 are available. Each element defined in section 3.1.1.10 as "{Config_Element_Name}" has been initialized from its corresponding data element "OnNextRestart_{Config_Element_Name}" upon CA startup.

Certificate Services Remote Administration Protocol server implementations that also implement the Windows Client Certificate Enrollment Protocol or the ICertPassage Remote Protocol use the same configuration data elements for those implementations as those defined in sections 3.1.1.6, 3.1.1.7, 3.1.1.8, 3.1.1.9, and 3.1.1.10.

1.6   Applicability Statement

The Certificate Services Remote Administration Protocol provides clients with the capability to interact with CA for the purpose of managing X.509 certificates, as specified in [X509], or a CA configuration.

1.7   Versioning and Capability Negotiation

The Certificate Services Remote Administration Protocol is based on DCOM technology, as specified in [MS-DCOM], which provides capabilities to query for interface versions. Clients use the **IUnknown.QueryInterface** method to determine the supported server interface version. If **Certificate Services** supports **ICertAdminD2**, then **ICertAdminD2** is used; otherwise, **ICertAdminD** is used.

1.8 Vendor-Extensible Fields

This protocol uses HRESULT values as defined in [MS-ERREF] section 2.1.1. Vendors can define their own HRESULT values, provided they set the C bit (0x20000000) for each vendor-defined value, indicating the value is a customer code.

1.9 Standards Assignments

No standards assignments have been received for the Certificate Services Remote Administration Protocol described in this document.

All values used in these extensions are in private ranges. The following table contains the **remote procedure call (RPC)** interface **universally unique identifiers (UUIDs)** for all the interfaces that are part of the Certificate Services Remote Administration Protocol object model.

Constant/value	Description
d99e6e71-fc88-11d0-b498-00a0c90312f3	UUID for the ICertAdminD interface
7fe0d935-dda6-443f-85d0-1cfb58fe41dd	UUID for the ICertAdminD2 interface

2 Messages

2.1 Transport

DCOM, as specified in [MS-DCOM], is used as the transport protocol.

This protocol uses the DCOM Remote Protocol, to create and use DCOM object references to server objects.

Certificate Services Remote Administration Protocol clients initialize a connection to the Certificate Services Remote Administration server by creating and executing a DCOM activation request. As a result of this DCOM activation, the Certificate Services Remote Administration client can use the DCOM client to call the methods specified in this document. The activation process is detailed in [MS-DCOM] section 3.2.4.

The RPC version number for all interfaces MUST be 0.0.

[MS-DCOM] section 3.2.4.1 specifies the various elements that a DCOM-using application passes to the DCOM client as part of the initial activation request. Below are the values the Certificate Services Remote Administration Protocol client sends to the DCOM layer.

General DCOM settings:

- Remote server name, which is the application-supplied remote server name as specified in [MS-DCOM] section 3.2.4.1. The Certificate Services Remote Administration Protocol client sends the name of the CA server.

- Class identifier (CLSID) of the object requested. This value is d99e6e73-fc88-11d0-b498-00a0c90312f3.

- Interface identifier(s) (IID) of interface(s) requested.

  - ICertAdminD: d99e6e71-fc88-11d0-b498-00a0c90312f3

  - ICertAdminD2: 7fe0d935-dda6-443f-85d0-1cfb58fe41dd<2>

Security settings ([MS-DCOM] section 3.2.4.1.1.2)

- Security provider: RPC_C_AUTHN_GSS_NEGOTIATE (9)

- Authentication level: RPC_C_AUTHN_LEVEL_PKT_PRIVACY (6).

  As a result of the security provider and authentication level used, there is a negotiation between the client and server security providers that results in either NTLM, as specified in [MS-NLMP], or Kerberos, as specified in [RFC4120] and [MS-KILE], being used as the authentication method.

- Impersonation level: RPC_C_IMP_LEVEL_IMPERSONATE (3).

  This means the server can use the client's security context while acting on behalf of the client, to access local resources such as files on the server.

- Authentication identity and credentials: NULL.

Passing NULL authentication identity and credentials for the RPC_C_AUTHN_GSS_NEGOTIATE security provider means that the OPRC call uses the identity and credentials of the higher-layer application.<3>

Default values, as specified in [MS-DCOM], are used for all DCOM inputs not specified above, such as Security Principal Name (SPN), and client and prototype context property buffers and their context property identifiers.

2.2 Common Data Types

2.2.1 Common Structures

This section defines the structures used by the Certificate Services Remote Administration Protocol. These structures are used when performing various operations (using interface methods specified in section 3.1.4) on the server and as part of the server's response. This protocol shares a number of structures with the Windows Client Certificate Enrollment Protocol (as specified in [MS-WCCE]), which are specified in the following sections.

2.2.1.1 BYTE

The **BYTE** type specifies an 8-bit data item that corresponds to a single octet in a network protocol.

This type is declared as follows:

```
typedef byte BYTE;
```

2.2.1.2 VARIANT

The VARIANT type is implemented as specified in [MS-OAUT] section 2.2.29.

2.2.1.3 CERTVIEWRESTRICTION

The **CERTVIEWRESTRICTION** structure is used to restrict the data set that is returned by the CA server during calls to the **OpenView** method for the **ICertAdminD** interface.

This structure is passed by RPC technology, as specified in [MS-RPCE], and does not need special marshaling.

```
typedef struct _CERTVIEWRESTRICTION {
  DWORD ColumnIndex;
  LONG SeekOperator;
  LONG SortOrder;
  [size_is(cbValue), unique] BYTE* pbValue;
  DWORD cbValue;
} CERTVIEWRESTRICTION;
```

ColumnIndex: An unsigned integer value that specifies the identifier for the database column that is receiving the restriction.

SeekOperator: An integer value that specifies the logical operator of the data-query qualifier for the column. This parameter MUST be set to one of the following values.

Value	Meaning
0x00000001	Equal to
0x00000002	Less than
0x00000004	Less than or equal to
0x00000008	Greater than or equal to
0x00000010	Greater than

SortOrder: An integer value that specifies the sort order for the column. This parameter MUST be set to one of the following values.

Value	Meaning
0x00000000	No sort order
0x00000001	Ascending
0x00000002	Descending

pbValue: A pointer to a byte array that specifies the value against which the value in the corresponding column (specified by **ColumnIndex**) is compared, using SeekOperator.

cbValue: An unsigned integer value that specifies the length of the byte array that is pointed to by the **pbValue** field.

2.2.1.4   CERTTRANSBLOB

The **CERTTRANSBLOB** structure defines a byte buffer that is used to store and request certificates, transmit responses, manipulate Unicode strings, and marshal property values.

```
typedef struct _CERTTRANSBLOB {
  ULONG cb;
  [size_is(cb), unique] BYTE* pb;
} CERTTRANSBLOB;
```

cb: An unsigned integer value that MUST contain the length, in bytes, of the buffer that is pointed to by **pb**.

pb: The **BYTE** buffer that contains the binary contents being transported in this **CERTTRANSBLOB**. That content consists of any of the following entities:

- A certificate.

- A certificate request.

- CA properties.

- Any common structure that is defined in section 2.2.1 other than VARIANT or **CERTVIEWRESTRICTION**.

- Any common structure that is defined in [MS-WCCE] section 2.2.2.

The **CERTTRANSBLOB** structure is empty when **cb** is set to 0 and **pb** is set to NULL.

The marshaling of other structures that can be passed in the **pb** byte buffer of **CERTTRANSBLOB** is defined in [MS-WCCE] section 2.2.2.

All instances of **CERTTRANSBLOB** that are used by this protocol MUST use the marshaling rules that are described in the following sections or in [MS-WCCE] section 2.2.2.

2.2.1.5 CATRANSPROP

The CATRANSPROP structure encapsulates information about a CA property. The CATRANSPROP structure and the marshaling of one or more CATRANSPROP structures into a **CERTTRANSBLOB** structure is specified in [MS-WCCE] section 2.2.2.3.

2.2.1.6 CAINFO

Defines a basic informational block describing a CA. The structure of **CAINFO** is specified in [MS-WCCE] section 2.2.2.4. The marshaling of CAINFO into a **CERTTRANSBLOB** structure is specified in [MS-WCCE] section 2.2.2.2.5.

2.2.1.7 CERTTRANSDBCOLUMN

The **CERTTRANSDBCOLUMN** structure is encoded within a **CERTTRANSBLOB** structure. The **CERTTRANSDBCOLUMN** structure contains **schema** information about a particular database column that is associated with a specific table to the client. This associated table is invoked when the client queries the **EnumViewColumn** or **EnumViewColumnTable** method of the **ICertAdminD** and **ICertAdminD2** interfaces, respectively.

```
typedef struct _CERTTRANSDBCOLUMN {
  DWORD Type;
  DWORD Index;
  DWORD cbMax;
  ULONG obwszName;
  ULONG obwszDisplayName;
} CERTTRANSDBCOLUMN;
```

Type: This field describes the column. The high and low WORDs are split and used separately.

0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5	6	7	8	9	3 0	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	I	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Where the bits are defined as:

Value	Description
I Column indexing WORD	The high WORD of the **Type** field is a bit field. Only bit 15 is used. If the **Type** field is set to 1, it indicates that the column is indexed for lookup purposes.

Value	Description
0 Reserved byte	Reserved byte
0 Column value type byte	The low WORD of the **Type** field is divided into two bytes: ▪ The high byte of the low WORD MUST be set to 0 and MUST be ignored by the server on receipt. ▪ The low byte of the low WORD MUST specify the value type for the column that is associated with a specific table by using one of the following values.

Value	Meaning
0x01	The Column type is a signed integer.
0x02	The Column type is a date.
0x03	The Column type is binary data.
0x04	The Column type is a string.

Index: An unsigned integer value that specifies the identifier for the column in the server database.

cbMax: An unsigned integer value that specifies the maximum data size, in bytes, that this column can contain.

obwszName: An integer that contains the offset from the beginning of the byte array buffer that is pointed to by the **pb** field in the containing **CERTTRANSBLOB** structure, to where the string that contains the name of this column can be found. The string format is a null-terminated UNICODE string. The offset MUST be divisible by 4.

obwszDisplayName: An integer that contains the offset from the beginning of the byte array buffer that is pointed to by the **pb** field in the containing **CERTTRANSBLOB** structure, to where the string that contains the display name of this column can be found. The string format is a null-terminated UNICODE string. The offset MUST be divisible by 4.

2.2.1.7.1   CERTTRANSDBCOLUMN Marshaling Format

The **CERTTRANSDBCOLUMN** structure is encoded within the byte array that is referenced by the **pb** member of a **CERTTRANSBLOB** structure.

A packet that contains an array of N **CERTTRANSDBCOLUMN** structures is specified in the following table.

The value of "N" is a separate return parameter for the EnumViewColumn and EnumViewColumnTable methods.

The **Type**, **Index**, **cbMax**, **obmszName**, and **obwszDisplayName** fields are included in the **CERTTRANSDBCOLUMN** structure. These structures MUST be contiguous and MUST NOT be padded. All structures MUST appear prior to any information about the column schema data, which appears in the **Column_Schema_Data** byte array at the end of the packet.

0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5	6	7	8	9	3 0	1
Column_1_Type_Value																														
Column_1_Index_Value																														
Column_1_cbMax_Value																														
Column_1_obwzName_Offset																														
Column_1_obwzDisplayName_Offset (variable)																														
...																														
Column_N_Type_Value																														
Column_N_Index_Value																														
Column_N_cbMax_Value																														
Column_N_obwzName_Offset																														
Column_N_obwzDisplayName_Offset																														
Column_Schema_Data (variable)																														
...																														

Column_1_Type_Value (4 bytes): The value indicating the type for the first column. The value MUST be **little-endian** encoded.

Column_1_Index_Value (4 bytes): The value indicating the ID for the first column. The value MUST be little-endian encoded.

Column_1_cbMax_Value (4 bytes): The maximum length of data that this column can contain. The value MUST be little-endian encoded.

Column_1_obwzName_Offset (4 bytes): The offset from the beginning of the byte array buffer that is pointed to by the **pb** field in the containing **CERTTRANSBLOB** structure to where the string that contains the name of this column can be found. The format is a null-terminated Unicode string. The offset MUST be divisible by 4. The offset value MUST be little-endian encoded.

Column_1_obwzDisplayName_Offset (variable): The offset from the beginning of the byte array buffer that is pointed to by the **pb** field in the containing **CERTTRANSBLOB** structure to where the string that contains the display name of this column can be found. The format is a null-terminated Unicode string. The offset MUST be divisible by 4. The offset value MUST be little-endian encoded.

Column_N_Type_Value (4 bytes): The value indicating the type for the Nth column. The value MUST be little-endian encoded.

Column_N_Index_Value (4 bytes): The value indicating the ID for the Nth column. The value MUST be little-endian encoded.

Column_N_cbMax_Value (4 bytes): The maximum length of data that this column can contain. The value MUST be little-endian encoded.

Column_N_obwzName_Offset (4 bytes): The offset from the beginning of the byte array buffer that is pointed to by the **pb** field in the containing **CERTTRANSBLOB** structure to where the string that contains the name of this column can be found. The format is a null-terminated Unicode string. The offset MUST be divisible by 4. The offset value MUST be little-endian encoded.

Column_N_obwzDisplayName_Offset (4 bytes): The offset from the beginning of the byte array buffer that is pointed to by the **pb** field in the containing **CERTTRANSBLOB** structure to where the string that contains the display name of this column can be found. The format is a null-terminated Unicode string. The offset MUST be divisible by 4. The offset value MUST be little-endian encoded.

Column_Schema_Data (variable): Contains the schema data for all columns that are referenced by the obwzName and obwszDisplayName fields of the **CERTTRANSDBCOLUMN** structures. Schema data for an individual column MUST NOT overlap with any other data. Arbitrary padding can be inserted between data values. Schema data MUST be little-endian encoded for each character of the null-terminated UNICODE string.

2.2.1.8   CERTTRANSDBATTRIBUTE

The **CERTTRANSDBATTRIBUTE** structure is encoded within a **CERTTRANSBLOB** structure. The **CERTTRANSDBATTRIBUTE** structure is used by the server to return attribute information that is associated with a request to the client (upon the client's query via invocation of the **EnumAttributesOrExtensions** method of the **ICertAdminD** interface).

```
typedef struct _CERTTRANSDBATTRIBUTE {
  ULONG obwszName;
  ULONG obwszValue;
} CERTTRANSDBATTRIBUTE;
```

obwszName: An integer that contains the offset from the beginning of the byte array buffer that is pointed to by the **pb** field in the containing **CERTTRANSBLOB** structure to where the string that contains the name of this attribute can be found. The format is a null-terminated UNICODE string. The offset MUST be divisible by 4.

obwszValue: An integer that contains the offset from the beginning of the byte array buffer that is pointed to by the **pb** field in the containing **CERTTRANSBLOB** structure to where the string that contains the value of this attribute can be found. The format is a null-terminated UNICODE string. The offset MUST be divisible by 4.

2.2.1.8.1   CERTTRANSDBATTRIBUTE Marshaling Format

A packet containing an array of N **CERTTRANSDBATTRIBUTE** structures is specified below.

The value of "N" is a separate return parameter for the EnumAttributesOrExtensions method.

The **obwzName** and **obwzValue** fields constitute the attribute header. Attribute headers MUST be contiguous and MUST NOT be padded. All attribute headers MUST appear prior to any column data, which appears in the attribute data byte array at the end of the packet.

0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5	6	7	8	9	3 0	1
Attribute 1 obwzName																															
Attribute 1 obwzValue (variable)																															
...																															
Attribute N obwzName																															
Attribute N obwzValue																															
Attribute Data (variable)																															
...																															

Attribute 1 obwzName (4 bytes): The offset from the beginning of the byte array buffer that is pointed to by the **pb** field in the containing **CERTTRANSBLOB** structure to where the string that contains the name of this attribute can be found. The format is a null-terminated UNICODE string. The offset MUST be divisible by 4. The offset MUST be little-endian encoded.

Attribute 1 obwzValue (variable): The offset from the beginning of the byte array buffer that is pointed to by the **pb** field in the containing **CERTTRANSBLOB** structure to where the string that contains the value of this attribute can be found. The format is a null-terminated UNICODE string. The offset MUST be divisible by 4. The offset MUST be little-endian encoded.

Attribute N obwzName (4 bytes): The offset from the beginning of the byte array buffer that is pointed to by the **pb** field in the containing **CERTTRANSBLOB** structure to where the string that contains the name of this attribute can be found. The format is a null-terminated UNICODE string. The offset MUST be divisible by 4. The offset MUST be little-endian encoded.

Attribute N obwzValue (4 bytes): The offset from the beginning of the byte array buffer that is pointed to by the **pb** field in the containing **CERTTRANSBLOB** structure to where the string that contains the value of this attribute can be found. The format is a null-terminated UNICODE string. The offset MUST be divisible by 4. The offset MUST be little-endian encoded.

Attribute Data (variable): Contains the data for all attributes. Data for individual attributes MUST NOT overlap with any other attribute data. Attribute name and value are string type. The data MUST use little-endian encoding format for a null-terminated UNICODE string.

2.2.1.9   CERTTRANSDBEXTENSION

The **CERTTRANSDBEXTENSION** structure is encoded within a **CERTTRANSBLOB** structure. The **CERTTRANSDBEXTENSION** structure is used by the server to return certificate extension information, as specified in [RFC3280] section 4, that is associated with a request. This associated request to the client occurs when the client performs a query by invoking the **EnumAttributesOrExtensions** method of the **ICertAdminD** interface.

```
typedef struct _CERTTRANSDBEXTENSION {
  ULONG obwszName;
  LONG ExtFlags;
  DWORD cbValue;
  ULONG obValue;
} CERTTRANSDBEXTENSION;
```

obwszName: An unsigned integer that contains the offset from the beginning of the byte array buffer that is pointed to by the **pb** field in the containing **CERTTRANSBLOB** structure to the string representation of an OID of this extension (as specified in [X680]). The string format is a null-terminated UNICODE string. The offset MUST be divisible by 4.

ExtFlags: An integer value that specifies the flags that are associated with the extension. The following table shows its contents.

0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5	6	7	8	9	3 0	1
C	D	0	0	0	0	0	0	0	0	0	0	0	0	0	0					Nigiro											

Mirrored (Nigiro) byte	Meaning
0x8000	The extension comes from the request.
0x4000	The extension was added by the CA. The CA assigns a value of 2 if the extension was added by the policy module of the CA.
0xC000	The extension was added by the CA. The CA assigns a value of 3 if the extension was added interactively by a human administrator of the CA.
0x2000	The extension was added by the CA. The CA assigns a value of 4 if the extension was added by the certificate server engine and not the policy module component of the CA.
0xA000	The extension was in the preceding certificate, which might occur, for example, when a certificate is renewed.
0x6000	The extension comes from an imported certificate (a certificate that was imported into the CA database).
0xE000	The extension comes from a PKCS7 request.
0x1000	The extension comes from a CMC request.
0x9000	The extension comes from the current CA signing certificate.

The **Flags** byte uses the 2 least-significant bits as follows:

- C is the ExtensionCriticalFlag, as defined in section 3.1.1.3.

    - A value of 0 means the extension is not critical.

- A value of 1 means the extension is critical.

- D is the ExtensionDisabledFlag as defined in section 3.1.1.3.

  - A value of 0 means the extension is not disabled.

  - A value of 1 means the extension is disabled.

cbValue: An unsigned integer value that contains the length, in bytes, of data that is referenced by the *obValue* parameter.

obValue: An unsigned integer that contains the offset from the beginning of the byte array buffer that is pointed to by the **pb** field in the containing **CERTTRANSBLOB** structure to where the value for this extension can be found. The length of the value is specified in the **cbValue** field. The value is in ASN.1 Distinguished Encoding Rules (DER) format for the extension, as specified in [X660]. The offset MUST be divisible by 4.

2.2.1.9.1 CERTTRANSDBEXTENSION Marshaling Format

A packet containing an array of N **CERTTRANSDBEXTENSION** structures is specified in the following packet diagram.

The value of "N" is a separate return parameter for the EnumAttributesOrExtensions method.

The extension header must include the **obwzName**, **extFlag**, **cbValue**, and **obValue** fields. Extension headers MUST be contiguous and MUST NOT be padded. All extension headers MUST appear prior to any extension data, which appears in the **Extension_Data** byte array at the end of the packet.

0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5	6	7	8	9	3 0	1
Extension_1_obwzName																															
Extension_1_extFlag																															
Extension_1_cbValue																															
Extension_1_obValue (variable)																															
...																															
Extension_N_obwzName																															
Extension_N_extFlag																															
Extension_N_cbValue																															
Extension_N_obValue																															
Extension_Data (variable)																															
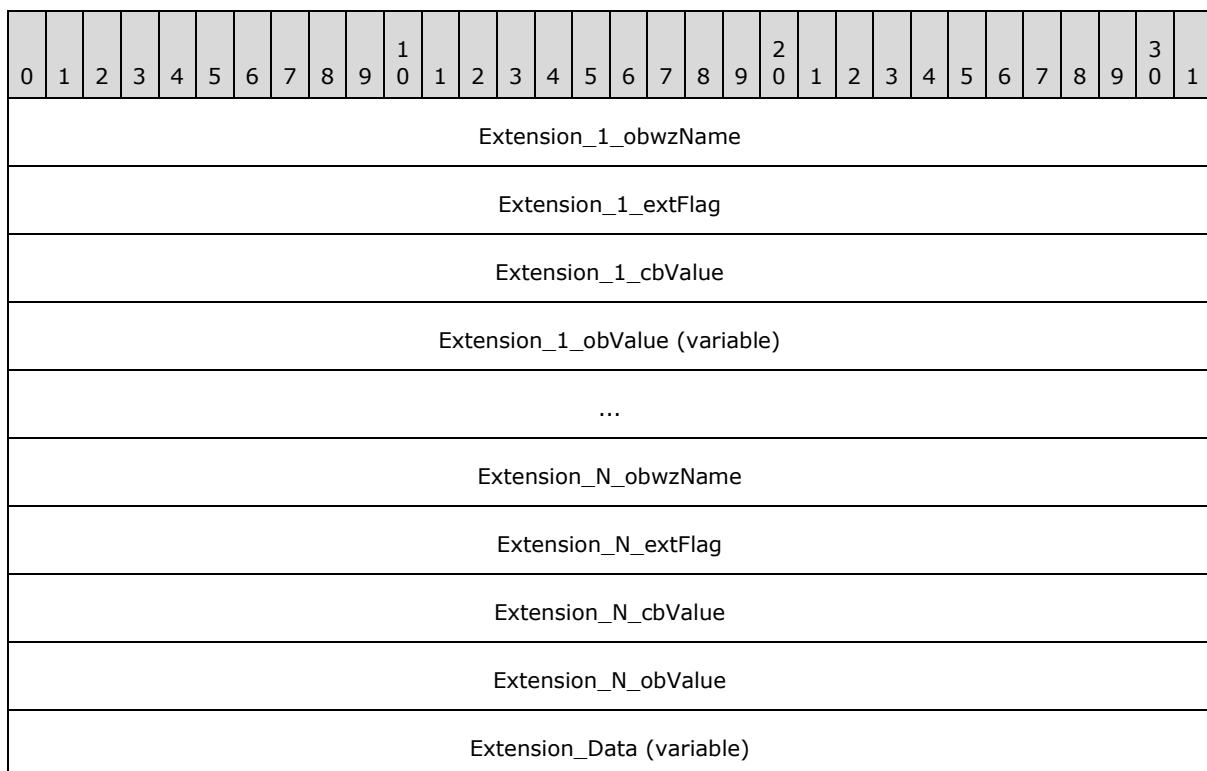
Extension_1_obwzName (4 bytes): The offset from the beginning of the byte array buffer that is pointed to by the **pb** field in the containing **CERTTRANSBLOB** structure to the string that contains the name of this **CERTTRANSDBEXTENSION**. The offset MUST be little-endian encoded. The offset MUST be divisible by 4.

Extension_1_extFlag (4 bytes): An integer value that specifies the flags that are associated with the extension. The value MUST be little-endian encoded.

Extension_1_cbValue (4 bytes): The length of the data in **Extension_1** referenced by **obValue** (offset). The value MUST be little-endian encoded.

Extension_1_obValue (variable): The offset from the beginning of the byte array buffer that is pointed to by the **pb** field in the containing **CERTTRANSBLOB** structure to where the value for this extension can be found. The offset MUST be divisible by 4. The offset value MUST be little-endian encoded.

Extension_N_obwzName (4 bytes): The offset from the beginning of the byte array buffer that is pointed to by the **pb** field in the containing **CERTTRANSBLOB** structure to the string that contains the name of this extension. The offset MUST be little-endian encoded. The offset MUST be divisible by 4.

Extension_N_extFlag (4 bytes): An integer value that specifies the flags that are associated with the extension. The value MUST be little-endian encoded.

Extension_N_cbValue (4 bytes): The length of the data in **Extension_N** referenced by **obValue** (offset). The length value MUST use little-endian encoding format.

Extension_N_obValue (4 bytes): The offset from the beginning of the byte array buffer that is pointed to by the **pb** field in the containing **CERTTRANSBLOB** structure to where the value for this extension can be found. The offset MUST be divisible by 4. The offset value MUST be little-endian encoded.

Extension_Data (variable): Contains the data for all extensions. Data for individual extensions MUST NOT overlap with any other extension data. An extension name data MUST use little-endian encoding format for a null-terminated UNICODE string. An extension value data is a byte array.

2.2.1.10   CERTTRANSDBRESULTCOLUMN

The **CERTTRANSDBRESULTCOLUMN** structure is encoded within a **CERTTRANSBLOB** structure. The **CERTTRANSDBRESULTCOLUMN** structure is used by the server to return the result of a CA database query that is done by the client (upon the client's query via invocation of the **OpenView** or **EnumView** method of the **ICertAdminD** interface).

The **OpenView** and **EnumView** methods return data in the form of a CERTTRANSBLOB structure whose **pb** member points to an array of one or more **CERTTRANSDBRESULTROW** structures. Each **CERTTRANSDBRESULTROW** structure contains one or more **CERTTRANSDBRESULTCOLUMN** structures.

The **CERTTRANSDBRESULTCOLUMN** structure contains data for a specific column in a specific row.

```
typedef struct _CERTTRANSDBRESULTCOLUMN {
```

```
    DWORD Type;
    DWORD Index;
    ULONG obValue;
    DWORD cbValue;
} CERTTRANSDBRESULTCOLUMN;
```

Type: This field describes the column. The high and low WORDs are split and used separately.

0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5	6	7	8	9	3 0	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	I	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Where the bits are defined as:

Value	Description
I Column indexing WORD	The high WORD of the **Type** field is a bit field. Only bit 15 is used. If the **Type** field is set to 1, it indicates that the column is indexed for lookup purposes.
0 Reserved byte	Reserved byte
0 Column value type byte	The low WORD of the **Type** field is divided into two bytes: ▪ The high byte of the low WORD MUST be set to 0 and MUST be ignored by the server on receipt. ▪ The low byte of the low WORD MUST specify the value type for the column that is associated with a specific table by using one of the following values.

Value	Meaning
0x01	The Column type is a signed integer.
0x02	The Column type is a date.
0x03	The Column type is binary data.
0x04	The Column type is a string.

Index: An unsigned integer value that specifies the identifier for the column in the relevant table.

obValue: An unsigned integer that contains the offset from the beginning of the corresponding **CERTTRANSDBRESULTROW** structure to where the value for this column can be found. The length of the value is specified in the **cbValue** field. The offset MUST be DWORD aligned.

cbValue: An unsigned integer value that specifies the length, in bytes, of the value for the specific column.

2.2.1.10.1   CERTTRANSDBRESULTCOLUMN Marshaling Format

The **CERTTRANSDBRESULTCOLUMN** structure is encoded within a CERTTRANSBLOB structure such that the **pb** member of the CERTTRANSBLOB points to the beginning of an array of one or more **CERTTRANSDBRESULTROW** structures, each of which contains one or more **CERTTRANSDBRESULTCOLUMN** structures.

A packet containing an array of N **CERTTRANSDBRESULTCOLUMN** structures is specified as follows. N is the value of the corresponding **cCol** member of **CERTTRANSDBRESULTROW**.

0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5	6	7	8	9	3 0	1
Result_Column_1_Type																															
Result_Column_1_Index																															
Result_Column_1_obValue																															
Result_Column_1_cbValue (variable)																															
...																															
Result_Column_N_Type																															
Result_Column_N_Index																															
Result_Column_N_obValue																															
Result_Column_N_cbValue																															
Result_Column_Data (variable)																															
...																															

Result_Column_1_Type (4 bytes): The value indicating the type for the first column. The value MUST be little-endian encoded.

Result_Column_1_Index (4 bytes): The value indicating the ID for the first column. The value MUST be little-endian encoded.

Result_Column_1_obValue (4 bytes): The offset from the start of the corresponding **CERTTRANSDBRESULTROW** structure to the Result Column 1 data. The offset MUST be little-endian encoded. The offset MUST be divisible by 4.

Result_Column_1_cbValue (variable): The length of the data in Result Column 1 referenced by **obValue** (offset). The length value MUST be little-endian encoded.

Result_Column_N_Type (4 bytes): The value indicating the type for the Nth column. The value MUST be little-endian encoded.

Result_Column_N_Index (4 bytes): The value indicating the ID for the Nth column. The value MUST be little-endian encoded.

Result_Column_N_obValue (4 bytes): The offset from the start of the corresponding **CERTTRANSDBRESULTROW** structure to the data for Result Column N. The offset MUST be little-endian encoded. The offset MUST be divisible by 4.

Result_Column_N_cbValue (4 bytes): The length of the data in Result Column #N referenced by obValue (Offset). The length value MUST be little-endian encoded.

Result_Column_Data (variable): Contains the data for all columns. Data for individual columns MUST NOT overlap with any other column data. Arbitrary padding can be inserted between data values. Based on the value of the **Type** field, the data value for the column MUST be encoded as follows.

Type field value	Column type	Data encoding
0x01	Integer	MUST use little-endian encoding format.
0x02	Date	MUST use little-endian encoding format.
0x03	Binary	
0x04	String	MUST use little-endian encoding format for each character of the null-terminated UNICODE string.

2.2.1.11   Officer and Enrollment Agent Access Rights

Officer and Enrollment Agent access rights structures are used by the server to return the results of a client query; for example, the client's invocation of the GetOfficerRights method of the ICertAdminD2 interface.

Officer rights and Enrollment Agent rights are security descriptors. Security descriptor structures (SID structures) are defined in [MS-DTYP] section 2.4.2. Officer rights and Enrollment Agent rights security descriptors have the following properties:

1. Each **access control entry (ACE)** in the **discretionary access control list (DACL)** MUST have:

    1. AceType 0x9 (ACCESS_ALLOWED_CALLBACK_ACE_TYPE)

    2. AccessMask 0x0001000

2. The ACE contains additional application data following the SID.

The format for the additional application data is as follows.

0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5	6	7	8	9	3 0	1
SIDCount																															
Array of SIDs (variable)																															

...
TemplateName (variable)
...

SIDCount (4 bytes): A little-endian encoded **DWORD** that contains the count of the SID structures following it.

Array of SIDs (variable): An array of SID structures that identify either (i) **principals** for whom the officer can approve requests; or (ii) principals on whose behalf the enrollment agent can obtain certificates. For an Officer rights security descriptor, case (i) applies. For an Enrollment Agent rights security descriptor, case (ii) applies. SID structures are as defined in [MS-DTYP] section 2.4.2.

TemplateName (variable): A little-endian encoded Unicode string that identifies the common name (CN) of the template (as defined in [MS-CRTD]) for which the officer is authorized to approve requests.

2.2.1.11.1 Marshaling Format for Officer and Enrollment Agent Rights

The marshaling of Officer rights and Enrollment Agent rights into a **CERTTRANSBLOB** structure depends on whether the server supports Enrollment Agent rights. CA implementors can determine whether to support Enrollment Agent rights. There is no requirement to support them.

If Enrollment Agent rights are not supported by the server, the **pb** member of the **CERTTRANSBLOB** structure refers to the Officer rights security descriptor (as defined in [MS-DTYP] section 2.4.2), and the **cb** member contains the length of the marshaled data.

If Enrollment Agent rights are supported by the server, the **CERTTRANSBLOB** structure is created as follows:

1. If Officer rights are enabled and Enrollment Agent rights are disabled, the **pb** member of the **CERTTRANSBLOB** structure refers to the following marshaled structure.

```
struct {
SECURITY_DESCRIPTOR OfficerRights;
DWORD bEARightDisabled;
};
```

Member	Value
OfficerRights	
bEARightsDisabled	(0x00000000)

OfficerRights: A marshaled security descriptor for officer rights.

bEARightsDisabled: A little-endian encoded **DWORD**. This value MUST be 0x00000000.

There is no padding or **DWORD** boundary requirement.

Release: Friday, October 25, 2013

2. If Officer rights are disabled and Enrollment Agent rights are disabled, the **pb** member of the **CERTTRANSBLOB** structure refers to the following marshaled structure.

```
struct {
DWORD bEARightDisabled;
};
```

Member	Value
bEARightsDisabled	(0x00000000)

bEARightsDisabled: A little-endian encoded **DWORD**. This value MUST be 0x00000000.

There is no padding or **DWORD** boundary requirement.

3. If Officer rights are disabled and Enrollment Agent rights are enabled, the **pb** member of the **CERTTRANSBLOB** structure refers to the following marshaled structure.

```
struct {
DWORD bEARightEnabled;
SECURITY_DESCRIPTOR EnrollmentAgentRights;
};
```

Member	Value
bEARightsEnabled	Must be nonzero.
EnrollAgentRights	

bEARightsEnabled: A little-endian encoded **DWORD**. This value MUST be nonzero.

Enrollment Agent Rights: A marshaled security descriptor for Enrollment Agent rights.

There is no padding or **DWORD** boundary requirement.

4. If Officer rights are enabled and Enrollment Agent rights are enabled, the **pb** member of the **CERTTRANSBLOB** structure refers to the following marshaled structure.

```
struct {
SECURITY_DESCRIPTOR OfficerRights;
DWORD bEARightEnabled;
SECURITY_DESCRIPTOR EnrollmentAgentRights;
};
```

Member	Value
OfficerRights	
bEARightsEnabled	Must be nonzero.
EnrollAgentRights	

OfficerRights: A marshaled security descriptor for Officer rights.

bEARightsEnabled: A little-endian encoded **DWORD**. The value MUST be nonzero.

Enrollment Agent Rights: A marshaled security descriptor for Enrollment Agent rights.

There is no padding or **DWORD** boundary requirement.

2.2.1.12   CERTTIME

The CERTTIME type consists of a 64-bit value that represents the number of 100-nanosecond intervals since January 1, 1601 (the start of the Gregorian calendar), according to **Coordinated Universal Time (UTC)**.

2.2.2   Certificate Requirements

2.2.2.1   CA Exchange Certificate

The Certificate Services Enrollment Protocol requires that the CA provide a CA **exchange certificate** for the purpose of client private **key archival** during the certificate enrollment process. A CA exchange certificate must be provided in the form of an X.509 digital certificate.

2.2.2.2   Key Recovery Certificate

A key recovery certificate is a prerequisite for certificate enrollment that encapsulates a private key for the purposes of **key escrow** to a CA.<4> A CA MAY use one or more locally configured and specified **key recovery certificates** to encrypt the private key of a client submitted to the CA encapsulated in a certificate enrollment request.

A key recovery certificate contains the following X.509v1 fields:

- Version

- Serial Number

- Signature Algorithm

- Valid From

- Valid To

- Subject

- Issuer

- Public Key

A key recovery certificate contains the following X.509v3 extensions identified in section 4.2.1 of [RFC3280]:

- Authority Key Identifier

- Subject Key Identifier

- Authority Information Access

- Key Usage (Key Encipherment = 0x20)

- Subject Alternative Name

- CDP (CRL Distribution Point)

- Extended Key Usage (Key Recovery OID = 1.3.6.1.4.1.311.21.6)

2.2.3   CERTTRANSDBRESULTROW

The **CERTTRANSDBRESULTROW** structure is encoded within a **CERTTRANSBLOB** structure. The **CERTTRANSDBRESULTROW** structure is used by the server to return the result of the database query done by the client (upon the client's query via invocation of **OpenView** or **EnumView** methods of the **ICertAdminD** interface). This structure contains data for a specific row.

```
typedef struct _CERTTRANSDBRESULTROW {
  DWORD RowId;
  DWORD ccol;
  ULONG cbrow;
} CERTTRANSDBRESULTROW;
```

RowId: Unsigned integer value that specifies the identifier for the row.

ccol: Unsigned integer value that specifies the count of **CERTTRANSDBRESULTCOLUMN** structures. Each structure contains the value of a specific column in the row identified by **RowId**.

cbrow: Unsigned integer value that specifies the total size of row data (in bytes). This is the sum of the size of **CERTTRANSDBRESULTROW** structure, size of each **CERTTRANSDBRESULTCOLUMN** structure for the row (the count of which is specified by ccol), and the DWORD-rounded-up size of each column value.

2.2.3.1   CERTTRANSDBRESULTROW Marshaling Format

The **CERTTRANSDBRESULTROW** packet is specified as follows.

The Row header includes the **RowId**, **ccol**, **Type**, and **cbRow** fields. A complete Row header MUST appear prior to any row data, which appears in the row data byte array at the end of the packet. Row data is composed of one or more **CERTTRANSDBRESULTCOLUMN** structures. The count of structures is specified in the **cCol** field value.

0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5	6	7	8	9	3 0	1
Row_1_RowID																																
Row_1_ccol																																
Row_1_cbRow																																
Array_of_CERTTRANSDBRESULTCOLUMN (variable)																																
...																																
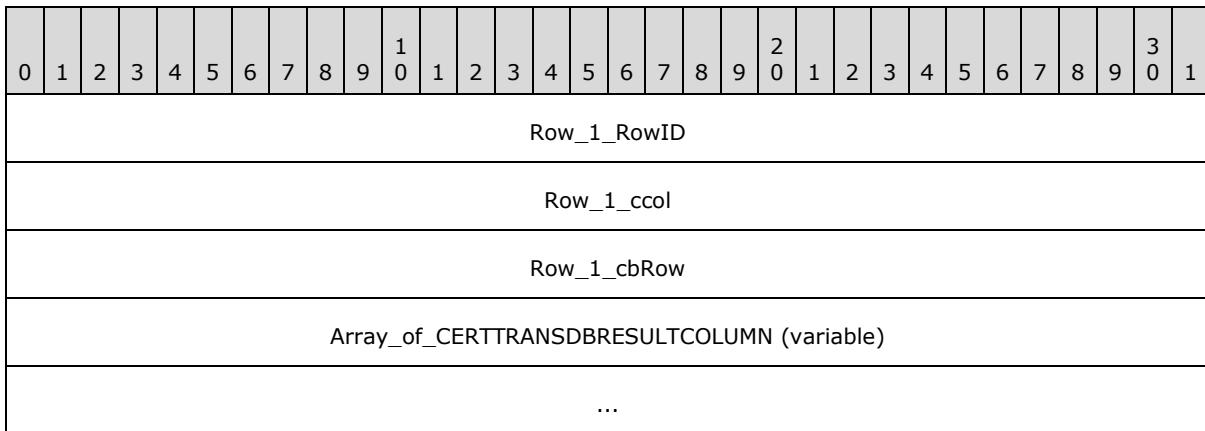
Row_1_RowID (4 bytes): The value identifying the row. The value MUST be little-endian encoded.

Row_1_ccol (4 bytes): The value indicating the number of columns in **Row_1**. The value MUST be little-endian encoded.

Row_1_cbRow (4 bytes): The total length, in bytes, of the **Row_1** data. The length value MUST be little-endian encoded.

Array_of_CERTTRANSDBRESULTCOLUMN (variable): An array of **CERTTRANSDBRESULTCOLUMN** structures, as specified in section 2.2.1.10.

2.2.4   Database File Name Structure

A specific format is used for representing the names of the database files with this protocol. The string is defined as follows.

db-filename = file-type-code UNC-path

file-type-code = "!" / "D"

UNC-path = path as specified by **UNC**

The file name strings MUST be null-terminated Unicode strings. The file-type-code is specified as follows.

Value	Meaning	Example
'D'	Database files	"D\\server\sharepoint\...path..."
'!'	Database **log files**	"!\\server\sharepoint\...path..."

2.2.5   Common Error Codes

The following error codes are used by this protocol to indicate specific error conditions. Other error values are possible and are implementation-specific.

Note Error names in this table that also appear in [MS-ERREF] have been redefined for use when implementing this protocol.

Return value/code	Description
0x80070057 ERROR_INVALID_PARAMETER	The parameter is incorrect.
0x80070006 ERROR_INVALID_HANDLE	The handle is invalid.
0x8000FFFF ERROR_UNEXPECTED_ERROR	An unexpected error occurred.
0x80004003 ERROR_INVALID_POINTER	An invalid pointer.
0x80071392 ERROR_OBJECT_EXISTS	An object already exists.

Release: Friday, October 25, 2013

Return value/code	Description
0x00000001 ERROR_ARITHMETIC_OVERFLOW	Arithmetic overflow.
0xc800020D ERROR_INVALID_BACKUP_SEQUENCE	An invalid backup sequence.
0xc800042D ERROR_OUT_OF_MEMORY	Out of memory.

2.3 Directory Service Schema Elements

This protocol accesses the directory service schema classes and attributes that are listed in the following table. For the syntactic specifications of the following class or class/attribute pairs, refer to Active Directory Domain Services (AD DS) in [MS-ADA1], [MS-ADA2], [MS-ADA3], and [MS-ADSC].

Class	Attribute
cRLDistributionPoint	certificateRevocationList deltaRevocationList
pKIEnrollmentService	certificateTemplate

3   Protocol Details

The Certificate Services Remote Administration Protocol is a request-response protocol. The client performs a server method invocation and the server responds with the requested data or a detailed disposition value. The primary usage of this protocol is CA management. Except where specified in the following section the protocol is a single message followed by a single reply.

3.1   Server Details

3.1.1   Abstract Data Model

This section details the data that is maintained by the CA. The Request table, Attribute table, Extension table, and CRL table data elements are implemented in the form of tables with columns. Each column has a numerical column identifier that is unique across all the data tables and is stored as an unsigned integer. Each column has a column name that is identified in the following sections.<5>

3.1.1.1   Request Table

The Request table holds the history of all CA requests, both completed and pending, with one row per request.

3.1.1.1.1   Request Table Required Data Elements

Values for the following elements are required in the Request table:

Request_Request_ID: Column name "Request.RequestID". A field that is used to uniquely identify the request in the table and to link to the Attribute and Extension tables. This field MUST have a positive value.

Request_Raw_Request: Column name "Request.RawRequest". The raw request, as delivered by ICertRequestD or ICertRequestD2, as specified in [MS-WCCE] section 2.2.2.6.

Request_Disposition: Column name "Request.Disposition". Identifies the certificate status.

Possible values are listed as follows. This specification refers to these abstract values as these strings. An implementation is free to use any representation for these values.<6>

Value	Description
Request failed	A certificate was never issued in response to the certificate request. The request cannot be resubmitted.
Request denied	A certificate was never issued in response to the certificate request. The request can be resubmitted by using the ResubmitRequest method.
Request pending	The request is in a state in which the decision whether to issue a certificate has not yet been made. The request can be denied by using the DenyRequest method. The request can be submitted again for issuance by using the ResubmitRequest method.
Certificate issued	A certificate was issued in response to the certificate request.
Certificate revoked	A certificate was issued, and subsequently the RevokeCertificate function was called, specifying the serial number of the certificate, in such a way that server

Value	Description
	processing rule #6 of the RevokeCertificate function was executed.
Foreign certificate	A certificate was issued by a different CA and then imported into the CA by using the **ImportCertificate** method.

Raw_Certificate: Column name "RawCertificate". The certificate that was issued for a request (if a certificate was issued).

Request_Raw_Archived_Key: Column name "Request.RawArchivedKey". Any private key that is archived as part of a certificate request. Archived keys are generally encrypted with a **key recovery agent (KRA)** key. The format for the encrypted private key is specified in [MS-WCCE] section 3.1.1.4.3.5.

Request_Revocation_Date: Column name "Request.RevokedEffectiveWhen". This CERTTIME field is used as the revocationDate for a certificate in a CRL (CRL as specified in [RFC3280] section 5.1). This field is initialized as NULL and updated by the **RevokeCertificate** method. The CA does not put a certificate serial number in a base or delta certificate revocation list (CRL) until the time specified in the Request_Revocation_Date has passed for that certificate.

The Request_Revocation_Date field does not follow all the **revocationDate** rules that are defined in [RFC3280] because the Certificate Services Remote Administration Protocol client can specify and change this field when necessary. This flexibility is achieved by client calls to the **RevokeCertificate** functionality of the server, which is described in this protocol specification.

Certificates may be retroactively revoked; they may be revoked again by specifying any revocation date. They may also be changed to a state in which applications that verify revocation do not recognize the certificate as revoked. This state is achieved by specifying a future date for the revocation, including future dates that are subsequent to the expiry date that is recorded in the certificate.

Multiple requirements of [RFC3280] can be violated by the preceding behaviors. The Request_Revocation_Date can differ from the "date on which the revocation occurred..." (referenced in [RFC3280] section 5.1.2.6) or can differ from "...the date at which the CA processed the revocation" (referenced in [RFC3280] section 5.3.3). Changing revocation to a future date that is beyond the date of the next CRL violates the requirements of [RFC3280] section 3.3. This section states in part, "An entry MUST NOT be removed from the CRL until it appears on one regularly scheduled CRL issued beyond the revoked certificate's validity period". Note that a CRL cannot list a certificate that is not yet revoked, as determined by its revocation date. Retroactive revocation - that is, using a revocation date that is prior to the issuance of one or more existing CRLs-violates the rule that the "revocation date SHOULD NOT precede the date of issue of earlier CRLs", as defined in [RFC3280] section 5.3.3.

Request_Revoked_Reason: Column name "Request.RevokedReason". When a certificate has been revoked, this element provides the reason for the revocation.

The Request_Revoked_Reason is similar to the reasonCode that is specified in [RFC3280] section 5.3.1, except that the protocol supports only a subset of the values that the RFC defines, and it supports one additional value, as shown in the following table.

Revocation reason	Value
unspecified	0x00000000

Release: Friday, October 25, 2013

Revocation reason	Value
keyCompromise	0x00000001
cACompromise	0x00000002
affiliationChanged	0x00000003
superseded	0x00000004
cessationOfOperation	0x00000005
certificateHold	0x00000006
removeFromCRL	0x00000008
Release from hold	0xffffffff

Serial_Number: Column name "SerialNumber". The issued certificate serial number.

Publish_Expired_Cert_In_CRL: Column name "PublishExpiredCertInCRL". This Request table column specifies whether the certificate whose serial number is identified in Serial_Number should be included in CRLs if the certificate is revoked, even after it has expired.

This Request table column is a Boolean value, as shown in the following table.

Value	Description
1	The certificate whose serial number is identified in Serial_Number should be included in CRLs if it is revoked, even after the certificate has expired.
0	The revoked certificate should not be included in CRLs after it has expired.

3.1.1.1.2 Request Table Optional Data Elements

Values for the following elements of the Request table SHOULD be maintained by the CA:

Request_Key_Recovery_Hashes: Column name "Request.KeyRecoveryHashes". Unique identifiers of the key recovery agent (KRA) certificates that are required to retrieve an archived private key.

Request_Raw_Old_Certificate: Column name "Request.RawOldCertificate". In the case of a renewal, the preceding certificate.

Request_Request_Attributes: Column name "Request.RequestAttributes". The certificate request attributes as defined in [MS-WCCE].

Request_Request_Type: Column name "Request.RequestType". The type or format of a certificate request, such as PKCS#10 or the Cryptographic Message Syntax (CMS) standard with Common Messaging Calls (CMC) as specified in [RFC2797].

Request_Request_Flags: Column name "Request.RequestFlags". Additional certificate request information.

The following are examples of request flag values. These flag values can be used in any combination.

Name	Value	Description
CR_FLG_FORCETELETEX	0x00000001	For encoding of the subject information in the certificate, a T61String type is used for elements that contain a Unicode character in the value.
CR_FLG_RENEWAL	0x00000002	The certificate request is a renewal request.
CR_FLG_FORCEUTF8	0x00000004	For encoding of the subject information in the certificate, a UTF8String type is used for elements that contain a Unicode character in the value.
CR_FLG_CAXCHGCERT	0x00000008	The certificate is the exchange certificate of the CA.
CR_FLG_ENROLLONBEHALFOF	0x00000010	The certificate request is an Enroll-on-behalf-of request.
CR_FLG_SUBJECTUNMODIFIED	0x00000020	The subject information in the certificate is an unmodified binary copy of the subject information from the certificate request.
CR_FLG_VALIDENCRYPTEDKEYHASH	0x00000040	For a certificate request with key archival, the CMC Full response includes the szOID_ENCRYPTED_KEY_HASH attribute.
CR_FLG_CACROSSCERT	0x00000080	The certificate is the cross certificate of the CA.
CR_FLG_ENFORCEUTF8	0x00000100	For encoding of the subject information in the certificate, a UTF8String type is used for directory string elements.
CR_FLG_DEFINEDCACERT	0x00000200	The certificate request contains an Authority Key Identifier extension that identifies the desired CA signing key for the certificate.
CR_FLG_CHALLENGEPENDING *	0x00000400	An attestation challenge ([MS-WCCE] section 3.2.2.6.2.1.2.6) for the corresponding certificate request has been sent to the client, and the server is waiting for a response.
CR_FLG_CHALLENGESATISFIED *	0x00000800	The client responded correctly to the attestation challenge for the corresponding certificate request ([MS-WCCE] section 3.2.2.6.2.1.2.7).
CR_FLG_TRUSTONUSE *	0x00001000	Verification of the requester's credentials for key attestation has succeeded ([MS-WCCE] section 3.2.2.6.2.1.2.5).
CR_FLG_TRUSTEKCERT *	0x00002000	Verification of the client's TPM endorsement certificate for key attestation has succeeded ([MS-WCCE] section 3.2.2.6.2.1.2.5).
CR_FLG_TRUSTEKKEY *	0x00004000	Verification of the public key of the client's TPM's endorsement key pair for key

Name	Value	Description
		attestation has succeeded ([MS-WCCE] section 3.2.2.6.2.1.2.5).
CR_FLG_PUBLISHERROR	0x80000000	The CA had difficulty publishing the certificate to the directory that is specified in the **userCertificate** attribute of the entity.

