[MS-OCSPA]:

Microsoft OCSP Administration Protocol

Intellectual Property Rights Notice for Open Specifications Documentation

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

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

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

Revision Summary

Date	Revision History	Revision Class	Comments
1/25/2008	0.1	Major	MCPP RSAT Initial Availability
3/14/2008	0.1.1	Editorial	Changed language and formatting in the technical content.
5/16/2008	0.1.2	Editorial	Changed language and formatting in the technical content.
6/20/2008	0.1.3	Editorial	Changed language and formatting in the technical content.
7/25/2008	0.1.4	Editorial	Changed language and formatting in the technical content.
8/29/2008	0.1.5	Editorial	Changed language and formatting in the technical content.
10/24/2008	1.0	Major	Updated and revised the technical content.
12/5/2008	2.0	Major	Updated and revised the technical content.
1/16/2009	3.0	Major	Updated and revised the technical content.
2/27/2009	4.0	Major	Updated and revised the technical content.
4/10/2009	4.0.1	Editorial	Changed language and formatting in the technical content.
5/22/2009	4.0.2	Editorial	Changed language and formatting in the technical content.
7/2/2009	4.1	Minor	Clarified the meaning of the technical content.
8/14/2009	4.1.1	Editorial	Changed language and formatting in the technical content.
9/25/2009	4.2	Minor	Clarified the meaning of the technical content.
11/6/2009	4.2.1	Editorial	Changed language and formatting in the technical content.
12/18/2009	4.2.2	Editorial	Changed language and formatting in the technical content.
1/29/2010	4.3	Minor	Clarified the meaning of the technical content.
3/12/2010	4.3.1	Editorial	Changed language and formatting in the technical content.
4/23/2010	4.4	Minor	Clarified the meaning of the technical content.
6/4/2010	4.4.1	Editorial	Changed language and formatting in the technical content.
7/16/2010	4.4.1	None	No changes to the meaning, language, or formatting of the technical content.
8/27/2010	4.4.1	None	No changes to the meaning, language, or formatting of the technical content.
10/8/2010	4.4.1	None	No changes to the meaning, language, or formatting of the technical content.
11/19/2010	4.4.1	None	No changes to the meaning, language, or formatting of the technical content.
1/7/2011	5.0	Major	Updated and revised the technical content.
2/11/2011	5.0	None	No changes to the meaning, language, or formatting of the technical content.
3/25/2011	5.0	None	No changes to the meaning, language, or formatting of the

Date	Revision History	Revision Class	Comments
			technical content.
5/6/2011	5.0	None	No changes to the meaning, language, or formatting of the technical content.
6/17/2011	5.1	Minor	Clarified the meaning of the technical content.
9/23