* Support for these flags is specified in the following product behavior note.<7>

Request_Status_Code: Column name "Request.StatusCode". Indicates whether the request was successful.

The value is 0 if the request processed successfully. Otherwise, this field contains an error code that results from request processing. Error codes are as specified in section 2.2.5 of this document and in [MS-ERREF].

Request_Disposition_Message: Column name "Request.DispositionMessage". The text description of Request_Disposition. Request_Disposition_Message is for presentation to a user and can contain any text string, including NULL, that the implementer considers informative.

Request_Submitted_When: Column name "Request.SubmittedWhen". The CERTTIME that a request was received by the CA.

Request_Resolved_When: Column name "Request.ResolvedWhen". The CERTTIME that the CA completed request processing (whether successfully or unsuccessfully).

Request_Revoked_When: Column name "Request.RevokedWhen". The CERTTIME that the CA processed a call to the **ICertAdminD::RevokeCertificate** function. This field is initialized as NULL and updated by the **ICertAdminD::RevokeCertificate** function.

Request_Requester_Name: Column name "Request.RequesterName". The RequesterName that is included in the certificate request.

Request_Caller_Name: Column name "Request.CallerName". The user or machine context that submitted the certificate request to the CA.

Request_Signer_Policies: Column name "Request.SignerPolicies". The list of valid certificate policy OIDs for each signer certificate from the certificate request.

Request_Signer_Application_Policies: Column name "Request.SignerApplicationPolicies". The list of valid Extended Key Usage OIDs for each signer certificate from the certificate request.

Request_Officer: Column name "Request.Officer". Indicates whether the caller is the certificate manager of the entity that corresponds to the Request_Requester_Name.

Request_Distinguished_Name: Column name "Request.DistinguishedName". The **distinguished name (DN)** from the Subject attribute of the certificate request (string representation).

Request_Raw_Name: Column name "Request.RawName". Subject information from the certificate request (ASN.1 DER encoded).

Request_Country: Column name "Request.Country". The country attribute of the DN from the Subject of the certificate request.

Request_Organization: Column name "Request.Organization". The organization attribute of the DN from the Subject of the certificate request.

Request_Org_Unit: Column name "Request.OrgUnit". The organizational-unit attribute of the DN from the Subject of the certificate request.

Request_Common_Name: Column name "Request.CommonName". The common name attribute of the DN from the Subject of the certificate request.

Request_Locality: Column name "Request.Locality". The locality attribute of the DN from the Subject of the certificate request.

Request_State: Column name "Request.State". The state or province name attribute of the DN from the Subject of the certificate request.

Request_Title: Column name "Request.Title". The title attribute of the DN from the Subject of the certificate request.

Request_Given_Name: Column name "Request.GivenName". The given name (also called first name) attribute of the DN from the Subject of the certificate request.

Request_Initials: Column name "Request.Initials". The initials attribute of the DN from the Subject of the certificate request.

Request_SurName: Column name "Request.SurName". The surname attribute of the DN from the Subject of the certificate request.

Request_Domain_Component: Column name "Request.DomainComponent". The domainComponent attribute of the DN from the Subject of the certificate request.

Request_Email: Column name "Request.EMail". The EmailAddress attribute of the DN from the Subject of the certificate request.

Request_Street_Address: Column name "Request.StreetAddress". The street address attribute of the DN from the Subject of the certificate request.

Request_Unstructured_Name: Column name "Request.UnstructuredName". The unstructured name attribute of the DN from the Subject of the certificate request.

Request_Unstructured_Address: Column name "Request.UnstructuredAddress". The unstructured address attribute of the DN from the Subject of the certificate request.

Request_Device_Serial_Number: Column name "Request.DeviceSerialNumber". The device serial number attribute of the DN from the Subject of the certificate request.

Request_RequesterName_From_Old_Certificate: Column name "Request.RequesterNameFromOldCertificate". For a renewal request that is signed by the previously issued certificate, the subject name of the old certificate.<8>

Request_Attestation_Challenge: Column name "Request.AttestationChallenge". The secret passed to the client in the attestation challenge message, encrypted with the CA exchange certificate.

Request_Endorsement_Key_Hash: Column name "Request.EndorsementKeyHash". The SHA-2 hash of the public key that was used to TPM-attest the request.

Request_Endorsement_Certificate_Hash: Column name "Request.EndorsementCertificateHash". The SHA2 hash of the endorsement certificate used to TPM-attest the request.

Request_ID: Column name "RequestID". The RequestID that corresponds to an issued certificate.

Certificate_Hash: Column name "CertificateHash". The **SHA-1 hash** over the value of the Raw_Certificate column.

Certificate_Template: Column name "CertificateTemplate". extnValue of extension with OID 1.3.6.1.4.1.311.20.2 of issued certificate.

Enrollment_Flags: Column name "EnrollmentFlags". The values that are defined in "EnrollmentFlags" from [MS-CRTD].

General_Flags: Column name "GeneralFlags". The values that are defined in "GeneralFlags" from [MS-CRTD].

Issuer_Name_Id: Column name "IssuerNameId". A sequential number that indicates which CA key signed the issued certificate.

Not_Before: Column name "NotBefore". The CERTTIME that provides the value for the Validity->notBefore field ([RFC3280] section 4.1.2.5) of the issued certificate.

Not_After: Column name "NotAfter". The CERTTIME that provides the value for the Validity->notAfter field ([RFC3280] section 4.1.2.5) of the issued certificate.

Subject_Key_Identifier: Column name "SubjectKeyIdentifier". The SubjectKeyIdentifier extension ([RFC3280] section 4.2.1.2) of the issued certificate.

Raw_Public_Key: Column name "RawPublicKey". The SubjectPublicKeyInfo->subjectPublicKey field of the issued certificate.

Public_Key_Length: Column name "PublicKeyLength". The length of the SubjectPublicKeyInfo->subjectPublicKey field of the issued certificate.

Public_Key_Algorithm: Column name "PublicKeyAlgorithm". The SubjectPublicKeyInfo->algorithm->algorithm field of the issued certificate.

Raw_Public_Key_Algorithm_Parameters: Column name "RawPublicKeyAlgorithmParameters". The SubjectPublicKeyInfo->algorithm->parameters field of the issued certificate.

UPN: Column name "UPN". The UPN alternate name entry from the SubjectAltName extension in the certificate.

Distinguished_Name: Column name "DistinguishedName". The Subject field ([RFC3280] section 4.1.2.6) of the issued certificate (string representation).

Raw_Name: Column name "RawName". The Subject information of the issued certificate (ASN.1 DER encoded).

Country: Column name "Country". The country attribute of the certificate Subject.

Organization: Column name "Organization". The organization attribute of the certificate Subject.

Org_Unit: Column name "OrgUnit". The organizational-unit attribute of the certificate Subject.

Common_Name: Column name "CommonName". The common name attribute of the certificate Subject.

Locality: Column name "Locality". The locality attribute of the certificate Subject.

State: Column name "State". The state or province name attribute of the certificate Subject.

Title: Column name "Title". The title attribute of the certificate Subject.

Given_Name: Column name "GivenName". The given name attribute of the certificate Subject.

Initials: Column name "Initials". The initials attribute of the certificate Subject.

SurName: Column name "SurName". The surname attribute of the certificate Subject.

Domain_Component: Column name "DomainComponent". The domainComponent attribute of the certificate Subject.

Email: Column name "EMail". The [RFC822] Name attribute from the Subject Alternative Name of the issued certificate.

Street_Address: Column name "StreetAddress". The street address attribute of the certificate Subject.

Unstructured_Name: Column name "UnstructuredName". The unstructured name attribute of the certificate Subject.

Unstructured_Address: Column name "UnstructuredAddress". The unstructured address attribute of the certificate Subject.

Device_Serial_Number: Column name "DeviceSerialNumber". The serial number attribute of the certificate Subject.

3.1.1.2   Attribute Table

A request includes an arbitrary number of attributes, as specified in [MS-WCCE] section 2.2.2.7, that are parsed and listed in the Attribute table.

Each entry in the table represents an attribute that is associated with a request and has the following required data elements:

Attribute_Request_ID: Column name "AttributeRequestId". This is the same unique identifier as the Request_Request_ID that is associated with a request in the Request table.

Attribute_Name: Column name "AttributeName". Contains the name of the attribute.

For any rows that have the same Attribute_Request_ID, the Attribute_Name MUST be unique for those rows within the table.

Attribute_Value: Column name "AttributeValue". Contains the value of the attribute.

3.1.1.3   Extension Table

A request includes an arbitrary number of X.509v3 extensions, as specified in [X509]. These extensions are parsed and listed in the Extension table.

Each entry in the table represents an extension that is associated with a request and has the following required data elements:

Extension_Request_ID: Column name "ExtensionRequestId". This is the same unique identifier as the Request_Request_ID that is associated with a request in the [Request](#) table.

Extension_Name: Column name "ExtensionName". An OID, as specified in [X680], that contains the name of the extension. The Extension_Name must be the string representation of the OID that is associated with the extension. If two rows have the same value in the Extension_Name field, they MUST NOT have the same value in the Request_ID field.

Extension_Value: Column name "ExtensionValue". The value of the extension.

The value of the extension is in ASN.1 DER format (as specified in [X660] and [X690]).

Extension_Raw_Value: Column name "ExtensionRawValue". The value of the extension.

Extension_Flags: Column name "ExtensionFlags". A flag that indicates the following:

- Whether the specified extension is to be marked critical.

- Whether this extension is enabled.

3.1.1.4 Certificate Revocation List (CRL) Table

CRL_Local: A multi-valued element that contains one or more CRLs.

The CA MUST also maintain a data element called **CRL_Local** that can contain one or more CRLs. This element is read and populated by the processing rules detailed in section [3.1.4.1.6](#). The **CRL_Local** data element is stored in the default local registry location specified in section [3.1.1.8](#) for the elements **Config_CA_CDP_Publish_To_Base** and **Config_CA_CDP_Publish_To_Delta**.[<9>](#)

Each entry in the table represents a CRL and has associated properties, which are described in the following sections.

3.1.1.4.1 CRL Table Required Data Elements

If the CA maintains a [CRL table](#), values for the following elements of the CRL table MUST be maintained:

CRL_Row_Id: Column name "CRLRowId". The unique identifier for the CRL in the table.

CRL_Name_Id: Column name "CRLNameId". The sequential number that indicates which CA key the CRL is for. For example, if a CA certificate has been renewed with a new key three times and the CA issues a CRL for each key, the **CRLNameId** field can be used to distinguish each of the four issued CRLs.

CRL_Raw_CRL: Column name "CRLRawCRL". The CRL that was issued.

CRL_Min_Base: Column name "CRLMinBase". The CRL_Number of the CRL, complete for a given scope, that was used as the starting point in the generation of this delta CRL.

If the value is 0, the CRL is a base CRL.

If the value is not 0, the CRL is a delta CRL.

CRL_Publish_Status_Code: Column name "CRLPublishStatusCode". An informational field that identifies whether the CA was able to publish the CRL to locations external to the CA server.

If the CRL was published successfully, this field contains 0. Otherwise, the field contains the return code from the first CRL publishing location to which publishing failed, following the last publishing attempt for the CRL. Common error codes are as specified in section 2.2.5; other error values are specified in [MS-ERREF].

CRL_This_Publish: Column name "CRLThisPublish". The CERTTIME at which a CRL is first created and published.

CRL_Propagation_Complete: Column name "CRLPropagationComplete". The estimated CERTTIME when the CRL is expected to have propagated to all servers after publishing. This data element is updated when a CRL is created and is used when creating a delta CRL. The delta CRL is based on the last base CRL that completely propagated.

CRL_Number: Column name "CRLNumber". The sequential number that is incremented each time a new base CRL is created. If the CA creates a single base CRL for multiple CA keys, the CRLs for all the associated CA keys have the same CRL number.

CRL_Count: Column name "CRLCount". The count of CRL entries in the CRL.

CRL_This_Update: Column name "CRLThisUpdate". The CERTTIME that provides the value of the thisUpdate field of the CRL. This value is also used to restrict the selection of revoked certificates whose serial numbers should be included on a delta CRL as specified in section 3.1.4.1.6.

CRL_Next_Update: Column name "CRLNextUpdate". The CERTTIME that provides the value of the nextUpdate field of the CRL.

CRL_Next_Publish: Column name "CRLNextPublish". The CERTTIME that provides the value of the nextPublish extension of the CRL.

CRL_Publish_Flags: Column name "CRLPublishFlags". Additional CRL information that was sent to the PublishCRLs method or returned from attempts to publish CRLs.

For the CRL_Publish_Flags element, the following values are defined:

Flag	Value
CPF_BASE	0x1 – A base CRL.
CPF_DELTA	0x2 – A delta CRL.
CPF_COMPLETE	0x4 – The CRL published successfully.
CPF_MANUAL	0x40 – The caller who initiated the generation of the CRL (via the **PublishCRLs** method) was running as an interactive user and was not launched by a timer on the CA.
CPF_SHADOW	0x8 – A blank delta CRL with new delta CRL indicator extension (CRL_Min_Base value). When delta CRLs have just been disabled (Config_Delta_CRL_Validity_Period has just been set to 0), the CA publishes this type of CRL to force clients to retrieve a new base CRL.
CPF_BADURL_ERROR	0x20 – A **URI** that does not meet the format

Flag	Value
	requirements specified in section 3.1.1.8 for Config_CA_CDP_Publish_To_Base and Config_CA_CDP_Publish_To_Delta was encountered during publishing of the CRL.
CPF_FILE_ERROR	0x200 – A file URI that does not meet the format requirements specified in section 3.1.1.8 for Config_CA_CDP_Publish_To_Base and Config_CA_CDP_Publish_To_Delta for a file location was encountered during publishing of the CRL, or the CA encountered an error trying to write the CRL to a file location.
CPF_HTTP_ERROR	0x800 – An HTTP URI was encountered during publishing of the CRL. The Windows CA does not write to http:// locations, so any http:// CRL publish attempt will cause this flag.
CPF_FTP_ERROR	0x400 – An FTP URI was encountered during publishing of the CRL. The Windows CA does not write to ftp:// locations, so any ftp:// CRL publish attempt will cause this flag.
CPF_LDAP_ERROR	0x100 –The CA encountered an error trying to write the CRL to an LDAP location.
CPF_POSTPONED_BASE_LDAP_ERROR	0x1000 – Postponed publishing a delta CRL due to a failure in publishing a base CRL to an ldap:/// location. For example, the Microsoft CA sends this flag with a call to publish a delta CRL when the corresponding base CRL could not be published to an LDAP location because of an error.
CPF_POSTPONED_BASE_FILE_ERROR	0x2000 – Postponed publishing a delta CRL due to a failure in publishing a base CRL to a file:// location. For example, the Microsoft CA sends this flag with a call to publish a delta CRL when the corresponding base CRL could not be published to a FILE location because of an error.
CPF_SIGNATURE_ERROR	0x80 – An error occurred when verifying the signature of the generated CRL prior to attempting to publish the CRL.
CPF_CASTORE_ERROR	0x10 – An error occurred when publishing the generated CRL to the default local registry location.

3.1.1.4.2   CRL Table Recommended Data Elements

If the CA maintains a CRL table, values for the following elements of the CRL table SHOULD be maintained by the CA:

CRL_Effective: Column name "CRLEffective". The CERTTIME at which the validity period of the CRL identified by CRL_Min_Base began. Used only for delta CRLs.

CRL_Publish_Attempts: Column name "CRLPublishAttempts". The number of times an attempt to publish the CRL was made.

CRL_Last_Published: Column name "CRLLastPublish". The CERTTIME at which the CRL was last published, or the last attempt to publish the CRL was made.

CRL_Publish_Error: Column name "CRLPublishError". CRL_Publish_Error contains the user name(s) in whose context the CRLs was published and any CRLs publishing location(s) to which the CRLs could not successfully be published.

3.1.1.5 Schema Table

The Schema table contains information about the data elements and their properties, which were defined in the preceding sections, and information about any vendor-defined elements.

The Schema table has the following required elements:

Schema_Column_Name: The internal name of the column. An abstract data model (ADM) element name or another unique name can be used.

Schema_Column_Display_Name: The display name of the column.

Schema_Column_ID: The database unique numeric identifier of the column.

Schema_Column_Type: The data type in the column.

This data element has four possible values, as shown in the following table.

Value	Data type
1	Integer
2	Date
3	Binary
4	String

Schema_Column_Type_Index: The indexing type, that is, whether the column is indexed.

This data element has two values.

Value	Description
0	The column is not indexed.
1	The column is indexed.

Schema_Column_Type_Max_Value: The maximum size, in bytes, of the data that the column can contain.

Schema_Column_In_View: The views (or "default column sets") the column is in.

This data element can have any of the following values in any combination:

- Issued certificates

- Pending requests

- Failed requests

- Extensions

- Attributes

- Certificate revocation lists (CRLs)

- Revoked certificates

3.1.1.6 Datum - DB View

Every CA that conforms to this specification MUST maintain a datum called **Config_Database_View_Open**, which is shared from the Config_Database_View_Open Boolean defined in [MS-WCCE] section 3.2.1.1.4, for each DCOM instance.

Certificate Services Remote Administration Protocol server implementations that also implement the Windows Client Certificate Enrollment Protocol or the ICertPassage Remote Protocol use the same data element for those implementations.

The Config_Database_View_Open datum indicates whether a caller has opened a view to the database. This datum has two possible values:

- **False**

- **True**

Every CA that conforms to this specification MUST initialize the value of Config_Database_View_Open to the default value of **False**.

3.1.1.7 Permissions

The CA SHOULD store the following sets of permissions. Certificate Services Remote Administration Protocol server implementations that also implement the Windows Client Certificate Enrollment Protocol or the ICertPassage Remote Protocol use the same configuration data element, defined here, for those implementations:

Config_Permissions_CA_Security: A list, shared from the Config_Permissions_CA_Security list defined in [MS-WCCE] section 3.2.1.1.4.

Config_Permissions_Officer_Rights: A list, shared from the Config_Permissions_Officer_Rights list defined in [MS-WCCE] section 3.2.1.1.4.

Config_Permissions_Enrollment_Agent_Rights: A list, shared from the Config_Permissions_Enrollment_Agent_Rights list defined in [MS-WCCE] section 3.2.1.1.4.

The permissions are used to enforce that the caller has particular permissions for any method specified in section 3.1.4.

A Windows CA defines six permissions: Enroll, Read, Officer, Administrator, Operator, and Auditor.<10>

For CA security (GetCASecurity, SetCASecurity, and GetMyRoles), the Microsoft CA assigns permissions to principals (identified by the access control entry (ACE)) in the following manner.

Permission	Access Mask Bit value
Read	0x00000100

Permission	Access Mask Bit value
Enroll	0x00000200
Officer	0x00000002
Administrator	0x00000001
Auditor	0x00000004
Operator	0x00000008

If a principal has Enroll, Officer, or Administrator permission, Read permission is implied and does not need to be explicitly set.

For the CA Operator role that is defined in [CIMC-PP], a principal must have Read permission (implicit or explicit) and must also have the SeBackupPrivilege, as specified in [MS-LSAD] section 3.1.1.2.1.

For the CA Auditor role that is defined in [CIMC-PP], a principal must have Read permission (implicit or explicit) and must also have the SeSecurityPrivilege, as specified in [MS-LSAD] section 3.1.1.2.1.

The following table specifies the method name and the list of permissions required by the caller. With the exception of where mentioned, the caller only needs to possess at least one of these access permissions for the call to be allowed by the CA.

Method name	Permissions required
ICertRequestD::Request	Enroll
ICertRequestD:GetCACert	Enroll
ICertRequestD2::Request2	Enroll
ICertRequestD2::GetCAProperty	Enroll
ICertRequestD2::GetCAPropertyInfo	Enroll
ICertRequestD2::GetCAProperty	Enroll
ICertRequestD2::GetCAPropertyInfo	Enroll
ICertAdminD::GetCRL	Administrator, Officer, Read
ICertAdminD2::GetCAProperty	Administrator, Officer, Read
ICertAdminD2::GetCAPropertyInfo	Administrator, Officer, Read
ICertAdminD::GetViewDefaultColumnSet	Administrator, Officer, Read
ICertAdminD::EnumAttributesOrExtensions	Administrator, Officer, Read
ICertAdminD::OpenView	Administrator, Officer, Read
ICertAdminD::IsValidCertificate	Administrator, Officer, Read
ICertAdminD::GetServerState	None required
ICertAdminD2::GetCASecurity	Administrator, Officer, Read

Method name	Permissions required
ICertAdminD2::GetAuditFilter	Administrator, Officer, Read
ICertAdminD2::GetOfficerRights	Administrator, Officer, Read
ICertAdminD2::GetConfigEntry	Administrator, Officer, Read
ICertAdminD2::EnumViewColumnTable	Administrator, Officer, Read
ICertAdminD2::GetMyRoles	Administrator, Officer, Read
ICertAdminD2::GetArchivedKey	Officer
ICertAdminD::SetExtension	Officer
ICertAdminD::SetAttributes	Officer
ICertAdminD::DenyRequest	Officer
ICertAdminD::ReSubmitRequest	Officer
ICertAdminD::RevokeCertificate	Officer
ICertAdminD::ImportCertificate	Officer
ICertAdmin:D2:ImportKey	Officer
ICertAdminD2::PublishCRLs	Administrator
ICertAdminD::ServerControl	Administrator, Operator
ICertAdminD::Ping	Administrator
ICertAdminD::Ping2	Administrator
ICertAdminD2::SetCASecurity	Administrator
ICertAdminD2::SetCAProperty	Administrator
ICertAdminD2::SetAuditFilter	Administrator, Auditor (either of these is checked based on a CA setting that denotes the permissions to check for SetAuditFilter)
ICertAdminD2::SetOfficerRights	Administrator
ICertAdminD2::SetConfigEntry	Administrator
ICertAdminD2::DeleteRow	Both Administrator and Officer must be present.
ICertAdminD::PublishCRL	Administrator
ICertAdminD::BackupPrepare	Operator
ICertAdminD::BackupEnd	Operator
ICertAdminD::RestoreGetDatabaseLocations	Operator
ICertAdminD::BackupGetAttachedInformation	Operator
ICertAdminD::BackupGetBackupLogs	Operator

Method name	Permissions required
ICertAdminD::BackupGetDynamicFiles	Operator
ICertAdminD::BackupOpenFile	Operator
ICertAdminD::BackupReadFile	Operator
ICertAdminD::BackupCloseFile	Operator
ICertAdminD::BackupTruncateLogs	Operator

The CA SHOULD enforce Officer rights for any of the following methods:

- ICertAdminD2::GetArchivedKey

- ICertAdminD::SetExtension

- ICertAdminD::SetAttributes

- ICertAdminD::DenyRequest

- ICertAdminD::ReSubmitRequest

- ICertAdminD::RevokeCertificate

The CA SHOULD enforce the Enrollment Agent rights for ICertRequestD::Request

3.1.1.8 CRL Publishing Locations

These data elements each contain a list of one or more CRL publishing locations, as defined in [MS-WCCE] section 3.2.1.1.4.

Certificate Services Remote Administration Protocol server implementations that also implement the Windows Client Certificate Enrollment Protocol or the ICertPassage Remote Protocol use the same configuration data elements, defined here, for those implementations.

Each of the following elements is used in the **PublishCRL** and **PublishCRLs** methods:

- **Config_CA_CDP_Publish_To_Base**

  This element is initialized to contain a local registry location:

  HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\SystemCertificates\CA\CRLs\

- **Config_CA_CDP_Publish_To_Delta**

  This element is initialized to contain a local registry location:

  HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\SystemCertificates\CA\CRLs\

- **Config_CA_CDP_Include_In_CRL_Publish_Locations_Extension**

- **Config_CA_CDP_Include_In_CRL_Freshest_CRL_Extension**

- **Config_CA_CDP_Include_In_CRL_IDP_Extension**

CRL publishing locations MUST be in the following format:

Config_CA_CDP_Publish_To_Base and **Config_CA_CDP_Publish_To_Delta**

For a file location, a valid local file path starting with a single-letter drive designator (for example, "D:"), a UNC network file path, or a URI in one of the following formats:

file://{DriveLetter}:\{Path}\{Filename}

file:///{DriveLetter}:\{Path}\{Filename}

file://\\{HostName}\{Path}\{Filename}

file:///\\{HostName}\{Path}\{Filename}

where

- DriveLetter is a single-letter drive designator on the local machine.

- HostName is the **FQDN** or NetBIOS name of a host on the network.

- Path is a file path that is available on the drive DriveLetter on the local machine or on the host HostName over the network.

For an LDAP location, a path that starts with "ldap:"

- The Config_CA_CDP_Include_In_CRL_Publish_Locations_Extension MUST start with "ldap:".

- The Config_CA_CDP_Include_In_CRL_Freshest_CRL_Extension CRL publishing location MUST be a valid UNC network path or start with "http:", "ftp:", "ldap:", or "file:".

- The Config_CA_CDP_Include_In_CRL_IDP_Extension CRL publishing location MUST start with "http:", "ftp:", or "ldap:".

Each CRL publishing location that references a directory path or other hierarchical location MUST specify a file name or attribute name in addition to a directory path or schema location.<11>

- **CRL_Publish_AD_Connection**: An ADConnection handle as defined in [MS-ADSO] section 6.2.3. This element is used each time the CA establishes an Active Directory connection to publish a CRL to an ldap: location.

3.1.1.9   CRL Validity Period

If the CA implements revocation via a CRL, the CA SHOULD maintain a list of the following data elements to describe validity periods for CRLs.

Certificate Services Remote Administration Protocol server implementations that also implement the Windows Client Certificate Enrollment Protocol or the ICertPassage Remote Protocol use the same configuration data elements, defined here, for those implementations.

Config_Base_CRL_Validity_Period: A period of time, shared from the Config_Base_CRL_Validity_Period defined in [MS-WCCE] section 3.2.1.1.4.

Config_Base_CRL_Overlap_Period: A period of time, shared from the Config_Base_CRL_Overlap_Period defined in [MS-WCCE] section 3.2.1.1.4.

Config_Delta_CRL_Validity_Period: A period of time, shared from the Config_Delta_CRL_Validity_Period defined in [MS-WCCE] section 3.2.1.1.4.

Previous_Delta_CRL_Validity_Period: A period of time that expresses the validity period of the previous delta CRL.

Config_Delta_CRL_Overlap_Period: A period of time, shared from the Config_Delta_CRL_Overlap_Period defined in [MS-WCCE] section 3.2.1.1.4.

Config_CA_Clock_Skew_Minutes: A period of time, shared from the Config_CA_Clock_Skew_Minutes defined in [MS-WCCE] section 3.2.1.1.4.

3.1.1.10 Configuration Data

The CA MUST maintain the following ADM elements.<12> Certificate Services Remote Administration Protocol server implementations that also implement the Windows Client Certificate Enrollment Protocol or the ICertPassage Remote Protocol use the same configuration data elements, defined here, for those implementations. If either Windows Client Certificate Enrollment Protocol or ICertPassage Remote Protocol or both are also implemented, access to the configuration data elements from either or both of these protocols SHOULD be serialized.

Config_CA_KRA_Cert_List: An indexed list of KRA certificates shared from the Config_CA_KRA_Cert_List list as defined in [MS-WCCE] section 3.2.1.1.4.

Config_CA_KRA_Cert_Count: A numeral shared from the Config_CA_KRA_Cert_Count element defined in [MS-WCCE] section 3.2.1.1.4.

Config_Configuration_Directory: A UNC path shared from the Config_Configuration_Directory path defined in [MS-WCCE] section 3.2.1.1.4.

Config_CA_Parent_DNS: An FQDN of the parent CA, shared from the Config_CA_Parent_DNS element defined in [MS-WCCE] section 3.2.1.1.4.

Config_CA_Role_Separation: An indicator of role separation state, shared from the Config_CA_Role_Separation element defined in [MS-WCCE] section 3.2.1.1.4.

Config_CA_Exchange_Cert: A list of SHA-1 hash values of all currently valid CA exchange certificates, shared from the Config_CA_Exchange_Cert element defined in [MS-WCCE] section 3.2.1.1.4.

Config_CA_CDP_Publish_To_Base: A list of one or more CRL publishing locations, shared from the list Config_CA_CDP_Publish_To_Base [MS-WCCE] section 3.2.1.1.4. The format requirements for each list value are specified in section 3.1.1.8.

Config_CA_CDP_Publish_To_Delta: A list of one or more delta CRL publishing locations, shared from the list Config_CA_CDP_Publish_To_Delta defined in [MS-WCCE] section 3.2.1.1.4. The format requirements for each list value are specified in section 3.1.1.8.

Config_CA_CDP_Include_In_CRL_Publish_Locations_Extension: A list of one or more CRL publishing locations, shared from the list Config_CA_CDP_Include_In_CRL_Publish_Locations_Extension defined in [MS-WCCE] section 3.2.1.1.4. The format requirements for each list value are specified in section 3.1.1.8.

Config_CA_CDP_Include_In_CRL_Freshest_CRL_Extension: A list of one or more delta CRL publishing locations, shared from the list Config_CA_CDP_Include_In_CRL_Freshest_CRL_Extension defined in [MS-WCCE] section 3.2.1.1.4. The format requirements for each list value are specified in section 3.1.1.8.

Config_CA_CDP_Include_In_CRL_IDP_Extension: A list of one or more CRL publishing locations, shared from the list Config_CA_CDP_Include_In_CRL_IDP_Extension defined in

[MS-WCCE] section 3.2.1.1.4. The format requirements for each list value are specified in section 3.1.1.8.

Config_CA_CDP_Include_In_Cert: A list shared from the Config_CA_CDP_Include_In_Cert list defined in [MS-WCCE] section 3.2.1.1.4.

Config_CA_AIA_Include_In_Cert: A list shared from the Config_CA_CDP_Include_In_Cert list defined in [MS-WCCE] section 3.2.1.1.4.

Config_CA_OCSP_Include_In_Cert: A list shared from the Config_CA_OCSP_Include_In_Cert list defined in [MS-WCCE] section 3.2.1.1.4.

Config_CA_Policy_Algorithm_Implementation: The name of the policy algorithm, shared from the Config_CA_Policy_Algorithm_Implementation element defined in [MS-WCCE] section 3.2.1.1.4.

Config_CA_Exit_Algorithm_Implementation_List: A list of names of the exit algorithms, shared from the list Config_CA_Exit_Algorithm_Implementation_List defined in [MS-WCCE] section 3.2.1.1.4.

Config_CA_Exit_Count: A numeral shared from the Config_CA_Exit_Count element defined in [MS-WCCE] section 3.2.1.1.4.

Config_CA_Accept_Request_Attributes_ValidityTime: A Boolean value shared from the Config_CA_Accept_Request_Attributes_ValidityTime element defined in [MS-WCCE] section 3.2.1.1.4.

Config_CA_Accept_Request_Attributes_Extensions: A Boolean value shared from the Config_CA_Accept_Request_Attributes_Extensions element defined in [MS-WCCE] section 3.2.1.1.4.

Config_CA_Accept_Request_Attributes_SAN: A Boolean value shared from the Config_CA_Accept_Request_Attributes_SAN element defined in [MS-WCCE] section 3.2.1.1.4.

Config_CA_Accept_Request_Attributes_Other: A Boolean value shared from the Config_CA_Accept_Request_Attributes_Other element defined in [MS-WCCE] section 3.2.1.1.4.

Config_CA_Accept_Request_Attributes_CertPath: A Boolean value shared from the Config_CA_Accept_Request_Attributes_CertPath element defined in [MS-WCCE] section 3.2.1.1.4.

Config_CA_Allow_RenewOnBehalfOf_Requests: A Boolean value shared from the Config_CA_Allow_RenewOnBehalfOf_Requests element defined in [MS-WCCE] section 3.2.1.1.4,<13>

Config_CA_Requests_Disposition: A 4-byte integer value shared from the Config_CA_Requests_Disposition element defined in [MS-WCCE] section 3.2.1.1.4.

Config_CA_Audit_Filter: The list of, at most, 32 events for which the CA server will create local security audit log entries. Filter values are defined in 3.1.4.2.10.

Config_CA_CACert_Publish_To: The list of locations to which the CA will publish its own certificate.

Config_CA_Common_Name: A null-terminated **UNICODE** string that contains the name of the CA.

Config_CA_Interface_Flags: A set of flags, shared from the Config_CA_Interface_Flags element defined in [MS-WCCE] section 3.2.1.1.4.

Config_CA_KRA_Flags: A set of flags that implementers can use to affect server behavior.

Config_Product_Version: A numeral or series of numerals, shared with the Config_Product_Version element defined in [MS-WCCE] section 3.2.1.1.4.

Config_CA_Type: A CA type as defined in section 2.2.2.4 of [MS-WCCE] as CAType.

Config_CA_Use_DS: A numeral that indicates whether the CA uses Active Directory for CRL publishing or not. A value of 1 or greater means Active Directory is used. A value of 0 means it is not used.

Config_CSP_CNG_Hash_Algorithm: A string that contains the name of the Hash algorithm used by the CA.

Config_CSP_Hash_Algorithm: A numeral that identifies the Hash algorithm used by the CA.

Config_CSP_Provider: The name of the **cryptographic service provider (CSP)** used by the CA.

Config_CSP_ProviderType: A numeral that indicates the type of CSP used by the CA.

Config_Setup_Status: A numeral that indicates the current status of the CA installation, for example whether it is complete.

Config_CA_CRL_Next_Publish: The CERTTIME value of the timer specified in section 3.1.2.1.1 that indicates the next time the CA will publish base CRLs. This element is updated by the **PublishCRL** and **PublishCRLs** methods.

Config_CA_CRL_Delta_Next_Publish: The CERTTIME value of the timer specified in section 3.1.2.1.2 that indicates the next time the CA will publish delta CRLs. This element is updated by the **PublishCRL** and **PublishCRLs** methods.

Config_CA_CRL_Attempt_Republish: This data element is a non-negative integer that, by its value, indicates:

- Whether or not the server will attempt to republish CRLs.

    - If the value is 0 or is greater than or equal to 10, no retries will happen. If the value is greater than 0 and less than 10, then retries will happen.

- How many previous unsuccessful attempts to publish CRLs have occurred since the last regularly scheduled CRL publishing attempt (including the last regularly scheduled publishing attempt, if that attempt was unsuccessful).

Config_CA_LDAP_Flags: This data element has two possible values:

- 0 means that port 389 will be used when opening an Active Directory connection to publish CRLs to ldap:/// locations.

- 1 means that port 636 will be used when opening an Active Directory connection to publish CRLs to ldap:/// locations.

Config_High_Serial_Number: A 4-byte integer, shared with the Config_High_Serial_Number element defined in [MS-WCCE] section 3.2.1.1.4, that is used in generating certificate serial numbers.

Config_High_Serial_String: A string value, shared with the Config_High_Serial_String element defined in [MS-WCCE] section 3.2.1.1.4 that is used in generating certificate serial numbers.

Config_Max_Number_Of_AD_Connections: A 4-byte integer that indicates the maximum number of cached **ADConnection** handles.

Config_AD_Connection_Referral: A flag that indicates whether referral option for **ADConnection** is set to TRUE.

Config_Endorsement_Key_List_Directories: A list of strings, each one a UNC or local file path to a folder that contains empty files. Each file name is the SHA2 hash of an **EK public key** trusted by the CA for key attestation by public key. The CA has read access to this location.

Each element defined below as "OnNextRestart_Config_Element_Name" will lend its value to the corresponding data element "Config_Element_Name" upon next CA restart.

The "OnNextRestart_..." element's value is available to the method GetConfigEntry at any time, regardless of CA restart.

Any configuration elements defined above that do not have corresponding values defined below always have equivalent on-disk values (available to GetConfigEntry) and in-memory values (used by CA).

OnNextRestart_Config_Max_Number_Of_AD_Connections: The value the **Config_Max_Number_Of_AD_Connections** data element will attain on next CA restart.

OnNextRestart_Config_AD_Connection_Referral: The value the **Config_AD_Connection_Referral** data element will attain on next CA restart.

OnNextRestart_Config_Permissions_CA_Security: The value the **Config_Permissions_CA_Security** data element will attain on next CA restart.

OnNextRestart_Config_Setup_Status: The value the **Config_Setup_Status** data element will attain on next CA restart.

OnNextRestart_Config_CA_Use_DS: The value the **Config_CA_Use_DS** data element will attain on next CA restart.

OnNextRestart_Config_CA_Type: The value the **Config_CA_Type** data element will attain on next CA restart.

OnNextRestart_Config_CA_KRA_Flags: The value the **Config_CA_KRA_Flags** data element will attain on next CA restart.

OnNextRestart_Config_Product_Version: The value the **Config_Product_Version** data element will attain on next CA restart.

OnNextRestart_Config_CA_Common_Name: The value the **Config_CA_Common_Name** data element will attain on next CA restart.

OnNextRestart_Config_CA_Interface_Flags: The value the **Config_CA_Interface_Flags** data element will attain on next CA restart.

OnNextRestart_Config_CSP_Provider: The value the **Config_CSP_Provider** data element will attain on next CA restart.

OnNextRestart_Config_CSP_ProviderType: The value the **Config_CSP_ProviderType** data element will attain on next CA restart.

OnNextRestart_Config_CSP_Hash_Algorithm: The value the **Config_CSP_Hash_Algorithm** data element will attain on next CA restart.

OnNextRestart_Config_CSP_CNG_Hash_Algorithm: The value the **Config_CSP_CNG_Hash_Algorithm** data element will attain on next CA restart.

OnNextRestart_Config_CA_CRL_Next_Publish: The value the **Config_CA_CRL_Next_Publish** data element will attain upon next CA restart. This element is updated by, but not read by, the CA's PublishCRL and PublishCRLs methods.

OnNextRestart_Config_CA_CRL_Delta_Next_Publish: The value the **Config_CA_CRL_Delta_Next_Publish** data element will attain on next CA restart. This element is updated by, but not read by, the CA's PublishCRL and PublishCRLs methods.

OnNextRestart_Config_CA_Audit_Filter: The value the **Config_CA_Audit_Filter** data element will attain on next CA restart.

OnNextRestart_Config_CA_Policy_Algorithm_Implementation: The value the **Config_CA_Policy_Algorithm_Implementation** data element will attain on next CA restart.

OnNextRestart_Config_CA_Exit_Algorithm_Implementation_List: The value the **Config_CA_Exit_Algorithm_Implementation_List** data element will attain on next CA restart.

OnNextRestart_Config_CA_CDP_Publish_To_Base: The value the **Config_CA_CDP_Publish_To_Base** data element will attain on next CA restart.

OnNextRestart_Config_CA_CDP_Publish_To_Delta: The value the **Config_CA_CDP_Publish_To_Delta** data element will attain on next CA restart.

OnNextRestart_Config_CA_CDP_Include_In_Cert: The value the **Config_CA_CDP_Include_In_Cert** data element will attain on next CA restart.

OnNextRestart_Config_CA_CDP_Include_In_CRL_Publish_Locations_Extension: The value the **Config_CA_CDP_Include_In_CRL_Publish_Locations_Extension** data element will attain on next CA restart.

OnNextRestart_Config_CA_CDP_Include_In_CRL_Freshest_CRL_Extension: The value the **Config_CA_CDP_Include_In_CRL_Freshest_CRL_Extension** data element will attain on next CA restart.

OnNextRestart_Config_CA_CDP_Include_In_CRL_IDP_Extension: The value the **Config_CA_CDP_Include_In_CRL_IDP_Extension** data element will attain on next CA restart.

OnNextRestart_Config_CA_AIA_Include_In_Cert: The value the **Config_CA_AIA_Include_In_Cert** data element will attain on next CA restart.

OnNextRestart_Config_CA_CACert_Publish_To: The value the **Config_CA_CACert_Publish_To** data element will attain on next CA restart.

OnNextRestart_Config_CA_OCSP_Include_In_Cert: The value the **Config_CA_OCSP_Include_In_Cert** data element will attain on next CA restart.

OnNextRestart_Config_CA_LDAP_Flags: The value that the **Config_CA_LDAP_Flags** data element will attain on next CA restart.

OnNextRestart_Config_High_Serial_Number: The value the **Config_High_Serial_Number** data element will attain on next CA restart.

OnNextRestart_Config_High_Serial_String: The value the **Config_High_Serial_String** data element will attain on next CA restart.

OnNextRestart_Config_CA_CRL_Attempt_Republish: The value the **Config_CA_CRL_Attempt_Republish** data element will attain on next CA restart.

OnNextRestart_Config_CA_Requests_Disposition: The value of the **Config_CA_Requests_Disposition** data element will attain on next CA restart.

3.1.1.11   Signing_Cert Table

Signing_Cert Table: An ordered list of CA private key and associated other data, including **public key** certificates, shared from the Signing_Cert Table defined in [MS-WCCE] section 3.2.1.1.2.

The **Signing_Cert_Certificate** in the last row of the Signing_Cert table is the current CA signing certificate.

3.1.1.12   CA Exchange Certificates

Current_CA_Exchange_Cert: An X.509 certificate shared from the Current_CA_Exchange_Cert element defined in [MS-WCCE] section 3.2.1.1.4.

Store_CA_Exchange_Cert: A list of X.509 certificates shared from the Store_CA_Exchange_Cert element defined in [MS-WCCE] section 3.2.1.1.4.

3.1.1.13   Client User Identity Token

uToken: An identity token of the type "**Token/AuthorizationContext**" as specified by [MS-DTYP] section 2.5.2

The initial value is NULL.

3.1.2   Timers

3.1.2.1   CRL Next Publish Timers

Each time a base or delta CRL is generated and published to all CRL publishing locations, either by an invocation of ICertAdminD::PublishCRL, an invocation of ICertAdminD2::PublishCRLs, or as a result of automatic invocation via another timer, a timer's value is set to time out at the next CRL publishing time. There is one timer for base CRLs and one timer for delta CRLs.

CRL creation and/or publishing that results in a nonzero CRL_Publish_Status_Code results in an additional timer setting. In this case, the timers for base or delta CRLs are set as described in section 3.1.2.2.

3.1.2.1.1 Base CRL Next Publish Timer

This timer represents the next time the CA will wake up and publish a base CRL again. The next timeout value for the Base CRL Next Publish Timer is contained in the configuration data element Config_CA_CRL_Next_Publish, which is updated at base CRL creation time.

For details about how the CA computes the Config_CA_CRL_Next_Publish value upon base CRL creation, see the server processing rules for the PublishCRL method (section 3.1.4.1.6) or the PublishCRLs method (section 3.1.4.2.1).

3.1.2.1.2 Delta CRL Next Publish Timer

This timer represents the next time the CA will wake up and publish a delta again. The next timeout value for the Delta CRL Next Publish Timer is contained in the configuration data element Config_CA_CRL_Delta_Next_Publish, which is updated at delta CRL creation time.

For details about how the CA computes the Config_CA_CRL_Delta_Next_Publish value upon delta CRL creation, see the server processing rules for the PublishCRL method (section 3.1.4.1.6) or the PublishCRLs method (section 3.1.4.2.1).

3.1.2.2 CRL Publication Retry Timer

If a CRL publishing attempt, either by invocation of ICertAdminD::PublishCRL or ICertAdminD2::PublishCRLs, or as a result of automatic invocation via another timer, does not return "0", then the CA MUST set this retry timer to an interval of 10 minutes.

For each subsequent invocation of ICertAdminD::PublishCRL or ICertAdminD2::PublishCRLs that does not return "0", and for each subsequent CRL publishing retry attempt that does not result in a CRL_Publish_Status_Code of "0" for all CRLs, the CA MUST update the CRL Publication Retry Timer's next timeout value.

The logic that is executed when the timer reaches its next timeout value is specified in section 3.1.5.2.

3.1.3 Initialization

Interface Initialization: On startup, the CA MUST ensure that remote clients have permissions to activate and call DCOM objects on the CA. Subsequently, DCOM object and interface initialization is performed by the DCOM object exporter in response to an activation request from the DCOM client. The Certificate Services Remote Administration Protocol client calls the DCOM client to initiate the activation request to the server. As a result, the DCOM server returns an object reference to the DCOM client, and the Certificate Services Remote Administration Protocol client can use this client object reference to make calls to the Certificate Services Remote Administration Protocol server methods specified in this document. The details of DCOM object initialization on the server, in response to client activation requests and **ORPC** calls, are specified in [MS-DCOM] sections 3.1.1.3, 3.1.1.5.1, and 3.1.1.5.4.<14>

Cryptographic Initialization: The CA MUST validate each of the CA signing certificate(s) represented by the **Signing_Cert_Certificate** column in the Signing_Cert Table (section 3.1.1.11). The certificates are validated in the order in which they occur in the Signing_Cert Table from first to last, with the latest Signing_Cert entry evaluated last.

The certificate path validation MUST be performed in accordance with the algorithm specified in [RFC3280] section 6.1. To invoke this algorithm, the CA MUST use the following inputs as defined in [RFC3280] section 6.1.1.

- Prospective certification path: This is the path from the CA signing certificate (**Signing_Cert_Certificate** column from the Signing_Cert Table) to a root certificate. The path is passed in to the algorithm. Certificates needed to build this path SHOULD be obtained either from the location specified in the Authority Information Access extension (as specified in [RFC3280] section 4.2.2.1) in the signing certificate, or from local storage, if certificates have previously been cached as described in [RFC3280] section 3.1.

  - For every certificate in the path starting with the signing certificate, its issuer certificate is determined either by matching its Authority Key Identifier (as specified in [RFC3280] section 4.2.1.1) with the Subject Key Identifier (as specified in [RFC3280] section 4.2.1.2) of the issuer certificate, or by matching its issuer field (as specified in [RFC3280] section 4.1.2.4) with the subject field (as specified in [RFC3280] section 4.1.2.6) of the issuer certificate.

- Current date and time: This is the current system date and time.

- User-initial-policy set: The user-initial-policy-set contains the value any-policy.

- Trust Anchors: The set of root certificates that are passed in to the algorithm from implementation-specific local storage.

- Initial-policy-mapping-inhibit: This input parameter is not set.

- Initial-explicit-policy: This input parameter is not set.

- Initial-any-policy-inhibit: This input parameter is not set.

Based on the above inputs, the algorithm is followed as specified in the other subsections of section 6.1 of [RFC3280]. The Microsoft CA does not check for optional certificate extensions such as the policy mapping extension, which is consistent with the standard specified in section 6.1 of [RFC3280].

If none of the CA signing certificates in the Signing_Cert table passes Certificate Path Validation (see [RFC3280] section 6) based on any of the trust anchors, the CA MUST NOT start processing messages. For example:

- If there are six certificates and only one passes, the CA starts.

- If there are six certificates and five pass, the CA starts.

- If there are six certificates and none of them passes, the CA MUST NOT start processing messages.

Database Initialization: The CA MUST ensure that the database is available for use and that the tables and fields that are defined in section 3.1.1 exist. If the database is not available, the CA MUST NOT start processing messages.

Configuration Initialization: The CA MUST initialize the configuration data elements defined in section 3.1.1.10. Each element defined in section 3.1.1.10 as "OnNextRestart_{Config_Element_Name}" will be used to initialize the corresponding data element "{Config_Element_Name}" upon CA startup. <15>

If the CA fails to complete any of the initialization steps, the CA MUST NOT start.

3.1.4   Message Processing Events and Sequencing Rules

The Certificate Services Remote Administration Protocol defines the following interfaces.

Interface	Description
ICertAdminD	Defines methods that enable a client to manage a CA.
ICertAdminD2	Extends the **ICertAdminD** interface.

3.1.4.1 Processing Rules for ICertAdminD

The **ICertAdminD** interface provides an application programming interface for a client<16> to manage a certificate authority.

The **ICertAdminD** interface inherits the **IUnknown** interface.

The version number for **IUnknown** is 1.0. The universally unique identifier (UUID) for the ICertAdminD interface is "d99e6e71-fc88-11d0-b498-00a0c90312f3". Method opnum field values start with 3; opnum values 0 through 2 represent the IUnknown methods: QueryInterface, AddRef, and Release methods, respectively, as specified in [MS-DCOM].

Methods in RPC Opnum Order

Method	Description
SetExtension	The **SetExtension** method allows adding, modifying, or disabling extensions, as specified in [RFC3280]; a CA can include an extension in an issued certificate for a particular pending request. Opnum: 3
SetAttributes	The **SetAttributes** method sets attributes in the specified pending certificate request. Opnum: 4
ResubmitRequest	The **ResubmitRequest** method resubmits a specific pending or denied certificate request to the CA. Opnum: 5
DenyRequest	The **DenyRequest** method denies a specific certificate request that is pending. Opnum: 6
IsValidCertificate	The **IsValidCertificate** method verifies the certificate against the CA key and checks that the certificate has not been revoked. Opnum: 7
PublishCRL	The **PublishCRL** method sends a request to the CA server to publish a new CRL, as specified in [RFC3280] section 5. Opnum: 8
GetCRL	The **GetCRL** method retrieves the current CRL for the CA server. Opnum: 9
RevokeCertificate	The **RevokeCertificate** method revokes a certificate either immediately or on a specified date. Opnum: 10
EnumViewColumn	The **EnumViewColumn** method returns an array of column

Release: Friday, October 25, 2013

Method	Description
	information. Opnum: 11
GetViewDefaultColumnSet	The **GetViewDefaultColumnSet** method returns an array of the column set identifiers associated with a specific view. Opnum: 12
EnumAttributesOrExtensions	The **EnumAttributesOrExtensions** method is used to access sets of attributes or extensions for a particular row ID. Opnum: 13
OpenView	The **OpenView** method opens a view into the database and returns a set of resultant rows data. Opnum: 14
EnumView	The **EnumView** method returns a set of resultant rows data for the opened view. Opnum: 15
CloseView	The **CloseView** method closes a view previously opened with the **OpenView** method call. Opnum: 16
ServerControl	The **ServerControl** method is used to force the CA server to unregister the ICertAdminD and ICertAdminD2 interfaces. Opnum: 17
Ping	The **Ping** method is used to test whether the Certificate server is alive. Opnum: 18
GetServerState	The **GetServerState** method is used to validate "Read". Opnum: 19
BackupPrepare	The **BackupPrepare** method is used to prepare the database for performing further backup operations. Opnum: 20
BackupEnd	The **BackupEnd** method completes the backup process started via a call to **ICertAdminD::BackupPrepare**. Opnum: 21
BackupGetAttachmentInformation	The **BackupGetAttachmentInformation** method is used to query the CA for the names of database files that should become part of the backup file set. Opnum: 22
BackupGetBackupLogs	The **BackupGetBackupLogs** method queries the CA for the names of database transaction log files that should become part of the backup file set. Opnum: 23
BackupOpenFile	The **BackupOpenFile** method opens a file for backup.

Method	Description
	Opnum: 24
BackupReadFile	The **BackupReadFile** method reads the database file and loads the content into the buffer provided. Opnum: 25
BackupCloseFile	The **BackupCloseFile** method closes the database file that was initialized by a prior call to **BackupOpenFile**. Opnum: 26
BackupTruncateLogs	The **BackupTruncateLogs** method function eliminates redundant records from the log files and reduces the disk storage space used by log files. Opnum: 27
ImportCertificate	The **ImportCertificate** method imports a certificate into the CA database. Opnum: 28
BackupGetDynamicFiles	The **BackupGetDynamicFiles** method retrieves the list of CA dynamic file names that need to be backed up. Opnum: 29
RestoreGetDatabaseLocations	The **RestoreGetDatabaseLocation** method retrieves the list of CA server database location names for all the database files being restored. Opnum: 30

All methods MUST NOT throw exceptions.

The CA MUST execute the following processing rules for each invocation of each of the methods listed below in this section. Then the CA MUST proceed to execute the processing rules listed for each individual method below.

The CA MUST obtain the SID in the RPC_SID form of the caller from the value of the element **uToken.Sids[uToken.UserIndex]**. The ADM element **uToken** is initialized by retrieving the identity token for the current **execution context** by calling the abstract interface **GetRpcImpersonationAccessToken()** as specified in [MS-RPCE] section 3.3.3.4.3.1. The value of the **uToken.Sids** array element indexed at **uToken.UserIndex** is the SID of the caller. If the caller cannot be identified (**uToken.Sids[uToken.UserIndex]** is NULL), the CA MUST refuse to establish a connection, returning an error.<17>

If Config_CA_Interface_Flags contains the value IF_ENFORCEENCRYPTICERTADMIN and the RPC_C_AUTHN_LEVEL_PKT_PRIVACY authentication level, as defined in [MS-RPCE] section 2.2.1.1.8, is not specified on the RPC connection from the client, the CA MUST refuse to establish a connection with the client by returning an error.<18>

If Config_CA_Interface_Flags contains the value IF_NOREMOTEICERTADMIN, the CA SHOULD return an error for any of the methods listed in this section.<19>

3.1.4.1.1 ICertAdminD::SetExtension (Opnum 3)

The **SetExtension** method allows adding, modifying, or disabling of extensions, as specified in [RFC3280]. A CA can include an extension in an issued certificate for a particular pending request.

Release: Friday, October 25, 2013

```
HRESULT SetExtension(
  [in, string, unique] wchar_t const* pwszAuthority,
  [in] DWORD dwRequestId,
  [in, string, unique] wchar_t const* pwszExtensionName,
  [in] DWORD dwType,
  [in] DWORD dwFlags,
  [in, ref] CERTTRANSBLOB* pctbValue
);
```

pwszAuthority: A null-terminated Unicode string that contains the name of the certificate server. The pwszAuthority is a Unicode string in the form of a distinguished name (DN) value, such as "CAName", where CAName MUST be the full common name (CN) or sanitized name of the CA, as specified in [MS-WCCE] section 3.1.1.4.1.1.

dwRequestId: A 32-bit nonzero unsigned integer value that specifies the ID of the certificate request.

pwszExtensionName: A null-terminated Unicode string that specifies the OID for the extension to set, as specified in [X680]. The string MUST be 31 or fewer characters in length and the characters MUST NOT be NULL.

dwType: An unsigned integer value that specifies the type of extension being set. The *dwType* parameter MUST agree with the data type of the pb member of the *pctbValue* parameter. This parameter can be one of the following values.

Value	Meaning
0x00000001	Unsigned long data
0x00000002	Date/time
0x00000003	Binary data
0x00000004	Unicode

dwFlags: An unsigned integer value that specifies the flags for the extension being set. This parameter can be one of the following values.

Value	Meaning
1	This is a critical extension.
2	The extension MUST NOT be used on issued certificates.

pctbValue: A pointer to a **CERTTRANSBLOB** structure. The **pb** member MUST point to the binary data for the extension and the cb field MUST contain the length, in bytes, of the value. Depending on the value of the *dwType* parameter, the format of the binary data that is pointed to by the **pb** member is shown in the following table.

Value of dwType	Meaning
0x00000001	The **CERTTRANSBLOB** structure **pb** member MUST point to an unsigned long data value in little-endian format.

Value of dwType	Meaning
0x00000002	The **CERTTRANSBLOB** structure **pb** member MUST point to data using little-endian encoding format.
0x00000003	The **CERTTRANSBLOB** structure **pb** member MUST point to an array of bytes that are not in need of endian forcing.
0x00000004	The **CERTTRANSBLOB** structure **pb** member MUST point to a null-terminated Unicode string in little-endian format.

This method instructs the CA to add, modify, or disable an extension that is associated with a previously submitted certificate request that is in a pending state, as specified in [MS-WCCE] section 3.2.1.4.2.1.3. If the certificate request does not contain an extension with the name specified in *pwszExtensionName*, then the extension is added to the certificate request. If the request already contains an extension of that name, then the extension specified in the **SetExtension** call will replace the old one, hence modifying the contents. To disable an extension, a value of 2 can be specified in *dwFlags* parameter, described above, when calling **SetExtension**.

The following processing rules apply:

1. The CA MUST look up the request based on the provided *dwRequestId* parameter in the request table. If the request is not found, the CA MUST fail the request.

2. If the request is found in the CA database, the CA MUST verify that the value of the Request_Disposition column is "request pending". If the value of the Request_Disposition column is not "request pending", the CA MUST fail the request.

3. The CA MUST verify if the *pwszExtensionName* parameter contains a valid OID, as specified in [X680]. If the OID is not valid, the CA MUST fail the request.

4. The CA MUST associate the specified extension and flags with the pending request, for possible inclusion later in the issued certificate, by adding the entry to the Extension table:

   ▪ If dwType is 1 (PROPTYPE_LONG), the CA MUST encode the LONG value into an ASN.1 integer, as specified in [X660], and save the encoded value as the extension value.

   ▪ If dwType is 2 (PROPTYPE_DATE), the CA MUST encode the FILETIME value into an ASN.1 Choice-of-Time, as specified in [X660], and save the encoded value as the extension value.

   ▪ If dwType is 3 (PROPTYPE_BINARY), the CA MUST save the specified value as the extension value.

   ▪ If dwType is 4 (PROPTYPE_STRING), the CA MUST encode the Unicode string value into an IA5String, as specified in [X660], and save the encoded value as the extension value.

   ▪ If dwType is any other value, the CA MUST fail the request. The error code SHOULD be ERROR_INVALID_PARAMETER (0x80070057).

3.1.4.1.2 ICertAdminD::SetAttributes (Opnum 4)

The **SetAttributes** method sets attributes in the specified pending certificate request.

```
HRESULT SetAttributes(
  [in, string, unique] wchar_t const* pwszAuthority,
  [in] DWORD dwRequestId,
```

```
    [in, string, unique] wchar_t const* pwszAttributes
);
```

pwszAuthority: See the *pwszAuthority* definition in **ICertAdminD::SetExtension (section 3.1.4.1.1)**.

dwRequestId: A 32-bit nonzero unsigned integer value that specifies the ID of the certificate request.

pwszAttributes: A null-terminated Unicode string. The value of the string MUST have the same format as specified in [MS-WCCE] section 3.2.1.4.2.1.1.

This method instructs the CA to add or modify a name-value pair that is associated with a previously submitted certificate request that is in a pending state. Information about a pending certificate request is specified in [MS-WCCE] section 3.2.1.4.2.1.4.2.

The following processing rules apply:

1. The CA MUST look up the request based on the provided *dwRequestId* parameter in the Request table. If the request is not found, the CA MUST fail the request.

2. If the request is found in the CA database, the CA MUST verify that the value of the Request_Disposition column is "request pending". If the value of the Request_Disposition column is not "request pending", the CA MUST fail the request.

3. The CA MUST parse the **pwszAttributes** string, as is done for the **ICertRequestD::Request** and **ICertRequestD2::Request2** methods as specified in [MS-WCCE] section 3.2.1.4.3.

4. The CA MUST ignore invalid name-value pair entries.

5. The CA MUST associate the valid name-value pair entries with the pending requests, for possible later impact on the issued certificate, by adding the entries in the Attribute table.

3.1.4.1.3   ICertAdminD::ResubmitRequest (Opnum 5)

The **ResubmitRequest** method resubmits a specific pending or denied certificate request to the CA.

```
HRESULT ResubmitRequest(
    [in, string, unique] wchar_t const* pwszAuthority,
    [in] DWORD dwRequestId,
    [out] DWORD* pdwDisposition
);
```

pwszAuthority: See the *pwszAuthority* definition in **ICertAdminD::SetExtension (section 3.1.4.1.1)**.

dwRequestId: A 32-bit nonzero unsigned integer value that specifies the ID of the certificate request.

pdwDisposition: A pointer to an unsigned integer value that receives the disposition status of the certificate (upon resubmission).

This method instructs the CA to try again to process a previously submitted certificate request, which is in a pending or denied state.

The following processing rules apply:

1. The CA MUST validate that the Unicode string referenced by *pwszAuthority* matches (case-insensitive) the full CN (as defined in [MS-GLOS]) or the sanitized name of the CA. Sanitized name is defined in [MS-WCCE] sections 1.3.2.4 and 3.1.1.4.1.1. If the value does not match, the server MUST fail the request. The error code SHOULD be 0x80070057.

2. The CA MUST look up the request based on the provided *dwRequestId* parameter in the request table:

   - If the request is not found, the CA MUST place 0x80094004 in the *pdwDisposition* parameter and return successfully.<20>

   - If the request is found in the CA database, the row is referred to as the identified row in the following processing rules.

   - The CA MUST verify that the value of the Request_Disposition column in the identified row is "request pending" or "request denied".

   - If the value of the Request_Disposition column in the identified row is not "request pending" or "request denied", the CA MUST place 0x80094003 in the *pdwDisposition* parameter and return successfully.

   - If the value of the Request_Disposition column in the identified row is "request denied" and the invoker of the method is not the CA administrator, the CA MUST place 0x80094003 in the *pdwDisposition* parameter and return successfully.

3. The CA MUST try to process the request as if it is a new request, as specified in [MS-WCCE] section 3.2.1.4.2.1.4, ignoring step one in [MS-WCCE] section 3.2.1.4.2.1.4.4.

4. If the request processing results in the CA issuing the certificate, the CA MUST place a 3 in the *pdwDisposition* parameter and return successfully.

5. If the request processing results in the denial of the certificate by the **CA policy algorithm**, the CA MUST set the Request_Disposition column of the identified row to "request denied", place a nonzero-zero error code in the *pdwDisposition* parameter, and MUST return successfully. Error codes are specified in [MS-ERREF]. All nonzero values of *pdwDisposition* SHOULD be treated equivalently by the client.

6. If the request processing results in the CA pending the certificate, the CA MUST place a 5 in the *pdwDisposition* parameter and return successfully.

7. If the request processing results in an error on the CA or in the policy algorithm, the CA MUST set the Request_Disposition column of the identified row to "request failed", MUST place an error code that is not equal to 2, 3, or 5 in the *pdwDisposition* parameter, and MUST return successfully. Error codes are specified in [MS-ERREF]. All nonzero values of *pdwDisposition* other than 2, 3, or 5 SHOULD be treated equivalently by the client.

8. The CA SHOULD set the Request_Disposition_Message column of the identified row to any value the implementer considers informative for presentation to a human reader.

   All disposition messages contain text in the system language of the server.

   The disposition message provides additional information, if available, about the reason for the assignment of a particular disposition value or the details of the certificate disposition:

If the return value is Error (30), the disposition messages will include one or more of the following:

- Error archiving private key. - This occurs if the CA encountered an error or was otherwise unable to archive a private key sent with the request.

- Error parsing request. - This occurs if the request is malformed.

- Error verifying request signature or signing certificate. - This occurs if the signature signing the request or the signature on a request signing certificate could not be verified.

- Resubmitted by {domain\name}, where {domain\name} is replaced with the user name of the caller if the request was submitted by using the **ResubmitRequest** method of this protocol.

If the return value is Denied (31), the disposition messages will include one or more of the following:

- Denied by {domain\name}, where {domain\name} is replaced with the user name of the caller if the request was submitted by using the **[DenyRequest](#)** method of this protocol.

- Denied by policy module. - This occurs if the policy module processing failed one or more of the checks required to issue a certificate.

- Denied by policy module, combined with a descriptive error message. - This occurs when the policy module processing failed one or more of the checks required to issue a certificate and an additional error code was generated. Other Windows error messages are as documented in [MS-ERREF].

- Requested by {domain\name}, where {domain\name} is replaced with the user name of the caller if the request was formerly in a pending state and was issued by using the **ResubmitRequest** method of this protocol.

If the return value is Issued (20), the disposition messages include the following:

- Requested by {domain\name}, where {domain\name} is replaced with the user name of the caller.

- Issued. - This occurs if the certificate was issued and no additional information is required.

- Issued, combined with a descriptive informational message from the policy algorithm. - This occurs if additional information relevant to the certificate issuance is available, for example, if a certificate validity period was truncated from the requested length.

- Resubmitted by {domain\name}, where {domain\name} is replaced with the user name of the caller if the request was formerly in a pending state and was issued by using the **ResubmitRequest** method of this protocol.

If the return value is Pending (9), the disposition messages include the following:

- Taken under submission. - This occurs if the CA or the certificate template is configured to require manager approval for certificate issuance. Prior to manager approval, the certificate will be in a pending or "taken under submission" disposition.

3.1.4.1.4   ICertAdminD::DenyRequest (Opnum 6)

The **DenyRequest** method denies a specific certificate request that is pending.

```
HRESULT DenyRequest(
  [in, string, unique] wchar_t const* pwszAuthority,
  [in] DWORD dwRequestId
);
```

pwszAuthority: See the *pwszAuthority* definition in **ICertAdminD::SetExtension (section 3.1.4.1.1)**.

dwRequestId: A 32-bit nonzero unsigned integer value that specifies the ID of the certificate request.

The following processing rules apply:

1. The CA MUST look up the request based on the provided *dwRequestId* parameter in the Request table. If the request is not found, the CA MUST fail the request. If the request is found, the selected row is referred to as the identified row in the following processing rules.

2. If the value of the Request_Disposition column in the identified row is not "request pending", the CA MUST fail the request.

3. If the value of the Request_Disposition column in the identified row is "request pending":

   1. The CA MUST set the value of the Request_Disposition column in the identified row to "request denied", and set the Request_Status_Code to 0x80094014 (CERTSRV_E_ADMIN_DENIED_REQUEST).<21>

   2. The CA SHOULD set the value of the Request_Disposition_Message column in the identified row to any value that the implementer considers human-readable. The Microsoft CA sets Request_Disposition_Message in this case to "Denied by {username}" where "{username}" is replaced with the user name of the caller.

3.1.4.1.5   ICertAdminD::IsValidCertificate (Opnum 7)

The **IsValidCertificate** method verifies the certificate against the CA key and verifies that the certificate has not been revoked.

```
HRESULT IsValidCertificate(
  [in, string, unique] wchar_t const* pwszAuthority,
  [in, string, unique] wchar_t const* pSerialNumber,
  [out] LONG* pRevocationReason,
  [out] LONG* pDisposition
);
```

pwszAuthority: See the *pwszAuthority* definition in **ICertAdminD::SetExtension (section 3.1.4.1.1)**.

pSerialNumber: A null-terminated Unicode string specifying a serial number that identifies the certificate to be reviewed. The string MUST specify the serial number as an even number of hexadecimal digits. If necessary, a zero can be prefixed to the number to produce an even number of digits. The string MUST NOT contain more than one leading zero. Information about the serial number is as specified in [RFC3280] section 4.1.2.2.

pRevocationReason: A pointer to a **LONG** value that receives the revocation reason code. The revocation reason code MUST be one of the following values that are defined for CRLReason, as specified in [RFC3280] section 5.3.1.

Value	Meaning
0	unspecified
1	keyCompromise
2	cACompromise
3	affiliationChanged
4	superseded
5	cessationOfOperation
6	certificateHold

pDisposition: A pointer to a **LONG** that receives the disposition status of the request. This parameter MUST be one of the following values.

Value	Meaning
0x00000002	The certificate has been revoked.
0x00000003	The certificate is still valid.
0x00000004	The certificate was never issued.

The following processing rules apply:

1. Unless otherwise specified in the processing rules that follow, the value that is returned as pRevocationReason SHOULD be 0.

2. The CA MUST look up a row in the Request table where the value of the Serial_Number column is identical to the value that is provided in the *pSerialNumber* parameter:

   ▪ If a row is not found, the CA MUST return 4 in the *pDisposition* parameter.

   ▪ If a row is found, this row is referred to as the identified row in the following processing rules.

3. The CA MUST inspect the value of the Request_Disposition column in the identified row and apply the following rules:

   ▪ If the value is "certificate issued", the CA MUST return 3 in the *pDisposition* parameter.

   ▪ If the value is "certificate revoked" and the value in Request_Revocation_Date is greater than the current time, the CA MUST return 3 in the *pDisposition* parameter.

   ▪ If the value is "certificate revoked" and the value in Request_Revocation_Date contains a value that is less than or equal to the current time, the CA MUST return 2 in the pDisposition parameter, read the Request_Revoked_Reason property from the Request table, and return the value as the pRevocationReason argument.

3.1.4.1.6 ICertAdminD::PublishCRL (Opnum 8)

The **PublishCRL** method sends a request to the CA server to publish a new CRL.

```
HRESULT PublishCRL(
  [in, string, unique] wchar_t const* pwszAuthority,
  [in] FILETIME FileTime
);
```

pwszAuthority: See the *pwszAuthority* definition in **ICertAdminD::SetExtension (section 3.1.4.1.1)**.

FileTime: Contains a 64-bit value that represents the number of 100-nanosecond intervals since January 1, 1601, according to Coordinated Universal Time (UTC). This is used to calculate the nextUpdate value of the CRL as specified in [RFC3280] section 5 in UTC-Greenwich Mean Time.

The following processing rules apply:

Note Although [RFC3280] specifies that revocation is a requirement, this protocol differs from that requirement.

1. When this method is invoked, the CA SHOULD create a new base and/or delta CRL for each CA Signing_Private_Key in the Signing Cert Table, as specified in the following steps. The type of CRL to create (base, delta, or both) for each CA key is determined as follows:<22>

    The CA SHOULD create a new base CRL for each CA key.

    If the CA has enabled delta CRLs, as indicated by a nonzero Config_Delta_CRL_Validity_Period value, the CA MUST create a new delta CRL in addition to a new base CRL for each CA key.

    If delta CRLs are currently disabled (Config_Delta_CRL_Validity_Period is 0) and were enabled previously (Previous_Delta_CRL_Validity_Period is not equal to zero), the CA MUST create a new delta CRL in addition to a new base CRL for each CA key.

2. The CA server MUST check the Request Table to determine which certificates to include in a CRL. If Request_Revocation_Date is not NULL and is in the past:

    ▪ If the Request_Disposition is not "certificate revoked", do not include the certificate Serial_Number in the CRL.

       An implication of this rule is that, if a previously revoked certificate is both released from hold and expires between the publishing times of two successive regularly published CRLs, the serial number of the certificate does not appear in the latter CRL. This outcome does not comply with [RFC3280] section 3.3, which specifies: "An entry MUST NOT be removed from the CRL until it appears on one regularly scheduled CRL issued beyond the revoked certificate's validity period."

    ▪ If the Request_Disposition is "certificate revoked":

       1. For a delta CRL, exclude certificates whose value for the **Request_Revoked_When** field is before the CRL_This_Update of the oldest unexpired base CRL.

       2. If the Publish_Expired_Cert_In_CRL column is set to 1, the certificate Serial_Number MUST be included in CRLs regardless of the certificate's **NotAfter** time.

       3. If the Publish_Expired_Cert_in_CRL column is not set to 1:

          ▪ If the certificate's **NotAfter** time is before the CRL_This_Publish of the last CRL, do not include the certificate's Serial_Number in the CRL.

- If the certificate's **NotAfter** time is after the CRL_This_Publish of the last CRL, include the certificate's Serial_Number in the CRL.

3. At CRL creation time, the CA SHOULD create a new CRL table entry for each type of CRL (base, and if enabled, delta) for each CA key. For each CRL table entry it creates, the CA MUST use the following rules to populate the individual data elements:<23>

   - **CRL_Number:** The CA MUST set this value to the CRL_Number of the last CRL created plus one.

   - **CRL_Name_Id:** The CA MUST set this value to the number that uniquely identifies the CA key for which the CRL is being created.

   - **CRL_Min_Base:** (only for a delta CRL). If all of the following conditions are true:

     - A delta CRL is being published.

     - Delta CRLs are disabled (Config_Delta_CRL_Validity_Period is 0).

     - Delta CRLs were enabled previously (Previous_Delta_CRL_Validity_Period is not equal to zero).

     The CA MUST set this value to the CRL_Number of the most recent base CRL (identified by selecting the CRL_Number from the CRL database row with CRL_Min_Base = 0, CRL_Name_ID matching the current row, and the highest CRL_Number).

     If the three conditions listed above are not all true, the CA MUST set this value to the CRL_Number of the latest fully propagated base CRL, selected based on a **CRL_Propagation_Complete** value that is in the past and the most recent CRL_This_Update value. If there are no fully propagated CRLs available, use the oldest unexpired base CRL as the CRL_Min_Base.

   - **CRL_Effective:** (only for a delta CRL): If this element is present, the CA MUST set its value to the CRL_This_Update value of the CRL table row corresponding to the base CRL whose CRL_Number is equal to the current delta CRL's CRL_Min_Base value.

   - **CRL_Count:** The CA MUST set this value to the number of serial numbers that will be included on the CRL.

   - **CRL_This_Publish:** The CA MUST set this value to the current time.

   - **CRL_This_Update:** The CA MUST set this value to the current time minus Config_CA_Clock_Skew_Minutes.<24> The value of CRL_This_Update MUST NOT be less than the NotBefore date of the current CA signing certificate. If the calculated value of CRL_This_Update is less than the NotBefore date of the current CA signing certificate, the CA MUST replace the CRL_This_Update value with the NotBefore date of the current CA signing certificate.

   - **CRL_Next_Publish:** The CA MUST set this data element to the current time plus Config_Base_CRL_Validity_Period for a base CRL or plus Config_Delta_CRL_Validity_Period for a delta CRL.

   - **CRL_Propagation_Complete:** The CA MUST set this data element to the current time plus the overlap period.

     The Microsoft CA computes the overlap period based on Config_Base_CRL_Overlap_Period for a base CRL and Config_Delta_CRL_Overlap_Period for a delta CRL as follows:

Release: Friday, October 25, 2013

1. If the registry value for Config_Base_CRL_Overlap_Period is configured and valid (valid means the units are correct and the value is present), this value is used as is. Similarly, if a valid Config_Delta_CRL_Overlap_Period exists, it is used as is.<25>

2. If either of these values is not present or not valid, the CA uses the following algorithms to determine a value:

   For a base CRL:

   1. Interim_Base_CRL_Overlap_Period = MinimumOf(0.1*(Registry_Base_CRL_Validity_Period), 12Hours)

   2. Interim_Base_CRL_Overlap_Period = GreaterOf(Interim_Base_CRL_Overlap_Period, 1.5*(Config_CA_Clock_Skew_Minutes))

   3. Interim_Base_CRL_Overlap_Period = MinimumOf(Interim_Base_CRL_Overlap_Period, Registry_Base_CRL_Validity_Period)

   4. Config_Base_CRL_Overlap_Period = Interim_Base_CRL_Overlap_Period + Config_CA_Clock_Skew_Minutes

   For a delta CRL:

   1. Interim_Delta_CRL_Overlap_Period = MinimumOf(Registry_Delta_CRL_Validity_Period, 12Hours)

   2. Interim_Delta_CRL_Overlap_Period = GreaterOf(Interim_Delta_CRL_Overlap_Period, 1.5*(Config_CA_Clock_Skew_Minutes))

   3. Interim_Delta_CRL_Overlap_Period = MinimumOf(Interim_Delta_CRL_Overlap_Period, Registry_Base_CRL_Validity_Period)

   4. Config_Delta_CRL_Overlap_Period = Interim_Delta_CRL_Overlap_Period + Config_CA_Clock_Skew_Minutes

- If a file time is provided, which is indicated by a nonzero *FileTime* parameter value, the CA MUST use it in the following manner:

  - If the *FileTime* value is less than the current time, the CA MUST fail with an error code ERROR_INVALID_PARAMETER.

  - Otherwise, the CA MUST use the value that is provided in the *FileTime* parameter, add overlap period plus the clock skew to it, and set it as the CRL_Next_Update.

- If a file time is not provided, the CA MUST set CRL_Next_Update to the current time plus Config_Base_CRL_Validity_Period for a base CRL or Config_Delta_CRL_Validity_Period for a delta CRL plus overlap period.

  The CA SHOULD add a clock skew.<26>

- **CRL_Publish_Flags**: The CA MUST update this data element with a bitwise OR of all values from the list below that apply, based on their descriptions in section 3.1.1.4.1.

  - CPF_BASE: Set this value for a base CRL.

  - CPF_DELTA: Set this value for a delta CRL.

Release: Friday, October 25, 2013

- CPF_MANUAL: Set this value if the **uToken.Sids[uToken.UserIndex]** is not NULL. This means that the identity of the caller who initiated the generation of the CRL is available.

- CPF_SHADOW: If a delta CRL is being published, delta CRLs are disabled (Config_Delta_CRL_Validity_Period is 0), and if they were enabled previously (Previous_Delta_CRL_Validity_Period is not equal to zero), the CA MUST set this value.

4. For each type of CRL (base and/or delta) required, and for each CA key, the CA MUST create a CRL and sign it with the corresponding Signing_Private_Key. The CA MUST allocate data to the various CRL fields whose structure and syntax are defined in [RFC3280] section 5.1, to the CRL extensions whose structure and syntax are defined in [RFC3280] section 5.2, and to the CRL entry extensions whose structure and syntax are defined in [RFC3280] section 5.3:

   - The signatureAlgorithm value is based on the public key and hash algorithm of the CA key and is derived from and has the syntax as specified in [RFC3280] section 5.1.1.2.

   - As specified in [RFC3280] section 5.1.1.3, a digital signature that is computed by using the ASN.1 DER–encoded tbsCertList MUST be used as the **signatureValue** field. The tbsCertList value is composed of the fields specified in the following list.

      - "V2" MUST be used for the version field (within the TBSCertList field), and its syntax is as specified in [RFC3280] section 5.1.

      - The same value previously used for signatureAlgorithm MUST be used for the signature field (within the TBSCertList field).

      - The distinguished name (DN) of the CA MUST be used for the issuer field (within the TBSCertList field). This is the same value as the value that is contained in the Issuer field in the CA certificate associated with the signing key. The syntax is as specified in [RFC3280] section 5.1.2.3.

      - CRL_This_Update MUST be used for the **thisUpdate** field of the CRL, and its syntax is as specified in [RFC3280] section 5.1.2.4.

      - CRL_Next_Update MUST be used for the **nextUpdate** field of the CRL, and its syntax is as specified in [RFC3280] section 5.1.2.5.

      - For each certificate that is determined in the preceding rule 3 to be included in the CRL, an entry in the revokedCertificates list MUST be created using the following logic:

         - If Request_Revoked_Reason is release from hold (0xffffffff) and a base CRL is being constructed, the certificate MUST NOT be included as an entry in the revokedCertificates list.

         - If Request_Revoked_Reason is release from hold (0xffffffff) and a delta CRL is being constructed, 8 MUST be used for the reasonCode CRL entry extension.

         - Serial_Number MUST be used for the userCertificate field.

         - Request_Revocation_Date MUST be used as the revocationDate field.

         - If Request_Revoked_Reason is not 0, it MUST be used for the reasonCode CRL entry extension.

         - The fields and CRL entry extensions within revokedCertificates have the ASN syntax as defined in [RFC3280] section 5.1.

Release: Friday, October 25, 2013

- The subject key identifier field of the CA certificate for which the CRL is being generated MUST be used for the AuthorityKeyIdentifier (AKI) CRL extension. The syntax is as defined in [RFC3280] section 5.2.1 (which in turn points to section 4.2.1.1 of the same RFC).

- The CRL_Number SHOULD be used for the CRL number extension, which has the syntax that is defined in [RFC3280] section 5.2.3.<27>

- For a delta CRL only, CRL_Min_Base MUST be used for the delta CRL indicator extension, which has the syntax that is defined in [RFC3280] section 5.2.3.<28>

- The issuing distribution point is a CRL extension that MUST be populated with all entries in the Config_CA_CDP_Include_In_CRL_IDP_Extension element that is defined in [MS-WCCE] section 3.2.1.1.4. This CRL extension has the syntax that is defined in [RFC3280] section 5.2.5.

- The freshest CRL extension (also known as the delta CRL distribution point) MUST be populated with all entries in the Config_CA_CDP_Include_In_CRL_Freshest_CRL_Extension element, as defined in [MS-WCCE] section 3.2.1.1.4. This CRL extension has the syntax that is defined in [RFC3280] section 5.2.6.

5. In addition to the preceding extensions, the following custom extensions MUST be added to the CRL.

- CRL_Next_Publish MUST be used for the CRL Next Publish custom extension, which is defined as follows:

  The OID of the CRL Next Publish extension is 1.3.6.1.4.1.311.21.4.

  The extension value is DER-encoded and is defined in ASN.1 [X509] as the following:

  ```
  CHOICE {
      utcTime      UTCTime,
      generalTime    GeneralizedTime
  }
  ```

  If the time is after 1950 and before 2050, it is UTC time encoded with a two-digit year. Otherwise, it is Generalized time that is encoded with a four-digit year. The date is precise to seconds.

- The CA Version extension MUST be populated with the value "{Signing_Cert_Version_ID}{Signing_Private_Key_Version_ID}" where Signing_Cert_Version_ID and Signing_Private_Key_Version_ID are replaced with their values as defined in [MS-WCCE] section 3.2.1.1.2.

  The OID of the CA Version extension is 1.3.6.1.4.1.311.21.1.

  This extension MUST NOT be marked critical.

  The extension value is DER-encoded and is defined in ASN.1 [X509] as INTEGER.

- The Published CRL Locations extension (an Active Directory path) MUST be populated with all entries from the **Config_CA_CDP_Include_In_CRL_Publish_Locations_Extension** ADM element, as defined in [MS-WCCE] section 3.2.1.1.4.

  The OID of the Published CRL Locations extension is 1.3.6.1.4.1.311.21.14.

  This extension MUST NOT be marked critical.

The extension value is DER-encoded and is defined in ASN.1 [X509] with the same syntax as the CRL distribution points certificate extension, as defined in [RFC3280] section 4.2.1.14.

6. After each CRL is created, the CA MUST save it to the **CRL_Raw_CRL** data element of the corresponding CRL table entry.

7. For each CRL created, immediately prior to publishing, the CA MUST verify the signature in the **signatureValue** field of the CRL. The inputs to signature verification are as follows:

   ▪ The signature value itself.

   ▪ The original message that was signed (in this case the CRL tbsCertList as defined in [RFC3280] section 5.1.1.1).

   ▪ The CA public key (from the Subject Public Key Information field of the Signing_Cert_Certificate corresponding to the Signing_Private_Key the CA used to sign the CRL).

   ▪ The signature algorithm, as identified in the **signatureAlgorithm** field of the CRL.

   The detailed signature verification process depends on the algorithm. Algorithms for signing CRLs are described in [RFC3279] section 2.2. If the verification is not successful, the CA MUST skip to step 8 below and use 0x80090006 for the error code.

8. The CA MUST then consult the **Config_CA_CDP_Publish_To_Base** and **Config_CA_CDP_Publish_To_Delta** ADM elements and attempt to publish base CRLs and delta CRLs to all of the locations identified in the respective elements, including the local registry location defined in section 3.1.1.8, noting the success or failure of each publish attempt. In the case of failure, the CA MUST note the error code generated by the CA or returned from the lower layer system for each publishing attempt. Steps 8 and 9 of this section are performed once for each CRL and location combination. When writing to the local registry location, the server uses methods consistent with the communications specified in the Windows Remote Registry Protocol (see [MS-RRP] section 4.2) to write a new registry value. The operations to write a new registry value are as follows:

   ▪ First, read all CRLs in the **CRL_Local** element to determine whether a CRL of the same type (base or delta) and signed by the Signing_Private_Key that signed the CRL being published exists already. If one is found, compare its content to that of the current CRL being published. If the content is identical, do nothing. If not, delete the existing CRL from **CRL_Local**. When all CRLs signed by the CRL being published have been evaluated, if no exact match was found, add the current CRL to **CRL_Local** as follows:

      ▪ Obtain a handle to the root key HKEY_LOCAL_MACHINE by using the **OpenLocalMachine** method [MS-RRP] section 3.1.5.3.

         ▪ Open the HKEY_LOCAL_MACHINE root key with the following input:

         samDesired: 0x6

      ▪ Use the handle returned from **OpenLocalMachine** with the **BaseRegOpenKey** method ([MS-RRP] section 3.1.5.15) to open a subkey. The **BaseRegOpenKey** method returns a handle to the subkey.

         ▪ Open the subkey with the following input:

         dwOptions: 0 (nonvolatile)

Release: Friday, October 25, 2013

samDesired: 0x6

- Use the handle to the subkey returned from **BaseRegOpenKey** with the **BaseRegCreateKey** method ([MS-RRP] section 3.1.5.7) to create a new subkey.

  - Create the new subkey with the following input:

  lpSubKey: The SHA-1 hash of the DER encoded CRL, including tbsCertList and appended signature (the 20 bytes of binary data output from the hash algorithm are converted to a string of 40 hexadecimal ASCII digits).

  lpClass: NULL

  dwOptions: 0 (nonvolatile)

  samDesired: 0x6

- Use the handle to a subkey returned from **BaseRegCreateKey** to write values under the subkey by using the **BaseRegSetValue** ([MS-RRP] section 3.1.5.22) method.

  - Create the new value with the following input:

  lpValueName: "Blob"

  dwType: 3 (binary)

  lpData: Points to a buffer containing the DER encoded CRL. The buffer is constructed as follows: The first 12 bytes contain the values represented in hexadecimal as 03 00 00 00 01 00 00 00 14 00 00 00, followed by the 20-byte SHA-1 hash value used above. The SHA-1 hash value is followed by 8 bytes that contain the value represented in hexadecimal as 21 00 00 00 01 00 00 00, then 4 bytes that contain the value of the length of the CRL, followed by the DER encoded CRL.

  cbData: The length, in bytes, of the data referenced by lpdata.

- After all required keys and values have been created and written, the client closes the open handles by using the **BaseRegCloseKey** method [MS-RRP] section 3.1.5.6.

If a nonzero return value is returned from any of the above methods, the CA continues to attempt to publish CRLs as follows, and the CPF_CASTORE_ERROR flag is set in step 9 below.

For each combination of new CRL table entry (created in preceding rule 3) and CDP location, there will be a publishing attempt, in which the CA attempts to write the CRL file to the location specified. Specifically, the publishing attempts are done as follows: <29>

- If, for a delta CRL and a location that begins with "file://", the CPF_FILE_ERROR flag value was set for the corresponding base CRL, do not attempt to publish the delta CRL to the file:// location. The CA MUST generate a nonzero error code instead. The CA SHOULD use the error code E_ABORT (0x80004004). The server will mark the CPF_POSTPONED_BASE_FILE_ERROR flag when it updates the corresponding CRL table entry below. The CA MUST then preserve this error code and skip to step 9.

- If, for a delta CRL and a location that begins with "ldap:///", the CPF_LDAP_ERROR flag value was set for the corresponding base CRL, do not attempt to publish the delta CRL to the ldap:/// location. The CA MUST generate a nonzero error code instead. The CA SHOULD use the error code E_ABORT (0x80004004). The server will mark the CPF_POSTPONED_BASE_LDAP_ERROR flag when it updates the corresponding CRL table entry below. The CA MUST then preserve this error code and skip to step 9.

- If any CRL Publishing Locations that have the invalid prefix "ftp:", "http:", or any locations with a prefix other than "file:" that consists of 2, 3, or 4 characters followed by a colon (":"), are encountered, the CA MUST generate a nonzero error code. The CA SHOULD use the error code ERROR_BAD_PATHNAME (0x800700a1). The CA MUST then preserve this error code and skip to step 9.

- When writing to a local file path, UNC path, or file:// location, a file system is invoked to write the CRL as a DER-encoded binary file to the path and file name specified. When the CRL publishing location starts with the prefix "file://", but the remaining portion of the location is not a path starting with "\\", the entire location is passed to the file system. When the location starts with the prefix "file://", and the remaining portion is a path starting with "\\", only the remaining portion is passed to the file system. When the location starts with "{DriveLetter}:" or with "\\", it is passed as is to the file system. A status or Win32 error code, when one was returned from the file system, is returned to the invoker (the CA). When a Win32 error code is returned from the file system, the server SHOULD convert this error code to a 4-byte HRESULT value using the pattern 0x8007XXXX, where XXXX is the first two bytes of the Win32 hex value 0x0000XXXX. Then, the server SHOULD preserve this HRESULT value, and go to step 9 (where the CA will set the CPF_FILE_ERROR publishing flag). For more information on writing remote files, see [MS-FASOD].<30>

- For any ldap:/// locations identified, within the object and attribute specified, the CRL is DER encoded. The steps the CA uses are as follows:

  1. If the value of the **CRL_Publish_AD_Connection** is not NULL, then a previous Active Directory connection exists. Go to step 4 and use that Active Directory connection for publishing the CRLs.

     Otherwise, invoke the task "Initialize an ADConnection", as defined in [MS-ADSO] section 6.2.6.1.1, with the following parameters:

     - **TaskInputTargetName**: NULL.

     - **TaskInputPortNumber**: If the value of **Config_CA_LDAP_Flags** is 0, use 389; if it is 1, use port 636.

     - Store the new **TaskReturnADConnection** returned from the task as **CRL_Publish_AD_Connection**.

  2. Invoke the task "Setting an LDAP Option on an ADConnection", as defined in [MS-ADSO] section 6.2.6.1.2, on the Active Directory connection **CRL_Publish_AD_Connection**. Invoke the task one time for each of the following options. For each of these, the *TaskInputAdConnection* parameter is **CRL_Publish_AD_Connection**.

     TaskInputOptionName: LDAP_OPT_GETDSNAME_FLAGS

     TaskInputOptionValue: Bit R as defined by [MS-NRPC] section 3.5.4.3.1

     TaskInputOptionName: LDAP_OPT_SIGN

     TaskInputOptionValue: TRUE

     TaskInputOptionName: LDAP_OPT_DNSDOMAIN_NAME

     TaskInputOptionValue: Set to the domain DNS name of the joined domain.

3. After the Active Directory connection is initialized and the options are set, the CA invokes the "Performing an LDAP Bind on an ADConnection" task, as specified in [MS-ADSO] section 6.2.6.1.4, on the connection **CRL_Publish_AD_Connection**, with the following parameters:

   TaskInputADConnection: **CRL_Publish_AD_Connection**

   If the **TaskReturnStatus** returned is not 0, skip to the error processing logic in step 5 below. Otherwise, continue.

4. The CA invokes the "Performing an LDAP Operation on an ADConnection" task, as specified in [MS-ADSO] section 6.2.6.1.6, with the following parameters:

   TaskInputADConnection: **CRL_Publish_AD_Connection**

   TaskInputRequestMessage: **protocolOp** is set to **modifyRequest** ([RFC2251] section 4.6)

   The parameters of the ModifyRequest are set as follows:

   - The **object** field contains the Config_CA_CDP_Publish_To_Base or Config_CA_CDP_Publish_To_Delta entry that specifies an LDAP location (without the "ldap:///" prefix)

   - The modification sequence has one list entry whose **operation** field has the value "replace" and whose **modification** field contains the following:

     - The **type** field contains either "certificateRevocationList", for a base CRL, or "deltaRevocationList" for a delta CRL.

     - The **vals** field contains the CRL. <31>

   TaskOutputResultMessages: A task output parameter that is a list of **LDAPMessage**, as specified in [MS-ADSO] section 6.2.6.1.6, that contains the response from the directory server. The result of a modify request is a modify response, which, as specified in [RFC2251], is an **LDAPResult**, which contains a **resultCode** and an **errorMessage**.

   If the **TaskReturnStatus** returned is not 0, perform to the error processing logic in step 5 below. Otherwise, skip step 5 and continue with step 9 below.

5. When invoking an LDAP higher layer triggered event, a status or error code, when one was returned from the lower layer system, is returned to the invoker (the CA).

   - If a nonzero **TaskReturnStatus** is returned from step 3 above (while attempting to bind to the ADConnection), the CA SHOULD close the Active Directory connection by invoking the "Performing an LDAP Unbind on an ADConnection" task defined in [MS-ADSO] section 6.2.6.1.5, convert the **TaskReturnStatus** (an LDAP **resultCode**) that was returned from step 3 to a 4-byte HRESULT value as specified in the Conversion of LDAP results at the end of this step. Then, preserve this HRESULT value, clear the **CRL_Publish_AD_Connection** element, and go to step 9 below.

   - If a nonzero **TaskOutputResultMessages.resultCode** value is returned from step 4 above (the LDAP modify operation), that indicates that the Active Directory server is down, unavailable, or that there is a timeout, the CA MUST close the Active Directory

connection by invoking the "Performing an LDAP Unbind on an ADConnection" task defined in [MS-ADSO] section 6.2.6.1.5, clear the **CRL_Publish_AD_Connection** element, and re-attempt one time to execute steps 1-4 above. If a nonzero **TaskReturnStatus** is returned from step 3 or 4 upon this single retry, the CA MUST close the Active Directory connection by invoking the "Performing an LDAP Unbind on an ADConnection" task defined in [MS-ADSO] section 6.2.6.1.5, convert the returned **TaskReturnStatus** from step 3 (in the case of a failure upon bind) or **TaskOutputResultMessages.errorMessage** from step 4 (in the case of a failure upon modify) to a 4-byte HRESULT value as specified in the Conversion of LDAP results at the end of this step, preserve this HRESULT value, clear the CRL_Publish_AD_Connection element, and go to step 9.<32>

- If any other nonzero **TaskOutputResultMessages.resultCode** value is returned from step 4 above (the LDAP modify operation), the server MUST convert the returned **TaskOutputResultMessages.errorMessage** to a 4-byte HRESULT value, as specified in the Conversion of LDAP results at the end of this step, preserve this HRESULT value, and go to step 9.

Conversion of LDAP results (**TaskReturnStatus** or **TaskOutputResultMessages**) to HRESULT is as follows:

- If **TaskOutputResultMessages.errorMessage** is at least 8 bytes long:

  - If each of the first 8 bytes is between '0' and '9' inclusive, or between 'a' and 'f' inclusive, or between 'A' and 'F' inclusive, then use this value as the hexadecimal representation of a Win32 error, which will be the input to the following HRESULT conversion.

  - If each of the first 8 bytes is 0, use ERROR_DS_GENERIC_ERROR (defined in [MS-ERREF] section 2.2, "Win32 Error Codes") as the Win32 error that will be input to the following HRESULT conversion.

- Otherwise, use **TaskOutputResultMessages.resultCode**. Convert it to a Win32 error using the conversion specified in [MS-ERREF] section 2.4, "LDAP Error to Win32 Error Mapping", and use this value as the Win32 error, which will be the input to the following HRESULT conversion.

- If there is no **TaskOutputResultMessages** (as in the case of the Bind task), use **TaskReturnStatus** (which is an LDAP resultCode). Convert this value to a Win32 error using the conversion specified in [MS-ERREF] section 2.4, "LDAP Error to Win32 Error Mapping", and use this value as input to the following HRESULT conversion.

- Convert the Win32 obtained above to an HRESULT using the pattern 0x8007XXXX, where XXXX is the first two bytes of the Win32 hex value 0x0000XXXX.

9. After each attempt to publish a CRL (that is, each combination of CRL and publishing location), the CA SHOULD update the corresponding CRL table row. For each row updated, the CA MUST use the following rules to update the individual data elements:<33>

- **CRL_Publish_Error:** If this element is present, populate CRL_Publish_Error with the user name associated with the caller identity, in the form "Published by {domain\UserName}", where:

  - Domain is replaced with the domain name of the domain in which the user account exists.

  - UserName is the user name associated with the caller identity.

The domain and UserName are found as follows:

1. Call **LsarOpenPolicy** [MS-LSAT] section 3.1.4.2 with the following inputs:

   ▪ SystemName: NULL

   ▪ DesiredAccess: contains the bit value 0x00000800 for POLICY_LOOKUP_NAMES

2. Call **LsarLookupSids** [MS-LSAT] section 3.1.4.11 on the returned PolicyHandle.

   ▪ PolicyHandle: the PolicyHandle returned from LsarOpenPolicy above.

   ▪ SidEnumBuffer: contains the SID from the ADM element **uToken.Sids[UserIndex]**.

      The return values from the LsarLookupSIDs are as follows:

      ▪ ReferencedDomains list: the domain name is found in the **Name** field of the Domains structure of the list entry whose index matches the DomainIndex of the Names structure of the entry in the TranslatedNames list that corresponds to the SID in question.

      ▪ TranslatedNames contains the UserName in the **Name** field of the Names structure of the entry in the list corresponding to the SID in question (from the SidEnumBuffer input list).

The Microsoft CA uses "–" instead of "Published by" for publish attempts made in the CA machine context, such as automatically scheduled CRL publishing attempts.

If step 8 resulted in an error code or nonzero HRESULT, the caller identity information SHOULD be followed by the CRL publishing location to which the CRL could not successfully be published. The Microsoft CA appends the username information with a space, two hyphens, a space, and an integer. The integer indicates the index (in the list of CRL publishing locations that is written to the same data element) of the location to which publishing failed. If more than one location fails for one CRL table entry, then the index of each failed location is appended to this line, separated by spaces. This is followed by two line feed characters, followed by the path of each CRL publishing location to which the CRL could not successfully be published, separated by newline characters.

▪ **CRL_Publish_Flags:** The CA MUST update this data element with a bitwise OR of its existing value and all values from the list below that apply, based on their descriptions in section 3.1.1.4.1:

   ▪ CPF_BADURL_ERROR: Set this value if a URL that does not meet the format requirements specified in section 3.1.1.8 for Config_CA_CDP_Publish_To_Base and Config_CA_CDP_Publish_To_Delta was encountered during publishing of the CRL.

   ▪ CPF_FILE_ERROR: Set this value if a local file path, UNC path, or file:// URI that does not meet the format requirements specified in section 3.1.1.8 for Config_CA_CDP_Publish_To_Base and Config_CA_CDP_Publish_To_Delta for a file location, was encountered during publishing of the CRL, or if step 8 resulted in an error code or nonzero HRESULT as a result of the publishing attempt to a file location.

   ▪ CPF_HTTP_ERROR: Set this value if an HTTP URI (a URI with the prefix "http:") was encountered during publishing of the CRL.

   ▪ CPF_FTP_ERROR: Set this value if an FTP URI (a URI with the prefix "ftp:") was encountered during publishing of the CRL.

Release: Friday, October 25, 2013

- CPF_LDAP_ERROR: Set this value if step 8 resulted in an error code or nonzero HRESULT as a result of the publishing attempt to an LDAP location.

- CPF_POSTPONED_BASE_LDAP_ERROR: Set this value if the server postponed publishing a delta CRL due to a failure in publishing a base CRL to an ldap:/// location (see rule 7 above), for example, when the corresponding base CRL could not be published to an LDAP location because of an error.

- CPF_POSTPONED_BASE_FILE_ERROR: Set this value if the server postponed publishing a delta CRL due to a failure in publishing a base CRL to a file:// location (see rule 7 above), for example, when the corresponding base CRL could not be published to a FILE location because of an error.

- CPF_SIGNATURE_ERROR: Set this value if an error occurred when verifying the signature of the generated CRL prior to attempting to publish the CRL.

- CPF_CASTORE_ERROR: Set this value if an error occurred when publishing the generated CRL to the default local registry location.

10. After publishing attempts have been made for all CRL publishing locations for a given CRL, the CA MUST use the following rules to update the following CRL table data elements in the corresponding CRL table row:

- CRL_Publish_Status_Code:

    - If an error code or nonzero HRESULT value was generated by the CA or returned from a lower layer system upon any of the previous attempts to publish the CRL (steps 8 and 9 above), the CA MUST set this value to the value generated or returned. If more than one error was generated or returned, the CA SHOULD set this value to the error code that resulted from the first unsuccessful publishing attempt. Error codes are specified in [MS-ERREF].

    - Otherwise, the CA MUST set this value to 0.

- CRL_Publish_Flags:

    - CPF_COMPLETE: The CA MUST set this value if the CRL published successfully to all locations (that is, the CRL_Publish_Status_Code was set to 0).

- **CRL_Last_Published** (for a base CRL only): If this element is present, the CA MUST set its value to the current time.

- **CRL_Publish_Attempts**: If this element is present, the CA MUST set its value to 1 to indicate the publish attempt for the newly created CRL.

11. After all publishing attempts have completed (for all base and delta CRLs for all CA keys to all locations), the CA MUST update the Config_CA_CRL_Next_Publish, OnNextRestart_Config_CA_CRL_Next_Publish, Config_CA_CRL_Delta_Next_Publish, and OnNextRestart_Config_CA_CRL_Delta_Next_Publish data elements. The CA MUST update the Config_CA_CRL_Next_Publish and OnNextRestart_Config_CA_CRL_Next_Publish configuration data elements with the value that was calculated for CRL_Next_Publish for the base CRL(s) (the current time plus Config_Base_CRL_Validity_Period), and the CA MUST update the Config_CA_CRL_Delta_Next_Publish and OnNextRestart_Config_CA_CRL_Delta_Next_Publish configuration data elements with the value that was calculated for CRL_Next_Publish for any delta CRL(s)created above (the current time plus Config_Delta_CRL_Validity_Period for a delta CRL).

12. If steps 8 or 9 above resulted in an error code or nonzero HRESULT value from any attempt to publish any of the CRLs, set the value of the ADM elements **Config_CA_CRL_Attempt_Republish** and **OnNextRestart_Config_CA_CRL_Attempt_Republish** to "1" and set the timer documented in section 3.1.2.2. If none of the attempts to publish the CRLs failed, reset the ADM elements **Config_CA_CRL_Attempt_Republish** and **OnNextRestart_Config_CA_CRL_Attempt_Republish** to "0".

13. The CA MUST then set the value of Previous_Delta_CRL_Validity_Period to the current value of Config_Delta_CRL_Validity_Period.

Return value: The method returns the first error code returned from the first CRL write attempt that failed or that was aborted. If none of the CRL write attempts failed, the method returns 0.

3.1.4.1.7 ICertAdminD::GetCRL (Opnum 9)

The **GetCRL** method instructs the CA to return the current base CRL for the current CA key.

```
HRESULT GetCRL(
  [in, string, unique] wchar_t const* pwszAuthority,
  [out, ref] CERTTRANSBLOB* pctbCRL
);
```

pwszAuthority: See the *pwszAuthority* definition in section 3.1.4.1.1.

pctbCRL: If the function succeeds, this method MUST return a **CERTTRANSBLOB** structure that contains the ASN.1 DER (as specified in [X660] and [X690]) encoded CRL (CRLRawCRL) for the CA server's current signing certificate.

The **GetCRL** method instructs the CA to return the recent base CRL, which is signed with the current CA key to the caller. If a CRL cannot be found, the CA MUST return ERROR_FILE_NOT_FOUND, as specified in [MS-ERREF].<34>

3.1.4.1.8 ICertAdminD::RevokeCertificate (Opnum 10)

The **RevokeCertificate** method revokes a certificate either immediately or on a specified date. It instructs the CA to revoke a certificate based on the certificate's serial number and reason code.

```
HRESULT RevokeCertificate(
  [in, string, unique] wchar_t const* pwszAuthority,
  [in, string, unique] wchar_t const* pwszSerialNumber,
  [in] DWORD Reason,
  [in] FILETIME FileTime
);
```

pwszAuthority: See the *pwszAuthority* definition in **ICertAdminD::SetExtension (section 3.1.4.1.1)**.

pwszSerialNumber: A null-terminated Unicode string that specifies a serial number that identifies the certificate to be revoked. The string MUST specify the serial number as plain hexadecimal digits (no leading 0x) as specified in [RFC3280] section 4.1.2.2.<35>

Reason: An unsigned integer value that specifies the revocation reason code. The revocation reason code MUST be either one of the values listed in the following table (and specified in [RFC3280] section 5.3.1), or one of the following values: 0xfffffffd, 0xfffffffe, or 0xffffffff.

Value	Meaning
0	unspecified
1	keyCompromise
2	cACompromise
3	affiliationChanged
4	superseded
5	cessationOfOperation
6	certificateHold
8	removeFromCRL
0xfffffffd	See processing rules, beginning with rule 2.
0xfffffffe	See processing rules, beginning with rule 3.
0xffffffff	Released from hold. (See processing rules, beginning with rule 4.)

FileTime: Contains a 64-bit value that represents the number of 100-nanosecond intervals since January 1, 1601 (UTC). This value specifies the date, according to Greenwich Mean Time, when the certificate became invalid. The *FileTime* corresponds to the Request_Revocation_Date that is defined in section 3.1.1.1.

The following processing rules apply:

1. The CA MUST find a row in the request table that contains the certificate that needs to be revoked in this method invocation. The CA finds the row by comparing the value of the *pwszSerialNumber* parameter to the values of the Serial_Number column in the request table. This is a case-sensitive string comparison. If none of the rows in the Request table have a Serial_Number value that is identical to the value that is passed in the *pwszSerialNumber* parameter, the CA MUST fail the request. The error returned SHOULD be 0x80070057. When a row that has an identical serial number, as previously specified, is found, the CA SHOULD continue with the following processing rules.

   Note These processing rules refer to this row as the identified row.

2. If the *Reason* parameter is 0xfffffffd, the CA MUST set the Publish_Expired_Cert_In_CRL column of the identified row to 0 and return successfully.<36>

3. If the *Reason* parameter is 0xfffffffe, the CA MUST set the Publish_Expired_Cert_In_CRL column of the identified row to 1 and return successfully.<37>

4. If the value for the *Reason* parameter is 0xffffffff and the value of the Request_Revoked_Reason column in the identified row is not "certificateHold", the CA MUST fail the request, and the error code SHOULD be ERROR_INVALID_DATA.

5. If the *Reason* parameter is not 0, 1, 2, 3, 4, 5, 6, 8, 0xfffffffd, 0xfffffffe, or 0xffffffff, the CA MUST fail the request. The error code SHOULD be E_INVALIDARG (0x80070057), as specified in [MS-ERREF] section 2.1. Otherwise, the CA MUST continue with the following processing rules. The Windows mnemonic for reason code 6 is CRL_REASON_CERTIFICATE_HOLD.

6. If the Request_Disposition column of the identified row equals "certificate issued", the CA MUST set the Request_Disposition column of the identified row to "certificate revoked", and the CA SHOULD set the value of the Request_Disposition_Message column in the identified row to any value that the implementer considers informative for presentation to a human. In this case, the Microsoft CA sets Request_Disposition_Message to "Revoked by *{username}*" where "*{username}*" is replaced with the user name of the caller.

7. If the Request_Disposition column of the identified row equals "certificate revoked":

   1. If the Request_Revoked_Reason column of the identified row equals "certificateHold":

      ▪ If the *Reason* parameter is 0xffffffff, the CA MUST set the Request_Disposition column to "certificate issued", and the CA SHOULD set the value of the Request_Disposition_Message column in the identified row to any value the implementer considers informative for presentation to a human. In this case, the Microsoft CA sets Request_Disposition_Message to "Revoked by *{username}*" where "*{username}*" is replaced with the user name of the caller.

      ▪ The CA MUST continue with processing, following rules 9 and 10.

   2. If the Request_Revoked_Reason column of the identified row does not equal "certificateHold":

      ▪ If the *Reason* parameter is 0xffffffff, the CA MUST fail the request, as specified in the preceding rule 4.

      ▪ If the *Reason* parameter is "certificateHold", the CA MUST fail the request. The error code SHOULD be ERROR_INVALID_DATA (0x8007000D), as specified in [MS-ERREF] section 2.1.

      ▪ If the *Reason* parameter does not equal 0xffffffff and does not equal "certificateHold", the CA MUST continue with processing, following rules 9 and 10.

         One implication of this rule is that the Windows CA allows updating of the Request_Revoked_Reason and Request_Revocation_Date values of a certificate that has already been revoked. Although [RFC3280] section 5.3.3 describes revocationDate as the "date at which the CA processed the revocation", the Windows CA allows a different date to be used.

8. If the Request_Disposition column of the identified row does not equal "certificate issued" and the Request_Disposition column of the identified row does not equal "certificate revoked", the CA MUST fail the request. The error code SHOULD be ERROR_INVALID_DATA, as specified in [MS-ERREF] section 2.1.

9. The CA MUST set the Request_Revoked_Reason column of the identified row to the value passed in the *Reason* parameter.

10. The CA MUST set the Request_Revocation_Date column of the identified row to the value passed in the *FileTime* parameter.

11. The CA MUST set the Request_Revoked_When column of the identified row to the current time.

3.1.4.1.9   ICertAdminD::EnumViewColumn (Opnum 11)

The **EnumViewColumn** method returns an array of column information.

```
HRESULT EnumViewColumn(
  [in, string, unique] wchar_t const* pwszAuthority,
  [in] DWORD iColumn,
```

```
  [in] DWORD cColumn,
  [out] DWORD* pcColumn,
  [out, ref] CERTTRANSBLOB* pctbColumnInfo
);
```

pwszAuthority: See the definition of the *pwszAuthority* parameter in
ICertAdminD::SetExtension (section 3.1.4.1.1).

iColumn: An unsigned integer that specifies the identifier of the column with which to begin the
enumeration. Valid values are from 0 to one less than the maximum number of columns for
the Request table.

cColumn: An unsigned integer that specifies the requested number of columns to return.

pcColumn: A pointer to an unsigned integer that receives the actual number of
CERTTRANSDBCOLUMN structures returned by the server in the *pctbColumnInfo* parameter.

pctbColumnInfo: A pointer to a **CERTTRANSBLOB** structure. Upon return, the **pb** member of
this structure points to an array of the marshaled **CERTTRANSDBCOLUMN** structures as
described in section 2.2.1.7.

The **EnumViewColumn** method returns information about the columns that are associated with the
Request table to the client. The processing rules for this method are the same as for the
EnumViewColumnTable method with the *iTable* parameter set to 0x00000000.

3.1.4.1.10 ICertAdminD::GetViewDefaultColumnSet (Opnum 12)

The **GetViewDefaultColumnSet** method returns an array of column identifiers that are associated
with a specific view.

```
HRESULT GetViewDefaultColumnSet(
  [in, string, unique] wchar_t const* pwszAuthority,
  [in] DWORD iColumnSetDefault,
  [out] DWORD* pcColumn,
  [out, ref] CERTTRANSBLOB* pctbColumnInfo
);
```

pwszAuthority: See the definition of the *pwszAuthority* parameter in
ICertAdminD::SetExtension (section 3.1.4.1.1).

iColumnSetDefault: An unsigned integer value that specifies the requested default column set
to get. The value MUST be one of the values in the following table. If a value other than one of
the listed values is used, the error E_INVALIDARG is returned.

Value	Meaning
0xFFFFFFFF	The caller attempts to retrieve the list of column identifiers that are useful for viewing pending requests.
0xFFFFFFFE	The caller attempts to retrieve the list of column identifiers that are useful for viewing issued certificates.
0xFFFFFFFD	The caller attempts to retrieve the list of column identifiers that are useful for viewing failed requests.

Release: Friday, October 25, 2013

Value	Meaning
0xFFFFFFFC	The caller attempts to retrieve the list of column identifiers that are useful for viewing extensions.
0xFFFFFFFB	The caller attempts to retrieve the list of column identifiers that are useful for viewing attributes.
0xFFFFFFFA	The caller attempts to retrieve the list of column identifiers that are useful for viewing CRLs.
0xFFFFFFF9	The caller attempts to retrieve the list of column identifiers that are useful for viewing revoked certificates.

pcColumn: A pointer to the unsigned integer that receives the count of Column identifiers that are returned by the server in the *pctbColumnInfo* parameter.

pctbColumnInfo: A pointer to a **CERTTRANSBLOB** structure. Its **cb member** field MUST contain the length, in bytes, of the array that is referenced by the **pb member** field. The **pb member** field MUST point to an array of DWORDs, where each DWORD value represents the identifier for a column. Each **DWORD** in the array is marshaled by using little-endian format.

The CA server MUST return the array of associated column identifiers in the following manner.

Value of the iColumnSetDefault parameter	Processing rule
0xFFFFFFFF	The CA MUST return a subset of column identifiers for a pending request view from the Request table.<38>
0xFFFFFFFE	The CA MUST return a subset of column identifiers for issued and revoked certificate views and failed request view from the Request table.<39>
0xFFFFFFFD	The CA MUST return a subset of column identifiers for a failed request view from the Request table.<40>
0xFFFFFFFC	The CA MUST return a subset of column identifiers for an extension view from the Extension table.<41>
0xFFFFFFFB	The CA MUST return a subset of column identifiers for an attribute view from the Attribute table.<42>
0xFFFFFFFA	The CA MUST return a subset of column identifiers for a CRL view from the CRL table.<43>
0xFFFFFFF9	The CA MUST return a subset of column identifiers for a revoked certificate view from the Request table.<44>

3.1.4.1.11   ICertAdminD::EnumAttributesOrExtensions (Opnum 13)

The **EnumAttributesOrExtensions** method is used to access sets of attributes or extensions for a particular row ID.

```
HRESULT EnumAttributesOrExtensions(
   [in, string, unique] wchar_t const* pwszAuthority,
```

```
  [in] DWORD RowId,
  [in] DWORD Flags,
  [in, string, unique] wchar_t const* pwszLast,
  [in] DWORD celt,
  [out] DWORD* pceltFetched,
  [out, ref] CERTTRANSBLOB* pctbOut
);
```

pwszAuthority: See the *pwszAuthority* definition in section **ICertAdminD::SetExtension (section 3.1.4.1.1)**.

RowId: An unsigned integer that specifies the RequestID value of the row to retrieve attributes or extensions.

Flags: An unsigned integer value that MUST take either of the following values.

Value	Meaning
0x00000000	Enumerate attributes
0x00000001	Enumerate extensions

pwszLast: A pointer to a null-terminated UNICODE string that specifies the name of the attribute or extension beyond which the data is requested. If the value of **Flags** is 1, the name MUST be an OID string as specified in [X680].

celt: An unsigned integer value that specifies the requested count of attributes (**CERTTRANSDBATTRIBUTE**) or extensions (**CERTTRANSDBEXTENSION**) structures to be returned to the client.

pceltFetched: A pointer to the unsigned integer that receives the actual count of the attributes (**CERTTRANSDBATTRIBUTE**) or extensions (**CERTTRANSDBEXTENSION**) structure data returned by the server in the *pctbOut* parameter.

pctbOut: A pointer to the **CERTTRANSBLOB** structure. The data returned is marshaled **CERTTRANSDBATTRIBUTE** or **CERTTRANSDBEXTENSION** structure array as described in **CERTTRANSDBATTRIBUTE** and **CERTTRANSDBEXTENSION**.

The **EnumAttributesOrExtensions** method obtains information about the attributes or extensions (as specified in [MS-WCCE] section 2.2.2.7) that are associated with a specific request in the Request table.

The CA server MUST apply the following ordered processing rules. If an error is encountered, the CA SHOULD return the specified error and terminate the processing of the method:

1. The CA server MUST enforce that the *Flags* parameter is either 0 or 1; otherwise, it MUST return an error. The error SHOULD be ERROR_INVALID_PARAMETER.

2. The CA server MUST enforce that the *RowId* parameter value is greater than 0 and that a row exists in the Request table with the specified *RowId* in the Request_Request_ID column. Otherwise, the CA Server MUST return an error code. If the *RowId* parameter value is not greater than 0, the error code SHOULD be ERROR_INVALID_PARAMETER. If the row does not exist, the error SHOULD be 0x80094004.

3. If the value of the *Flags* parameter is 0x00000000, the CA MUST compute the set of rows from the Attribute table, where each row MUST have the same value in its Request_ID_Attribute

Release: Friday, October 25, 2013

column as the *RowId* parameter value. If the *pwszLast* parameter is not NULL, the CA MUST find the row in the set that has the same value in the AttributeName as the value of the *pwszLast* parameter (case-insensitive comparison). If the row is not found, the CA MUST fail. The error code SHOULD be 0x80094004. If the row is found, the CA MUST remove the rows prior to this row (including this row) from the set of rows to return to the client.

> **Note** The computed set of rows from this step is referred to as the "resultant set" in steps 5 and 7.

4. If the value of the *Flags* parameter is 0x00000001, the CA MUST compute the set of rows from the Extension table, where each row MUST have the same value in its Request_ID_Extension column as the *RowId* parameter value. If the *pwszLast* parameter is not NULL, the CA MUST find the row in the set that has the same value in the ExtensionName as the value of the *pwszLast* parameter (case-insensitive comparison). If the row is not found, the CA MUST fail. The error code SHOULD be ERROR_INVALID_PARAMETER. If the row is found, the CA MUST remove the rows prior to this row (including this row) from the set of rows to return to the client.

> **Note** The computed set of rows from this step is referred to as the resultant set in steps 5 and 7.

5. The value of the *celt* parameter is referred to as RequestedRows. If RequestedRows is smaller than the number of the rows in the resultant set, the CA MUST return in step 7 only the first RequestedRows rows in the sorted resultant set, and remove the rest of the rows from the resultant set.

6. The value of the **pceltFetched* parameter MUST be set to the number of the rows in the resultant rows set returned in step 7.

7. The CA MUST return the resultant set in the *pctbOut* parameter. The format and marshaling for the value of *pctbOut* is described in **CERTTRANSDBATTRIBUTE** and **CERTTRANSDBEXTENSION**.

3.1.4.1.12   ICertAdminD::OpenView (Opnum 14)

The **OpenView** method opens a view into the database and returns a set of resultant row data.

```
HRESULT OpenView(
   [in, string, unique] wchar_t const* pwszAuthority,
   [in] DWORD ccvr,
   [in, size_is(ccvr)] CERTVIEWRESTRICTION const* acvr,
   [in] DWORD ccolOut,
   [in, size_is(ccolOut)] DWORD const* acolOut,
   [in] DWORD ielt,
   [in] DWORD celt,
   [out] DWORD* pceltFetched,
   [out, ref] CERTTRANSBLOB* pctbResultRows
);
```

pwszAuthority: See the definition of the *pwszAuthority* parameter in section 3.1.4.1.1.

ccvr: An unsigned integer value that specifies the count of a **CERTVIEWRESTRICTION** structure array that is pointed to by the *acvr* parameter.

acvr: A pointer to an array of **CERTVIEWRESTRICTION** structures. For more information, see section 2.2.1.3.

ccolOut: An unsigned integer value that specifies the count of a DWORD array that is pointed to by the *acolOut* parameter.

acolOut: A pointer to an array of DWORDs. Each DWORD value specifies the column identifier for the resultant set of rows.

ielt: An unsigned integer value that specifies the identifier of the first row to return from the resultant set of rows.

celt: An unsigned integer value that specifies the requested count of the row data to be returned from the resultant set of rows.

pceltFetched: A pointer to an unsigned integer value that receives the actual count of row data that is returned by the server in the *pctbResultRows* parameter.

pctbResultRows: A pointer to a **CERTTRANSBLOB** structure. The *pb* byte array of the CERTTRANSBLOB structure MUST contain (on successful return) an array of *n* marshaled **CERTTRANSDBRESULTROW** structures (section 2.2.3), where *n* is the value returned in *pceltFetched*. Each **CERTTRANSDBRESULTROW** contains one or more **CERTTRANSDBRESULTCOLUMN** structures (section 2.2.1.10).

The **OpenView** method opens a view into the database and returns a set of resultant row data.

The CA server MUST enforce the following sequencing rules:

1. If the **OpenView** method is called when the value of Config_Database_View_Open is **False**, the server MUST set Config_Database_View_Open to **True** and continue processing by using the following rules.

2. If the **OpenView** method is called when the value of Config_Database_View_Open is **True**, the server MUST return an error. The error code SHOULD be ERROR_UNEXPECTED_ERROR.

The CA server MUST apply the following processing rules:

1. The CA server MUST ensure that sortOrder is specified in only one of the restrictions that are specified in the *acvr* parameter. If more than one column specifies sort order, the server MUST reject the request by using the error ERROR_INVALID_PARAMETER.

2. The CA server MUST also ensure that all the column identifiers that are specified in the restrictions and the *acolOut* parameter are valid and associated with only one database table. The table MUST be either the Request table (section 3.1.1.1) or the CRL table (section 3.1.1.4). If the table is not the Request table or the CRL table, the CA MUST return ERROR_INVALID_PARAMETER.

3. The CA server MUST compute the resultant set of rows to return, after applying the restrictions on the row set that is associated with the table and sorting the resultant rows based on the restriction information. If no sort ordering is provided in the restriction set, the sorting MUST be done based on the primary index column of the table. If no restriction set is given, the resultant set of rows to return is the entire row set that is associated with the table.

   Each restriction MUST be processed in the following manner:

   1. Based on the value type of the column, which is identified by the *columnIndex* parameter, the pbValue MUST be decoded. The value MUST be in the format and encoded as specified in section 2.2.1.3; otherwise, the CA server MUST fail with the error ERROR_INVALID_PARAMETER.

2. Based on the seek operator that is specified in the restriction, the value for the associated column in each row of the resultant set MUST satisfy the rules that follow.

3. For each row in the resultant set (after sorting), only the columns that are identified by column identifiers in the acolOut array MUST be retained. The rest of the columns MUST be removed from the resultant set.

4. See section 3.1.4.1.13 for the processing of the *ielt* and *celt* parameters and the data returned via the *pceltFetched* and *pctbResultRows* out parameters.

5. The value for the associated column in each row of the resultant set MUST be compared, based on the seek operator, to the value specified in the restriction. If the comparison fails, the row MUST be removed from the resultant set. For column type 3 (BINARY), the seek operator MUST be 0x00000001; otherwise, the call MUST fail with error ERROR_INVALID_PARAMETER.

3.1.4.1.13 ICertAdminD::EnumView (Opnum 15)

The **EnumView** method returns a set of resultant row data for the opened view.

```
HRESULT EnumView(
   [in, string, unique] wchar_t const* pwszAuthority,
   [in] DWORD ielt,
   [in] DWORD celt,
   [out] DWORD* pceltFetched,
   [out, ref] CERTTRANSBLOB*& pctbResultRows
);
```

pwszAuthority: See the definition of the *pwszAuthority* parameter in section 3.1.4.1.1.

ielt: An unsigned integer value that specifies the identifier of the first row to return from the resultant set of rows.

celt: An unsigned integer value that specifies the requested count of the row data to be returned from the resultant set of rows.

pceltFetched: A pointer to an unsigned integer value that receives the actual count of row data that is returned by the server in the *pctbResultRows* parameter.

pctbResultRows: A pointer to a **CERTTRANSBLOB** structure. The *pb* byte array of the **CERTTRANSBLOB** structure MUST contain (on successful return) an array of *n* marshaled **CERTTRANSDBRESULTROW** structures (section 2.2.3), where *n* is the value returned in *pceltFetched*. Each **CERTTRANSDBRESULTROW** contains one or more **CERTTRANSDBRESULTCOLUMN** structures (section 2.2.1.10). In addition, an extra **CERTTRANSDBRESULTROW** structure is included in the array when the server encounters the end of the enumeration, as described in the following rules.

The CA server MUST enforce the following sequencing rules:

1. If the **EnumView** method is called when the value of Config_Database_View_Open is **False**, the server MUST return an error. The error code SHOULD be ERROR_INVALID_HANDLE.

2. If the **EnumView** method is called when the value of Config_Database_View_Open is **True**, the server MUST continue with the following processing rules.

3. The CA server MUST use the resultant set of rows as obtained via the **OpenView** method call.

4. The CA server MUST use the value of *ielt* as an **index** to this resultant set of rows.

5. The number of resultant rows returned MUST be a minimum of the *celt* parameter value and the remaining number of rows in the set (starting from *ielt*). The value of the **pceltFetched* parameter MUST be set to the number of resultant rows returned.

6. When the server encounters the end of the enumeration, the server MUST add an extra **CERTTRANSDBRESULTROW** structure at the end of the array.

   ▪ The count returned in *pceltFetched* MUST NOT include the extra **CERTTRANSDBRESULTROW** structure. The **RowId** field for the extra **CERTTRANSDBRESULTROW** structure MUST be set to the total row count, and the **ccol** field MUST be set to its bitwise inverse.

7. The rules for marshaling a **CERTTRANSDBRESULTROW** into a CERTTRANSBLOB are described in section 2.2.3.1. The rules for marshaling a **CERTTRANSDBRESULTCOLUMN** into a **CERTTRANSBLOB** are described in section 2.2.1.10.1.

3.1.4.1.14 ICertAdminD::CloseView (Opnum 16)

The **CloseView** method closes a view that was previously opened by using the **OpenView** method call.

```
HRESULT CloseView(
  [in, string, unique] wchar_t const* pwszAuthority
);
```

pwszAuthority: See the *pwszAuthority* definition in section 3.1.4.1.1.

The CA server MUST release the resources associated with storing the resultant set of rows obtained via the **OpenView** method call. If the DCOM connection to a Windows CA on which **OpenView** was called is terminated before a call to **CloseView** is made, DCOM eventually releases the resources, but may not release the resources immediately.

The CA server MUST enforce the following sequencing rules:

1. If the **CloseView** method is called when the value of Config_Database_View_Open is **False**, the server MUST return an error. The error code SHOULD be ERROR_INVALID_HANDLE.

2. If the **CloseView** method is called when the value of Config_Database_View_Open is **True**, the server MUST set the value of Config_Database_View_Open to **False**.

3.1.4.1.15 ICertAdminD::ServerControl (Opnum 17)

The **ServerControl** method is used to force the CA server to unregister the **ICertAdminD** and **ICertAdminD2** interfaces.

```
HRESULT ServerControl(
  [in, string, unique] wchar_t const* pwszAuthority,
  [in] DWORD dwControlFlags,
  [out, ref] CERTTRANSBLOB* pctbOut
);
```

pwszAuthority: See the *pwszAuthority* definition in section 3.1.4.1.1.

dwControlFlags: An unsigned integer value that specifies the control to be sent to the certificate server. It MUST take one of the following values.

Value	Meaning
0x000000001	Request unregister for DCOM interfaces for the certificate server.
0x000000002	Not currently used.
0x000000003	Not currently used.

pctbOut: All fields of this parameter MUST be set to 0 on return.

The following processing rules apply.

The CA MUST check the control flags and MUST take the following action:

1. If the value of the *dwControlFlags* parameter is 1, the CA MUST unregister the **ICertAdminD** and **ICertAdminD2** interfaces.

2. If the value of the *dwControlFlags* parameter is 2 or 3, the CA returns successfully.

3. If the value of the *dwControlFlags* parameter is not 1, 2, or 3, the CA MUST return an error. The error code SHOULD be "0x80070057".

3.1.4.1.16 ICertAdminD::Ping (Opnum 18)

The **Ping** method is used to test whether the certificate server is alive.

```
HRESULT Ping(
   [in, string, unique] wchar_t const* pwszAuthority
);
```

pwszAuthority: See the *pwszAuthority* definition in section 3.1.4.1.1.

Windows formats return values per the definition of HRESULT as specified in [MS-ERREF]. Negative values indicate errors, positive values indicate success. Specific values are as specified in [MS-ERREF].

The **ICertAdminD::Ping** method is as specified in [MS-WCCE] section 3.2.1.4.2.3.

3.1.4.1.17 ICertAdminD::GetServerState (Opnum 19)

The **GetServerState** method is used to validate that the caller has permission to read the CA database.

```
HRESULT GetServerState(
   [in, string, unique] wchar_t const* pwszAuthority,
   [out] DWORD* pdwState
);
```

pwszAuthority: See the *pwszAuthority* definition in section 3.1.4.1.1.

pdwState: A pointer to an unsigned integer value that specifies whether the caller has permission to read from the CA database.

The CA MUST return 1 for *pdwState* if the caller has permission to read from the CA database. Otherwise, the CA MUST return 0.

3.1.4.1.18 ICertAdminD::BackupPrepare (Opnum 20)

The **BackupPrepare** method is used to prepare the database for performing further backup operations, such as **BackupEnd**, **BackupGetAttachmentInformation**, **BackupGetBackupLogs**, **BackupOpenFile**, **BackupReadFile**, **BackupCloseFile**, and **BackupTruncateLogs**.

```
HRESULT BackupPrepare(
   [in, string, unique] wchar_t const* pwszAuthority,
   [in] unsigned long grbitJet,
   [in] unsigned long dwBackupFlags,
   [in] const WCHAR* pwszBackupAnnotation,
   [in] DWORD dwClientIdentifier
);
```

pwszAuthority: See the *pwszAuthority* definition in section 3.1.4.1.1.

grbitJet: An unsigned long value. This MUST be one of the following values.

Value	Meaning
0x00000000	Prepare for full backup of the CA database.
0x00000001	Prepare for an incremental backup as opposed to a full backup. This means that only the log files since the last full or incremental backup will be backed up.

dwBackupFlags: An unsigned long value. MUST be 0. MUST be ignored on receipt.

pwszBackupAnnotation: Not Used. MUST be empty string (L""). MUST be ignored on receipt.

dwClientIdentifier: An unsigned long value. Not used. MUST be 0. MUST be ignored on receipt.

If Config_CA_Interface_Flags contains the value IF_NOREMOTEICERTADMINBACKUP, the server SHOULD return an error.<45>

The CA server MUST enforce the following sequencing rules:<46>

- Before a CA backup can occur, **BackupPrepare** MUST be called by the client to notify the CA that a backup of the CA is about to happen:

  - If **BackupGetAttachmentInformation** is called without a previous call to **BackupPrepare** with the *grbitJet* parameter set to 0 (for a full backup), the CA MUST fail. The error code SHOULD be 0xc8000209.

  - If **BackupPrepare** is not called, the remaining database backup-related APIs (**BackupGetAttachmentInformation**, **BackupGetBackupLogs**, **BackupOpenFile**/**BackupReadFile**, **BackupCloseFile**, and **BackupEnd**) MUST fail.

- If **BackupPrepare** is called again by the same client (before calling **BackupEnd**), the CA MUST fail.

- If **BackupReadFile** is called before **BackupOpenFile**, the CA MUST fail.

- If **BackupCloseFile** is called before **BackupOpenFile**, the CA MUST fail.

- If **BackupOpenFile** is called again (before calling **BackupCloseFile**), the CA MUST fail.

- **BackupReadFile** MUST be called after **BackupOpenFile** and before **BackupCloseFile**; otherwise, the CA MUST fail.

- **BackupEnd** MUST be the final API for a backup session.

- If **BackupTruncateLogs** is called before all the log files returned by **BackupGetBackupLogs** and all of the database files returned by **BackupGetAttachmentInformation** are backed up, the CA MUST fail.

If the preceding sequencing rules are not met, the server MUST return ERROR_UNEXPECTED_ERROR.

The CA server MUST take into account the *grbitJet* value to account for an incremental backup versus a full backup. If a full backup has not taken place, the CA MUST return failure if the method is invoked for an incremental backup (*grbitJet* parameter value 1).

3.1.4.1.19 ICertAdminD::BackupEnd (Opnum 21)

The **BackupEnd** method completes the backup process that is started via a call to **ICertAdminD::BackupPrepare**.

```
HRESULT BackupEnd();
```

This method has no parameters.

If Config_CA_Interface_Flags contains the value IF_NOREMOTEICERTADMINBACKUP, the server SHOULD return an error.<47>

The CA server MUST enforce the sequencing rules for **BackupEnd** as specified in section 3.1.4.1.18.<48>

3.1.4.1.20 ICertAdminD::BackupGetAttachmentInformation (Opnum 22)

The **BackupGetAttachmentInformation** method is used to query the CA for the names of database files that should become part of the backup file set.

```
HRESULT BackupGetAttachmentInformation(
  [out, size_is(, *pcwcDBFiles)] WCHAR** ppwszzDBFiles,
  [out] LONG* pcwcDBFiles
);
```

ppwszzDBFiles: A pointer to a WCHAR pointer that will receive the list of null-terminated database file names. Detailed information on database file name structure formatting is specified in section 2.2.4.

pcwcDBFiles: A pointer to an integer value that contains the total length, in characters, of all strings (including a NULL-terminator character) returned in *ppwszzDBFiles*.

The CA server MUST enforce the sequencing rules for **BackupGetAttachmentInformation** as specified in section 3.1.4.1.18.<49>

The CA server MUST apply the following processing rules:

- If Config_CA_Interface_Flags contains the value IF_NOREMOTEICERTADMINBACKUP, the server SHOULD return an error. <50>

- The CA server MUST return a list of file names associated with the CA database data files required for backup. The files corresponding to the names that are returned MUST be accessible to the CA. Each file name MUST be in UNC format and MUST be prefixed with the "D" character. If there are no database files, the CA MUST set the value of the *pcwcDBFiles* parameter to 0 and return successfully.

3.1.4.1.21   ICertAdminD::BackupGetBackupLogs (Opnum 23)

The **BackupGetBackupLogs** method queries the CA for the names of database transaction log files that should become part of the backup file set.

```
HRESULT BackupGetBackupLogs(
   [out, size_is(, *pcwcLogFiles)]
     WCHAR** ppwszzLogFiles,
   [out] LONG* pcwcLogFiles
);
```

ppwszzLogFiles: A pointer to the WCHAR pointer that receives the list of null-terminated log file names. Detailed information about database file name structure formatting is specified in section 2.2.4.

pcwcLogFiles: A pointer to an integer value that contains the total length, in characters, of all strings (including the NULL terminator character) returned in *ppwszzLogFiles*.

The CA server MUST enforce the sequencing rules for **BackupGetBackupLogs**, as specified in section 3.1.4.1.18.<51>

The CA server MUST apply the following processing rules:

- If Config_CA_Interface_Flags contains the value IF_NOREMOTEICERTADMINBACKUP, the server SHOULD return an error. <52>

- The CA server MUST return a list of file names associated with the databases log files required for backup. The list of files MUST be accessible to the CA. Each file name MUST be in UNC format and MUST be prefixed with an exclamation point "!". If there are no database log files, the CA MUST set 0 as the value of the **pcwcLogFiles* parameter and return successfully.

3.1.4.1.22   ICertAdminD::BackupOpenFile (Opnum 24)

The **BackupOpenFile** method opens a file for backup.

```
HRESULT BackupOpenFile(
   [in, string, unique] wchar_t const* pwszPath,
   [out] unsigned hyper* pliLength
);
```

pwszPath: A null-terminated UNICODE string that specifies the path to the targeted file. The file name MUST be UNC form, for example: "\\server\sharepoint\...path...\filename.ext".

pliLength: A pointer to a signed 64-bit integer that receives the size, in bytes, of the targeted file.

The CA server MUST enforce the sequencing rules for **BackupOpenFile** as specified in section 3.1.4.1.18.<53>

The CA server MUST apply the following processing rules:

▪ If Config_CA_Interface_Flags contains the value IF_NOREMOTEICERTADMINBACKUP, the server SHOULD return an error.<54>

▪ The CA server MUST enforce that FileName is one of the file names (without the prefix) that could be returned via a call to **BackupGetAttachmentInformation** or **BackupGetBackupLogs**.

▪ The CA server MUST enforce that the file corresponding to FileName is accessible to the CA.

▪ Upon successful return, the CA MUST return the size, in bytes, of the file content in the *pliLength* parameter.

3.1.4.1.23 ICertAdminD::BackupReadFile (Opnum 25)

The **BackupReadFile** method reads the database file and loads the contents into the buffer that is provided. The file MUST be initialized by a prior call to **BackupOpenFile**.

```
HRESULT BackupReadFile(
  [ref, out, size_is(cbBuffer)] BYTE* pbBuffer,
  [in] LONG cbBuffer,
  [out] LONG* pcbRead
);
```

pbBuffer: A pointer to the buffer that receives the read data.

cbBuffer: The size, in bytes, of the preceding buffer. This parameter MUST be a multiple of the page size of the operating system.

pcbRead: A pointer to an integer that receives the actual number of bytes read.

The CA server MUST enforce the sequencing rules for the **BackupReadFile**, as specified in section 3.1.4.1.18.<55>

The CA server MUST apply the following processing rules:

▪ If Config_CA_Interface_Flags contains the value IF_NOREMOTEICERTADMINBACKUP, the server SHOULD return an error. <56>

▪ If the **BackupReadFile** is called for the first time after the **BackupOpenFile**, the CA MUST read the content, in bytes, from the start of the file up to, at maximum, the value of the *cbBuffer* parameter.

▪ Upon a subsequent call to the **BackupReadFile**, the CA MUST read the content, starting from a 1-byte offset of the last byte read in the previous call to the **BackupReadFile**.

▪ If the CA has already reached end-of-file, the call MUST succeed with 0 as the value of *pcbRead*; otherwise, *pcbRead* MUST contain the actual number of bytes read from the file.

3.1.4.1.24   ICertAdminD::BackupCloseFile (Opnum 26)

The **BackupCloseFile** method closes the database file that was initialized by a prior call to the **BackupOpenFile**.

```
HRESULT BackupCloseFile();
```

This method has no parameters.

If Config_CA_Interface_Flags contains the value IF_NOREMOTEICERTADMINBACKUP, the server SHOULD return an error.<57>

The CA server MUST enforce the sequencing rules for the **BackupCloseFile**, as specified in section 3.1.4.1.18.

3.1.4.1.25   ICertAdminD::BackupTruncateLogs (Opnum 27)

The **BackupTruncateLogs** method function eliminates redundant records from the log files and reduces the disk storage space that is used by log files.

```
HRESULT BackupTruncateLogs();
```

This method has no parameters.

If Config_CA_Interface_Flags contains the value IF_NOREMOTEICERTADMINBACKUP, the server SHOULD return an error.<58>

The CA server MUST enforce the sequencing rules for **BackupTruncateLogs** as specified in section 3.1.4.1.18.

The CA server MUST remove the redundant records in the log files (records that are present in the database also are defined as redundant), thereby decreasing the disk space used to store the log files.<59>

3.1.4.1.26   ICertAdminD::ImportCertificate (Opnum 28)

The **ImportCertificate** method imports a certificate into the CA database.

```
HRESULT ImportCertificate(
  [in, string, unique] wchar_t const* pwszAuthority,
  [in, ref] CERTTRANSBLOB* pctbCertificate,
  [in] LONG dwFlags,
  [out] LONG* pdwRequestId
);
```

pwszAuthority: See the *pwszAuthority* definition in section 3.1.4.1.1.

pctbCertificate: A **CERTTRANSBLOB** that contains an ASN.1 DER–encoded (as specified in [X660] and [X690]) certificate that is inserted into the CA database.

dwFlags: A **LONG** value that MUST take one of the following values.

Value	Meaning
0	If this value is set, the CA server does not allow certificates that are not issued by it to be imported into its database.
FLAG_ALLOW_IMPORT_FOREIGN 0x00010000	A request to the CA server to allow certificates that are not issued by it to be imported into its database.
ICF_EXISTINGROW 0x00020000	A request to the CA to associate the imported certificates with an existing request row.

pdwRequestId: Returns the request ID for the imported certificate. This is used to refer to the certificate after it is imported into the database.

ImportCertificate imports a certificate into the CA database Request table.

The CA server MUST apply the following processing rules:

1. The CA server MUST enforce that the *pctbCertificate* parameter value represents an ASN.1 DER–encoded certificate (as specified in [X660]). If not, it MUST fail with the error ERROR_INVALID_DATA (0x8007000D).

2. The CA server MUST validate the signature on the certificate by using the public key that is associated with the CA's signing certificates.

3. If the ICF_EXISTINGROW flag is not passed as the value of *dwFlags*, if the signature validation succeeds (at step 2), and if the certificate does not already exist in the Request table (this is checked by searching on the Serial_Number in the certificate in the database), the certificate MUST be added to the Request table as a new row and the CA MUST return the resulting Request_Request_ID to the client. For processing rules for each data element in the Request table, see the ImportCertificate data element in the following table.

4. If the ICF_EXISTINGROW flag is not passed as the value of *dwFlags*, if the signature validation succeeds (at step 2), and if the certificate is already present in the Request table, the CA MUST fail with the error ERROR_OBJECT_EXISTS.

5. If the ICF_EXISTINGROW flag is passed as the value of *dwFlags*, if the signature validation succeeds (at step 2), and if the certificate does not already exist in the Request table (this is checked by searching on the Serial_Number in the certificate in the database):

   ▪ The CA MUST locate an entry in the Request table whose Request_Disposition is "request pending" and whose Request_ID has an entry in the Extensions table with a Subject Key Identifier equal to the SubjectKeyIdentifier extension of the certificate that is passed as *pctbCertificate*.

      1. If the entry is found in the Request Table, the CA MUST update the fields from the information in the certificate. For processing rules for each data element in the Request table, see the ImportCertificate processing rules in the following table. The CA MUST return the Request_Request_ID of the updated Request table row as the *pdwRequestId* parameter.

      2. If the entry is not found in the Request table, the CA MUST fail with the error 0x80092009 (CRYPT_E_NO_MATCH).

6. If the signature validation fails (at step 2) and FLAG_ALLOW_IMPORT_FOREIGN is not passed as a value of *dwFlags*, the CA MUST fail with the error 0x800b0107, according to the ImportCertificate data element processing rules in the following table.

7. If the signature validation fails (at step 2) and FLAG_ALLOW_IMPORT_FOREIGN is passed as value of *dwFlags* and the certificate is already present in the Request table, the CA SHOULD return the resulting Request_Request_ID to the client. For processing rules for each data element in the Request table, see the ImportCertificate processing rules in the following table.<60>

8. If the signature validation fails (at step 2), if FLAG_ALLOW_IMPORT_FOREIGN is passed as the value of *dwFlags*, and if the certificate is not already present in the Request table, the certificate SHOULD be added to the Request table as a new row and the CA SHOULD return the resulting Request_Request_ID to the client. For processing rules for each data element in the Request table, see the ImportCertificate data element in the following table.<61>

The certificate fields and extensions SHOULD be processed and stored in individual Request table fields according to the rules in the following table.

Data type	Maximum size of data	Data element name	Processing rule or x.509 certificate field or extension
0x10001 long indexed	4 bytes	Request_Request_ID	The next sequential number after Request_Request_ID of the last database row.
0x1 long	4 bytes	"Request_Status_Code"	If the import is successful, the CA SHOULD set this value to 0.
0x10001 long indexed	4 bytes	"Request_Disposition"	If the certificate being imported was issued by a foreign CA, set Request_Disposition to "foreign certificate". The Microsoft CA uses a DB_DISP_FOREIGN value that is equal to decimal 12 for "foreign certificate". Otherwise, set to "certificate issued".
0x4 string	8192 bytes	"Request_Disposition_Message"	The CA SHOULD set the value of the Request_Disposition_Message column to any value the implementer considers informative for presentation to a human. The Microsoft CA sets Request_Disposition_Message in this case to "certificate issued".
0x10002 date indexed	8 bytes	"Request_Submitted_When"	The time when the method is invoked.
0x10002 date indexed	8 bytes	"Request_Resolved_When"	The time when the method is invoked.
0x10004 string indexed	2048 bytes	"Request_Requester_Name"	The identity of the caller invoking the method.
0x10004 string	2048 bytes	"Request_Caller_Name"	The identity of the caller invoking the method.

Release: Friday, October 25, 2013

Data type	Maximum size of data	Data element name	Processing rule or x.509 certificate field or extension
indexed			
0x3 bin	4096 bytes	"Request_Raw_Name"	The raw subject information from the certificate.
0x4 string	8192 bytes	"Request_Country"	The country attribute of the certificate Subject.
0x4 string	8192 bytes	"Request_Organization"	The organization attribute of the certificate Subject.
0x4 string	8192 bytes	"Request_Org_Unit"	The organizational-unit attribute of the certificate Subject.
0x4 string	8192 bytes	"Request_Common_Name"	The common name attribute of the certificate Subject.
0x4 string	8192 bytes	"Request_Locality"	The locality attribute of the certificate Subject.
0x4 string	8192 bytes	"Request_State"	The state or province name attribute of the certificate Subject.
0x4 string	8192 bytes	"Request_Title"	The title attribute of the certificate Subject.
0x4 string	8192 bytes	"Request_Given_Name"	The given name attribute of the certificate Subject.
0x4 string	8192 bytes	"Request_Initials"	The initials attribute of the certificate Subject.
0x4 string	8192 bytes	"Request_SurName"	The surname attribute of the certificate Subject.
0x4 string	8192 bytes	"Request_Domain_Component"	The domainComponent attribute of the certificate Subject.
0x4 string	8192 bytes	"Request_EMail"	The [RFC822] Name from Subject Alternative Name.
0x4 string	8192 bytes	"Request_Device_Serial_Number"	The serial number attribute of the certificate Subject.
0x10001 long indexed	4 bytes	"Request_ID"	The next sequential number after Request_Request_ID of the last database row.
0x3 bin	16384 bytes	"Raw_Certificate"	The byte stream that is pointed to in the parameter *pctbCertificate* ->pb member.
0x10004 string indexed	128 bytes	"Certificate_Hash"	The SHA-1 hash over the value of "Raw_Certificate".

Data type	Maximum size of data	Data element name	Processing rule or x.509 certificate field or extension
0x10004 string indexed	254 bytes	"Certificate_Template"	The extnValue of extension with OID 1.3.6.1.4.1.311.20.2.
0x10004 string indexed	128 bytes	"Serial_Number"	The serial number ([RFC3280] section 4.1.2.2) of the certificate.
0x2 date	8 bytes	"Not_Before"	The Validity::notBefore ([RFC3280] section 4.1.2.5) field of the certificate.
0x10002 date indexed	8 bytes	"Not_After"	The Validity::notAfter ([RFC3280] section 4.1.2.5) field of the certificate.
0x4 string	128 bytes	"Subject_Key_Identifier"	The Value of the SubjectKeyIdentifier ([RFC3280] section 4.2.1.2) extension of the certificate.
0x3 bin	4096 bytes	"Raw_Public_Key"	The raw value of the public key that is associated with the certificate. (SubjectPublicKeyInfo->subjectPublicKey).
0x1 long	4 bytes	"Public_Key_Length"	The length, in bits, of the public key that is associated with the certificate (SubjectPublicKeyInfo->subjectPublicKey).
0x4 string	254 bytes	"Public_Key_Algorithm"	The name of the algorithm that is associated with the public key of the certificate (SubjectPublicKeyInfo->algorithm->algorithm).
0x3 bin	4096 bytes	"Raw_Public_Key_Algorithm_Parameters"	The raw value of the parameters that are associated with the public key of the certificate (SubjectPublicKeyInfo->algorithm->parameters).
0x4 string	8192 bytes	"Distinguished_Name"	The Subject ([RFC3280] section 4.1.2.6) field of the certificate.
0x4 string	8192 bytes	"Country"	The country attribute of the certificate Subject.
0x4 string	8192 bytes	"Organization"	The organization attribute of the certificate Subject.
0x4 string	8192 bytes	"OrgUnit"	The organizational-unit attribute of the certificate Subject.
0x10004	8192	"Common_Name"	The common name (CN) attribute

Release: Friday, October 25, 2013

Data type	Maximum size of data	Data element name	Processing rule or x.509 certificate field or extension
string indexed	bytes		of the certificate Subject.
0x4 string	8192 bytes	"Locality"	The locality attribute of the certificate Subject.
0x4 string	8192 bytes	"State"	The state or province name attribute of the certificate Subject.
0x4 string	8192 bytes	"Title"	The title attribute of the certificate Subject.
0x4 string	8192 bytes	"Given_Name"	The given name attribute of the certificate Subject.
0x4 string	8192 bytes	"Initials"	The initials attribute of the certificate Subject.
0x4 string	8192 bytes	"SurName"	The surname attribute of the certificate Subject.
0x4 string	8192 bytes	"Domain_Component"	The DomainComponent attribute of the certificate Subject.
0x4 string	8192 bytes	"EMail"	The [RFC822] Name from Subject Alternative Name.
0x4 string	8192 bytes	"Device_Serial_Number"	The serial number attribute of the certificate Subject.

3.1.4.1.27   ICertAdminD::BackupGetDynamicFiles (Opnum 29)

The **BackupGetDynamicFiles** method retrieves the list of CA dynamic file names that need to be backed up. The dynamic files are those that are not included in the CA database backup and are created dynamically by the CA, for example: CRL files created by the CA. Note that **BackupOpenFile** and **BackupReadFile** cannot be used to open and read the dynamic files whose names are returned by this method. Dynamic files must be backed up by means outside this protocol.

```
HRESULT BackupGetDynamicFiles(
  [out, size_is(, *pcwcFiles)] WCHAR** ppwszzFiles,
  [out] LONG* pcwcFiles
);
```

ppwszzFiles: A pointer to a **WCHAR** pointer that receives the list of null-terminated dynamic file names that are used by a CA.

pcwcFiles: A pointer to the **LONG** value that specifies the number of characters in *ppwszzFiles*.

The CA server MUST apply the following processing rules:

▪ If Config_CA_Interface_Flags contains the value IF_NOREMOTEICERTADMINBACKUP, the server SHOULD return an error.<62>

- The CA server MUST return a list of file names that are not part of the database but which the CA deems necessary to be part of backup. An example of a CA dynamic file is the CRL. The files MUST be accessible to the CA. Each file name MUST be in UNC format. If no files are deemed necessary, the CA MUST set 0 as the value of the *pcwcFiles* parameter and return successfully.

3.1.4.1.28   ICertAdminD::RestoreGetDatabaseLocations (Opnum 30)

The **RestoreGetDatabaseLocation** method retrieves the list of CA server database location names for all the database files being restored.

```
HRESULT RestoreGetDatabaseLocations(
  [out, size_is(, *pcwcPaths)] WCHAR** ppwszzDatabaseLocations,
  [out] LONG* pcwcPaths
);
```

ppwszzDatabaseLocations: A pointer to a **WCHAR** pointer that will receive the list of null-terminated database location names and log directory name. Detailed information about database file name structure formatting is specified in section 2.2.4.

pcwcPaths: A pointer to the **LONG** value that specifies the number of bytes in *ppwszzDatabaseLocations*.

The CA server MUST apply the following processing rules:

- If Config_CA_Interface_Flags contains the value IF_NOREMOTEICERTADMINBACKUP, the server SHOULD return an error.<63>

- The CA server MUST return a list that includes the CA database file name and the locations to which the CA log and system files will be restored. The locations MUST be accessible to the CA. The database file name MUST be in UNC format and MUST be prefixed with "D". The log file location MUST be in UNC format and prefixed with a character whose value is 130. The system file location MUST be in UNC format and prefixed with a character whose value is 131.

3.1.4.2   Processing Rules for ICertAdminD2

The **ICertAdminD2** interface extends the **ICertAdminD** interface described in the preceding section.<64>

The version number for this interface is "1.0". The UUID for this interface is: "7fe0d935-dda6-443f-85d0-1cfb58fe41dd".

Opnum values start with the value subsequent to the last opnum value in the last inherited method. Therefore, opnum for this interface starts with 31.

Methods in RPC Opnum Order

Method	Description
PublishCRLs	The **PublishCRLs** method forces a CA to publish CRLs and delta CRLs. Opnum: 31
GetCAProperty	The **GetCAProperty** method is used to retrieve a given property's value from the CA. Opnum: 32

Method	Description
SetCAProperty	The **SetCAProperty** method is used to set CA properties. Opnum: 33
GetCAPropertyInfo	The **GetCAPropertyInfo** method is used to retrieve information about a property on the CA, such as its type and length. Opnum: 34
EnumViewColumnTable	The **EnumViewColumnTable** method retrieves information about one or more columns from the specified CA database table. Opnum: 35
GetCASecurity	The **GetCASecurity** method is used to retrieve CA server's security descriptor. Opnum: 36
SetCASecurity	The **SetCASecurity** method is used to set CA server's security descriptor. Opnum: 37
Ping2	The **Ping2** method is used to determine whether the CA service is started and responding. Opnum: 38
GetArchivedKey	The **GetArchivedKey** method is used to retrieve an archived private key and the associated certificate. Opnum: 39
GetAuditFilter	The **GetAuditFilter** method retrieves the list of events for which the CA server is currently set to create security audit events, as specified in [CIMC-PP]. Opnum: 40
SetAuditFilter	The **SetAuditFilter** method sets the list of events that the CA server MUST create security audit events, as specified in [CIMC-PP]. Opnum: 41
GetOfficerRights	The **GetOfficerRights** method is used to retrieve the Officer rights, as specified in [CIMC-PP]. Opnum: 42
SetOfficerRights	The **SetOfficerRights** method is used to set the Officer rights. Opnum: 43
GetConfigEntry	The **GetConfigEntry** method retrieves the CA configuration data. Opnum: 44
SetConfigEntry	The **SetConfigEntry** method is used to set the configuration information for the Certificate server. Opnum: 45
ImportKey	The **ImportKey** method adds an encrypted key set to an item in the CA database. Opnum: 46

Method	Description
GetMyRoles	The **GetMyRoles** method retrieves the roles, as specified in [CIMC-PP], assigned to the user who calls the method. Opnum: 47
DeleteRow	The **DeleteRow** method deletes a row or set of rows from a database table. Opnum: 48

All methods MUST NOT throw exceptions.

The CA MUST execute the following processing rules for each invocation of the methods listed below in this section. Then the CA MUST proceed to execute the processing rules listed for each method.

The CA MUST determine the identity of the caller by checking the value of the element **uToken.Sids[uToken.UserIndex]**. The ADM element **uToken** is initialized by retrieving the identity token for the current execution context by calling the abstract interface **GetRpcImpersonationAccessToken()** as specified in [MS-RPCE] section 3.3.3.4.3.1. The SID of the caller is the value of the **uToken.Sids** array element indexed at **uToken.UserIndex**. If the caller cannot be identified, the CA MUST refuse to establish a connection, returning an error.<65>

If Config_CA_Interface_Flags contains the value IF_ENFORCEENCRYPTICERTADMIN and the RPC_C_AUTHN_LEVEL_PKT_PRIVACY authentication level, as defined in [MS-RPCE] section 2.2.1.1.8, is not specified on the RPC connection from the client, the CA MUST refuse to establish a connection with the client by returning an error. In Windows the error is E_ACCESSDENIED (0x80070005).

If Config_CA_Interface_Flags contains the value IF_NOREMOTEICERTADMIN, the CA SHOULD return an error for any of the methods listed in this section.<66>

3.1.4.2.1   ICertAdminD2::PublishCRLs (Opnum 31)

The **PublishCRLs** method instructs a CA to publish CRLs and delta CRLs. This call can either cause the republishing of the current CRLs or cause the CA to create and publish new CRLs.

```
HRESULT PublishCRLs(
  [in, string, unique] wchar_t const* pwszAuthority,
  [in] FILETIME FileTime,
  [in] DWORD Flags
);
```

pwszAuthority: See the definition of the *pwszAuthority* parameter in section 3.1.4.1.1.

FileTime: Contains a 64-bit value that represents the number of 100-nanosecond intervals since January 1, 1601 (UTC). Specifies the nextUpdate value of the CRL, as specified in [RFC3280] section 5.1.2.5, in Greenwich Mean Time.

Flags: An unsigned integer value that specifies the type of CRL to publish and the publishing parameters. This parameter MUST be set to a combination of the following values. *Flags* uses B as the least-significant bit. It uses B, D and F as shown in the following table.

0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5	6	7	8	9	3 0	1
B	D	0	0	F	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Value	Description
B	If 1, the CA MUST publish a base CRL.
D	If 1, the CA MUST publish a delta CRL.
F	If 1, the CA MUST republish the existing CRLs.

The CA server MUST apply the following processing rules:

1. If the F bit is set in *Flags*, the *FileTime* parameter is ignored and the following MUST occur:

   - If the B bit is set in *Flags*, the CA MUST republish the most recent base CRL (the CRL identified by the CRL table row with CRL_Min_Base of 0 and the highest CRL_Number) for each valid CA key (CRL_Name_ID) to the locations that are identified in Config_CA_CDP_Publish_To_Base using the logic in section 3.1.5.2, rules 2 and 3 only.<67>

   - If the D bit is set in *Flags*, the CA MUST publish the most recent delta CRL (the CRL identified by the CRL table row with CRL_Min_Base not equal to 0 and the highest CRL_Number) for each valid CA key (CRL_Name_ID) to the locations that are identified in Config_CA_CDP_Publish_To_Delta using the logic in section 3.1.5.2, rules 2 and 3 only.<68>

   - If neither the B bit nor the D bit is set in *Flags*, the CA MUST return an error. The error SHOULD be ERROR_INVALID_PARAMETER.

2. If the F bit is NOT set in *Flags*, the following SHOULD occur:

   - The CA MUST create a CRL for each valid CA key using the logic in section 3.1.4.1.6, rules 2 through 7. The CRL type is determined as follows:

     - If the B bit is set in *Flags*, the type of CRL that the CA creates for each valid CA key MUST be a new base CRL and, if delta CRLs are enabled, a delta CRL.

     - If the D bit is set in *Flags*, the type of CRL that the CA creates for each valid CA key MUST be a new delta CRL.

     - If neither the B bit nor the D bit is set in *Flags*, the CA MUST return an error. The error SHOULD be ERROR_INVALID_PARAMETER.

3. The CA MUST then publish the CRLs using the logic in section 3.1.4.1.6, rules 8 through 13.

Return value: The method returns the first error code returned from the first CRL write operation that failed or was aborted. If none of the CRL write operations failed, the method returns 0.

3.1.4.2.2   ICertAdminD2::GetCAProperty (Opnum 32)

The **GetCAProperty** method is used to retrieve the value of a specific property from the CA.

```
HRESULT GetCAProperty(
  [in, string, unique] wchar_t const* pwszAuthority,
  [in] LONG PropId,
  [in] LONG PropIndex,
  [in] LONG PropType,
  [out, ref] CERTTRANSBLOB* pctbPropertyValue
);
```

pwszAuthority: See the *pwszAuthority* definition in section 3.1.4.1.1.

PropId: An integer value specifying the property to be returned. The *PropID* value MUST be one of the values in the table labeled PropId in [MS-WCCE] section 3.2.1.4.3.2.

PropIndex: Some of these properties (the ones labeled "indexed" in the table in [MS-WCCE] section 3.2.1.4.3.2) have arrays of values. This parameter MUST be used as the index into such an array. For properties that are not arrays, this parameter MUST be ignored.

PropType: An integer value that specifies the property data type.

Value	Meaning
PROPTYPE_LONG 0x00000001	The property type is a signed long integer or a byte array.
PROPTYPE_BINARY 0x00000003	The property type is binary data.
PROPTYPE_STRING 0x00000004	The property type is a Unicode string.

pctbPropertyValue: If the function succeeds, this method MUST return a **CERTTRANSBLOB** structure that contains the property value. If the function fails, the contents are undefined.

> **Note** The numeric values for the constants listed in this topic are defined in the table for the *PropID* parameter.

The data type of the value returned depends on the value specified in the *PropType* parameter and the property specified in the *PropID* parameter:

- If PROPTYPE_STRING is specified in the *PropType* parameter, pctbPropertyValue MUST be a pointer to a **CERTTRANSBLOB** structure. The **pb** member of the structure points to the little-endian encoded Unicode string. The length, in bytes, of the string MUST be contained in the cb member.

- If PROPTYPE_LONG is specified in the *PropType* parameter, there are two possible return types depending on the *PropID*. The first type is the return of a CAINFO structure (as specified in [MS-WCCE] section 2.2.2.4) and the second type is for the return of a BYTE array:

    - If the value passed in PropId maps to one of the following properties, *pctbPropertyValue* is a pointer to a **CERTTRANSBLOB** structure, and the **pb** member of that structure MUST contain a pointer to a CAINFO structure that contains the values of the properties listed as follows. The marshaling rules for a CAINFO structure in a CERTTRANSBLOB are specified in [MS-WCCE] section 2.2.2.2.5:

        - CR_PROP_CATYPE

Release: Friday, October 25, 2013

- CR_PROP_CASIGCERTCOUNT

- CR_PROP_CAXCHGCERTCOUNT

- CR_PROP_EXITCOUNT

- CR_PROP_CAPROPIDMAX

- CR_PROP_KRACERTUSEDCOUNT

- CR_PROP_ROLESEPARATIONENABLED

- CR_PROP_KRACERTCOUNT

- CR_PROP_ADVANCEDSERVER

- If the value passed in *PropId* maps to one of the following properties, *pctbPropertyValue* is a pointer to a **CERTTRANSBLOB** structure, and the **pb** member of the structure points to a byte array containing the value for the requested property. The marshaling rules for each property are specified in the subsection of [MS-WCCE] section 3.2.1.4.3.2 that corresponds to the property name. The **cb** member contains the length of the byte array:

  - CR_PROP_CACERTSTATE

  - CR_PROP_CRLSTATE

  - CR_PROP_KRACERTSTATE

  - CR_PROP_BASECRLPUBLISHSTATE

  - CR_PROP_DELTACRLPUBLISHSTATE

  - CR_PROP_CACERTSTATUSCODE

  - CR_PROP_CAFORWARDCROSSCERTSTATE

  - CR_PROP_CABACKWARDCROSSCERTSTATE

- If PROPTYPE_BINARY is specified in the *PropType* parameter, *pctbPropertyValue* MUST be a pointer to a **CERTTRANSBLOB** structure. The pb member of the structure points to the requested binary large object (BLOB).

  Based on the property identifier passed in *PropId*, the binary data pointed to by the *pb* member MUST be populated as follows:

  - CR_PROP_CASIGCERT: MUST be an X.509 certificate encoded using DER, as specified in [X660].

  - CR_PROP_BASECRL: MUST be a X.509 CRL encoded using DER, as specified in [X660].

  - CR_PROP_CAFORWARDCROSSCERT: MUST be a X.509 certificate encoded using DER, as specified in [X660].

  - CR_PROP_CABACKWARDCROSSCERT: MUST be a X.509 certificate encoded using DER, as specified in [X660].

- CR_PROP_CAXCHGCERT: MUST be a X.509 certificate encoded using DER, as specified in [X660].

The CA MUST execute the processing rules specified in [MS-WCCE] section 3.2.1.4.3.2.15, "PropID = 0x0000000F (CR_PROP_CAXCHGCERT) "CA Exchange Certificate"".

- CR_PROP_CAXCHGCERTCHAIN: MUST be a CMS message, as specified in [RFC2797]encoded using DER, as specified in [X660].

The CA MUST execute the processing rules specified in [MS-WCCE] section 3.2.1.4.3.2.16, "PropID = 0x00000010 (CR_PROP_CAXCHGCERTCHAIN) "CA Exchange Certificate Chain"".

- CR_PROP_CASIGCERTCHAIN: MUST be a CMS message [RFC2797] encoded using DER. [X660].

- CR_PROP_CASIGCERTCRLCHAIN: MUST be a CMS message, as specified in [RFC2797], encoded using DER, as specified in [X660].

- CR_PROP_CASIGCERTCRLCHAIN: MUST be a CMS message, as specified in [RFC2797], encoded using DER, as specified in [X660].

- CR_PROP_CAXCHGCERTCRLCHAIN: CR_PROP_CASIGCERTCRLCHAIN: MUST be a CMS message, as specified in [RFC2797], encoded using DER, as specified in [X660].

The CA MUST execute the processing rules specified in [MS-WCCE] section 3.2.1.4.3.2.33, "PropID = 0x00000021 (CR_PROP_CAXCHGCERTCRLCHAIN) "CA Exchange Certificate Chain and CRL"".

- CR_PROP_DELTACRL: MUST be a X.509 CRL encoded using DER [X660].

- CR_PROP_KRACERT: MUST be a X.509 CRL encoded using DER, as specified in [X660].

The marshaling rules for each of the preceding properties into a CERTTRANSBLOB are specified in [MS-WCCE] sections 2.2.2.2.2 (for X.509 certificate), 2.2.2.2.3 (X.509 CRL), and 2.2.2.2.4 (CMS message).

3.1.4.2.3 ICertAdminD2::SetCAProperty (Opnum 33)

The **SetCAProperty** method is used to set CA properties.

```
HRESULT SetCAProperty(
   [in, string, unique] wchar_t const* pwszAuthority,
   [in] LONG PropId,
   [in] LONG PropIndex,
   [in] LONG PropType,
   [in] CERTTRANSBLOB* pctbPropertyValue
);
```

pwszAuthority: See *pwszAuthority* definition in section 3.1.4.1.1.

PropId: A **LONG** value that specifies one and exactly one of the following property identifiers. The use of PropIds, is as specified in [MS-WCCE] section 3.2.1.4.3.2.

Value	Meaning
0x0000001a	A binary object that contains the CA's key recovery agent (KRA) certificate to be added at the index specified by *PropIndex* parameter.
0x00000019	The maximum number of KRA certificates available on the CA.
0x00000018	The minimum number of KRAs to use when archiving a private key. For more information on KRA usage, see [MSFT-ARCHIVE].
0x0000001d	A collection of name and OID pairs that identify the templates supported by a CA.

PropIndex: A **LONG** value for the index of the KRA certificate to set when the provided PropId is 0x1a. For other *PropId* values, it MUST be 0.

PropType: A **LONG** value that specifies the type of the property. This parameter MUST be one of the following values.

Value	Meaning
PROPTYPE_LONG 0x00000001	Signed LONG data
PROPTYPE_BINARY 0x00000003	Binary data
PROPTYPE_STRING 0x00000004	Unicode String data

pctbPropertyValue: A pointer to **CERTTRANSBLOB** that specifies the new property value. The format for the value contained in **CERTTRANSBLOB** is specific to the propId defined as follows.

Value of PropID	Format for values in CERTTRANSBLOB
CR_PROP_KRACERTUSEDCOUNT 0x00000018	The **pb** member of **CERTTRANSBLOB** MUST point to an unsigned integer value (little-endian format) and the **cb** member of **CERTTRANSBLOB** MUST contain the length of the bytes containing the value.
CR_PROP_KRACERTCOUNT 0x00000019	The **pb** member of **CERTTRANSBLOB** MUST point to an unsigned integer value (little-endian format) and the **cb** member of **CERTTRANSBLOB** MUST contain the length of the bytes containing the value.
CR_PROP_KRACERT 0x0000001a	The **pb** member of **CERTTRANSBLOB** MUST point to an ASN.1 DER (as specified in [ITUX690]) encoded byte array of **Certificate**. The **cb** member of **CERTTRANSBLOB** MUST contain the length of the array.
CR_PROP_TEMPLATES 0x0000001d	As specified in [MS-WCCE] section 3.2.1.4.3.2.29.

The following table defines the values that MUST be set for PropIndex and PropType for each one of the property values passed via *PropID*.

PropID value	PropIndex MUST be	PropType MUST be
0x0000001a	The minimum index is 0.	0x00000003
0x00000019	0x00000000	0x00000001
0x00000018	0x00000000	0x00000001
0x0000001d	0x00000000	0x00000004

When processing the **SetCAProperty** method, the server determines its behavior based on the requested property ID (*PropID* parameter). All valid property IDs are listed in the preceding table. The CA MUST return the error value ERROR_INVALID_PARAMETER if any of the following conditions are met:

▪ The value of *PropID* is not listed in the preceding table, or

▪ For a given *PropID* value the *PropIndex* value does not match the required values defined in the preceding table, or

▪ For a given *PropID* value the *PropType* value does not match the required values defined in the preceding table.

The CA server MUST use the property values to modify the data (as specified in Abstract Data Model in [MS-WCCE] section 3.2.1.1) maintained by CA as part of the configuration.

The CA server MUST apply the following processing rules:

1. The value of CR_PROP_KRACERTUSEDCOUNT MUST be between 1 and the current configured value of CR_PROP_KRACERTCOUNT property. The initial value for CR_PROP_KRACERTCOUNT property MUST be 0.

2. The value to which the CR_PROP_KRACERTCOUNT property is being set MUST be less than the currently configured value of CR_PROP_KRACERTCOUNT.

3. If the *PropIndex* is greater than or equal to the property CR_PROP_KRACERTCOUNT, then the CA must increase the value of CR_PROP_KRACERTCOUNT to the value of *PropIndex* plus 1 each time SetCAProperty (CR_PROP_KRACERT) is called. Else, if the *PropIndex* is less than CR_PROP_KRACERTCOUNT, then the value of CR_PROP_KRACERTCOUNT is not changed.

4. When SetCAProperty(CR_PROP_TEMPLATES) is called, the CA MUST apply the following processing rules:

    1. If the pctbPropertyValue, pb field doesn't have at least two separators identified by '\n', the CA MUST fail the request. The error code SHOULD be E_INVALIDARG (0x80070057), as specified in [MS-ERREF] section 2.1. Otherwise, the CA MUST continue with the following processing rules.

    2. The pctbPropertyValue, pb field contains the following string:"TemplateName1\nTemplateOID1\nTemplateName2\nTemplateOID2\... ", where

        ▪ TemplateName1 is one of the values of the **cn** attribute of the certificate template object that is stored in the Certificate_Template column.

        ▪ TemplateOID1 is the value of the msPKI-Template-Cert-Template-OID attribute of the certificate template stored in the Certificate_Template column.

Note The CA MUST ignore the TemplateOID values specified above.

3. The enrollment services container identified in [MS-WCCE] 2.2.2.11.2 is modified with the following attribute:

   ▪ **certificateTemplates** attribute: The CA MUST set this attribute with the list of template names specified in step 4.2 above.

3.1.4.2.4 ICertAdminD2::GetCAPropertyInfo (Opnum 34)

The **GetCAPropertyInfo** method is used to retrieve information about a property on the CA, such as its type and length.

```
HRESULT GetCAPropertyInfo(
   [in, string, unique] wchar_t const* pwszAuthority,
   [out] LONG* pcProperty,
   [out, ref] CERTTRANSBLOB* pctbPropInfo
);
```

pwszAuthority: See the *pwszAuthority* definition in **ICertAdminD::SetExtension**.

pcProperty: An integer value containing the number of property structures returned.

pctbPropInfo: A **CERTTRANSBLOB** structure containing zero or more **CATRANSPROP** structures. For more information on **CERTTRANSBLOB** and **CATRANSPROP** structures, see Common Structures.

The processing of the **ICertAdminD2::GetCAPropertyInfo** method is the same as that specified in [MS-WCCE] section 3.2.1.4.3.3.

3.1.4.2.5 ICertAdminD2::EnumViewColumnTable (Opnum 35)

The **EnumViewColumnTable** method retrieves information about one or more columns from the specified CA database table.

```
HRESULT EnumViewColumnTable(
   [in, string, unique] wchar_t const* pwszAuthority,
   [in] DWORD iTable,
   [in] DWORD iColumn,
   [in] DWORD cColumn,
   [out] DWORD* pcColumn,
   [out, ref] CERTTRANSBLOB* pctbColumnInfo
);
```

pwszAuthority: See the definition of the *pwszAuthority* parameter in section 3.1.4.1.1.

iTable: An unsigned integer that specifies the database table to be used for the enumeration. This MUST be set from the following values.

Value	Meaning
0x00000000	Request table
0x00003000	Extension table

Value	Meaning
0x00004000	Attribute table
0x00005000	CRL table

iColumn: An unsigned integer that specifies the column number with which to begin the enumeration. Valid values are from 0 to one less than the maximum number of columns for the table.

cColumn: An unsigned integer that specifies the requested number of columns to return.

pcColumn: A pointer to an unsigned integer that receives the actual number of **CERTTRANSDBCOLUMN** structures returned.

pctbColumnInfo: A pointer to a **CERTTRANSBLOB** structure. Upon return, the **pb** member of this structure points to an array of the marshaled **CERTTRANSDBCOLUMN** structures. The format and marshaling for the value of *pctbColumnInfo* MUST be as specified in section 2.2.1.7.

The **EnumViewColumnTable** method returns information to the client about columns that are associated with a specific table. The CA server MUST enforce the following processing rules:

▪ The CA server MUST enforce that the *iTable* parameter has a value as specified in the previous table; otherwise, it MUST fail with the error ERROR_INVALID_PARAMETER.

▪ The CA server MUST enforce that *iColumn* is less than the number of columns associated with the table; otherwise, it MUST fail with the error ERROR_ARITHMETIC_OVERFLOW.

▪ The CA server MUST enforce that *cColumn* is greater than 0; otherwise, it MUST fail with the error ERROR_INVALID_PARAMETER.<69>

▪ The CA server MUST use the value of **iColumn** to identify the column identifier that is associated with the table (identified by the value of the *iTable* parameter).

▪ The number of column information returned MUST be a minimum of the **cColumn** value and the remaining number of columns in the table (starting from **iColumn**). The value of *pcColumn MUST be set to the number of the column information returned.

3.1.4.2.6   ICertAdminD2::GetCASecurity (Opnum 36)

The **GetCASecurity** method is used to retrieve CA security, as defined in Abstract Data Model (section 3.1.1).

```
HRESULT GetCASecurity(
   [in, string, unique] wchar_t const* pwszAuthority,
   [out, ref] CERTTRANSBLOB* pctbSD
);
```

pwszAuthority: See the *pwszAuthority* definition in section 3.1.4.1.1.

pctbSD: A pointer to a **CERTTRANSBLOB** data structure that contains the CA's security descriptor. Security descriptors are specified in [MS-DTYP] section 2.4.6.

3.1.4.2.7 ICertAdminD2::SetCASecurity (Opnum 37)

The **SetCASecurity** method is used to set the CA security, as defined in the Abstract Data Model (section 3.1.1).

```
HRESULT SetCASecurity(
  [in, string, unique] wchar_t const* pwszAuthority,
  [in, ref] CERTTRANSBLOB* pctbSD
);
```

pwszAuthority: See the *pwszAuthority* definition in section 3.1.4.1.1.

pctbSD: A pointer to a **CERTTRANSBLOB** data structure that holds the security descriptor. Security descriptors are specified in [MS-DTYP] section 2.4.6.

The CA SHOULD use the permissions set in **pctbSD** to deny and allow permissions to CA functionality. Microsoft CA permissions are defined in section 3.1.1.7.

3.1.4.2.8 ICertAdminD2::Ping2 (Opnum 38)

The **Ping2** method is used to determine if the CA service is started and responding.

```
HRESULT Ping2(
  [in, string, unique] wchar_t const* pwszAuthority
);
```

pwszAuthority: See the *pwszAuthority* definition in section 3.1.4.1.1.

ICertAdminD2::Ping2 is as specified in [MS-WCCE] section 3.2.1.4.3.4.

3.1.4.2.9 ICertAdminD2::GetArchivedKey (Opnum 39)

The **GetArchivedKey** method is used to retrieve an archived private key and the associated certificate.

```
HRESULT GetArchivedKey(
  [in, string, unique] wchar_t const* pwszAuthority,
  [in] DWORD dwRequestId,
  [out, ref] CERTTRANSBLOB* pctbArchivedKey
);
```

pwszAuthority: See the *pwszAuthority* definition in section 3.1.4.1.1.

dwRequestId: An unsigned integer value that specifies the RequestId of the certificate request for which the archived private key and associated certificate are being requested.

pctbArchivedKey: A pointer to a **CERTTRANSBLOB** structure that MUST contain, on successful response, the archived private key and associated certificate.

ArchivedKey Property Value Processing and Format

The CA server MUST create the ArchivedKey property value by using the following processing rules:

1. The server MUST read the value from the Request_Raw_Archived_Key column of the Request table where the value of the Request_Request_ID column matches the value of the *dwRequestId* parameter. If no such record exists or the value of the Request_Raw_Archived_Key column is empty, the CA MUST return a nonzero error to the client.

2. The server MUST construct a PKCS #7 with the following requirements:

   ▪ ContentInfo: This field MUST have the following values:

     ▪ ContentType: This field MUST be SignedData (1.2.840.113549.1.7.2 PKCS#7 Signed).

     ▪ Content: This field MUST be the value that was read in step 1 from the Request_Raw_Archived_Key column.

   ▪ Certificates: This field MUST include the current CA signing certificate that is used to verify this PKCS#7 message.

   ▪ SignerInfos: The first SignerInfo in the SignerInfos collection MUST use the key that is associated with the current CA signing certificate.

3. The ASN.1 DER–encoded PKCS#7 signed data that was created in step 2 MUST be the value of the ArchivedKey property in the request table (see section 3.1.1).

The **GetArchivedKey** method is used to retrieve the archived private key and issued certificate from the CA's database.

The CA server MUST enforce the following processing rules:

▪ The CA MUST look up the request based on the provided *dwRequestId* parameter in the CA database Request table.

▪ If the request is not found, the CA MUST fail the request with the error CERTSRV_E_PROPERTY_EMPTY.

▪ If the request is found, the CA MUST ensure that the value of the Request.Disposition column in the identified row is "certificate issued".

▪ The CA MUST also ensure that the identified request has the ArchivedKey property value. Otherwise, the CA server MUST fail with the error CERTSRV_E_PROPERTY_EMPTY.

▪ The **pb** field of the *pctbArchivedBlob* parameter MUST reference the value of the ArchivedKey property, and the **cb** field of the *pctbArchivedBlob* parameter MUST contain the length, in bytes, of the ArchivedKey property value.

3.1.4.2.10   ICertAdminD2::GetAuditFilter (Opnum 40)

The **GetAuditFilter** method retrieves the list of events for which the CA server is currently set to create security audit events, as specified in [CIMC-PP].

```
HRESULT GetAuditFilter(
   [in, string, unique] wchar_t const* pwszAuthority,
   [out] DWORD* pdwFilter
);
```

   pwszAuthority: See *pwszAuthority* definition in section 3.1.4.1.1.

pdwFilter: An unsigned integer that specifies the current audit settings. This is a bitwise-OR combination of zero or more of the following values.

Value	Meaning
0x00000001	Audit CA server for the following events: ▪ ServerControl ▪ Registration of the **ICertAdminD** interface. ▪ Unregistration of the **ICertAdminD** interface.
0x00000002	Audit CA server for the following method calls: ▪ BackupPrepare ▪ BackupEnd
0x00000004	Audit CA server for the following method calls: ▪ ICertRequestD::Request ▪ ResubmitRequest ▪ DenyRequest ▪ SetAttributes ▪ SetExtensions ▪ ImportCertificate ▪ DeleteRow
0x00000008	Audit CA server for the following method calls: ▪ RevokeCertificate ▪ PublishCRL ▪ PublishCRLs
0x00000010	Audit CA server for the following method calls: ▪ SetCASecurity ▪ SetOfficerRights
0x00000020	Audit CA server for the following method calls: ▪ GetArchivedKey ▪ ImportKey
0x00000040	Audit CA server for the following method calls:

Value	Meaning
	▪ SetCAProperty ▪ SetConfigEntry

The **GetAuditFilter** method is used to retrieve the audit filter currently in use (initialize to 0 during the registration of the interfaces and can be modified by a call to the **SetAuditFilter** method).

3.1.4.2.11 ICertAdminD2::SetAuditFilter (Opnum 41)

The **SetAuditFilter** method sets the list of events for which the CA server MUST create security audit events, as specified in [CIMC-PP].

```
HRESULT SetAuditFilter(
  [in, string, unique] wchar_t const* pwszAuthority,
  [in] DWORD dwFilter
);
```

pwszAuthority: See the *pwszAuthority* definition in section 3.1.4.1.1.

dwFilter: An unsigned integer that specifies the events to be audited by the CA. For possible values, see section 3.1.4.2.10.

The **SetAuditFilter** method is used to set the audit filter value that is passed in by the client. The audit filter value is used to determine which actions are audited.

The CA server MUST start auditing the methods based on the value of the *dwFilter* parameter. The list of methods for the value is specified in section 3.1.4.2.10.

If Config_CA_Interface_Flags contains the value IF_ENABLEADMINASAUDITOR and the caller does not have administrator permissions (as defined in section 3.1.1.7), the server MUST return an error. In Windows, the error is E_ACCESSDENIED (0x80070005).

3.1.4.2.12 ICertAdminD2::GetOfficerRights (Opnum 42)

The **GetOfficerRights** method is used to retrieve the Officer rights, as specified in [CIMC-PP].

```
HRESULT GetOfficerRights(
  [in, string, unique] wchar_t const* pwszAuthority,
  [out] BOOL* pfEnabled,
  [out, ref] CERTTRANSBLOB* pctbSD
);
```

pwszAuthority: See the *pwszAuthority* definition in section 3.1.4.1.1.

pfEnabled: A pointer to a Boolean value.

pctbSD: A pointer to the **CERTTRANSBLOB** structure that contains the marshaled information specified in section 2.2.1.11.1.

The following processing rules apply:

1. If the CA server does not support Enrollment Agent rights:

1. If no Officer rights are configured, the server MUST set the value of *pfEnabled* to 0, the *pb* member of *pctbSD* to NULL, and the *cb* member to 0.

2. If Officer rights are configured on the CA server, the server MUST set the value of *pfEnabled* to nonzero and return the marshaled data specified in section 2.2.1.11.1 in pctbSD.

2. If the CA server supports Enrollment Agent rights:

1. If no Officer rights are configured (Config_Permissions_Officer_Rights) and no Enrollment Agent rights (Config_Permissions_Enrollment_Agent_Rights) are configured on the CA server, then the server MUST set the value of *pfEnabled* to 0 and the *pb* member of *pctbSD* MUST contain the marshaled data specified in section 2.2.1.11.1.

2. If no Officer rights are configured, but Enrollment Agent rights are configured on the CA server, then the server MUST set the value of *pfEnabled* to 0 and *pctbSD* MUST contain the marshaled data specified in section 2.2.1.11.1.

3. If Officer rights are configured, but no Enrollment Agent rights are configured on the CA server, then the server MUST set the value of *pfEnabled* to nonzero and the pb member of pctbSD MUST contain the marshaled data specified in section 2.2.1.11.1.

4. If Officer rights are configured and Enrollment Agent rights are configured on the CA server, then the server MUST set the value of *pfEnabled* to nonzero and pctbSD MUST contain the marshaled data specified in section 2.2.1.11.1.

3.1.4.2.13   ICertAdminD2::SetOfficerRights (Opnum 43)

The **SetOfficerRights** method is used to set Officer rights or Enrollment Agent rights. Information on **role separation** is specified in [CIMC-PP].

```
HRESULT SetOfficerRights(
  [in, string, unique] wchar_t const* pwszAuthority,
  [in] BOOL fEnable,
  [in, ref] CERTTRANSBLOB* pctbSD
);
```

pwszAuthority: See the *pwszAuthority* definition in section 3.1.4.1.1.

fEnable: A 32-bit **BOOL** parameter composed of two 16-bit fields. Each of these fields may be set to zero or to a nonzero value as follows.

0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5	6	7	8	9	3 0	1
F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

> **F - fRightsEnable:** If bits 0 through 15 are 0, then disable access rights (**officer** or **enrollment agent**) and ignore the value of *pctbSD*.

> **R - RightsType:** If bits 16 through 31 are 0, then the security descriptor in the *pctbSD* parameter is for **officer rights**.

> If bits 16 through 31 are nonzero, the security descriptor in the *pctbSD* is for the **enrollment agents**.

pctbSD: A pointer to the **CERTTRANSBLOB** structure that holds the marshaled security descriptor, as specified in [MS-DTYP] section 2.4.6.

The following processing rules apply:

1. If fRightsEnable is 0 and RightsType is 0, the server MUST disable Officer access rights (Config_Permissions_Officer_Rights) and ignore the value of *pctbSD*.

2. If fRightsEnable is 0 and RightsType is nonzero, the server MUST disable Enrollment Agent access rights (Config_Permissions_Enrollment_Agent_Rights) and ignore the value of *pctbSD*.

3. If fRightsEnable is nonzero and RightsType is 0, the server MUST set the security descriptor specified in *pctbSD* as officer access rights.

4. If fRightsEnable is nonzero and RightsType is nonzero, the server MUST set the security descriptor specified in *pctbSD* as enrollment agent access rights.

3.1.4.2.14   ICertAdminD2::GetConfigEntry (Opnum 44)

The **GetConfigEntry** method retrieves the CAs that persisted the configuration data listed in section 3.1.1.10. Configuration data is represented as a hierarchical data structure with the following format: [\pwszAuthority][\pwszNodePath][\pwszEntry].

```
HRESULT GetConfigEntry(
   [in, string, unique] wchar_t const* pwszAuthority,
   [in, string, unique] wchar_t const* pwszNodePath,
   [in, string, ref] wchar_t const* pwszEntry,
   [out, ref] VARIANT* pVariant
);
```

pwszAuthority: See the *pwszAuthority* definition in section 3.1.4.1.1.

pwszNodePath: A string value that represents the node path for the configuration information. This parameter can be an empty string and MUST NOT be NULL.<70>

pwszEntry: A string value that represents the name of the leaf entry whose information is being retrieved. This value can be an EMPTY string and MUST NOT be NULL.<71>

pVariant: A pointer to a VARIANT that receives the requested information.

The Windows CA uses these datatypes to set the data that it stores in the registry:

REG_BINARY – The vt member of VARIANT is set to VT_ARRAY|VT_UI1 and the **pArray** member references a single dimension SAFEARRAY the binary data. The number of elements of the SAFEARRAY reference by **pArray** is equal to the length of binary data.

REG_DWORD – The vt member of VARIANT is set to VT_I4 and the lVal member is the registry value.

REG_SZ – The vt member of VARIANT is set to VT_BSTR and the bstrVal member is set to BSTR for Unicode string in the registry value.

REG_MULTI_SZ – The vt member of VARIANT is set to VT_ARRAY|VT_BSTR and the **pArray** member references a single dimension SAFEARRAY. The number of elements of the SAFEARRAY referenced by pArray is equal to the number of the strings in the registry value.

For each string, there is an element in the SAFEARRAY referenced by pArray containing the BSTR for Unicode string value in the registry.

The **GetConfigEntry** method retrieves the CA configuration data or configuration data hierarchy information.

The following processing rules apply:

1. If *pwszAuthority* parameter is EMPTY and *pwszNodePath* parameter is EMPTY and *pwszNodeEntry* is EMPTY, the CA MUST return all available leaf properties' names that exist in the configuration's root node as a VARIANT array.

2. If *pwszAuthority* is EMPTY and *pwszNodePath* is EMPTY and *pwszNodeEntry* is not EMPTY, the CA must return the leaf property value identified by *pwszNodeEntry* that exists under the Configuration root node as a VARIANT.

3. If *pwszAuthority* is EMPTY and *pwszNodePath* is not EMPTY, for any value of *pwszNodeEntry* the CA MUST fail the call with an error code of 0x80070057.

4. If *pwszAuthority* parameter is not EMPTY and *pwszNodePath* is EMPTY and *pwszNodeEntry* is EMPTY, the CA MUST return all available leaf properties' names that exist under the *pwszAuthority* node as a VARIANT array.

5. If *pwszAuthority* parameter is not EMPTY and *pwszNodePath* is EMPTY and *pwszNodeEntry* is not EMPTY, the CA MUST return the leaf property value identified by *pwszNodeEntry* that exists under the *pwszAuthority* node as a VARIANT array.

6. If *pwszAuthority* parameter is not EMPTY and *pwszNodePath* is not EMPTY and *pwszEntry* is EMPTY, the CA MUST return all available leaf properties' names that exist under the *pwszNodePath* node as a VARIANT array.

7. If *pwszAuthority* parameter is not EMPTY and *pwszNodePath* is not EMPTY and *pwszEntry* is not EMPTY, the CA MUST return the leaf property value identified by *pwszNodeEntry* that exists under the *pwszNodePath* node as a VARIANT array.

8. For each input in the left column of the table below, the CA MUST perform the processing rules in the corresponding cell in the right column.

Input Parameters	Processing rule for pVariant
pwszNodePath is EMPTY and pwszEntry is "Security"	The CA MUST return the value of the OnNextRestart_Config_Permissions_CA_Security ADM element as a VARIANT. The vt member of VARIANT MUST be set to VT_ARRAY|VT_UI1 and the pArray member MUST reference a single dimension safearray. The number of elements of the safearray reference by pArray MUST be equal to the length of marshaled Security Descriptor. Security Descriptor is as specified in [MS-DTYP] section 2.4.6.
pwszNodePath is EMPTY and pwszEntry is "SetupStatus"	The CA MUST return the value of the OnNextRestart_Config_Setup_Status ADM element as a VARIANT. The vt member of the VARIANT MUST be set to VT_I4 and the lVal member MUST be either 0 or a bitwise OR of the following values:

Input Parameters	Processing rule for pVariant
	0x00000004 – incomplete installation
	0x00000008 – new cert requested
	0x00000800 – still need to upgrade security from Windows 2000
	0x00001000 – permissions changed while the CA was down and the CA will need to update the directory service when it restarts
pwszNodePath is EMPTY and pwszEntry is "UseDS"	The CA MUST return the value of the OnNextRestart_Config_CA_Use_DS ADM element as a VARIANT.
	The vt member of the VARIANT MUST be set to VT_I4 and the lVal member MUST be set to one of the following values:
	0 – The CA is not using Active Directory for CRL publication
	Any nonzero value: The CA is using AD for CRL publication
pwszNodePath is EMPTY and pwszEntry is "CAType"	The CA MUST return the value of the OnNextRestart_Config_CA_Type ADM element as a VARIANT.
	The vt member of the VARIANT MUST be set to VT_I4 and the lVal member MUST be set to one of the CAType values specified in [MS-WCCE] section 2.2.2.4.
pwszNodePath is EMPTY and pwszEntry is "KRAFlags"	The CA MUST return the value of the OnNextRestart_Config_CA_KRA_Flags ADM element as a VARIANT.
	The vt member of the VARIANT MUST be set to VT_I4 and the lVal member MUST be one the following values:
	0
	0x00000001 – allow foreign certificate key archival
pwszNodePath is EMPTY and pwszEntry is "Version"	The CA MUST return the value of the OnNextRestart_Config_Product_Version ADM element as a VARIANT.
	The vt member of the VARIANT MUST be set to VT_I4 and the lVal member MUST be set to the one of the following values:
	0x00010001 – Server is Windows 2000 Server
	0x00020002 – Server is Windows Server 2003
	0x00030001 – Server is Windows Server 2008
	0x00040001 – Server is Windows Server 2008 R2
	0x00050001 – Server is Windows Server 2012
	0x00060001 – Server is Windows Server 2012 R2
pwszNodePath is EMPTY and pwszEntry is "CommonName"	The CA MUST return the value of the **OnNextRestart_Config_CA_Common_Name** ADM element as a VARIANT.
	The vt member of the VARIANT MUST be set to VT_BSTR and the bstrVal member MUST be BSTR for the Unicode string value of the common name of the CA.
pwszNodePath is EMPTY and pwszEntry is "InterfaceFlags"	The CA MUST return the value of the **OnNextRestart_Config_CA_Interface_Flags** ADM element as a VARIANT.

Input Parameters	Processing rule for pVariant
	The vt member of the VARIANT MUST be set to VT_I4 and the lVal member MUST be either 0 or a bitwise-OR of the following values:

IF_LOCKICERTREQUEST = 0x1

This value has no effect.

IF_NOREMOTEICERTREQUEST = 0x2

The CA will not issue any certificates or hold pending any requests for remote users.

IF_NOLOCALICERTREQUEST = 0x4

The CA will not issue any certificates or hold pending any requests for local users.

IF_NORPCICERTREQUEST = 0x8

The CA will not issue any certificates or hold pending any requests for callers using the ICertPassage interface.

IF_NOREMOTEICERTADMIN = 0x10 (16)

No access to Certificate Services Remote Administration Protocol methods for remote callers.

IF_NOLOCALICERTADMIN = 0x20 (32)

No access to Certificate Services Remote Administration Protocol methods for local callers.

IF_NOREMOTEICERTADMINBACKUP = 0x40 (64)

The CA restricts access to the backup-related methods of this protocol for remote callers.

IF_NOLOCALICERTADMINBACKUP = 0x80 (128)

The CA restricts access to the backup-related methods of this protocol for local callers.

IF_NOSNAPSHOTBACKUP = 0x100 (256)

The database files cannot be backed up using a mechanism other than the methods of this interface.

IF_ENFORCEENCRYPTICERTREQUEST = 0x200 (512)

RPC_C_AUTHN_LEVEL_PKT_PRIVACY, as defined in [MS-RPCE] section 2.2.1.1.8, should be defined for all RPC connections to the server for certificate-request operations.

IF_ENFORCEENCRYPTICERTADMIN = 0x400 (1024)

RPC_C_AUTHN_LEVEL_PKT_PRIVACY, as defined in [MS-RPCE] section 2.2.1.1.8, should be defined for all RPC connections to the server for certificate administrative operations (the methods defined in this interface).

IF_ENABLEEXITKEYRETRIEVAL = 0x800 (2048)

Enables an exit algorithm to retrieve the Encrypted private-Key Blob.

IF_ENABLEADMINASAUDITOR = 0x1000 (4096)

Only CA administrators can update the CA audit filter settings. |
| pwszNodePath is EMPTY and pwszEntry is "HighSerial" | If the value of the OnNextRestart_Config_High_Serial_String is not empty, the CA MUST return the value of the OnNextRestart_Config_High_Serial_String. Otherwise, the CA MUST return the value of the OnNextRestart_Config_High_Serial_Number. |

Input Parameters	Processing rule for pVariant
pwszEntry is "Provider" and pwszNodePath is "CSP"	The CA MUST return the value of the **OnNextRestart_Config_CSP_Provider** ADM element as a VARIANT. The vt member of the VARIANT MUST be set to VT_BSTR and the bstrVal member MUST be BSTR for the Unicode string value of the provider.
pwszEntry is "ProviderType" and pwszNodePath is "CSP"	The CA MUST return the value of the **OnNextRestart_Config_CSP_ProviderType** ADM element as a VARIANT. The vt member of VARIANT MUST be set to VT_I4 and the lVal member MUST be set to the provider type.
pwszEntry is "HashAlgorithm" and pwszNodePath is "CSP"	The CA MUST return the value of the **OnNextRestart_Config_CSP_Hash_Algorithm** ADM element as a VARIANT. The vt member of the VARIANT MUST be set to VT_I4 and the lVal member MUST be set to the algid (algorithm identifier) that corresponds to the hash algorithm used by the CA.
pwszEntry is "CNGHashAlgorithm" and pwszNodePath is "CSP"	The CA MUST return the value of the **OnNextRestart_Config_CSP_CNG_Hash_Algorithm** ADM element as a VARIANT. The vt member of the VARIANT MUST be set to VT_BSTR and the bstrVal member MUST be BSTR for the Unicode string value of the name of the CNG hash algorithm used by the CA.
pwszNodePath is EMPTY and pwszEntry is "CRLPeriodUnits"	The CA MUST return the numeric value of the **Config_Base_CRL_Validity_Period** ADM element as a VARIANT. The vt member of the VARIANT MUST be set to VT_I4 and the lVal member MUST be set to the CRL publication period value. A value of 0 means CRL publishing is disabled.
pwszNodePath is EMPTY and pwszEntry is "CRLPeriod"	The CA MUST return the value of the units of time with which the Config_Base_CRL_Validity_Period ADM element is counted as a VARIANT. The vt member of the VARIANT MUST be set to VT_BSTR and the bstrVal member MUST be BSTR for the Unicode string value of one of following: Years Months Weeks Days Hours Minutes Seconds
pwszNodePath is EMPTY and pwszEntry is "CRLDeltaPeriodUnits"	The CA MUST return the numeric value of the Config_Delta_CRL_Validity_Period ADM element as a VARIANT. The vt member of the VARIANT MUST be set to VT_I4 and the lVal member MUST be set to the delta CRL publication period.

Input Parameters	Processing rule for pVariant
	A value of 0 for means Delta CRL publishing is disabled.
pwszNodePath is EMPTY and pwszEntry is "CRLDeltaPeriod"	The CA MUST return the value of the units of time with which the Config_Delta_CRL_Validity_Period ADM element is counted as a VARIANT.
	The vt member of the VARIANT MUST be set to VT_BSTR and the bstrVal member MUST be BSTR for the Unicode string value of one of following:
	Years
	Months
	Weeks
	Days
	Hours
	Minutes
	Seconds
pwszNodePath is EMPTY and pwszEntry is "CRLNextPublish"	The CA MUST return the value of the ADM element OnNextRestart_Config_CA_CRL_Next_Publish as a VARIANT.
	The vt member of the VARIANT MUST be set to VT_ARRAY|VT_UI1 and the pArray member MUST reference a single dimension safearray. The number of elements of the safearray reference by pArray SHOULD be equal to 8 bytes. The value of the bytes should contain a 64-bit value that represents the number of 100-nanosecond intervals since January 1, 1601, according to Coordinated Universal Time (UTC) (encoded in little endian).
pwszNodePath is EMPTY and pwszEntry is "CRLDeltaNextPublish"	The CA MUST return the value of the ADM element OnNextRestart_Config_CA_CRL_Delta_Next_Publish as a VARIANT.
	The vt member of the VARIANT MUST be set to VT_ARRAY|VT_UI1 and the pArray member MUST reference a single dimension safearray. The number of elements of the safearray reference by pArray SHOULD be equal to 8 bytes. The value of the bytes should contain a 64-bit value that represents the number of 100-nanosecond intervals since January 1, 1601, according to Coordinated Universal Time (UTC) (encoded in little endian).
pwszNodePath is EMPTY and pwszEntry is "AuditFilter"	The CA MUST return the value of the OnNextRestart_Config_CA_Audit_Filter ADM element as a VARIANT.
	The vt member of VARIANT MUST be set to VT_I4 and the lVal member MUST be either 0 or bitwise OR of the following values.
	0x00000001 – Audit start/stop of the service.
	0x00000002 – Audit operations associated with backup/restore of the CA database.
	0x00000004 – Audit operations associated with certificate issuance.
	0x00000008 – Audit operations associated with certificate revocation.
	0x00000010 – Audit changes to the security settings on the

Input Parameters	Processing rule for pVariant
	Certificate Authority Service. 0x00000020 – Audit operations associated with Key Recovery. 0x00000040 – Audit operations associated with Changes in CA configuration.
pwszEntry is "Active" and pwszNodePath is "PolicyModules"	The CA MUST return the value of the OnNextRestart_Config_CA_Policy_Algorithm_Implementation ADM element as a VARIANT. The vt member of the VARIANT MUST be set to VT_BSTR and the bstrVal member MUST be BSTR for the Unicode string value of the name (progid) of the policy algorithm By default the Microsoft CA uses a policy module called "CertificateAuthority_MicrosoftDefault.Policy" as the policy algorithm. For more information on the policy algorithm, see [MS-WCCE] section 3.2.1.4.2.1.4.4.
pwszEntry is "Active" and pwszNodePath is "ExitModules"	The CA MUST return the value of the OnNextRestart_Config_CA_Exit_Algorithm_Implementation_List ADM element as a VARIANT. The vt member of the VARIANT MUST be set to VT_ARRAY|VT_BSTR and the pArray member MUST reference a single dimension safearray. The number of elements of the safearray referenced by pArray MUST be equal to the number of active exit algorithms on the CA. For each exit algorithm, there MUST be an element in the safearray referenced by pArray containing the BSTR for the Unicode string value of the name (progid) of the exit algorithm. By default, the Microsoft CA uses an exit module called "CertificateAuthority_MicrosoftDefault.Exit" as the default active exit algorithm. For more information on the exit algorithm, see [MS-WCCE] section 3.2.1.4.2.1.4.8.
pwszNodePath is EMPTY and pwszEntry is "CRLPublicationURLs"	The CA MUST use the values of the following ADM elements to create the VARIANT returned: **OnNextRestart_Config_CA_CDP_Publish_To_Base** **OnNextRestart_Config_CA_CDP_Publish_To_Delta** **OnNextRestart_Config_CA_CDP_Include_In_Cert** **OnNextRestart_Config_CA_CDP_Include_In_CRL_Publish_Locations_Extension** **OnNextRestart_Config_CA_CDP_Include_In_CRL_Freshest_CRL_Extension** **OnNextRestart_Config_CA_CDP_Include_In_CRL_IDP_Extension** The vt member of the VARIANT MUST be set to VT_ARRAY	VT_BSTR and the pArray member MUST reference a single dimension safearray. The number of elements of the safearray referenced by pArray MUST be equal to the number of URLs. For each URL, there MUST be an element in the safearray referenced by pArray containing the BSTR for the Unicode string value of the URI. Each URI is of the format "NumericPrefix:URI", where

Input Parameters	Processing rule for pVariant
	NumericPrefix is the decimal value corresponding to the combination of following flags:
	0x00000001 – The CA should publish the CRL to the URI (OnNextRestart_Config_CA_CDP_Publish_To_Base).
	0x00000002 – The URI should be added in the CDP extension of the certificate issued by the CA (OnNextRestart_Config_CA_CDP_Include_In_Cert).
	0x00000004 – The URI should be added in the FreshestCRL extension of the CRL issued by the CA (OnNextRestart_Config_CA_CDP_Include_In_CRL_Freshest_CRL_Extension).
	0x00000008 – The URI should be added in the CDP extension of the CRL issued by the CA (OnNextRestart_Config_CA_CDP_Include_In_CRL_Publish_Locations_Extension).
	0x00000010 – Not Used.
	0x00000040 – The CA should publish the Delta CRL to the URI (OnNextRestart_Config_CA_CDP_Publish_To_Delta).
	0x00000080 – The URI should be added in the IDP extension of the CRL issued by the CA (OnNextRestart_Config_CA_CDP_Include_In_CRL_IDP_Extension).
pwszNodePath is EMPTY and pwszEntry is "CACertPublicationURLs"	The CA MUST use the values of the following ADM elements to create the VARIANT returned:
	OnNextRestart_Config_CA_AIA_Include_In_Cert
	OnNextRestart_Config_CA_CACert_Publish_To
	The vt member of the VARIANT MUST be set to VT_ARRAY | VT_BSTR and the pArray member MUST reference a single dimension safearray.
	The number of elements of the safearray referenced by pArray MUST be equal to the number of URLs. For each URL, there MUST be an element in the safearray referenced by pArray containing the BSTR for the Unicode string value of the URI.
	Each URI is of the format "NumericPrefix:URI", where NumericPrefix is the decimal value corresponding to the combination of following flags:
	0x00000001 – The CA should publish the CA certificate(s) to the URI (OnNextRestart_Config_CA_CACert_Publish_To).
	0x00000002 – The URI should be added in the AIA extension of the certificates issued by the CA (OnNextRestart_Config_CA_AIA_Include_In_Cert).
pwszNodePath is EMPTY and *pwszEntry is* "OCSPURLs"	The CA MUST return the value of the **OnNextRestart_Config_CA_OCSP_Include_In_Cert** ADM element as a VARIANT.
	The vt member of the VARIANT MUST be set to VT_ARRAY | VT_BSTR and the **pArray** member MUST reference a single dimension safearray.
	The number of elements of the safearray referenced by **pArray** MUST be equal to the number of machines running Online Responder Service with the same configuration

Input Parameters	Processing rule for pVariant
	information. For each machine, there MUST be an element in the safearray referenced by **pArray** containing the BSTR for the Unicode string value of the URI of the machine.
pwszNodePath is "PolicyModules\CertificateAuthority_Micro softDefault.Policy" and pwszEntry is "RequestDisposition"	The CA MUST return the value of the **OnNextRestart_Config_CA_Requests_Disposition** as a VARIANT. The vt member of VARIANT MUST be set to VT_I4 and the lVal member MUST be the value of the **OnNextRestart_Config_CA_Requests_Disposition** ADM element. The value of this ADM element determines whether the CA sets all requests to pending, accepts all requests, or denies all requests.

3.1.4.2.15   ICertAdminD2::SetConfigEntry (Opnum 45)

The **SetConfigEntry** method is used to set the CA's persisted configuration data that is listed in section 3.1.1.10.

```
HRESULT SetConfigEntry(
   [in, string, unique] wchar_t const* pwszAuthority,
   [in, string, unique] wchar_t const* pwszNodePath,
   [in, string, ref] wchar_t const* pwszEntry,
   [in, ref] VARIANT* pVariant
);
```

pwszAuthority: See the *pwszAuthority* definition in section 3.1.4.1.1.

pwszNodePath: A string value that represents the node path for the configuration information. This parameter can be an EMPTY string and MUST NOT be NULL.

pwszEntry: A string value that represents the name of the leaf entry whose information is being set. This value can be an EMPTY string and MUST NOT be NULL.

pVariant: A pointer to VARIANT that specifies the information to set. If this value is EMPTY, the indicated entry MUST be deleted.

The following processing rules apply:

1. If all arguments are provided, the CA MUST update the configuration with the value provided.

2. If the configuration value parameter passed in is empty, the indicated configuration entry MUST be deleted.

3. For each input in the left column of the table below, the CA MUST perform the processing rules in the corresponding cell in the right column. Unless otherwise specified below, changes to these ADM elements made through this method require a CA restart to take effect.

Input	Store information as ADM element
pwszNodePath is EMPTY and pwszEntry is "Security"	OnNextRestart_Config_Permissions_CA_Security
pwszNodePath is EMPTY and pwszEntry is	OnNextRestart_Config_Setup_Status

Release: Friday, October 25, 2013

Input	Store information as ADM element
"SetupStatus"	
pwszNodePath is EMPTY and pwszEntry is "UseDS"	OnNextRestart_Config_CA_Use_DS
pwszNodePath is EMPTY and pwszEntry is "CAType"	OnNextRestart_Config_CA_Type
pwszNodePath is EMPTY and pwszEntry is "KRAFlags"	OnNextRestart_Config_CA_KRA_Flags
pwszNodePath is EMPTY and pwszEntry is "Version"	OnNextRestart_Config_Product_Version
pwszNodePath is EMPTY and pwszEntry is "CommonName"	OnNextRestart_Config_CA_Common_Name
pwszNodePath is EMPTY and pwszEntry is "InterfaceFlags"	OnNextRestart_Config_CA_Interface_Flags
pwszEntry is "Provider" and pwszNodePath is "CSP"	OnNextRestart_Config_CSP_Provider
pwszEntry is "ProviderType" and pwszNodePath is "CSP"	OnNextRestart_Config_CSP_ProviderType
pwszEntry is "HashAlgorithm" and pwszNodePath is "CSP"	OnNextRestart_Config_CSP_Hash_Algorithm
pwszEntry is "CNGHashAlgorithm" and pwszNodePath is "CSP"	OnNextRestart_Config_CSP_CNG_Hash_Algorithm
pwszNodePath is EMPTY and pwszEntry is "CRLPeriodUnits"	The numeric value of the Config_Base_CRL_Validity_Period ADM element. Changing the value with this method MUST affect the validity period of published base CRLs (the use of this ADM element in sections 3.1.4.1.6 and 3.1.4.2.1) immediately, without a CA restart.
pwszNodePath is EMPTY and pwszEntry is "CRLPeriod"	The units of time with which the Config_Base_CRL_Validity_Period ADM element is counted. Changing the value with this method MUST affect the validity period of published base CRLs (the use of this ADM element in sections 3.1.4.1.6 and 3.1.4.2.1) immediately, without a CA restart.
pwszNodePath is EMPTY and pwszEntry is "CRLDeltaPeriodUnits"	The numeric value of the Config_Delta_CRL_Validity_Period ADM element. Changing the value with this method MUST affect the validity period of published delta CRLs (the use of this ADM element in sections 3.1.4.1.6 and 3.1.4.2.1) immediately, without a CA restart.
pwszNodePath is EMPTY and pwszEntry is "CRLDeltaPeriod"	The units of time with which the Config_Delta_CRL_Validity_Period ADM element is counted. Changing the value with this method MUST affect the

Input	Store information as ADM element
	validity period of published delta CRLs (the use of this ADM element in sections 3.1.4.1.6 and 3.1.4.2.1) immediately, without a CA restart.
pwszNodePath is EMPTY and pwszEntry is "CRLNextPublish"	OnNextRestart_Config_CA_CRL_Next_Publish
pwszNodePath is EMPTY and pwszEntry is "CRLDeltaNextPublish"	OnNextRestart_Config_CA_CRL_Delta_Next_Publish
pwszNodePath is EMPTY and pwszEntry is "AuditFilter"	OnNextRestart_Config_CA_Audit_Filter
pwszEntry is "Active" and pwszNodePath is "PolicyModules"	OnNextRestart_Config_CA_Policy_Algorithm_Implementation
pwszEntry is "Active" and pwszNodePath is "ExitModules"	OnNextRestart_Config_CA_Exit_Algorithm_Implementation_List
pwszNodePath is EMPTY and pwszEntry is "CRLPublicationURLs"	OnNextRestart_Config_CA_CDP_Publish_To_Base OnNextRestart_Config_CA_CDP_Publish_To_Delta OnNextRestart_Config_CA_CDP_Include_In_Cert OnNextRestart_Config_CA_CDP_Include_In_CRL_Publish_Locations_Extension OnNextRestart_Config_CA_CDP_Include_In_CRL_Freshest_CRL_Extension OnNextRestart_Config_CA_CDP_Include_In_CRL_IDP_Extension
pwszNodePath is EMPTY and pwszEntry is "CACertPublicationURLs"	OnNextRestart_Config_CA_AIA_Include_In_Cert OnNextRestart_Config_CA_CACert_Publish_To
pwszNodePath is EMPTY and pwszEntry is "OCSPURLs"	OnNextRestart_Config_CA_OCSP_Include_In_Cert
pwszNodePath is EMPTY and pwszEntry is "CRLAttemptRepublish"	OnNextRestart_Config_CA_CRL_Attempt_Republish
pwszNodePath is "PolicyModules\CertificateAuthority_MicrosoftDefault.Policy" and pwszEntry is "RequestDisposition"	OnNextRestart_Config_CA_Requests_Disposition

3.1.4.2.16   ICertAdminD2::ImportKey (Opnum 46)

The **ImportKey** method adds an encrypted key set to an item in the CA database.

```
HRESULT ImportKey(
   [in, string, unique] wchar_t const* pwszAuthority,
   [in] DWORD dwRequestId,
   [in, string, unique] wchar_t const* pwszCertHash,
   [in] DWORD dwFlags,
   [in, ref] CERTTRANSBLOB* pctbKey
);
```

pwszAuthority: See the *pwszAuthority* definition in section 3.1.4.1.1.

dwRequestId: An unsigned integer value that represents the certificate request ID in the CA database.

pwszCertHash: A null-terminated Unicode string value that represents the SHA-1 hash of the ASN.1 DER–encoded certificate data (as specified in [X660]) and that is formatted as a hexadecimal string.

dwFlags: An unsigned integer that specifies the optional flags for this method.

Value	Meaning
0x00010000	Overwrite the existing archived key, if present.

pctbKey: A **CERTTRANSBLOB** structure that contains the ASN.1 DER–encoded (as specified in [X660] and [X690]) PKCS#7 message (as specified in [RFC2315]) that contains the private key to be archived. The content of the enveloped PKCS#7 is as specified in [MS-WCCE] section 3.2.1.4.2.1.4.

The following processing rules apply:

1. The CA MUST process the enveloped PKCS#7 in the *pctbKey* parameter as specified in [MS-WCCE] section 3.2.1.4.2.1.4.

2. If the Request ID is 0 or 4294967295 and *pwszCertHash* is null, the CA server MUST fail with the error ERROR_INVALID_PARAMETER.

3. If *pwszCertHash* is non-null, the CA MUST look up the request based on the provided *pwszCertHash* parameter value in the CA Request table by computing the SHA-1 hash of each issued certificate in the Request table and comparing the hexadecimal string form of it with the value that is specified in *pwszCertHash*.

   1. If the request is not found, the CA server MUST fail the request with the error ERROR_INVALID_PARAMETER.

   2. If the request is found, the CA MUST verify that the private key (decrypted in step 1) is cryptographically related to the public key in the certificate. If the keys are not related, the method MUST fail with the error ERROR_INVALID_PARAMETER.

4. If *pwszCertHash* is null, the CA MUST look up the request based on the provided *dwRequestId* parameter in the CA database Request table:

   1. If the request is not found, the method MUST fail with the error CERTSRV_E_PROPERTY_EMPTY.

   2. If the request is found, the CA MUST verify that the private key (decrypted in step 1) is cryptographically related to the public key in the private key. If the keys are not related, the method MUST fail with the error ERROR_INVALID_PARAMETER.

5. If the request is found, has an encrypted private key associated with it, and the value of *dwFlags* is not 0x00010000, the CA MUST fail with the error ERROR_INVALID_PARAMETER.

6. If the request is found, has an encrypted private key associated with it, and the value of *dwFlags* is 0x00010000, the CA MUST replace the encrypted private key (in the request stored in the Request table) with the encrypted private key that is specified in the *pctbKey* parameter.

7. If the request is found and does not have an encrypted private key that is associated with it, the CA MUST replace the encrypted private key (in the request that is stored in the Request table) with the encrypted private key that is specified in the *pctbKey* parameter.

3.1.4.2.17 ICertAdminD2::GetMyRoles (Opnum 47)

The **GetMyRoles** method retrieves the CA roles, as specified in [CIMC-PP], assigned to the user who calls the method.

```
HRESULT GetMyRoles(
   [in, string, unique] wchar_t const* pwszAuthority,
   [out] LONG* pdwRoles
);
```

pwszAuthority: See the *pwszAuthority* definition in section 3.1.4.1.1.

pdwRoles: A bitwise-OR combination of zero or more CA security values based on the CA implementation. Microsoft CA permissions are defined in section 3.1.1.7.

For *pdwRoles*, the server MUST return a signed integer that represents the CA roles assigned to the caller.

3.1.4.2.18 ICertAdminD2::DeleteRow (Opnum 48)

The **DeleteRow** method deletes a row or set of rows from a database table.

```
HRESULT DeleteRow(
   [in, string, unique] wchar_t const* pwszAuthority,
   [in] DWORD dwFlags,
   [in] FILETIME FileTime,
   [in] DWORD dwTable,
   [in] DWORD dwRowId,
   [out, retval] LONG* pcDeleted
);
```

pwszAuthority: See the definition of the *pwszAuthority* parameter in section 3.1.4.1.1.

dwFlags: An unsigned integer value that specifies the type of rows to be deleted. This parameter can be one of the following values.

Value	Meaning
0x00000000	Delete the individual row.
0x00000001	Delete the rows that contain expired certificates.
0x00000002	Delete the rows that contain pending or failed requests.

FileTime: Contains a 64-bit value that represents the number of 100-nanosecond intervals since January 1, 1601 (UTC). The value is used to query for multiple rows to be deleted. It MUST contain all zeros if the *dwRowId* parameter is nonzero.

dwTable: An unsigned integer value that specifies the table in which to delete rows. This parameter can be one of the following values.

Release: Friday, October 25, 2013

Value	Meaning
0x00000000	Delete the Request table rows.
0x00003000	Delete the Extension table rows.
0x00004000	Delete the Attribute table rows.
0x00005000	Delete the CRL table rows.

dwRowId: An unsigned integer value that represents the row identifier in the CA data table. MUST be set to 0 if *FileTime* is nonzero.

pcDeleted: Returns the count of successfully deleted table rows.

The **DeleteRow** method is used to instruct the CA to delete rows from the specified table.

The following processing rules apply:

1. The CA MUST verify that exactly one of *dwRowId* or *FileTime* is zero. If both are zero or if neither is zero, the CA MUST fail the request with error code E_INVALIDARG (0x80070057).

2. The CA MUST verify that *dwTable* is set to one of the values defined for the *dwTable* parameter. If set to any other value, the CA MUST fail the request with error code E_INVALIDARG (0x80070057).

3. If dwTable is set to 0x00000000:

    1. If the *dwFlags* parameter is nonzero and not set to 0X00000001 or 0x00000002, the CA MUST fail the request.

    2. If the *dwRowId* parameter is nonzero:

        1. If *dwRowId* is not a valid **RequestId**, the CA MUST pass the request and return 0 in the *pcDeleted* parameter.

        2. The CA MUST delete the corresponding rows in the Request table and also delete all the associated rows in the Extension table and Attribute table.

    3. If *FileTime* is nonzero:

        ▪ The CA MUST delete all the rows in the Request table that match the following criteria and also delete all associated rows in the Extension table and Attribute table:

            1. If the *dwFlags* parameter is set to 0x00000001:

                ▪ Delete all rows that contain issued and revoked certificates that expire before *FileTime* and do not contain archived private keys in the Request_Raw_Archived_Key datum.

            2. If *dwFlags* is set to 0x00000002:

                ▪ Delete all rows that contain pending and failed requests that were last acted upon before *FileTime* and do not contain archived private keys in the Request_Raw_Archived_Key datum.

4. If *dwTable* is set to 0x00003000:

Release: Friday, October 25, 2013

1. If *dwRowId* is zero, the CA MUST fail the request.

2. If *dwFlags* is nonzero, the CA MUST fail the request.

3. The CA MUST delete the corresponding Extension table row.

5. If dwTable is set to 0x00004000:

    1. If *dwRowId* is zero, the CA MUST fail the request.

    2. If *dwFlags* is nonzero, the CA MUST fail the request.

    3. The CA MUST delete the corresponding Attribute table row.

6. If *dwTable* is set to 0x00005000:

    1. If *dwFlags* is nonzero and not set to 0x00000001, the CA MUST fail the request.

    2. If *dwFlags* is set to 0x00000000 or 0x00000001:

        1. If *dwRowId* is nonzero:

            1. If *dwRowId* is not a valid CRL table CRL_RowId, the CA MUST pass the request and return 0 in the *pcDeleted* parameter.

            2. The CA MUST delete the corresponding CRL table row.

        2. If *FileTime* is nonzero:

            - The CA MUST delete all CRL table rows that contain CRLs for which the value in the CRL_Next_Update column occurs before *FileTime*.

7. The CA MUST count all deleted rows and return that count in *pcDeleted. If the Windows CA fails to delete all rows that match a date restriction as previously specified, it returns an HRESULT value of ERROR_OUT_OF_MEMORY to indicate to the client that more rows matching the criteria may remain.

3.1.5 Timer Events

3.1.5.1 CRL Next Publish Timer Events

When either the Base or Delta CRL Next Publish Timer reaches its next timeout value, the CA SHOULD stop all republishing attempts that may be in progress as a result of the CRL Publication Retry Timer and rebuild and republish CRLs using the processing rules in **ICertAdminD2::PublishCRLs (Opnum 31) (section 3.1.4.2.1)**, with the following input values:

Filetime: This parameter is set to 0.

Flags: In this parameter, the F bit is not set. If the Base CRL Next Publish Timer (section 3.1.2.1.1) has been reached, the B bit is set. If the Delta CRL Next Publish Timer (section 3.1.2.1.2) has been reached, the D bit is set.

3.1.5.2 CRL Publication Retry Timer Events

When the CRL Publication Retry Timer reaches its timeout value, the CA MUST attempt to republish CRLs using the following steps.

1. The CA evaluates the ADM element **OnNextRestart_Config_CA_CRL_Attempt_Republish**, and if its value equals or exceeds 10, the CA does not execute the logic in this section. Rather, CRL creation and publishing will occur again when either the Base CRL Next Publish Timer (section 3.1.2.1.1) or the Delta CRL Next Publish Timer (section 3.1.2.1.2) reaches its next timeout value, whichever occurs first.

2. The CA increments the ADM elements **Config_CA_CRL_Attempt_Republish** and **OnNextRestart_Config_CA_CRL_Attempt_Republish** by 1.

3. The CA examines the CRL table row(s) corresponding to the most recent base and, if enabled, delta CRLs, for all CA keys. For each found CRL table row:

   ▪ The CA reattempts publishing of that CRL to all Config_CA_CDP_Publish_To_Base or Config_CA_CDP_Publish_To_Delta CRL publishing locations as applicable, and, following this attempt, update the following data elements in the corresponding CRL table row:

      ▪ CRL_Publish_Status_Code: Use the same logic as in section 3.1.4.1.6, rule 11.

      ▪ **CRL_Publish_Error**: If this element is present, update it as follows: If the retry is not successful, preserve the existing username and CRL location information in the data element and append username and CRL location information for the current failed retry. If the retry is successful, preserve the username information, append the username information for the current, successful retry, and remove the CRL location information for previous failed publishing attempts.

        Once publishing is successful, the Microsoft CA removes any existing CRL publishing location indices from each line of username information. If republishing is not successful, the Microsoft CA preserves the existing username and CRL location index information, up to and including the line feed after the last line of user information, and adds a new line containing user name followed by a space, two hyphens, and the numeric index of the CRL publishing location to which publication failed upon the current retry. If more than one location fails for one CRL table entry upon retry, then the index of each failed location is appended to this line, separated by spaces.

        The example following illustrates the contents of CRL_Publish_Error on a Microsoft CA after three publishing attempts, the first and second of which failed when writing to an LDAP path, and the third was successful. The first attempt was initiated in the user context of Administrator, and in the second and third attempts in the context of the CA:

        "Published by CORP\Administrator -- 0*{linefeed}*

        - -- 0*{linefeed}*

        -*{linefeed}*

        {linefeed}

        ldap:///CN=CA1,CN=host1,CN=CDP,CN=Public Key Services,CN=Services,CN=Configuration,DC=corp,DC=contoso,DC=com"

      ▪ **CRL_Publish_Attempts**: If this element is present, increment its value by 1.

      ▪ **CRL_Last_Published**: If this element is present, update its value to the current time.

4. If all CRL publishing retry attempts are successful, the CA resets the data elements **Config_CA_CRL_Attempt_Republish** and **OnNextRestart_Config_CA_CRL_Attempt_Republish** back to 0 and set the Base and Delta

CRL Next Publish Timers as follows: set the Base CRL Next Publish Timer to the value of the CRL_Next_Publish of the base CRL that was just published, and set the Delta CRL Next Publish Timer to the value of the CRL_Next_Publish of the delta CRL that was just published. Otherwise, reset the value of the CRL Publication Retry Timer specified in section 3.1.2.2.

3.1.6   Other Local Events

No other local events.

3.2   Client Details

3.2.1   Abstract Data Model

None.

3.2.2   Timers

The Certificate Services Remote Administration Protocol client contains no timers.

3.2.3   Initialization

The Certificate Services Remote Administration Protocol depends on DCOM for authentication, as specified in [MS-DCOM].

3.2.4   Message Processing Events and Sequencing Rules

The Certificate Services Remote Administration Protocol client invokes DCOM methods based on operator requests for the different methods available. The Certificate Services Remote Administration Protocol client invokes the DCOM client to establish a connection to the server as specified in section 2.1, initialize the server DCOM objects and interfaces, and enable the client to make subsequent calls to the methods defined in this document. Windows provides Certificate Services Remote Administration Protocol client functionality through the use of Microsoft Management Console (MMC) and through the use of command-line tools (certutil.exe).

Upon receiving a reply from the server in response to a method call, the client MUST validate the return code. Return codes from all method calls are of type **HRESULT**. If the **HRESULT** indicates success (0), the client may assume that any output parameters are present and valid. For any other return code (**HRESULT** is nonzero), the client MUST assume the method call failed. Unless redefined locally within this document, return codes are as specified in [MS-ERREF].

3.2.4.1   Processing Rules for ICertAdminD

3.2.4.1.1   ICertAdminD::SetExtension (Opnum 3)

No specific processing rules.

3.2.4.1.2   ICertAdminD::SetAttributes (Opnum 4)

No specific processing rules.

3.2.4.1.3   ICertAdminD::ResubmitRequest (Opnum 5)

Upon successful return from the ICertAdminD::ResubmitRequest method invocation, the client receives the *pdwDisposition* parameter as an output value.

If this value is 3, the CA issued the certificate.

If this value is 2, the CA denied the certificate request.

The value of 5 will not be assigned to the *pdwDisposition* parameter if the default **CA policy module** is used. If a custom CA policy module is used, then it is possible to have the *pdwDisposition* parameter set to a value of 5. See [MS-WCCE] sections 3.1.1.4.3.6 and 3.1.1.6.1 for information about how to retrieve pending requests from a CA.

If the value is any other nonzero value, the server has encountered an error.

The client SHOULD NOT rely on any specific value for its processing rules.

3.2.4.1.4   ICertAdminD::DenyRequest (Opnum 6)

No specific processing rules.

3.2.4.1.5   ICertAdminD::IsValidCertificate (Opnum 7)

No specific processing rules.

3.2.4.1.6   ICertAdminD::PublishCRL (Opnum 8)

No specific processing rules.

3.2.4.1.7   ICertAdminD::GetCRL (Opnum 9)

No specific processing rules.

3.2.4.1.8   ICertAdminD::RevokeCertificate (Opnum 10)

No specific processing rules.

### 3.2.4.1.9   ICertAdminD::EnumViewColumn (Opnum 11)

See **ICertAdminD2::EnumViewColumnTable**.

3.2.4.1.10   ICertAdminD::GetViewDefaultColumnSet (Opnum 12)

No specific processing rules.

3.2.4.1.11   ICertAdminD::EnumAttributesOrExtensions (Opnum 13)

No specific processing rules.

3.2.4.1.12   ICertAdminD::OpenView (Opnum 14)

The OpenView method obtains the column values for rows associated with a particular resultant set of rows from a table:

- *pwszAuthority*: See the *pwszAuthority* definition in section 3.1.4.1.1.

- *acvr*: Each structure in the array MUST contain the information for individual restrictions on a specific column to be applied to the associated table data before the resultant data is returned by the CA. The array of restrictions MUST be encoded as specified in **CERTVIEWRESTRICTION**.

▪ *acolOut*: Each DWORD value specifies the column identifier for the output. Column identifiers MUST be obtained by using one of the following methods: EnumViewColumn, EnumViewColumnTable, or GetViewDefaultColumnSet.

If the marshaled data buffer returned in *pctbResultRows* includes space for an extra **CERTTRANSDBRESULTROW** structure in addition to the row count returned in *pcelt, and if the extra **CERTTRANSDBRESULTROW** structure's **RowId** and **ccol** fields are bitwise inverses of each other, then the field can be relied upon by the client as the total number of rows in the currently active view.

The client SHOULD invoke calls to OpenView, EnumView, and CloseView in the proper sequence, defined as the sequence enforced by the server and documented in sequencing rules in sections 3.1.4.1.12, 3.1.4.1.13, and 3.1.4.1.14.

If the sequence is not followed, the server responds as documented in the server processing rules in section 3.1.4.1.12, 3.1.4.1.13, and 3.1.4.1.14.

3.2.4.1.13 ICertAdminD::EnumView (Opnum 15)

See section 3.1.4.1.12 for sequencing rules.

See ICertAdminD::OpenView for client processing rules.

3.2.4.1.14 ICertAdminD::CloseView (Opnum 16)

See section 3.1.4.1.12 for sequencing rules.

3.2.4.1.15 ICertAdminD::ServerControl (Opnum 17)

No specific processing rules.

3.2.4.1.16 ICertAdminD::Ping (Opnum 18)

The ICertAdminD::Ping method is as specified in [MS-WCCE] section 3.1.1.4.5.

3.2.4.1.17 ICertAdminD::GetServerState (Opnum 19)

No specific processing rules.

3.2.4.1.18 ICertAdminD::BackupPrepare (Opnum 20)

Sequencing rules are as follows:

1. Before a certificate services backup can occur, a call to BackupPrepare MUST be made to notify the CA that a backup of the CA is about to happen.

2. The functions BackupGetAttachmentInformation and BackupGetBackupLogs are used after BackupPrepare to retrieve the certificate services list of database file names and database log file names.

3. To open a file for backup purposes, BackupOpenFile MUST be used before BackupReadFile or BackupCloseFile.

4. After opening the file for backup purposes (using BackupOpenFile), BackupReadFile is used to retrieve the contents of the file and call an application-specific routine to write the contents to a backup medium.

5. Before reading another file, BackupCloseFile MUST be called to close the already read file.

6. When the backup session is completed, BackupEnd MUST be invoked.

The client MUST follow the preceding sequencing rules. If BackupPrepare returns a failure, the client MUST NOT make any further method calls related to backup. If the sequencing rules are not met, the server returns ERROR_UNEXPECTED_ERROR.

The client application MUST ensure that a full backup (*grbit* parameter with value 0) has already happened before calling the server for incremental backup (*grbit* parameter with value 1).

3.2.4.1.19   ICertAdminD::BackupEnd (Opnum 21)

The client MUST enforce the sequencing rules as specified in BackupPrepare (section 3.2.4.1.18).

3.2.4.1.20   ICertAdminD::BackupGetAttachmentInformation (Opnum 22)

The client MUST enforce the sequencing rules as described in BackupPrepare (section 3.2.4.1.18). After a call to BackupPrepare, the client MUST call BackupGetAttachmentInformation to obtain the list of database file names.

3.2.4.1.21   ICertAdminD::BackupGetBackupLogs (Opnum 23)

The client MUST enforce the sequencing rules as described in BackupPrepare (section 3.2.4.1.18). After a call to BackupPrepare, the client MUST call BackupGetBackupLogs to obtain the list of database log file names.

3.2.4.1.22   ICertAdminD::BackupOpenFile (Opnum 24)

The client MUST enforce the sequencing rules as described in BackupPrepare (section 3.2.4.1.18). The client MUST call this with the file names obtained from one of the following methods: BackupGetAttachmentInformation, BackupGetBackupLogs, or BackupGetDynamicFiles.

The client MUST remove the prefix "D" and "!" from the file names obtained through the method calls to BackupGetAttachmentInformation or BackupGetBackupLogs.

3.2.4.1.23   ICertAdminD::BackupReadFile (Opnum 25)

The client MUST enforce the sequencing rules as described in BackupPrepare (section 3.2.4.1.18). After opening the file for backup purposes (by using BackupOpenFile), the client MUST call BackupReadFile to retrieve the contents of the file and call an application-specific routine to write the contents to a backup medium. The client can call this API multiple times to read the entire content of the file. Upon successful return, if the value of *pcbRead* is less than *cbRead* or is 0, the client MUST assume that the entire contents of the file has been read and MUST call BackupCloseFile.

3.2.4.1.24   ICertAdminD::BackupCloseFile (Opnum 26)

The client MUST enforce the sequencing rules as described in BackupPrepare (section 3.2.4.1.18). The client MUST call BackupCloseFile for each corresponding BackupOpenFile (and after BackupReadFile is done).

3.2.4.1.25 ICertAdminD::BackupTruncateLogs (Opnum 27)

The client MUST enforce the sequencing rules as described in BackupPrepare (section 3.2.4.1.18). The client may call BackupTruncateLogs to truncate the log files and reduce the disk space occupied by the database files on the CA machine. Microsoft clients call the BackupTruncateLogs method after performing full backup operation.

3.2.4.1.26 ICertAdminD::ImportCertificate (Opnum 28)

No specific processing rules.

3.2.4.1.27 ICertAdminD::BackupGetDynamicFiles (Opnum 29)

The client MUST retrieve a list of file names that are not part of the database but are deemed necessary as part of the backup by the CA. An example of a CA dynamic file is the certificate revocation list (CRL).

3.2.4.1.28 ICertAdminD::RestoreGetDatabaseLocations (Opnum 30)

Before performing restoration of the database, the client MUST retrieve the location where the database files have to be placed. To do so, the client MUST call RestoreGetDatabaseLocations. The location names prefixed with "D" MUST be used to restore the database files. The location names prefixed with "!" MUST be used to restore the database log files.

3.2.4.2 Processing Rules for ICertAdminD2

3.2.4.2.1 ICertAdminD2:: PublishCRLs (Opnum 31)

No specific processing rules.

3.2.4.2.2 ICertAdminD2::GetCAProperty (Opnum 32)

The ICertAdminD2::GetCAProperty method is as specified in [MS-WCCE] section 3.1.1.4.6.

3.2.4.2.3 ICertAdminD2::SetCAProperty (Opnum 33)

No specific processing rules.

3.2.4.2.4 ICertAdminD2::GetCAPropertyInfo (Opnum 34)

The ICertAdminD2::GetCAPropertyInfo method is as specified in [MS-WCCE] section 3.1.1.4.7.

3.2.4.2.5 ICertAdminD2::EnumViewColumnTable (Opnum 35)

No specific processing rules.

3.2.4.2.6 ICertAdminD2::GetCASecurity (Opnum 36)

No specific processing rules.

3.2.4.2.7 ICertAdminD2::SetCASecurity (Opnum 37)

No specific processing rules.

3.2.4.2.8 ICertAdminD2::Ping2 (Opnum 38)

The ICertAdminD2::Ping2 method is as specified in [MS-WCCE] section 3.1.1.4.5.

3.2.4.2.9 ICertAdminD2::GetArchivedKey (Opnum 39)

No specific processing rules.

3.2.4.2.10 ICertAdminD2::GetAuditFilter (Opnum 40)

No specific processing rules.

3.2.4.2.11 ICertAdminD2::SetAuditFilter (Opnum 41)

No specific processing rules.

3.2.4.2.12 ICertAdminD2::GetOfficerRights (Opnum 42)

No specific processing rules.

3.2.4.2.13 ICertAdminD2::SetOfficerRights (Opnum 43)

No specific processing rules.

3.2.4.2.14 ICertAdminD2::GetConfigEntry (Opnum 44)

No specific processing rules.

3.2.4.2.15 ICertAdminD2::SetConfigEntry (Opnum 45)

No specific processing rules.

3.2.4.2.16 ICertAdminD2::ImportKey (Opnum 46)

For the *pctbKey* parameter, the client MUST create the encrypted private key as specified in [MS-WCCE] section 3.1.1.4.3.5.

3.2.4.2.17 ICertAdminD2::GetMyRoles (Opnum 47)

No specific processing rules.

3.2.4.2.18 ICertAdminD2::DeleteRow (Opnum 48)

No specific processing rules.

3.2.5 Timer Events

No client timers.

3.2.6 Other Local Events

None.

4   Protocol Examples
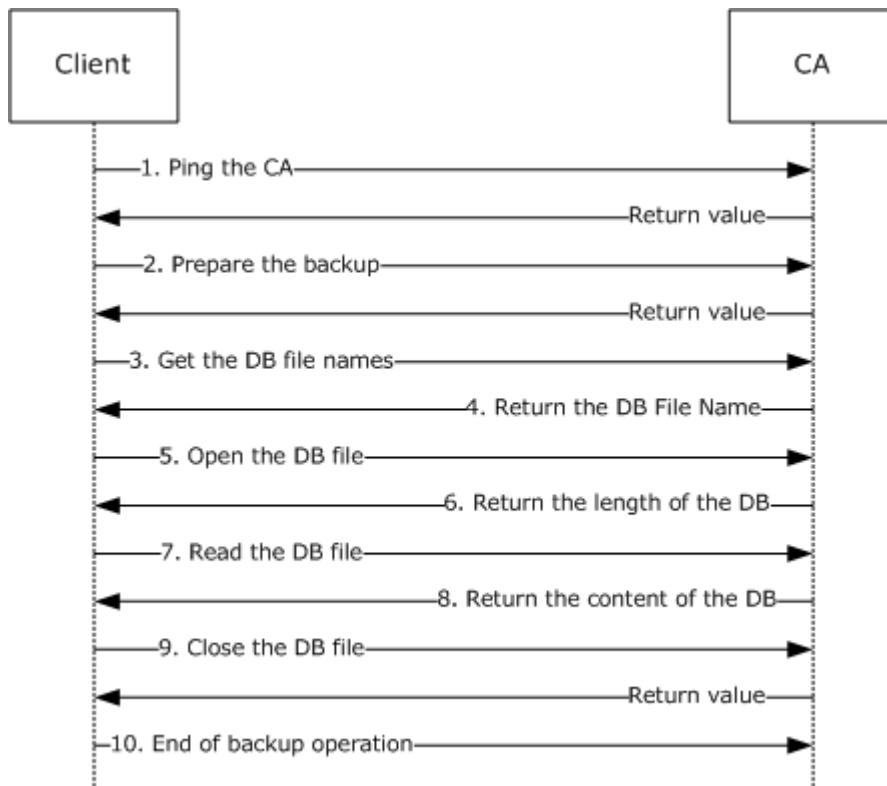
Figure 3: Backing up a CA database

A client determines that it is necessary to back up a CA database. To do so, the client needs to obtain the name of the CA to be backed up.

For this example, the CA name is "**RootCA**".

The following sequence of events needs to occur:

1. The client pings the CA and verifies that the CA service is online. The client calls the method Ping with the following parameters.

   ```
   ICertAdminD::Ping(L"RootCA")
   ```

   The client verifies that the return value from this method is S_OK (0).

2. The client sends a request to the CA to start a backup process. The client calls the BackupPrepare method with the following parameters.

   ```
   ICertAdminD::BackupPrepare( L"RootCA", 0, 0, NULL, 0 )
   ```

   The client verifies that the return value from this method is S_OK (0).

3. The client retrieves the database file names. The client calls the
   BackupGetAttachmentInformation method with the following parameters.

```
ICertAdminD::BackupGetAttachmentInformation
    (&pwszFileList,&cwList)
```

4. The server returns the number of database and log files by setting the out cwList parameter (in
   this example, it is set to 1) and pwszFileList parameter (in this example, it is set to
   L"\\servername\e$\winnt\system32\certlog\rootca.edb").

5. The client opens the specific database file returned in the preceding step. The client needs to
   learn the length of the file. The client calls the BackupOpenFile method with the following
   parameters.

```
ICertAdminD::BackupOpenFile
    (L"\\servername\e$\winnt\system32\certlog\rootca.edb",&Length)
```

6. The server returns the length of the requested file in the *Length* parameter. In this example, the
   length is 123456.

7. The client allocated the required buffer and reads the content of the DB file. The client calls the
   BackupReadFile method with the following parameters.

```
ICertAdminD::BackupReadFile(pbData,123456,&Read)
```

8. The server copies the content of the requested file to the *pbData* parameter and sets the length
   of the actual read operation into the *Read* parameter.

9. The client closes the DB file. The client calls BackupCloseFile with no parameters. The client
   verifies that the return value from this method is S_OK (0).

10.The client ends the backup operation by calling the BackupEnd method with no parameters.

5   Security

The Certificate Services Remote Administration Protocol allows an administrator to manipulate the CA in various ways.

The two most security-sensitive tasks that an administrator can apply to a CA are as follows:

- Manually approve **issuance** of a certificate.

- Recover an archived private key.

The CA has its own security requirements for preventing information tampering and keeping cryptographic keys secret, as specified in [MS-WCCE] section 5. All those requirements apply to an implementation of the CA. In addition, this protocol exposes a risk if the administrator is not authenticated properly, and this section lists only additional requirements about that function.

5.1   Security Considerations for Implementers

5.1.1   Strong Administrator Authentication

An administrator of the CA must authenticate strongly. This task can be accomplished by using a high-entropy password or some multiple-factor authentication method (such as a smart card). The CA administrator should use a login account that functions only for CA administration and not for any other purpose because use of the same credentials on a vulnerable computer while performing some other task exposes them to capture and misuse.

5.1.2   KDC Security

Because authentication of the administrator is by Kerberos, in this protocol, the KDC must itself be kept secure; that is, free from tampering and free from vulnerabilities that would allow privilege-elevation penetrations.

5.1.3   Administrator Console Security

The administrator console includes the applications that the administrator uses to run the client side of this protocol and the operating system in which that functionality runs. The administrator console must be kept secure from penetrations that would allow an attacker to act as the administrator.

5.1.4   Administrator Credential Issuance

The procedures used by a human CA administrator to control access must be kept free from penetration and human error. These procedures include the following:

1. Assign a name to the CA administrator. Kerberos domains assign a name to each CA administrator.

2. Add the name of the new CA administrator to a named group of administrators.

3. Add the named group of administrators to the **ACL** that is used by the CA.

The following list provides a few examples of security risks:

- Penetration: **Social engineering**

- Human error: Misspellings

- Human error: Unwarranted assumptions

5.2 Index of Security Parameters

Security settings for this protocol are described in section 2.1.

6   Appendix A: Full IDL

For ease of implementation, the full IDL is provided, where "ms-dtyp.idl" refers to the IDL found in [MS-DTYP] Appendix A, and "ms-oaut.idl" is the IDL found in [MS-OAUT] Appendix A.

```
import "ms-dtyp.idl";
import "ms-oaut.idl";


typedef byte BYTE;

typedef struct _CERTTRANSBLOB
{
    ULONG                  cb;
    [size_is(cb), unique] BYTE *pb;
} CERTTRANSBLOB;

typedef struct _CATRANSPROP {
  LONG lPropID;
  BYTE propType;
  BYTE Reserved;
  USHORT propFlags;
  ULONG obwszDisplayName;
} CATRANSPROP;


typedef enum _ENUM_CATYPES
{
  ENUM_ENTERPRISE_ROOTCA = 0x00000000,
  ENUM_ENTERPRISE_SUBCA = 0x00000001,
  ENUM_STANDALONE_ROOTCA = 0x00000003,
  ENUM_STANDALONE_SUBCA = 0x00000004
} ENUM_CATYPES;

typedef struct _CERTTRANSDBATTRIBUTE
{
    ULONG obwszName;
    ULONG obwszValue;
} CERTTRANSDBATTRIBUTE;

typedef struct _CERTTRANSDBEXTENSION
{
    ULONG obwszName;
    LONG  ExtFlags;
    DWORD cbValue;
    ULONG obValue;
} CERTTRANSDBEXTENSION;


typedef struct _CERTTRANSDBCOLUMN
{
    DWORD Type;
    DWORD Index;
    DWORD cbMax;
    ULONG obwszName;
    ULONG obwszDisplayName;
} CERTTRANSDBCOLUMN;

typedef struct _CERTTRANSDBRESULTCOLUMN
```

Release: Friday, October 25, 2013

```
{
    DWORD   Type;
    DWORD   Index;
    ULONG   obValue;
    DWORD   cbValue;
} CERTTRANSDBRESULTCOLUMN;

typedef struct _CERTTRANSDBRESULTROW// Marshaled form
{
    DWORD rowid;
    DWORD ccol;
    ULONG cbrow;
} CERTTRANSDBRESULTROW;


typedef struct _CERTVIEWRESTRICTION
{
    DWORD                           ColumnIndex;
    LONG                            SeekOperator;
    LONG                            SortOrder;
    [size_is(cbValue), unique] BYTE *pbValue;
    DWORD                           cbValue;
} CERTVIEWRESTRICTION;

/* Interface ICertAdminD */
[
    object,
    uuid(d99e6e71-fc88-11d0-b498-00a0c90312f3),
    helpstring("ICertAdmin DCOM Interface"),
    pointer_default(unique)
]
interface ICertAdminD: IUnknown
{
    HRESULT SetExtension(
        [in, string, unique] wchar_t const *pwszAuthority,
        [in]                 DWORD          dwRequestId,
        [in, string, unique] wchar_t const *pwszExtensionName,
        [in]                 DWORD          dwType,
        [in]                 DWORD          dwFlags,
        [in, ref]            CERTTRANSBLOB *pctbValue
    );

    HRESULT SetAttributes(
        [in, string, unique] wchar_t const *pwszAuthority,
        [in]                 DWORD          dwRequestId,
        [in, string, unique] wchar_t const *pwszAttributes
    );

    HRESULT ResubmitRequest(
        [in, string, unique] wchar_t const *pwszAuthority,
        [in]                 DWORD          dwRequestId,
        [out]                DWORD         *pdwDisposition
    );

    HRESULT DenyRequest(
        [in, string, unique] wchar_t const *pwszAuthority,
        [in]                 DWORD          dwRequestId
    );
```

```
HRESULT IsValidCertificate(
    [in, string, unique] wchar_t const *pwszAuthority,
    [in, string, unique] wchar_t const *pSerialNumber,
    [out]                LONG          *pRevocationReason,
    [out]                LONG          *pDisposition
);

HRESULT PublishCRL(
    [in, string, unique] wchar_t const *pwszAuthority,
    [in]                 FILETIME       FileTime
);

HRESULT GetCRL(
    [in, string, unique] wchar_t const *pwszAuthority,
    [out, ref]           CERTTRANSBLOB *pctbCRL
);

HRESULT RevokeCertificate(
    [in, string, unique] wchar_t const *pwszAuthority,
    [in, string, unique] wchar_t const *pwszSerialNumber,
    [in]                 DWORD          Reason,
    [in]                 FILETIME       FileTime
);

HRESULT EnumViewColumn(
    [in, string, unique] wchar_t const *pwszAuthority,
    [in]                 DWORD          iColumn,
    [in]                 DWORD          cColumn,
    [out]                DWORD         *pcColumn,
    [out, ref]           CERTTRANSBLOB *pctbColumnInfo
);

HRESULT GetViewDefaultColumnSet(
    [in, string, unique] wchar_t const *pwszAuthority,
    [in]                 DWORD          iColumnSetDefault,
    [out]                DWORD         *pcColumn,
    [out, ref]           CERTTRANSBLOB *pctbColumnInfo
);

HRESULT EnumAttributesOrExtensions(
    [in, string, unique] wchar_t const *pwszAuthority,
    [in]                 DWORD          RowId,
    [in]                 DWORD          Flags,
    [in, string, unique] wchar_t const *pwszLast,
    [in]                 DWORD          celt,
    [out]                DWORD         *pceltFetched,
    [out, ref]           CERTTRANSBLOB *pctbOut
);


HRESULT OpenView(
    [in, string, unique] wchar_t const         *pwszAuthority,
    [in]                 DWORD                            ccvr,
    [in, size_is(ccvr)]  CERTVIEWRESTRICTION const *acvr,
    [in]                 DWORD                            ccolOut,
    [in, size_is(ccolOut)] DWORD const               *acolOut,
    [in]                 DWORD                            ielt,
    [in]                 DWORD                            celt,
    [out]                DWORD                   *pceltFetched,
```

```
    [out, ref]              CERTTRANSBLOB      *pctbResultRows
);


HRESULT EnumView(
    [in, string, unique] wchar_t const *pwszAuthority,
    [in]                 DWORD          ielt,
    [in]                 DWORD          celt,
    [out]                DWORD         *pceltFetched,
    [out, ref]           CERTTRANSBLOB *pctbResultRows
);


HRESULT CloseView(
    [in, string, unique] wchar_t const *pwszAuthority
);

HRESULT ServerControl(
    [in, string, unique] wchar_t const *pwszAuthority,
    [in]                 DWORD          dwControlFlags,
    [out, ref]           CERTTRANSBLOB *pctbOut
);

/* this is a test function */
HRESULT Ping(
    [in, string, unique] wchar_t const *pwszAuthority
);

HRESULT GetServerState(
    [in, string, unique] wchar_t const *pwszAuthority,
    [out]                DWORD         *pdwState
);

HRESULT BackupPrepare(
    [in, string, unique] wchar_t const *pwszAuthority,
    [in]                 unsigned long  grbitJet,
    [in]                 unsigned long  dwBackupFlags,
    [in]                 WCHAR const   *pwszBackupAnnotation,
    [in]                 DWORD          dwClientIdentifier
);

HRESULT BackupEnd(
);

HRESULT BackupGetAttachmentInformation(
    [out, size_is( , *pcwcDBFiles)] WCHAR **ppwszzDBFiles,
    [out]                           LONG   *pcwcDBFiles
);

HRESULT BackupGetBackupLogs(
    [out, size_is( , *pcwcLogFiles)] WCHAR **ppwszzLogFiles,
    [out]                            LONG   *pcwcLogFiles
);

HRESULT BackupOpenFile(
    [in, string, unique] wchar_t const  *pwszPath,
    [out]                unsigned hyper *pliLength
);
```

```
    HRESULT BackupReadFile(
        [ref, out, size_is(cbBuffer)] BYTE *pbBuffer,
        [in]                          LONG  cbBuffer,
        [out]                         LONG *pcbRead
    );

    HRESULT BackupCloseFile(
    );

    HRESULT BackupTruncateLogs(
    );

    HRESULT ImportCertificate(
        [in, string, unique] wchar_t const *pwszAuthority,
        [in, ref]              CERTTRANSBLOB *pctbCertificate,
        [in]                   LONG           dwFlags,
        [out]                  LONG          *pdwRequestId
    );

    HRESULT BackupGetDynamicFiles(
        [out, size_is( , *pcwcFiles)] WCHAR **ppwszzFiles,
        [out]                         LONG   *pcwcFiles
    );

    HRESULT RestoreGetDatabaseLocations(
        [out, size_is( , *pcwcPaths)] WCHAR
                   **ppwszzDatabaseLocations,
        [out]                         LONG   *pcwcPaths
    );
};


/* Interface ICertAdminD2 */
[
    object,
    uuid(7fe0d935-dda6-443f-85d0-1cfb58fe41dd),
    helpstring("ICertAdmin2 DCOM Interface"),
    pointer_default(unique)
]
interface ICertAdminD2: ICertAdminD
{
    HRESULT PublishCRLs(
        [in, string, unique] wchar_t const *pwszAuthority,
        [in]                   FILETIME       FileTime,
        [in]                   DWORD          Flags
    );

    HRESULT GetCAProperty(
        [in, string, unique] wchar_t const *pwszAuthority,
        [in]                 LONG            PropId,
        [in]                 LONG            PropIndex,
        [in]                 LONG            PropType,
        [out, ref]           CERTTRANSBLOB *pctbPropertyValue
    );

    HRESULT SetCAProperty(
        [in, string, unique] wchar_t const *pwszAuthority,
        [in]                 LONG            PropId,
```

```
    [in]                    LONG            PropIndex,
    [in]                    LONG            PropType,
    [in]                    CERTTRANSBLOB *pctbPropertyValue
);

HRESULT GetCAPropertyInfo(
    [in, string, unique] wchar_t const *pwszAuthority,
    [out]                   LONG            *pcProperty,
    [out, ref]              CERTTRANSBLOB *pctbPropInfo
);

HRESULT EnumViewColumnTable(
    [in, string, unique] wchar_t const *pwszAuthority,
    [in]                    DWORD           iTable,
    [in]                    DWORD           iColumn,
    [in]                    DWORD           cColumn,
    [out]                   DWORD           *pcColumn,
    [out, ref]              CERTTRANSBLOB *pctbColumnInfo
);

HRESULT GetCASecurity(
    [in, string, unique] wchar_t const *pwszAuthority,
    [out, ref]              CERTTRANSBLOB *pctbSD
);

HRESULT SetCASecurity(
    [in, string, unique] wchar_t const *pwszAuthority,
    [in, ref]               CERTTRANSBLOB *pctbSD
);

/* this is a test function */
HRESULT Ping2(
    [in, string, unique] wchar_t const *pwszAuthority
);

HRESULT GetArchivedKey(
    [in, string, unique] wchar_t const *pwszAuthority,
    [in]                    DWORD           dwRequestId,
    [out, ref]              CERTTRANSBLOB *pctbArchivedKey
);


HRESULT GetAuditFilter(
    [in, string, unique] wchar_t const *pwszAuthority,
    [out]                   DWORD           *pdwFilter
);

HRESULT SetAuditFilter(
    [in, string, unique] wchar_t const *pwszAuthority,
    [in]                    DWORD           dwFilter
);

HRESULT GetOfficerRights(
    [in, string, unique] wchar_t const *pwszAuthority,
    [out]                   BOOL            *pfEnabled,
    [out, ref]              CERTTRANSBLOB *pctbSD
);

HRESULT SetOfficerRights(
```

```
        [in, string, unique] wchar_t const *pwszAuthority,
        [in]                     BOOL               fEnable,
        [in, ref]              CERTTRANSBLOB *pctbSD
    );

    HRESULT GetConfigEntry(
        [in, string, unique] wchar_t const *pwszAuthority,
        [in, string, unique] wchar_t const *pwszNodePath,
        [in, string, ref]    wchar_t const *pwszEntry,
        [out, ref]             VARIANT          *pVariant
    );

    HRESULT SetConfigEntry(
        [in, string, unique] wchar_t const *pwszAuthority,
        [in, string, unique] wchar_t const *pwszNodePath,
        [in, string, ref]    wchar_t const *pwszEntry,
        [in, ref]              VARIANT          *pVariant
    );

    HRESULT ImportKey(
        [in, string, unique] wchar_t const *pwszAuthority,
        [in]                     DWORD             dwRequestId,
        [in, string, unique] wchar_t const *pwszCertHash,
        [in]                     DWORD             dwFlags,
        [in, ref]              CERTTRANSBLOB *pctbKey
    );

    HRESULT GetMyRoles(
        [in, string, unique] wchar_t const *pwszAuthority,
        [out]                    LONG            *pdwRoles
    );

    HRESULT DeleteRow(
        [in, string, unique] wchar_t const *pwszAuthority,
        [in]                     DWORD             dwFlags,
        [in]                     FILETIME       FileTime,
        [in]                     DWORD           dwTable,
        [in]                     DWORD           dwRowId,
        [out, retval]        LONG           *pcDeleted
    );
}
```

7 Appendix B: Product Behavior

The information in this specification is applicable to the following Microsoft products or supplemental software. References to product versions include released service packs:

- Windows 2000 operating system

- Windows XP operating system

- Windows Server 2003 operating system

- Windows Vista operating system

- Windows Server 2008 operating system

- Windows 7 operating system

- Windows Server 2008 R2 operating system

- Windows 8 operating system

- Windows Server 2012 operating system

- Windows 8.1 operating system

- Windows Server 2012 R2 operating system

Exceptions, if any, are noted below. If a service pack or Quick Fix Engineering (QFE) number appears with the product version, behavior changed in that service pack or QFE. The new behavior also applies to subsequent service packs of the product 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 1.1: A representation of the history of Windows behavior: Windows Server 2003 stores request submissions and certificate revocations that have occurred since the last log file truncation or backup. Log file volume increases as database activity occurs. The log files can be decreased in size by performing a backup and then calling BackupTruncateLogs (as specified in section 2.2.2.1).

<2> Section 2.1: The following values are used by the Certificate Services Remote Administration Protocol client and server.

Constant/value	Description
d99e6e71-fc88-11d0-b498-00a0c90312f3	UUID for the ICertAdminD interface
7fe0d935-dda6-443f-85d0-1cfb58fe41dd	UUID for the ICertAdminD2 interface

<3> Section 2.1: On a Windows computer, if NULL authentication identity and credentials is passed, the RPC_C_AUTHN_GSS_NEGOTIATE security provider uses the identity and credentials from the process token of the process in which the higher layer application is running. This means the account on which the Certificate Services Remote Administration Protocol client is running is the account whose identity will be sent as the identity of the ORPC call.

Windows Server 2003 will use key recovery certificates that contain the following X.509v3 extensions specific to Windows:

- Application Policies (Policy Identifier = Key Recovery Agent)

- Certificate Template Information

Key recovery certificates, when issued by a Windows enterprise CA, are automatically written to the configuration container of Active Directory. The actual certificates are published to the userCertificate attribute (as specified in [RFC4523]) of the key recovery agent (KRA) object when issued to a member of the **domain** administrators group in Active Directory.

<5> Section 3.1.1: Windows implements the version-specific Request, Attribute, Extension, and CRL database tables as detailed in the following tables.

Request Tables

The following table details the Request table for Windows 2000 Server.

Column identifier	Data type	Maximum size of data	Column name (ADM element)*	Column display name
0x1000	0x10001	4 bytes	"Request.RequestID" (Request_Request_ID)	"Request ID"
0x1001	0x3	65536 bytes	"Request.RawRequest" (Request_Raw_Request)	"Binary Request"
0x1002	0x3	16384 bytes	"Request.RawOldCertificate"	"Old Certificate"
0x1003	0x4	32768 bytes	"Request.RequestAttributes" (Request_Request_Attributes)	"Request Attributes"
0x1004	0x1	4 bytes	"Request.RequestType" (Request_Request_Type)	"Request Type"
0x1005	0x1	4 bytes	"Request.RequestFlags" (Request_Request_Flags)	"Request Flags"
0x1006	0x1	4 bytes	"Request.Status"	"Request Status"
0x1007	0x1	4 bytes	"Request.StatusCode" (Request_Status_Code)	"Request Status Code"
0x1008	0x10001	4 bytes	"Request.Disposition" (Request_Disposition)	"Request Disposition"
0x1009	0x4	8192 bytes	"Request.DispositionMessage" (Request_Disposition_Message)	"Request Disposition Message"
0x100a	0x2	8 bytes	"Request.SubmittedWhen" (Request_Submitted_When)	"Request Submission Date"
0x100b	0x2	8 bytes	"Request.ResolvedWhen"	"Request

Column identifier	Data type	Maximum size of data	Column name (ADM element)*	Column display name
			(Request_Resolved_When)	Resolution Date"
0x100c	0x2	8 bytes	"Request.RevokedWhen" (Request_Revoked_When)	"Revocation Date"
0x100d	0x2	8 bytes	"Request.RevokedEffectiveWhen" (Request_Revocation_Date)	"Effective Revocation Date"
0x100e	0x1	4 bytes	"Request.RevokedReason" (Request_Revoked_Reason)	"Revocation Reason"
0x100f	0x4	2048 bytes	"Request.RequesterName" (Request_Requester_Name)	"Requester Name"
0x1010	0x4	2048 bytes	"Request.RequesterAddress"	"Requester Address"
0x1011	0x4	8192 bytes	"Request.DistinguishedName" (Request_Distinguished_Name)	"Request Distinguished Name"
0x1012	0x3	4096 bytes	"Request.RawName" (Request_Raw_Name)	"Request Binary Name"
0x1013	0x1	4 bytes	"Request.NameType"	"Request Name Type"
0x1014	0x4	8192 bytes	"Request.Country" (Request_Country)	"Request Country/Region"
0x1015	0x4	8192 bytes	"Request.Organization" (Request_Organization)	"Request Organization"
0x1016	0x4	8192 bytes	"Request.OrgUnit" (Request_Org_Unit)	"Request Organization Unit"
0x1017	0x4	8192 bytes	"Request.CommonName" (Request_Common_Name)	"Request Common Name"
0x1018	0x4	8192 bytes	"Request.Locality" (Request_Locality)	"Request City"
0x1019	0x4	8192 bytes	"Request.State" (Request_State)	"Request State"
0x101a	0x4	8192 bytes	"Request.Title" (Request_Title)	"Request Title"
0x101b	0x4	8192 bytes	"Request.GivenName" (Request_Given_Name)	"Request First Name"
0x101c	0x4	8192 bytes	"Request.Initials" (Request_Initials)	"Request Initials"

Column identifier	Data type	Maximum size of data	Column name (ADM element)*	Column display name
0x101d	0x4	8192 bytes	"Request.SurName" (Request_SurName)	"Request Last Name"
0x101e	0x4	8192 bytes	"Request.DomainComponent" (Request_Domain_Component)	"Request Domain Component"
0x101f	0x4	8192 bytes	"Request.EMail" (Request_Email)	"Request Email Address"
0x1020	0x4	8192 bytes	"Request.StreetAddress" (Request_Street_Address)	"Request Street Address"
0x1021	0x4	8192 bytes	"Request.UnstructuredName" (Request_Unstructured_Name)	"Request Unstructured Name"
0x1022	0x4	8192 bytes	"Request.UnstructuredAddress" (Request_Unstructured_Address)	"Request Unstructured Address"
0x1023	0x4	8192 bytes	"Request.DeviceSerialNumber" (Request_Device_Serial_Number)	"Request Device Serial Number"
0x2000	0x10001	4 bytes	"RequestID" (Request_ID)	"Issued Request ID"
0x2001	0x3	16384 bytes	"RawCertificate" (Raw_Certificate)	"Binary Certificate"
0x2002	0x4	128 bytes	"CertificateHash" (Certificate_Hash)	"Certificate Hash"
0x2003	0x4	254 bytes	"CertificateType" (Certificate_Template)	"Certificate Type"
0x2004	0x10004	128 bytes	"SerialNumber" (Serial_Number)	"Serial Number"
0x2005	0x1	4 bytes	"IssuerNameId" (Issuer_Name_Id)	"Issuer Name ID"
0x2006	0x2	8 bytes	"NotBefore" (Not_Before)	"Certificate Effective Date"
0x2007	0x10002	8 bytes	"NotAfter" (Not_After)	"Certificate Expiration Date"
0x2008	0x3	4096 bytes	"RawPublicKey" (Raw_Public_Key)	"Binary Public Key"
0x2009	0x4	254 bytes	"PublicKeyAlgorithm" (Public_Key_Algorithm)	"Public Key Algorithm"

Column identifier	Data type	Maximum size of data	Column name (ADM element)*	Column display name
0x200a	0x3	4096 bytes	"RawPublicKeyAlgorithmParameters" (Raw_Public_Key_Algorithm_Parameters)	"Public Key Algorithm Parameters"
0x200b	0x4	8192 bytes	"DistinguishedName" (Distinguished_Name)	"Issued Distinguished Name"
0x200c	0x3	4096bytes	"RawName" (Raw_Name)	"Issued Binary Name"
0x200d	0x1	4 bytes	"NameType"	"Issued Name Type"
0x200e	0x4	8192 bytes	"Country" (Country)	"Issued Country/Region"
0x200f	0x4	8192 bytes	"Organization" (Organization)	"Issued Organization"
0x2010	0x4	8192 bytes	"OrgUnit" (Org_Unit)	"Issued Organization Unit"
0x2011	0x10004	8192 bytes	"CommonName" (Common_Name)	"Issued Common Name"
0x2012	0x4	8192 bytes	"Locality" (Locality)	"Issued City"
0x2013	0x4	8192 bytes	"State" (State)	"Issued State"
0x2014	0x4	8192 bytes	"Title" (Title)	"Issued Title"
0x2015	0x4	8192 bytes	"GivenName" (Given_Name)	"Issued First Name"
0x2016	0x4	8192 bytes	"Initials" (Initials)	"Issued Initials"
0x2017	0x4	8192 bytes	"SurName" (SurName)	"Issued Last Name"
0x2018	0x4	8192 bytes	"DomainComponent" (Domain_Component)	"Issued Domain Component"
0x2019	0x4	8192 bytes	"EMail" (Email)	"Issued Email address"
0x201a	0x4	8192 bytes	"StreetAddress" (Street_Address)	"Issued Street Address"

Column identifier	Data type	Maximum size of data	Column name (ADM element)*	Column display name
0x201b	0x4	8192 bytes	"UnstructuredName" (Unstructured_Name)	"Issued Unstructured Name"
0x201c	0x4	8192 bytes	"UnstructuredAddress" (Unstructured_Address)	"Issued Unstructured Address"
0x201d	0x4	8192 bytes	"DeviceSerialNumber" (Device_Serial_Number)	"Issued Device Serial Number"
0x201e	0x3	16384 bytes	"RawSMIMECapabilities"	"Issued SMIME Capabilities"

* Not all database columns have associated ADM elements.

The following table details the [Request](#) table for Windows Server 2003.

Column identifier	Data Type	Maximum size of data	Column name (ADM element)	Column display name
0x1000	0x10001	4 bytes	"Request.RequestID" (Request_Request_ID)	"Request ID"
0x1001	0x3	65536 bytes	"Request.RawRequest" (Request_Raw_Request)	"Binary Request"
0x1002	0x3	65536 bytes	"Request.RawArchivedKey" (Request_Raw_ArchivedKey)	"Archived Key"
0x1003	0x4	8192 bytes	"Request.KeyRecoveryHashes" (Request_Key_Recovery_Hashes)	"Key Recovery Agent Hashes"
0x1004	0x3	16384 bytes	"Request.RawOldCertificate" (Request_Raw_Old_Certificate)	"Old Certificate"
0x1005	0x4	32768 bytes	"Request.RequestAttributes" (Request_Request_Attributes)	"Request Attributes"
0x1006	0x1	4 bytes	"Request.RequestType" (Request_Request_Type)	"Request Type"
0x1007	0x1	4 bytes	"Request.RequestFlags" (Request_Request_Flags)	"Request Flags"
0x1008	0x1	4 bytes	"Request.StatusCode" (Request_Status_Code)	"Request Status Code"
0x1009	0x10001	4 bytes	"Request.Disposition" (Request_Disposition)	"Request Disposition"
0x100a	0x4	8192 bytes	"Request.DispositionMessage"	"Request Disposition

Column identifier	Data Type	Maximum size of data	Column name (ADM element)	Column display name
			(Request_Disposition_Message)	Message"
0x100b	0x10002	8 bytes	"Request.SubmittedWhen" (Request_Submitted_When)	"Request Submission Date"
0x100c	0x10002	8 bytes	"Request.ResolvedWhen" (Request_Resolved_When)	"Request Resolution Date"
0x100d	0x2	8 bytes	"Request.RevokedWhen" (Request_Revoked_When)	"Revocation Date"
0x100e	0x10002	8 bytes	"Request.RevokedEffectiveWhen" (Request_Revocation_Date)	"Effective Revocation Date"
0x100f	0x1	4 bytes	"Request.RevokedReason" (Request_Revoked_Reason)	"Revocation Reason"
0x1010	0x10004	2048 bytes	"Request.RequesterName" (Request_Requester_Name)	"Requester Name"
0x1011	0x10004	2048 bytes	"Request.CallerName" (Request_Caller_Name)	"Caller Name"
0x1012	0x4	8192 bytes	"Request.SignerPolicies" (Request_Signer_Policies)	"Signer Policies"
0x1013	0x4	8192 bytes	"Request.SignerApplicationPolicies" (Request_Signer_Application_Policies)	"Signer Application Policies"
0x1014	0x1	4 bytes	"Request.Officer" (Request_Officer)	"Officer"
0x1015	0x4	8192 bytes	"Request.DistinguishedName" (Request_Distinguished_Name)	"Request Distinguished Name"
0x1016	0x3	4096 bytes	"Request.RawName" (Request_Raw_Name)	"Request Binary Name"
0x1017	0x4	8192 bytes	"Request.Country" (Request_Country)	"Request Country/Region"
0x1018	0x4	8192 bytes	"Request.Organization" (Request_Organization)	"Request Organization"
0x1019	0x4	8192 bytes	"Request.OrgUnit" (Request_Org_Unit)	"Request Organization Unit"
0x101a	0x4	8192 bytes	"Request.CommonName" (Request_Common_Name)	"Request Common Name"
0x101b	0x4	8192 bytes	"Request.Locality"	"Request City"

Column identifier	Data Type	Maximum size of data	Column name (ADM element)	Column display name
			(Request_Locality)	
0x101c	0x4	8192 bytes	"Request.State" (Request_State)	"Request State"
0x101d	0x4	8192 bytes	"Request.Title" (Request_Title)	"Request Title"
0x101e	0x4	8192 bytes	"Request.GivenName" (Request_Given_Name)	"Request First Name"
0x101f	0x4	8192 bytes	"Request.Initials" (Request_Initials)	"Request Initials"
0x1020	0x4	8192 bytes	"Request.SurName" (Request_SurName)	"Request Last Name"
0x1021	0x4	8192 bytes	"Request.DomainComponent" (Request_Domain_Component)	"Request Domain Component"
0x1022	0x4	8192 bytes	"Request.EMail" (Request_Email)	"Request Email Address"
0x1023	0x4	8192 bytes	"Request.StreetAddress" (Request_Street_Address)	"Request Street Address"
0x1024	0x4	8192 bytes	"Request.UnstructuredName" (Request_Unstructured_Name)	"Request Unstructured Name"
0x1025	0x4	8192 bytes	"Request.UnstructuredAddress" (Request_Unstructured_Address)	"Request Unstructured Address"
0x1026	0x4	8192 bytes	"Request.DeviceSerialNumber" (Request_Device_Serial_Number)	"Request Device Serial Number"
0x2000	0x10001	4 bytes	"RequestID" (Request_ID)	"Issued Request ID"
0x2001	0x3	16384 bytes	"RawCertificate" (Raw_Certificate)	"Binary Certificate"
0x2002	0x10004	128 bytes	"CertificateHash" (Certificate_Hash)	"Certificate Hash"
0x2003	0x10004	254 bytes	"CertificateTemplate" (Certificate_Template)	"Certificate Template"
0x2004	0x1	4 bytes	"EnrollmentFlags" (Enrollment_Flags)	"Template Enrollment Flags"
0x2005	0x1	4 bytes	"GeneralFlags"	"Template General

Release: Friday, October 25, 2013

Column identifier	Data Type	Maximum size of data	Column name (ADM element)	Column display name
			(General_Flags)	Flags"
0x2006	0x10004	128 bytes	"SerialNumber" (Serial_Number)	"Serial Number"
0x2007	0x1	4 bytes	"IssuerNameId" (Issuer_Name_Id)	"Issuer Name ID"
0x2008	0x2	8 bytes	"NotBefore" (Not_Before)	"Certificate Effective Date"
0x2009	0x10002	8 bytes	"NotAfter" (Not_After)	"Certificate Expiration Date"
0x200a	0x4	128 bytes	"SubjectKeyIdentifier" (Subject_Key_Identifier)	"Issued Subject Key Identifier"
0x200b	0x3	4096 bytes	"RawPublicKey" (Raw_Public_Key)	"Binary Public Key"
0x200c	0x1	4 bytes	"PublicKeyLength" (Public_Key_Length)	"Public Key Length"
0x200d	0x4	254 bytes	"PublicKeyAlgorithm" (Public_Key_Algorithm)	"Public Key Algorithm"
0x200e	0x3	4096 bytes	"RawPublicKeyAlgorithmParameters" (Raw_Public_Key_Algorithm_Parameters)	"Public Key Algorithm Parameters"
0x200f	0x10004	2048 bytes	"UPN" (UPN)	"User Principal Name"
0x2010	0x4	8192 bytes	"DistinguishedName" (Distinguished_Name)	"Issued Distinguished Name"
0x2011	0x3	4096 bytes	"RawName" (Raw_Name)	"Issued Binary Name"
0x2012	0x4	8192 bytes	"Country" (Country)	"Issued Country/Region"
0x2013	0x4	8192 bytes	"Organization" (Organization)	"Issued Organization"
0x2014	0x4	8192 bytes	"OrgUnit" (Org_Unit)	"Issued Organization Unit"
0x2015	0x10004	8192 bytes	"CommonName" (Common_Name)	"Issued Common Name"
0x2016	0x4	8192 bytes	"Locality"	"Issued City"

Column identifier	Data Type	Maximum size of data	Column name (ADM element)	Column display name
			(Locality)	
0x2017	0x4	8192 bytes	"State" (State)	"Issued State"
0x2018	0x4	8192 bytes	"Title" (Title)	"Issued Title"
0x2019	0x4	8192 bytes	"GivenName" (Given_Name)	"Issued First Name"
0x201a	0x4	8192 bytes	"Initials" (Initials)	"Issued Initials"
0x201b	0x4	8192 bytes	"SurName" (SurName)	"Issued Last Name"
0x201c	0x4	8192 bytes	"DomainComponent" (Domain_Component)	"Issued Domain Component"
0x201d	0x4	8192 bytes	"EMail" (Email)	"Issued Email address"
0x201e	0x4	8192 bytes	"StreetAddress" (Street_Address)	"Issued Street Address"
0x201f	0x4	8192 bytes	"UnstructuredName" (Unstructured_Name)	"Issued Unstructured Name"
0x2020	0x4	8192 bytes	"UnstructuredAddress" (Unstructured_Address)	"Issued Unstructured Address"
0x2021	0x4	8192 bytes	"DeviceSerialNumber" (Device_Serial_Number)	"Issued Device Serial Number"

The following table details the Request table for Windows Server 2008, Windows Server 2008 R2, Windows Server 2012, and Windows Server 2012 R2.

Column identifier	Data type	Maximum size of data	Column name (ADM element)	Column display name
0x1000	0x10001	4 bytes	Request.RequestID (Request_Request_ID)	"Request ID"
0x1001	0x3	65536 bytes	"Request.RawRequest" (Request_Raw_Request)	"Binary Request"
0x1002	0x3	65536 bytes	"Request.RawArchivedKey" (Request_Raw_Archived_Key)	"Archived Key"

Column identifier	Data type	Maximum size of data	Column name (ADM element)	Column display name
0x1003	0x4	8192 bytes	"Request.KeyRecoveryHashes" (Request_Key_Recovery_Hashes)	"Key Recovery Agent Hashes"
0x1004	0x3	16384 bytes	"Request.RawOldCertificate" (Request_Raw_Old_Certificate)	"Old Certificate"
0x1005	0x4	32768 bytes	"Request.RequestAttributes" (Request_Request_Attributes)	"Request Attributes"
0x1006	0x1	4 bytes	"Request.RequestType" (Request_Request_Type)	"Request Type"
0x1007	0x1	4 bytes	"Request.RequestFlags" (Request_Request_Flags)	"Request Flags"
0x1008	0x1	4 bytes	"Request.StatusCode" (Request_Status_Code)	"Request Status Code"
0x1009	0x10001	4 bytes	"Request.Disposition" (Request_Disposition)	"Request Disposition"
0x100a	0x4	8192 bytes	"Request.DispositionMessage" (Request_Disposition_Message)	"Request Disposition Message"
0x100b	0x10002	8 bytes	"Request.SubmittedWhen" (Request_Submitted_When)	"Request Submission Date"
0x100c	0x10002	8 bytes	"Request.ResolvedWhen" (Request_Resolved_When)	"Request Resolution Date"
0x100d	0x2	8 bytes	"Request.RevokedWhen" (Request_Revoked_When)	"Revocation Date"
0x100e	0x10002	8 bytes	"Request.RevokedEffectiveWhen" (Request_Revocation_Date)	"Effective Revocation Date"
0x100f	0x1	4 bytes	"Request.RevokedReason" (Request_Revoked_Reason)	"Revocation Reason"
0x1010	0x10004	2048 bytes	"Request.RequesterName" (Request_Requester_Name)	"Requester Name"
0x1011	0x10004	2048 bytes	"Request.CallerName" (Request_Caller_Name)	"Caller Name"
0x1012	0x4	8192 bytes	"Request.SignerPolicies" (Request_Signer_Policies)	"Signer Policies"
0x1013	0x4	8192 bytes	"Request.SignerApplicationPolicies" (Request_Signer_Application_Policies)	"Signer Application Policies"

Column identifier	Data type	Maximum size of data	Column name (ADM element)	Column display name
0x1014	0x1	4 bytes	"Request.Officer" (Request_Officer)	"Officer"
0x1015	0x4	8192 bytes	"Request.DistinguishedName" (Request_Distinguished_Name)	"Request Distinguished Name"
0x1016	0x3	4096 bytes	"Request.RawName" (Request_Raw_Name)	"Request Binary Name"
0x1017	0x4	8192 bytes	"Request.Country" (Request_Country)	"Request Country/Region"
0x1018	0x4	8192 bytes	"Request.Organization" (Request_Organization)	"Request Organization"
0x1019	0x4	8192 bytes	"Request.OrgUnit" (Request_Org_Unit)	"Request Organization Unit"
0x101a	0x4	8192 bytes	"Request.CommonName" (Request_Common_Name)	"Request Common Name"
0x101b	0x4	8192 bytes	"Request.Locality" (Request_Locality)	"Request City"
0x101c	0x4	8192 bytes	"Request.State" (Request_State)	"Request State"
0x101d	0x4	8192 bytes	"Request.Title" (Request_Title)	"Request Title"
0x101e	0x4	8192 bytes	"Request.GivenName" (Request_Given_Name)	"Request First Name"
0x101f	0x4	8192 bytes	"Request.Initials" (Request_Initials)	"Request Initials"
0x1020	0x4	8192 bytes	"Request.SurName" (Request_SurName)	"Request Last Name"
0x1021	0x4	8192 bytes	"Request.DomainComponent" (Request_Domain_Component)	"Request Domain Component"
0x1022	0x4	8192 bytes	"Request.EMail" (Request_Email)	"Request Email Address"
0x1023	0x4	8192 bytes	"Request.StreetAddress" (Request_Street_Address)	"Request Street Address"
0x1024	0x4	8192 bytes	"Request.UnstructuredName" (Request_Unstructured_Name)	"Request Unstructured

Column identifier	Data type	Maximum size of data	Column name (ADM element)	Column display name
				Name"
0x1025	0x4	8192 bytes	"Request.UnstructuredAddress" (Request_Unstructured_Address)	"Request Unstructured Address"
0x1026	0x4	8192 bytes	"Request.DeviceSerialNumber" (Request_Device_Serial_Number)	"Request Device Serial Number"
0x1027 *	0x3	4096 bytes	"Request.AttestationChallenge" (Request_Attestation_Challenge)	"Attestation Challenge"
0x1028 *	0x4	144 bytes	"Request.EndorsementKeyHash" (Request_Endorsement_Key_Hash)	"Endorsement Key Hash"
0x1029 *	0x4	144 bytes	"Request.EndorsementCertificateHash" (Request_Endorsement_Certificate_Hash)	"Endorsement Certificate Hash"
0x2000	0x10001	4 bytes	"RequestID" (Request_ID)	"Issued Request ID"
0x2001	0x3	16384 bytes	"RawCertificate" (Raw_Certificate)	"Binary Certificate"
0x2002	0x10004	128 bytes	"CertificateHash" (Certificate_Hash)	"Certificate Hash"
0x2003	0x10004	254 bytes	"CertificateTemplate" (Certificate_Template)	"Certificate Template"
0x2004	0x1	4 bytes	"EnrollmentFlags" (Enrollment_Flags)	"Template Enrollment Flags"
0x2005	0x1	4 bytes	"GeneralFlags" (General_Flags)	"Template General Flags"
0x2006	0x10004	128 bytes	"SerialNumber" (Serial_Number)	"Serial Number"
0x2007	0x1	4 bytes	"IssuerNameId" (Issuer_Name_Id)	"Issuer Name ID"
0x2008	0x2	8 bytes	"NotBefore" (Not_Before)	"Certificate Effective Date"
0x2009	0x10002	8 bytes	"NotAfter" (Not_After)	"Certificate Expiration Date"
0x200a	0x4	128 bytes	"SubjectKeyIdentifier" (Subject_Key_Identifier)	"Issued Subject Key Identifier"
0x200b	0x3	4096 bytes	"RawPublicKey"	"Binary Public

Column identifier	Data type	Maximum size of data	Column name (ADM element)	Column display name
			(Raw_Public_Key)	Key"
0x200c	0x1	4 bytes	"PublicKeyLength" (Public_Key_Length)	"Public Key Length"
0x200d	0x4	254 bytes	"PublicKeyAlgorithm" (Public_Key_Algorithm)	"Public Key Algorithm"
0x200e	0x3	4096 bytes	"RawPublicKeyAlgorithmParameters" (Raw_Public_Key_Algorithm_Parameters)	"Public Key Algorithm Parameters"
0x200f	0x1	4 bytes	"PublishExpiredCertInCRL" (Publish_Expired_Cert_In_CRL)	"Publish Expired Certificate in CRL"
0x2010	0x10004	2048 bytes	"UPN" (UPN)	"User Principal Name"
0x2011	0x4	8192 bytes	"DistinguishedName" (Distinguished_Name)	"Issued Distinguished Name"
0x2012	0x3	4096 bytes	"RawName" (Raw_Name)	"Issued Binary Name"
0x2013	0x4	8192 bytes	"Country" (Country)	"Issued Country/Region"
0x2014	0x4	8192 bytes	"Organization" (Organization)	"Issued Organization"
0x2015	0x4	8192 bytes	"OrgUnit" (Org_Unit)	"Issued Organization Unit"
0x2016	0x10004	8192 bytes	"CommonName" (Common_Name)	"Issued Common Name"
0x2017	0x4	8192 bytes	"Locality" (Locality)	"Issued City"
0x2018	0x4	8192 bytes	"State" (State)	"Issued State"
0x2019	0x4	8192 bytes	"Title" (Title)	"Issued Title"
0x201a	0x4	8192 bytes	"GivenName" (Given_Name)	"Issued First Name"
0x201b	0x4	8192 bytes	"Initials" (Initials)	"Issued Initials"

Column identifier	Data type	Maximum size of data	Column name (ADM element)	Column display name
0x201c	0x4	8192 bytes	"SurName" (SurName)	"Issued Last Name"
0x201d	0x4	8192 bytes	"DomainComponent" (Domain_Component)	"Issued Domain Component"
0x201e	0x4	8192 bytes	"EMail" (Email)	"Issued Email address"
0x201f	0x4	8192 bytes	"StreetAddress" (Street_Address)	"Issued Street Address"
0x2020	0x4	8192 bytes	"UnstructuredName" (Unstructured_Name)	"Issued Unstructured Name"
0x2021	0x4	8192 bytes	"UnstructuredAddress" (Unstructured_Address)	"Issued Unstructured Address"
0x2022	0x4	8192 bytes	"DeviceSerialNumber" (Device_Serial_Number)	"Issued Device Serial Number"

* These database columns are available only in Windows Server 2012 R2.

Extension Tables

The following table details the Extension table for Windows 2000 Server, Windows Server 2003, Windows Server 2008, Windows Server 2008 R2, Windows Server 2012, and Windows Server 2012 R2.

Column identifier	Data type	Maximum size of data	Column name	Column display name
0x4000	0x1001	4 bytes	"ExtensionRequestId" (Extension_Request_ID)	"Extension Request Id"
0x4001	0x4	254 bytes	"ExtensionName" (Extension_Name)	"Extension Name"
0x4002	0x1	4 bytes	"ExtensionFlags" (Extension_Flags)	"Extension Flags"
0x4003	0x3	4096 bytes	"ExtensionRawValue" (Extension_Raw_Value)	"Extension Raw Value"

Attribute Tables

The following table details the Attribute table for Windows 2000 Server, Windows Server 2003, Windows Server 2008, Windows Server 2008 R2, Windows Server 2012, and Windows Server 2012 R2.

Column identifier	Data type	Maximum size of data	Column name	Column display name
0x3000	0x10001	4 bytes	"AttributeRequestId" (Attribute_Request_ID)	"Attribute Request Id"
0x3001	0x4	254 bytes	"AttributeName" (Attribute_Name)	"Attribute Name"
0x3002	0x4	8192 bytes	"AttributeValue" (Attribute_Value)	"Attribute Value"

CRL Tables

The following table details the CRL table for Windows Server 2003, Windows Server 2008, Windows Server 2008 R2, Windows Server 2012, and Windows Server 2012 R2. (Windows 2000 Server does not have a CRL table.)

Column identifier	Data type	Maximum size of data	Column name (ADM element)	Column display name
0x5000	0x10001	4 bytes	CRLRowId (CRL_Row_ID)	"CRL Row ID"
0x5001	0x10001	4 bytes	"CRLNumber" (CRL_Number)	"CRL Number"
0x5002	0x1	4 bytes	"CRLMinBase" (CRL_Min_Base)	"CRL Minimum Base"
0x5003	0x1	4 bytes	"CRLNameId" (CRL_Name_Id)	"CRL Name ID"
0x5004	0x1	4 bytes	"CRLCount" (CRL_Count)	"CRL Count"
0x5005	0x2	8 bytes	"CRLThisUpdate" (CRL_This_Update)	"CRL This Update"
0x5006	0x10002	8 bytes	"CRLNextUpdate" (CRL_Next_Update)	"CRL Next Update"
0x5007	0x2	8 bytes	"CRLThisPublish" (CRL_This_Publish)	"CRL This Publish"
0x5008	0x10002	8 bytes	"CRLNextPublish" (CRL_Next_Publish)	"CRL Next Publish"
0x5009	0x2	8 bytes	"CRLEffective" (CRL_Effective)	"CRL Effective"
0x500a	0x10002	8 bytes	"CRLPropagationComplete" (CRL_Propagation_Complete)	"CRL Propagation Complete"

Column identifier	Data type	Maximum size of data	Column name (ADM element)	Column display name
0x500b	0x10002	8 bytes	"CRLLastPublish" (CRL_Last_Published)	"CRL Last Published"
0x500c	0x10001	4 bytes	"CRLPublishAttempts" (CRL_Publish_Attempts)	"CRL Publish Attempts"
0x500d	0x1	4 bytes	"CRLPublishFlags" (CRL_Publish_Flags)	"CRL Publish Flags"
0x500e	0x10001	4 bytes	"CRLPublishStatusCode" (CRL_Publish_Status_Code)	"CRL Publish Status Code"
0x500f	0x4	8192 bytes	"CRLPublishError" (CRL_Publish_Error)	"CRL Publish Error Information"
0x5010	0x3	536870912 bytes	"CRLRawCRL" (CRL_Raw_CRL)	"CRL Raw CRL"

<6> Section 3.1.1.1.1: Windows uses a **DWORD** number to represent these values. The following table shows how Windows internal values correspond to the preceding string representations.

Windows value	Abstract data model value
CR_DISP_ERROR 0x00000001	Request failed
CR_DISP_DENIED 0x00000002	Request denied
CR_DISP_ISSUED 0x00000003	Certificate issued
CR_DISP_UNDER_SUBMISSION 0x00000005	Request pending
CR_DISP_REVOKED 0x00000006	Certificate revoked

<7> Section 3.1.1.1.2: These flags are supported only in Windows Server 2012 R2.

<8> Section 3.1.1.1.2: Request_RequesterName_From_Old_Certificate is supported in Windows Server 2008 R2, Windows Server 2012, and Windows Server 2012 R2 only.

<9> Section 3.1.1.4: Windows 2000, Windows Server 2003, Windows Server 2008, Windows Server 2008 R2, Windows Server 2012, and Windows Server 2012 R2 CAs store this CRL in the registry location
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\SystemCertificates\CA\CRLs.

<10> Section 3.1.1.7: The permissions of Officer, Operator, and Auditor are supported on Windows Server 2003 Enterprise Edition, Windows Server 2003 Datacenter Edition, Windows Server 2008 Enterprise, Windows Server 2008 Datacenter, Windows Server 2008 R2 Enterprise Edition, Windows Server 2008 R2 Datacenter Edition, Windows Server 2012, and Windows Server 2012 R2.

The Microsoft CA keeps all CRL publishing locations in a registry multistring value.

```
                 HKLM\SYSTEM\CurrentControlSet\Services\CertSvc\Configuration\
   {CA Name}\CRLPublicationURLs
```

The Microsoft CA uses a numeric prefix that is stored with each entry of the CRL publishing location list in order to indicate which of the ADM elements the URL belongs to, as shown in the following table.

Numeric prefix value	ADM element
0x00000001	Config_CA_CDP_Publish_To_Base
0x00000040	Config_CA_CDP_Publish_To_Delta
0x00000008	Config_CA_CDP_Include_In_CRL_Publish_Locations_Extension
0x00000004	Config_CA_CDP_Include_In_CRL_Freshest_CRL_Extension
0x00000080	Config_CA_CDP_Include_In_CRL_IDP_Extension

Calling the ICertAdminD2::SetConfigEntry method with the pwszNodePath as an EMPTY string, the pwszEntry "CRLPublicationURLs", and the pVariant data that contains the required URLs allows manipulation of this list.

Also, the usual registry manipulation tools that are specified in [MS-RRP] can be used to update these values.

For Config_CA_CDP_Publish_To_Base and Config_CA_CDP_Publish_To_Delta, the default values that are used by the Microsoft CA are a local path on the CA machine,

```
                 {SYSTEM}\CertSrv\CertEnroll\{CATruncatedName}{CRLNameSuffix}
   {DeltaIndicator}.crl
```

a local path in the registry,

```
                 HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\SystemCertificates\CA\CRLs\
```

and the Active Directory path,

```
   ldap: ///CN={CATruncatedName}{CRLNameSuffix},CN={CAServerShortName},
   CN=CDP,CN=Public Key
```

```
Services,CN=Services,CN=Configuration,DC={contoso},DC=com{CDPObjectClass}
```

where:

- "{**SYSTEM**}" is replaced with the system directory of the CA machine, such as "C:\Windows\System32".

- "{**CATruncatedName**}" is replaced with the sanitized name of the CA, as defined in [MS-WCCE] sections 1.3.2.4 and 3.1.1.4.1.1.

- "{**DeltaIndicator**}" is replaced with NULL for a base CRL and "+" for a delta CRL.

- "{**CRLNameSuffix**}" is replaced with NULL if the CRL is signed by the first CA key (a CA key that has a key index of 0) and by "(n)" if the CRL is signed by any subsequent CA key.

  Where "(n)" is replaced with an integer that is equal to the identifier Signing_Private_Key_Version_ID, as defined in [MS-WCCE] section 3.2.1.1.2 and in the example in [MS-WCCE] section 3.2.1.4.3.2.34.

- "{**CAServerShortName**}" is replaced with the name of the host on which the CA is running.

- "**DC={contoso},DC=com**" is replaced by the distinguished name (DN) of the forest root domain naming context (NC) of the Active Directory forest in which the Microsoft CA is installed.

  - The forest root domain NC is defined in [MS-ADTS] section 1.1.

  - For example, the DN of the forest root domain NC of a forest called "corp.contoso.com" would be "DC=corp,DC=contoso,DC=com".

- "{**CDPObjectClass**}" is replaced with "?certificateRevocationList?base?objectClass=cRLDistributionPoint" for a base CRL and with "?deltaRevocationList?base?objectClass=cRLDistributionPoint" for a delta CRL.

  - The object class cRLDistributionPoint is as defined in [MS-ADSC].

  - The attribute certificateRevocationList is defined in [MS-ADA1].

  - The attribute deltaRevocationList is defined in [MS-ADA1].

The deltaRevocationList attribute is not used by the Windows 2000 version of the CA. The Windows Server 2003, Windows Server 2008, Windows Server 2008 R2, Windows Server 2012, and Windows Server 2012 R2 versions of the CA use both base CRL and delta CRL attributes.

Within the certificateRevocationList or deltaRevocationList attribute, the CRL is encoded by using Distinguished Encoding Rules (DER).

<12> Section 3.1.1.10: Microsoft CAs persist only a subset of the configuration data. They store the configuration data in the registry in the following locations:

Values under

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\CertSvc\Configuration\
```

DBSessionCount

> **ADM Datum:** Config_Max_Number_Of_AD_Connections and OnNextRestart_Config_Max_Number_Of_AD_Connections.

> **Registry Value Type:** REG_DWORD

> **Default Value:** 20

> **Registry Value Mapping to ADM:** The value in the registry equals the ADM datum value. The minimum value for this registry is 4 and the maximum value is 1024.

> **No Value Semantics:** The value always exists.

LDAPFlags

> **ADM Datum:** Config_CA_LDAP_Flags and OnNextRestart_Config_CA_LDAP_Flags.

> **Registry Value Type:** REG_DWORD

> **Default Value:** 0

> **Registry Value Mapping to ADM:** The value in the registry equals the ADM datum value.

> **No Value Semantics:** The value always exists.

Version

> **ADM Datum:** Config_Product_Version and OnNextRestart_Config_Product_Version

> **Registry Value Type:** REG_DWORD

> **Default Value:** By default, the value depends on the Windows version:

> - 0x00010001: Windows 2000 Server
> - 0x00020002: Windows Server 2003
> - 0x00030001: Windows Server 2008
> - 0x00040001: Windows Server 2008 R2
> - 0x00050001: Windows Server 2012
> - 0x00060001: Windows Server 2012 R2

> **Registry Value Mapping to ADM:** The value in the registry equals the ADM datum value.

> **No Value Semantics:** The value always exists.

Values under

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\CertSvc\Configuration\<CA_CN>
```

where <CA_CN> is replaced with the common name (CN) of the CA. The values are as follows:

ConfigurationDirectory

ADM Datum: Config_Configuration_Directory (defined in [MS-WCCE]).

Registry Value Type: REG_SZ

Default Value: By default, the value does not exist.

Registry Value Mapping to ADM: The value in the registry equals the ADM datum value.

No Value Semantics: The CA does not implement Config_Configuration_Directory datum (for details, see section 3.2.1.4.3.2.8).

ParentCAMachine

ADM Datum: Config_CA_Parent_DNS (defined in [MS-WCCE]).

Registry Value Type: REG_SZ

Default Value: By default, the value does not exist for the root CA. For the **subordinate CA**, the value is set to the FQDN (1) of the machine where the parent CA is installed.

Registry Value Mapping to ADM: The value in the registry equals the ADM datum value.

No Value Semantics: The CA does not have a parent CA.

RoleSeparationEnabled

ADM Datum: Config_CA_Role_Separation (defined in [MS-WCCE]).

Registry Value Type: REG_DWORD

Default Value: The value does not exist by default.

Registry Value Mapping to ADM: If the value in the registry is zero, the ADM datum is set to false. For any nonzero registry value, ADM datum is set to true.

No Value Semantics: Same as setting the value to zero.

CAXchgCertHash

ADM Datum: Config_CA_Exchange_Cert (defined in [MS-WCCE]).

Registry Value Type: REG_MULTI_SZ

Default Value: None.

Registry Value Mapping to ADM: Each value is an SHA-1 hash of the corresponding CA exchange certificate. The actual exchange certificates are stored in the Request table.

No Value Semantics: There are no CA exchange certificates configured on the server.

CACertPublicationURLs

- **ADM Datum**: Multiple, see **Registry Value Mapping to ADM**.

- **Registry Value Type**: REG_MULTI_SZ

- **Default Value**:

  1:C:\Windows\system32\CertSrv\CertEnroll\%1_%3%4.crt

3:ldap:///CN=%7,CN=AIA,CN=Public Key Services,CN=Services,%6%11

2:http://%1/CertEnroll/%1_%3%4.crt

0:file://%1/CertEnroll/%1_%3%4.crt

- **Registry Value Mapping to ADM**: The string format for each string in the list is

  <Numeric Prefix Value>:<Some Path>

  Where the <Numeric Prefix Value> is a binary OR of the values in the following table and is represented as a decimal value. And where <Some Path> is a string that is composed of literal strings and wild cards, that is defined in the following tables, and that represents an HTTP, FILE, or LDAP **URL)**, or a UNC path.

Numeric prefix value	ADM element
0x00000001	Config_CA_CACert_Publish_To and OnNextRestart_Config_CA_CACert_Publish_To. The CA publishes its signing certificate to this location.
0x00000002	Config_CA_AIA_Include_In_Cert and OnNextRestart_Config_CA_AIA_Include_In_Cert.
0x00000020	Config_CA_OCSP_Include_In_Cert and OnNextRestart_Config_CA_OCSP_Include_In_Cert.

Wild card	Meaning
%1	The NetBIOS name of the server on which the CA is installed.
%2	The FQDN (1) of the server on which the CA is installed.
%3	The name of the CA.
%4	The certificate file name suffix that corresponds to the key index for the CA signing key.
%6	The DN name of the configuration container.
%7	The truncated name of the CA.
%11	The class name for the AD object of the CA.

- **No Value Semantics**: The ADMs that correspond to the prefixes are empty.

CRLPublicationURLs

- **ADM Datum**: Multiple, see **Registry Value Mapping to ADM**.

- **Registry Value Type**: REG_MULTI_SZ

- **Default Value**:

  65:E:\Windows\system32\CertSrv\CertEnroll\%3%8%9.crl

  79:ldap:///CN=%7%8,CN=%2,CN=CDP,CN=Public Key Services,CN=Services,%6%10

6:http://%1/CertEnroll/%3%8%9.crl

0:file://%1/CertEnroll/%3%8%9.crl

- **Registry Value Mapping to ADM**: The string format for each string in the list is:

<Numeric Prefix Value>:<Some Path>

Where the <Numeric Prefix Value> is a binary OR of the values in the following table and is represented as a decimal value. And <Some Path> is a string that is composed of literal strings and wild cards, that is defined in the following tables, and that represents an HTTP, FILE, or LDAP URL, or a UNC path.

Numeric prefix value	ADM element
0x00000001	Config_CA_CDP_Publish_To_Base and OnNextRestart_Config_CA_CDP_Publish_To_Base
0x00000002	Config_CA_CDP_Include_In_Cert and OnNextRestart_Config_CA_CDP_Include_In_Cert
0x00000040	Config_CA_CDP_Publish_To_Delta and OnNextRestart_Config_CA_CDP_Publish_To_Delta
0x00000008	Config_CA_CDP_Include_In_CRL_Publish_Locations_Extension and OnNextRestart_Config_CA_CDP_Include_In_CRL_Publish_Locations_Extension
0x00000004	Config_CA_CDP_Include_In_CRL_Freshest_CRL_Extension and OnNextRestart_Config_CA_CDP_Include_In_CRL_Freshest_CRL_Extension
0x00000080	Config_CA_CDP_Include_In_CRL_IDP_Extension and OnNextRestart_Config_CA_CDP_Include_In_CRL_IDP_Extension

Wild card	Meaning
%1	The FQDN (1) of the server on which the CA is installed.
%2	The NetBIOS name of the server on which the CA is installed.
%3	The name of the CA.
%4	The certificate name.
%6	The DN name of the configuration container.
%7	The truncated name of the CA.
%8	The signing key identifier, which is enclosed in (), of the CA.
%9	Append + character suffix.
%10	The AD CRL and delta CRL object class.
%11	The class name for the AD object of the CA.

EnrollmentAgentRights

- **ADM Datum**: Config_Permissions_Enrollment_Agent_Rights (defined in [MS-WCCE]).

- **Registry Value Type**: REG_BINARY

- **Default Value**: None.

- **Registry Value Mapping to ADM**: A binary security descriptor that is defined in section 2.2.1.11.

- **No Value Semantics**: No Enrollment Agent permissions are defined.

OfficerRights

- **ADM Datum**: Config_Permissions_Officer_Rights (defined in [MS-WCCE]).

- **Registry Value Type**: REG_BINARY

- **Default Value**: None.

- **Registry Value Mapping to ADM**: A binary security descriptor that is defined in section 2.2.1.11.

- **No Value Semantics**: No Officer permissions are defined.

Security

- **ADM Datum**: Config_Permissions_CA_Security (defined in [MS-WCCE]), and OnNextRestart_Config_Permissions_CA_Security

- **Registry Value Type**: REG_BINARY

- **Default Value**:

  - Standalone CA

    - Builtin Administrators – Owner, Group, Administrator, Officer

    - Everyone: Enroll

  - Enterprise CA

    - Builtin Administrators – Owner, Group, Administrator, Officer

    - Domain Administrators – Administrator, Officer

    - Enterprise Administrators – Administrator, Officer

    - Authenticated Users – Enroll

- **Registry Value Mapping to ADM**: A binary security descriptor with permissions that are defined as follows.

Permission	Bit value
Read	0x00000100
Enroll	0x00000200
Officer	0x00000002
Administrator	0x00000001

Permission	Bit value
Auditor	0x00000004
Operator	0x00000008

- **No Value Semantics**: None defined.

AuditFilter

ADM Datum: Config_CA_Audit_Filter and OnNextRestart_Config_CA_Audit_Filter

Registry Value Type: REG_DWORD

Default Value: None.

Registry Value Mapping to ADM: Specified in section 3.1.4.2.10.

No Value Semantics: No auditing occurs.

CAType

ADM Datum: Config_CA_Type and OnNextRestart_Config_CA_Type

Registry Value Type: REG_DWORD

Default Value: None.

Provisioning: Populated by the CA installation, based upon a selection made by the administrator.

Registry Value Mapping to ADM:

- ENUM_ENTERPRISE_ROOTCA = 0x00000000
- ENUM_ENTERPRISE_SUBCA = 0x00000001
- ENUM_STANDALONE_ROOTCA = 0x00000003
- ENUM_STANDALONE_SUBCA = 0x00000004

No Value Semantics: Value always present.

CommonName

ADM Datum: Config_CA_Common_Name and OnNextRestart_Config_CA_Common_Name

Registry Value Type: REG_SZ

Default Value: None.

Provisioning: Populated by the CA installation, based upon a selection made by the administrator.

Registry Value Mapping to ADM: Value is set to installed CA name.

No Value Semantics: Value always present.

CRLDeltaNextPublish

ADM Datum: Config_CA_CRL_Delta_Next_Publish and
OnNextRestart_Config_CA_CRL_Delta_Next_Publish

Registry Value Type: REG_BINARY

Default Value: Installed filetime plus 1 day.

Registry Value Mapping to ADM: Specified in section 3.1.4.2.14.

No Value Semantics: Value always present is re-created based on calculations specified in
section 3.1.4.1.6.

CRLNextPublish

ADM Datum: Config_CA_CRL_Next_Publish and OnNextRestart_Config_CA_CRL_Next_Publish

Registry Value Type: REG_BINARY

Default Value: Installed filetime plus 1 week.

Registry Value Mapping to ADM: Specified in section 3.1.4.2.14.

No Value Semantics: Value is re-created based on calculations specified in section 3.1.4.1.6.

CRLDeltaPeriod and CRLDeltaPeriodUnits

ADM Datum: Config_Delta_CRL_Validity_Period

Registry Value Type: CRLDeltaPeriod is REG_SZ, and CRLDeltaPeriodUnits is REG_DWORD

Default Value: CRLDeltaPeriodUnits = 1; CRLDeltaPeriod = "Days"

Registry Value Mapping to ADM: CRLDeltaPeriod contains the string representation of the
"periods of time" in which the Config_Delta_CRL_Validity_Period is expressed. Valid values are
Seconds, Minutes, Hours, Days, Weeks, or Months. CRLDeltaPeriodUnits contains the number
of those periods of time. For example, see "Default Value" above.

No Value Semantics: Hardcoded default values equal to "Default Value", above, are used

CRLPeriod and CRLPeriodUnits

ADM Datum: Config_Base_CRL_Validity_Period

Registry Value Type: CRLPeriod is REG_SZ, and CRLPeriodUnits is REG_DWORD.

Default Value: CRLPeriodUnits = 1; CRLPeriod = "Weeks".

Registry Value Mapping to ADM: CRLPeriod contains the string representation of the "periods
of time" in which the Config_Base_CRL_Validity_Period is expressed. Valid values are
Seconds, Minutes, Hours, Days, Weeks, or Months. CRLPeriodUnits contains the number of
those periods of time. For example, see "Default Value" above.

No Value Semantics: Hardcoded default values equal to "Default Value" above, are used.

HighSerial

ADM Datum: Config_High_Serial_Number, Config_High_Serial_String,
OnNextRestart_Config_High_Serial_Number, and OnNextRestart_Config_High_Serial_String

Registry Value Type: REG_DWORD

Default Value: The value does not exist by default. It must be created manually using registry methods.

Registry Value Mapping to ADM: Defined in [MS-WCCE] section 3.2.1.4.2.1.4.5.

No Value Semantics: Same as registry value of 0.

InterfaceFlags

ADM Datum: Config_CA_Interface_Flags and OnNextRestart_Config_CA_Interface_Flags.

Registry Value Type: REG_DWORD

Default Value: 0x41

Registry Value Mapping to ADM: Defined in section 3.1.4.2.14.

No Value Semantics: No interface flags in effect.

KRAFlags

ADM Datum: Config_CA_KRA_Flags and OnNextRestart_Config_CA_KRA_Flags.

Registry Value Type: REG_DWORD

Default Value: 0

Registry Value Mapping to ADM: Defined in section 3.1.4.2.14.

No Value Semantics: No KRA flags in effect.

SetupStatus

ADM Datum: Config_Setup_Status and OnNextRestart_Config_Setup_Status.

Registry Value Type: REG_DWORD

Default Value: By default, for a complete CA installation, the value is 1.

Registry Value Mapping to ADM: The value in the registry equals the ADM datum value.

No Value Semantics: Value always present.

SubjectTemplate

ADM Datum: Config_CA_DN_Order_String

Registry Value Type: REG_MULTI_SZ

Default Value: Email

CommonName

OrganizationalUnit

Organization

Locality

State

DomainComponent

Country

Registry Value Mapping to ADM: The value in the registry equals the ADM datum value.

No Value Semantics: Value always present.

CRLDeltaOverlapPeriod and CRLDeltaOverlapUnits

ADM Datum: Config_Delta_CRL_Overlap_Period

Registry Value Type: CRLDeltaOverlapPeriod is REG_SZ, and CRLDeltaOverlapUnits is REG_DWORD.

Default Value: CRLDeltaOverlapUnits = 0; CRLDeltaOverlapPeriod = "Minutes".

Registry Value Mapping to ADM: CRLDeltaOverlapPeriod contains the string representation of the "periods of time" in which the Config_Delta_CRL_Overlap_Period is expressed. Valid values are Seconds, Minutes, Hours, Days, Weeks, or Months. CRLDeltaOverlapUnits contains the number of those periods of time. For example, see "Default Value" above.

No Value Semantics: Hardcoded default values equal to "Default Value", above, are used.

CRLOverlapPeriod and CRLOverlapUnits

ADM Datum: Config_Base_CRL_Overlap_Period

Registry Value Type: CRLOverlapPeriod is REG_SZ, and CRLOverlapUnits is REG_DWORD.

Default Value: CRLOverlapUnits = 0; CRLOverlapPeriod = "Hours"

Registry Value Mapping to ADM: CRLOverlapPeriod contains the string representation of the "periods of time" in which the Config_Base_CRL_Overlap_Period is expressed. Valid values are Seconds, Minutes, Hours, Days, Weeks, or Months. CRLOverlapUnits contains the number of those periods of time. For example, see "Default Value" above.

No Value Semantics: Hardcoded default values equal to "Default Value", above, are used.

CRLAttemptRepublish

ADM Datum: OnNextRestart_Config_CA_CRL_Attempt_Republish

Registry Value Type: REG_DWORD

Default Value: By default the element is absent. Windows instantiates the value upon the first unsuccessful CRL publishing attempt per the processing rules in section 3.1.4.1.6.

Registry Value Mapping to ADM: The value in the registry equals the ADM datum value.

No Value Semantics: Same as value of 0.

Maximum Value: 10

KRACertCount

ADM Datum: Config_CA_KRA_Cert_Count

Registry Value Type: REG_DWORD

Default Value: 0

Registry Value Mapping to ADM: The value in the registry equals the ADM datum value.

No Value Semantics: No KRA certificates available for encryption.

KRACertHash

ADM Datum: Config_CA_KRA_Cert_List

Registry Value Type: REG_MULTI_SZ

Default Value: Empty by default.

Registry Value Mapping to ADM: Each of the multiple values in the registry is the hash of one of the KRA certificates in the list.

No Value Semantics: No KRA certificates available for **encryption**.

UseDS

ADM Datum: Config_CA_Use_DS and OnNextRestart_Config_CA_Use_DS

Registry Value Type: REG_DWORD

Default Value: For an enterprise CA (Config_CA_Type of 0 or 1), the value is 1. For a standalone CA (Config_CA_Type of 3 or 4), the value is 0.

Registry Value Mapping to ADM: The value in the registry equals the ADM datum value.

No Value Semantics: Value always present.

CNGHashAlgorithm

ADM Datum: Config_CSP_CNG_Hash_Algorithm and OnNextRestart_Config_CSP_CNG_Hash_Algorithm

Registry Value Type: REG_SZ

Default Value: None

Provisioning: : If the administrator has selected a **CNG** provider as the Config_CSP_Provider, this value is populated by the CA installation, based upon another selection made by the administrator. If the administrator has selected a **CryptoAPI** CSP as the Config_CSP_Provider, this value is provisioned with no value (null).

Registry Value Mapping to ADM: The value in the registry equals the ADM datum value.

No Value Semantics: If there is no Config_CSP_CNG_Hash_Algorithm value, then this means the CA uses a CryptoAPI CSP and not a CNG provider. In this case, Config_CSP_Hash_Algorithm will contain the identifier of the CA hash algorithm.

HashAlgorithm

ADM Datum: Config_CSP_Hash_Algorithm and OnNextRestart_Config_CSP_Hash_Algorithm

Registry Value Type: REG_DWORD

Default Value: None

Provisioning: If the administrator has selected a CryptoAPI CSP as the Config_CSP_Provider, this value is populated by the CA installation, based upon another selection made by the administrator. If the administrator has not selected a CryptoAPI CSP as the Config_CSP_Provider, this value is initialized to 0xffffffff.

Registry Value Mapping to ADM: The value in the registry is either 0xffffffff or the algid (algorithm identifier) that corresponds to the hash algorithm used by the CA.

0xffffffff – no value. The CA has been installed with a CNG provider, so Config_CSP_CNG_Hash_Algorithm contains the name of the CA hash algorithm.

Algids of supported hash algorithms are as follows:

- MD2   0x8001

- MD5   0x8003

- SHA1   0x8004

- SHA256   0x8012

- SHA384   0x8013

- SHA512   0x8014

No Value Semantics: The element always exists.

Provider

ADM Datum: Config_CSP_Provider and OnNextRestart_Config_CSP_Provider

Registry Value Type: REG_SZ

Default Value: Populated by the CA installation, based upon a selection made by the administrator.

Registry Value Mapping to ADM: The value in the registry equals the ADM datum value.

No Value Semantics: Element always exists.

Values under:

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\CertSvc\Configuration\<CA_CN>\PolicyModu
les\CertificateAuthority_MicrosoftDefault.Policy\RequestDisposition
```

where <CA_CN> is replaced with the common name (CN) of the CA. The values are as follows:

RequestDisposition

ADM Datum: OnNextRestart_Config_CA_Requests_Disposition

Registry Value Type: REG_DWORD

Default Value: For an Enterprise CA (Config_CA_Type of 0 or 1), the value is 0x00000001. For a Standalone CA (Config_CA_Type of 3 or 4), the value is 0x00000101.

Registry Value Mapping to ADM: The value of the registry equals the value of the OnNextRestart_Config_CA_Requests_Disposition datum.

No Value Semantics: The OnNextRestart_Config_CA_Requests_Disposition datum value is set to 0x00000101 (REQDISP_PENDINGFIRST | REQDISP_ISSUE).

ProviderType

ADM Datum: Config_CSP_ProviderType and OnNextRestart_Config_CSP_ProviderType

Registry Value Type: REG_DWORD

Default Value: Populated by the CA installation based upon the selection of Provider made by the administrator. A value of 0 means the provider is a CNG provider. A nonzero value means the provider is a legacy CryptoAPI CSP.

Registry Value Mapping to ADM: The value in the registry equals the ADM datum value.

No Value Semantics: Element always exists.

Values under:

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\CertSvc\Configuration\<CA_CN>\PolicyModu
les\CertificateAuthority_MicrosoftDefault.Policy\EditFlags
```

where <CA_CN> is replaced with the CN of the CA. The values are as follows:

EditFlags

ADM Datum: Multiple, see **Registry Value Mapping to ADM**.

Registry Value Type: REG_DWORD

Default Value: The flags that are defined here for this value are not set by default.

Registry Value Mapping to ADM: 0x00000020 – If this bit is set, Config_CA_Accept_Request_Attributes_ValidityTime (defined in [MS-WCCE]) is set to true. Otherwise, it is set to false.

0x00008000 – If this bit is set, Config_CA_Accept_Request_Attributes_Extensions (defined in [MS-WCCE]) is set to true. Otherwise, it is set to false.

0x00040000 – If this bit is set, Config_CA_Accept_Request_Attributes_SAN (defined in [MS-WCCE]) is set to true. Otherwise, it is set to false.

0x00080000 – If this bit is set, Config_AD_Connection_Referral (defined in [MS-WCCE]) is set to true. Otherwise, it is set to false

0x02000000 – If this bit is set, Config_CA_Allow_RenewOnBehalfOf_Requests is set to true.

No Value Semantics: All ADM elements that are controlled by this value are set to false.

Values under:

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\CertSvc\Configuration\<CA_CN>\CRLFlags
```

where <CA_CN> is replaced with the CN of the CA. The values are as follows:

CRLEditFlag

> **ADM Datum:** Config_CA_Accept_Request_Attributes_Other (defined in [MS-WCCE])

> **Registry Value Type:** REG_DWORD

> **Default Value:** The flags that are defined here for this value are not set by default.

> **Registry Value Mapping to ADM:** 0x00010000 – If this bit is set, Config_CA_Accept_Request_Attributes_Other (defined in [MS-WCCE]) is set to true. Otherwise, it is set to false.

> **No Value Semantics:** All ADM elements that are controlled by this value are set to false.

Values under:

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\CertSvc\Configuration\<CA_CN>\ExitModule
s\CertificateAuthority_MicrosoftDefault.Exit\PublishCertFlags
```

where <CA_CN> is replaced with the CN of the CA. The values are as follows:

PublishCertFlags

> **ADM Datum:** Config_CA_Accept_Request_Attributes_CertPath (defined in [MS-WCCE])

> **Registry Value Type:** REG_DWORD

> **Default Value:** The flags that are defined here for this value are not set by default.

> **Registry Value Mapping to ADM:** 0x00000001 – If this bit is set, Config_CA_Accept_Request_Attributes_CertPath (defined in [MS-WCCE]) is set to true. Otherwise, it is set to false.

> **No Value Semantics:** All ADM elements that are controlled by this value are set to false.

Values under:

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\CertSvc\Configuration\<CA_CN>\ExitModule
s
```

where <CA_CN> is replaced with the CN of the CA. The values are as follows:

Active

> **ADM Datum:** See "Registry Value Mapping to ADM."

> **Registry Value Type:** REG_MULTI_SZ

> **Default Value:** CertificateAuthority_MicrosoftDefault.Exit

> **Registry Value Mapping to ADM:** The names of the COM classes implementing the ICertExit interface and used as exit algorithms on the CA comprise

Config_CA_Exit_Algorithm_Implementation_List and
OnNextRestart_Config_CA_Exit_Algorithm_Implementation_List.

The number of entries in this value sets the Config_CA_Exit_Count.

No Value Semantics: No exit algorithms are installed on the CA.

Values under:

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\CertSvc\Configuration\<CA_CN>\PolicyModu
les
```

where <CA_CN> is replaced with the CN of the CA. The values are as follows:

Active

ADM Datum: Config_CA_Policy_Algorithm_Implementation and
OnNextRestart_Config_CA_Policy_Algorithm_Implementation

Registry Value Type: REG_SZ

Default Value: CertificateAuthority_MicrosoftDefault.Policy

Registry Value Mapping to ADM: The registry value contains the name of the COM class that
implements the ICertPolicy interface and should be used at the policy algorithm on the CA.

No Value Semantics: Unsupported; a Windows CA always has at least one policy module.

<13> Section 3.1.1.10: **Config_CA_Allow_RenewOnBehalfOf_Requests** is supported in
Windows Server 2008 R2, Windows Server 2012, and Windows Server 2012 R2.

<14> Section 3.1.3: The DCOM security descriptor is accessed from the registry location
HKLM\SOFTWARE\Microsoft\Ole\:

Value: MachineAccessRestriction

Values added

COM_RIGHTS_EXECUTE | COM_RIGHTS_EXECUTE_LOCAL | COM_RIGHTS_EXECUTE_REMOTE

and

Value: MachineLaunchRestriction

Values added

COM_RIGHTS_EXECUTE | COM_RIGHTS_ACTIVATE_LOCAL | COM_RIGHTS_ACTIVATE_REMOTE

where

COM_RIGHTS_EXECUTE maps to the value of 1

COM_RIGHTS_EXECUTE_LOCAL maps to the value of 2

COM_RIGHTS_EXECUTE_REMOTE maps to the value of 4

COM_RIGHTS_ACTIVATE_LOCAL maps to the value of 8

COM_RIGHTS_ACTIVATE_REMOTE maps to the value of 16

<15> Section 3.1.3: The Microsoft Windows CA, upon CA service startup, reads the configuration values from the registry location "HKLM\SYSTEM\CurrentControlSet\Services\CertSvc\".

<16> Section 3.1.4.1: The supported clients are Windows 2000 Professional, Windows XP, Windows Vista with Admin Pack, Windows 7, Windows 8, and Windows 8.1. The supported servers are Windows 2000 Server, Windows Server 2003, Windows Server 2008, Windows Server 2008 R2, Windows Server 2012, and Windows Server 2012 R2.

<17> Section 3.1.4.1: In Windows 2000, Windows Server 2003, Windows Server 2008, Windows Server 2008 R2, Windows Server 2012, and Windows Server 2012 R2, the error is E_ACCESSDENIED (0x80070005).

<18> Section 3.1.4.1: In Windows Server 2003, Windows Server 2008, Windows Server 2008 R2, Windows Server 2012, and Windows Server 2012 R2, the error is E_ACCESSDENIED (0x80070005). Windows 2000 does not return an error.

<19> Section 3.1.4.1: In Windows Server 2003, Windows Server 2008, Windows Server 2008 R2, Windows Server 2012, and Windows Server 2012 R2, the error is E_ACCESSDENIED (0x80070005). Windows 2000 does not return an error.

<20> Section 3.1.4.1.3: The Windows Server 2003 CA places 0x80094004 in the *pdwDisposition* parameter and returns successfully. Windows Server 2008, Windows Server 2008 R2, Windows Server 2012, and Windows Server 2012 R2 CAs place 0 in the *pdwDisposition* parameter and return 0x80094004 as the error code.

<21> Section 3.1.4.1.4: Windows 2000, Windows XP, Windows Server 2003, Windows Vista, and Windows Server 2008 set the Request_Status_Code to 0x0 (S_OK).

<22> Section 3.1.4.1.6: In a Windows 2000 CA, CRL creation can be disabled by setting the Config_Base_CRL_Validity_Period to 0. In a Windows 2000 CA, if the Microsoft default exit module "CertificateAuthority_MicrosoftDefault.Exit" is not active (that is, not included in the ADM element **Config_CA_Exit_Algorithm_Implementation_List**), then no CRLs are published. The setting Config_CA_Exit_Algorithm_Implementation_List has no effect on PublishCRL behavior of Windows Server 2003, Windows Server 2008, Windows Server 2008 R2, Windows Server 2012, or Windows Server 2012 R2. If CRLs are disabled, certificates issued by the CA cannot be used for applications that require CRL−based revocation checking.

<23> Section 3.1.4.1.6: The Windows 2000 CA does not have a CRL table ; therefore, it does not create or update a CRL table entry.

<24> Section 3.1.4.1.6: The Microsoft CA for Windows uses a default clock skew (Config_CA_Clock_Skew_Minutes) of 10 minutes. The Microsoft CA defines this value in the registry as follows:

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services
\CertSvc\Configuration{CA Name}\ClockSkewMinutes  (REG_DWORD)
```

<25> Section 3.1.4.1.6: By default, the Config_Base_CRL_Overlap_Period and Config_Delta_CRL_Overlap_Period values are not defined. The Microsoft CA keeps these overlap periods in the following registry values:

```
HKLM\SYSTEM\CurrentControlSet\Services\CertSvc\Configuration\{CA Name}\
REG_SZ        CRLOverlapPeriod
REG_DWORD     CRLOverlapPeriodUnits
REG_SZ        CRLDeltaOverlapPeriod
REG_DWORD     CRLDeltaOverlapPeriodUnits
```

<26> Section 3.1.4.1.6: The Microsoft CA for Windows uses a default clock skew (Config_CA_Clock_Skew_Minutes) of 10 minutes. The Microsoft CA defines this value in the registry as follows:

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services\CertSvc\Configuration{CA
Name}\ClockSkewMinutes  (REG_DWORD)
```

<27> Section 3.1.4.1.6: Differing from [RFC3280] section 5, a Windows 2000 CA does not populate the CRL number extension.

<28> Section 3.1.4.1.6: A Windows 2000 CA does not create delta CRLs.

<29> Section 3.1.4.1.6: The Microsoft CA keeps this list of CDP locations in a registry multistring value.

```
HKLM\SYSTEM\CurrentControlSet\Services\CertSvc\Configuration\
{CA Name}\CRLPublicationURLs
```

A specific protocol method does not exist to manipulate this list. Instead, the Microsoft CA uses the typical registry manipulation tools.

The default values used by the Microsoft CA are a local path on the CA machine,

```
{SYSTEM}\CertSrv\CertEnroll\
```

a local path in the registry,

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\SystemCertificates\CA\CRLs\
```

and the Active Directory path,

```
ldap: ///CN={CAName}{CRLNameSuffix},CN={CAServerName},CN=CDP,CN=Public Key
Services,CN=Services,CN=Configuration,DC={contoso},DC=com
```

where:

- **SYSTEM** is replaced with the system directory of the CA machine, such as "C:\Windows\System32".

- **CAName** is replaced with the sanitized name of the CA, as defined in [MS-WCCE] sections 1.3.2.4 and 3.1.1.4.1.1.

- **{CRLNameSuffix}** is replaced with NULL if the CRL is signed by the first CA key (CA key with key index 0) and by "(n)" if the CRL is signed by any subsequent CA key, with {n} being an

integer equal to the identifier (Signing_Private_Key_Version_ID, as defined in [MS-WCCE] section 3.2.1.1.3) of the CA certificate private key.

- **CAServerName** is replaced with the name of the host on which the CA is running.

- **DC={contoso},DC=com** is replaced with the name space of the Active Directory domain in which the Microsoft CA is installed.

<30> Section 3.1.4.1.6: For the Microsoft CA, the error code will be in the form of a 2-byte WIN32 error code (as specified in [MS-ERREF] section 2.2), such as 0x2098, which means "Insufficient access rights to perform the operation". This will then be converted to an HRESULT (4 byte) error code (as specified in [MS-ERREF] section 2.1), such as 0x80072098. Note that the first 2 bytes, the "0x8007" portion of the HRESULT value, have nothing to do with the error condition and are determined by the Severity and Facility bits, as defined in [MS-ERREF] section 2.1.

<31> Section 3.1.4.1.6: The Microsoft CA publishes CRLs to LDAP paths in Active Directory as follows:

The path the server expects is

```
ldap: ///CN={CAName}{CRLNameSuffix},CN={CAServerShortName},CN=CDP,CN=Public Key
Services,CN=Services,CN=Configuration,DC={contoso},DC=com{CDPObjectClass}
```

where:

"{CAName}" is replaced with the sanitized name of the CA, as defined in [MS-WCCE] sections 1.3.2.4 and 3.1.1.4.1.1.

"{DeltaIndicator}" is replaced with NULL for a base CRL and "+" for a delta CRL.

"{CRLNameSuffix}" is replaced with NULL if the CRL is signed by the first CA key (CA key with key index 0) and by "(n)" if the CRL is signed by any subsequent CA key.

where "n" is replaced with an integer equal to the identifier (Signing_Private_Key_Version_ID, as defined in [MS-WCCE] section 3.2.1.1.2) of the CA certificate private key.

"{CAServerName}" is replaced with the name of the host on which the CA is running.

"DC={contoso},DC=com" is replaced with the DN of the **forest** root domain naming context (NC) of the Active Directory forest in which the Microsoft CA is installed.

The forest root domain NC is defined in section 1.1 of [MS-ADTS].

For example, the DN of the forest root domain NC of a forest called "corp.contoso.com" is "DC=corp,DC=contoso,DC=com".

{CDPObjectClass} is replaced with "?certificateRevocationList?base?objectClass=cRLDistributionPoint" for a base CRL and with "?deltaRevocationList?base?objectClass=cRLDistributionPoint" for a delta CRL.

The object class cRLDistributionPoint is as defined in [MS-ADSC]. The attribute certificateRevocationList is defined in [MS-ADA1].

The attribute deltaRevocationList is defined in [MS-ADA1].

Release: Friday, October 25, 2013

The deltaRevocationList attribute is not used by the Windows 2000 version of the CA. The Windows Server 2003, Windows Server 2008, Windows Server 2008 R2, Windows Server 2012, and Windows Server 2012 R2 versions of the CA use both base CRL and delta CRL attributes.

Within the certificateRevocationList or deltaRevocationList attribute, the CRL is encoded by using DER.

For any ldap:/// write operation, if the LDAP write operation returns an error that indicates the LDAP server is down or otherwise unavailable, the Microsoft Windows Server 2003, Windows Server 2008, Windows Server 2008 R2, Windows Server 2012, and Windows Server 2012 R2 CA will attempt to rebind (creating a new LDAP handle) and reattempt the LDAP write one time. The Microsoft CA in Windows 2000 does not perform this LDAP handle caching and single retry with a new LDAP handle.

<32> Section 3.1.4.1.6: A Windows Server 2003, Windows Server 2008, Windows Server 2008 R2, Windows Server 2012, or Windows Server 2012 R2 CA will perform this one-time retry logic for LDAP if the LDAP call returns one of the following ldap error codes: LDAP_SERVER_DOWN (0x51), or LDAP_UNAVAILABLE (0x34), or LDAP_TIMEOUT (0x55). The Windows 2000 CA does not perform this one-time LDAP retry logic.

<33> Section 3.1.4.1.6: The Windows 2000 CA does not have a CRL table. Therefore, it does not create or update data elements for a CRL table.

<34> Section 3.1.4.1.7: The Windows 2000 CA retrieves the most recent base CRL from the registry location HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\SystemCertificates\CA\CRLs\. The Windows Server 2003, Windows Server 2008, Windows Server 2008 R2, Windows Server 2012, and Windows Server 2012 R2 versions of the CA retrieve the most recent base CRL (CRLRawCRL) from the CRL table.

<35> Section 3.1.4.1.8: Windows allows serial numbers longer than 20 octets.

<36> Section 3.1.4.1.8: The parameter value 0xfffffffd is valid only on a Windows Server 2008, Windows Server 2008 R2, Windows Server 2012, or Windows Server 2012 R2 CA. If this value is used on a Windows Server 2003 CA, the CA fails with return code ERROR_INVALID_PARAMETER (0x80070057).

<37> Section 3.1.4.1.8: The parameter value 0xfffffffe is valid only on a Windows Server 2008, Windows Server 2008 R2, Windows Server 2012, or Windows Server 2012 R2 CA. If this value is used on a Windows Server 2003 CA, the CA fails with return code ERROR_INVALID_PARAMETER (0x80070057).

<38> Section 3.1.4.1.10: Windows Server 2003, Windows Server 2008, Windows Server 2008 R2, Windows Server 2012, and Windows Server 2012 R2 send the column identifiers as the following **DWORD** array.

```
{0x00001000, 0x00001010, 0x0000100b, 0x00001008, 0x0000100a,
 0x00002003, 0x0000101a, 0x00001022, 0x00001019, 0x00001018,
 0x0000101b,0x0000101c, 0x00001017, 0x00001001}
```

These identifiers correspond to the following columns in the Request table:

"Request.RequestID", "Request.RequesterName", "Request.SubmittedWhen", "Request.StatusCode", "Request.DispositionMessage", "CertificateTemplate", "Request.CommonName", "Request.EMail", "Request.OrgUnit", "Request.Organization", "Request.Locality", "Request.State", "Request.Country", "Request.RawRequest"

Windows 2000 Server sends the column identifiers as the following **DWORD** array.

```
{0x00001000, 0x0000100f, 0x00002004, 0x00002006, 0x00002007,
 0x00002011, 0x00002019, 0x00002010, 0x0000200f, 0x00002012
 0x00002013, 0x0000200e, 0x00002001}
```

These identifiers correspond to the following columns in the Request table:

"Request.RequestID", "Request.RequesterName", "SerialNumber", "NotBefore", "NotAfter", "CommonName", "EMail", "OrgUnit", "Organization", "Locality", "State", "Country", "RawCertificate"

<39> Section 3.1.4.1.10: Windows Server 2003, Windows Server 2008, Windows Server 2008 R2, Windows Server 2012, and Windows Server 2012 R2 send the column identifiers as the following **DWORD** array.

```
{0x00001000, 0x00001010, 0x00002006, 0x00002008, 0x00002009,
 0x00002003, 0x00002016, 0x0000201e, 0x00002015, 0x00002014,
 0x00002017, 0x00002018, 0x00002013, 0x00002001}
```

These identifiers correspond to the following columns in the Request table:

"Request.RequestID", "Request.RequesterName", "SerialNumber", "NotBefore", "NotAfter", "CertificateTemplate", "CommonName", "EMail", "OrgUnit","Organization", "Locality", "State", "Country", "RawCertificate"

Windows 2000 Server sends the column identifier as the following **DWORD** array.

```
{0x00001000, 0x0000100f, 0x00002004, 0x00002006, 0x00002007,
 0x00002011, 0x00002019, 0x00002010, 0x0000200f, 0x00002012,
 0x00002013, 0x0000200e, 0x00002001}
```

These identifiers correspond to the following columns in the Request table:

"Request.RequestID", "Request.RequesterName", "SerialNumber", "NotBefore", "NotAfter", "CommonName", "EMail", "OrgUnit","Organization", "Locality", "State", "Country", "RawCertificate"

<40> Section 3.1.4.1.10: Windows Server 2003, Windows Server 2008, Windows Server 2008 R2, Windows Server 2012, and Windows Server 2012 R2 send the column identifiers as the following **DWORD** array.

```
{0x00001000, 0x00001010, 0x0000100b, 0x00001008, 0x0000100a,
 0x00002003, 0x0000101a, 0x00001022, 0x00001019, 0x00001018,
 0x0000101b, 0x0000101c, 0x00001017, 0x00001001, }
```

These identifiers correspond to the following columns in the Request table:

"Request.RequestID", "Request.RequesterName", "Request.SubmittedWhen", "Request.StatusCode", "Request.DispositionMessage", "CertificateTemplate", "Request.CommonName", "Request.EMail",

"Request.OrgUnit", "Request.Organization", "Request.Locality", "Request.State", "Request.Country", "Request.RawRequest"

Windows 2000 Server sends the column identifiers as the following **DWORD** array.

```
{0x00001000, 0x0000100f, 0x0000100a, 0x00001009, 0x00001017,
 0x0000101f, 0x00001016, 0x00001015, 0x00001018, 0x00001019,
 0x00001014, 0x00001001}
```

These identifiers correspond to the following columns in the Request table:

"Request.RequestID", "Request.RequesterName", "Request.SubmittedWhen", "Request.DispositionMessage", "Request.CommonName", "Request.EMail", "Request.OrgUnit", "Request.Organization", "Request.Locality", "Request.State", "Request.Country", "Request.RawRequest"

<41> Section 3.1.4.1.10: Windows Server 2003, Windows Server 2008, Windows Server 2008 R2, Windows Server 2012, and Windows Server 2012 R2 send the column identifiers as the following **DWORD** array.

```
{0x00004000, 0x00004001, 0x00004002, 0x00004003}
```

These identifiers correspond to the following columns in the Extension table:

"ExtensionRequestId", "ExtensionName", "ExtensionFlags", "ExtensionRawValue"

Windows 2000 Server returns E_INVALIDARG for this value of the *iColumnSetDefault* parameter.

<42> Section 3.1.4.1.10: Windows Server 2003, Windows Server 2008, Windows Server 2008 R2, Windows Server 2012, and Windows Server 2012 R2 send the column identifiers as the following **DWORD** array.

```
{0x00003000, 0x00003001, 0x00003002}
```

These identifiers correspond to the following columns in the Attribute table:

"AttributeRequestId", "AttributeName", "AttributeValue"

Windows 2000 Server returns E_INVALIDARG for this value of the *iColumnSetDefault* parameter.

<43> Section 3.1.4.1.10: Windows Server 2003, Windows Server 2008, Windows Server 2008 R2, Windows Server 2012, and Windows Server 2012 R2 send the column identifiers as the following **DWORD** array.

```
{0x00005000, 0x00005001, 0x00005002, 0x00005003, 0x00005004,
 0x00005005, 0x00005006, 0x00005007, 0x00005008, 0x00005009,
 0x0000500a, 0x0000500b, 0x0000500c, 0x0000500d, 0x0000500e,
 0x0000500f, 0x00005010}
```

These identifiers correspond to the following columns in the CRL table:

"CRLRowId", "CRLNumber", "CrlMinBase", "CRLNameId", "CrlCount", "CRLThisUpdate", "CRLNextUpdate", "CRLThisPublish", "CRLNextPublish", "CRLEffective", "CRLPropagationComplete", "CRLLastPublish", "CRLPublishAttempts", "CRLPublishFlags", "CRLPublishStatusCode", "CRLPublishError", "CRLRawCRL"

Windows 2000 Server returns E_INVALIDARG for this value of the *iColumnSetDefault* parameter.

<44> Section 3.1.4.1.10: Windows Server 2003, Windows Server 2008, Windows Server 2008 R2, Windows Server 2012, and Windows Server 2012 R2 send the column identifiers as the following **DWORD** array.

```
{0x00001000, 0x00001010, 0x00002006, 0x00002008, 0x00002009,
 0x00002003, 0x00002016, 0x0000201e, 0x00002015,
 0x00002014, 0x00002017, 0x00002018, 0x00002013, 0x00002001,
 0x0000100d, 0x0000100e, 0x0000100f}
```

These identifiers correspond to the following columns in the Request table:

"Request.RequestID", "Request.RequesterName", "SerialNumber", "NotBefore", "NotAfter", "CertificateTemplate", "OrgUnit", "DomainComponent", "Organization", "Country", "CommonName", "Locality", "RawName", "RawCertificate", "Request.RevokedWhen", "Request.RevokedEffectiveWhen", "Request.RevokedReason"

Windows 2000 Server returns E_INVALIDARG for this value of the *iColumnSetDefault* parameter.

<45> Section 3.1.4.1.18: In Windows Server 2003, Windows Server 2008, Windows Server 2008 R2, Windows Server 2012, and Windows Server 2012 R2, the error is E_ACCESSDENIED (0x80070005). Windows 2000 does not return an error.

<46> Section 3.1.4.1.18: In Windows Server 2003, Windows Server 2008, Windows Server 2008 R2, Windows Server 2012, and Windows Server 2012 R2, the Windows CA defines local configuration to restrict programmatic access to some backup-related methods from a remote computer.

The Windows CA enforces this restriction based on the value of the following registry key: **HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\CertSvc\Configuration\{CA Name}\InterfaceFlags**.

Value	Meaning
0x00000000	The CA does not restrict access to the methods listed for the following servers.
0x00000040	The CA restricts access to the methods listed for the following servers.

<47> Section 3.1.4.1.19: In Windows Server 2003, Windows Server 2008, Windows Server 2008 R2, Windows Server 2012, and Windows Server 2012 R2, the error is E_ACCESSDENIED (0x80070005). Windows 2000 does not return an error.

<48> Section 3.1.4.1.19: In Windows Server 2003, Windows Server 2008, Windows Server 2008 R2, Windows Server 2012, and Windows Server 2012 R2, the Windows CA defines local configuration to restrict programmatic access to some backup-related methods from a remote computer.

The Windows CA enforces this restriction based on the value of the following registry key: **HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\CertSvc\Configuration\{C A Name}\InterfaceFlags**

Value	Meaning
0x00000000	The CA does not restrict access to the methods listed for the following servers.
0x00000040	The CA restricts access to the methods listed for the following servers.

<49> Section 3.1.4.1.20: In Windows Server, the Windows CA defines local configuration to restrict programmatic access to some backup-related methods from a remote computer.

The Windows CA enforces this restriction based on the value of the following registry key: **HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\CertSvc\Configuration\{C A Name}\InterfaceFlags**.

Value	Meaning
0x00000000	The CA does not restrict access to the methods listed for the following servers.
0x00000040	The CA restricts access to the methods listed for the following servers.

<50> Section 3.1.4.1.20: In Windows Server 2003 the error is ERROR_UNEXPECTED_ERROR (0x8000FFFF). In Windows Server 2008, Windows Server 2008 R2, Windows Server 2012, and Windows Server 2012 R2, the error is E_ACCESSDENIED (0x80070005). Windows 2000 does not return an error.

<51> Section 3.1.4.1.21: In Windows Server 2003, Windows Server 2008, Windows Server 2008 R2, Windows Server 2012, and Windows Server 2012 R2, the Windows CA defines local configuration to restrict programmatic access to some backup-related methods from a remote computer.

The Windows CA enforces this restriction based on the value of the following registry key: **HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\CertSvc\Configuration\{C A Name}\InterfaceFlags**

Value	Meaning
0x00000000	The CA does not restrict access to the methods listed for the following servers.
0x00000040	The CA does restrict access to the methods listed for the following servers.

<52> Section 3.1.4.1.21: In Windows Server 2003, the error is ERROR_UNEXPECTED_ERROR (0x8000FFFF). In Windows Server 2008, Windows Server 2008 R2, Windows Server 2012, and Windows Server 2012 R2, the error is E_ACCESSDENIED (0x80070005). Windows 2000 does not return an error.

<53> Section 3.1.4.1.22: In Windows Server 2003, Windows Server 2008, Windows Server 2008 R2, Windows Server 2012, and Windows Server 2012 R2, the Windows CA defines local configuration to restrict programmatic access to some backup-related methods from a remote computer.

The Windows CA enforces this restriction based upon the value of the following registry key:
**HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\CertSvc\Configuration\{C
A Name}\InterfaceFlags**

Value	Meaning
0x00000000	The CA does not restrict access to the methods listed for the following servers.
0x00000040	The CA restricts access to the methods listed for the following servers.

<54> Section 3.1.4.1.22: Windows 2000 does not return an error. In Windows Server 2003, the error is ERROR_UNEXPECTED_ERROR (0x8000FFFF). In Windows Server 2008, Windows Server 2008 R2, Windows Server 2012, and Windows Server 2012 R2, the error is E_ACCESSDENIED (0x80070005).

<55> Section 3.1.4.1.23: In Windows Server 2003, Windows Server 2008, Windows Server 2008 R2, Windows Server 2012, and Windows Server 2012 R2, the Windows CA defines local configuration to restrict programmatic access to some backup-related methods from a remote computer.

The Windows CA enforces this restriction based on the value of the following registry key:
**HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\CertSvc\Configuration\{C
A Name}\InterfaceFlags**.

Value	Meaning
0x00000000	The CA does not restrict access to the methods listed for the following servers.
0x00000040	The CA restricts access to the methods listed for the following servers.

<56> Section 3.1.4.1.23: In Windows Server 2003, the error is ERROR_UNEXPECTED_ERROR (0x8000FFFF). In Windows Server 2008, Windows Server 2008 R2, Windows Server 2012, and Windows Server 2012 R2, the error is E_ACCESSDENIED (0x80070005). Windows 2000 does not return an error.

<57> Section 3.1.4.1.24: In Windows Server 2008, Windows Server 2008 R2, Windows Server 2012, and Windows Server 2012 R2, the error is E_ACCESSDENIED (0x80070005). Windows Server 2003 and Windows 2000 do not return an error.

<58> Section 3.1.4.1.25: In Windows Server 2003, the error is ERROR_UNEXPECTED_ERROR (0x8000FFFF). In Windows Server 2008, Windows Server 2008 R2, Windows Server 2012, and Windows Server 2012 R2, the error is E_ACCESSDENIED (0x80070005). Windows 2000 does not return an error.

<59> Section 3.1.4.1.25: In Windows Server 2003, Windows Server 2008, Windows Server 2008 R2, Windows Server 2012, and Windows Server 2012 R2, the Windows CA defines local configuration to restrict programmatic access to some backup-related methods from a remote computer.

The Windows CA enforces this restriction based upon the value of the following registry key:**HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\CertSvc\Configuration
\{CA Name}\InterfaceFlags**

Value	Meaning
0x00000000	The CA does not restrict access to the methods listed for the following servers.
0x00000040	The CA restricts access to the methods listed for the following servers.

<60> Section 3.1.4.1.26: The Microsoft CA maintains local configuration to allow or prevent the importing of foreign certificates, regardless of the value of *dwFlags*. The configuration is stored in the registry at the location that is specified in the following code example. If the registry value is set to 1, the ImportCertificate method works as documented. If it is set to 0, the FLAG_ALLOW_IMPORT_FOREIGN flag that is passed as a parameter has no effect, and 0x800b0107 is returned.

```
HKLM\SYSTEM\CurrentControlSet\Services\CertSvc\Configuration\{CA Name}
\KRAFlags (REG_DWORD)
```

<61> Section 3.1.4.1.26: The Microsoft CA maintains local configuration to allow or prevent the importing of foreign certificates regardless of the value of *dwFlags*. The configuration is stored in the registry at the location specified in the following code example. If the registry value is set to 1, the ImportCertificate method works as documented. If it is set to 0, the FLAG_ALLOW_IMPORT_FOREIGN flag that is passed as a parameter does not have an effect, and 0x800b0107 is returned.

```
HKLM\SYSTEM\CurrentControlSet\Services\CertSvc\Configuration\{CA Name}
\KRAFlags (REG_DWORD)
```

<62> Section 3.1.4.1.27: In Windows Server 2008, Windows Server 2008 R2, Windows Server 2012, and Windows Server 2012 R2, the error is E_ACCESSDENIED (0x80070005). Windows Server 2003 and Windows 2000 do not return an error.

<63> Section 3.1.4.1.28: In Windows Server 2008, Windows Server 2008 R2, Windows Server 2012, and Windows Server 2012 R2, the error is E_ACCESSDENIED (0x80070005). Windows Server 2003 and Windows 2000 do not return an error.

<64> Section 3.1.4.2: The supported clients are Windows XP, Windows Vista, Windows 7, Windows 8, and Windows 8.1. The supported servers are Windows Server 2003, Windows Server 2008, Windows Server 2008 R2, Windows Server 2012, and Windows Server 2012 R2.

<65> Section 3.1.4.2: In Windows 2000, Windows Server 2003, Windows Server 2008, Windows Server 2008 R2, Windows Server 2012, and Windows Server 2012 R2, the error is E_ACCESSDENIED (0x80070005).

<66> Section 3.1.4.2: In Windows Server 2003, Windows Server 2008, Windows Server 2008 R2, Windows Server 2012, and Windows Server 2012 R2, the error is E_ACCESSDENIED (0x80070005). Windows 2000 does not return an error.

<67> Section 3.1.4.2.1: The Microsoft CA keeps this list in a registry multistring value.

```
HKLM\SYSTEM\CurrentControlSet\Services\CertSvc\Configuration\
{CA Name}\CRLPublicationURLs
```

A specific protocol method does not exist to manipulate this list. Instead, the Microsoft CA uses typical registry manipulation tools.

The default values that are used by the Microsoft CA are a local path on the CA machine,

```
{SYSTEM}\CertSrv\CertEnroll\
```

a local path in the registry,

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\SystemCertificates\CA\CRLs\
```

and the Active Directory path,

```
ldap: ///CN={CAName}{index},CN={CAServerName},CN=CDP,
CN=Public Key Services,CN=Services,CN=Configuration,DC={contoso},DC=com
```

where:

- **SYSTEM** is replaced with the system directory of the CA machine, such as "C:\Windows\System32".

- **CAName** is replaced with the sanitized name of the CA, as defined in [MS-WCCE] sections 1.3.2.4 and 3.1.1.4.1.1.

- **{CRLNameSuffix}** is replaced with NULL if the CRL is signed by the first CA key (the CA key that has a key index 0) and by "(n)" if the CRL is signed by any subsequent CA key, with {n} being an integer equal to the identifier (Signing_Private_Key_Version_ID, as defined in [MS-WCCE] section 3.2.1.1.3) of the CA certificate private key.

- **CAServerName** is replaced with the name of the host on which the CA is running.

- **DC={contoso},DC=com** is replaced with the namespace of the Active Directory domain in which the Microsoft CA is installed.

<68> Section 3.1.4.2.1: The Microsoft CA keeps this list in the following registry multistring value. Note that the value is the same as that specified in the preceding Windows behavior note.

```
HKLM\SYSTEM\CurrentControlSet\Services\CertSvc\Configuration\
{CA Name}\CRLPublicationURLs
```

<69> Section 3.1.4.2.5: This rule applies only to a Windows Server 2008, Windows Server 2008 R2, Windows Server 2012, or Windows Server 2012 R2 CA. In Windows 2000 and Windows Server 2003, a CA will not enforce that *cColumn* is greater than 0. Rather, when *cColumn* is equal to zero, it will set *pcColumn* equal to zero, *pctbColumnInfo->cb* equal to 0, *pctbColumnInfo->pb* will point to a zero-length item, and the function will return successfully.

<70> Section 3.1.4.2.14: The Windows CA uses subkeys that use the following key as a node path:

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\CertSvc\Configuration\<CA_
CN>
```

where <CA_CN> is replaced with the CN of the CA.

<71> Section 3.1.4.2.14: The Windows CA uses a registry value name under the registry key that is composed by adding the value of the pwszNodePath parameter to the registry key:

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\CertSvc\Configuration\<CA_
CN>
```

where <CA_CN> is replaced with the CA's common name (CN).

8 Change Tracking

This section identifies changes that were made to the [MS-CSRA] protocol document between the August 2013 and November 2013 releases. Changes are classified as New, Major, Minor, Editorial, or No change.

The revision class **New** means that a new document is being released.

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 or functionality.

- An extensive rewrite, addition, or deletion of major portions of content.

- The removal of a document from the documentation set.

- Changes made for template compliance.

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 **Editorial** means that the language and formatting in the technical content was changed. Editorial changes apply to grammatical, formatting, and style issues.

The revision class **No change** means that no new technical or language changes were introduced. The technical content of the document is identical to the last released version, but minor editorial and formatting changes, as well as updates to the header and footer information, and to the revision summary, may have been made.

Major and minor changes can be described further using the following change types:

- New content added.

- Content updated.

- Content removed.

- New product behavior note added.

- Product behavior note updated.

- Product behavior note removed.

- New protocol syntax added.

- Protocol syntax updated.

- Protocol syntax removed.

- New content added due to protocol revision.

- Content updated due to protocol revision.

- Content removed due to protocol revision.

- New protocol syntax added due to protocol revision.

- Protocol syntax updated due to protocol revision.

- Protocol syntax removed due to protocol revision.

- New content added for template compliance.

- Content updated for template compliance.

- Content removed for template compliance.

- Obsolete document removed.

Editorial changes are always classified with the change type **Editorially updated.**

Some important terms used in the change type descriptions are defined as follows:

- **Protocol syntax** refers to data elements (such as packets, structures, enumerations, and methods) as well as interfaces.

- **Protocol revision** refers to changes made to a protocol that affect the bits that are sent over the wire.

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

Section	Tracking number (if applicable) and description	Major change (Y or N)	Change type
2.1 Transport	69287 Updated section reference for remote server name DCOM setting.	N	Content updated.
3.2.4.1.3 ICertAdminD::ResubmitRequest (Opnum 5)	69297 Updated the name of the ICertRequestD interface to ICertAdminD.	Y	Content updated.

Release: Friday, October 25, 2013

9 Index

Release: Friday, October 25, 2013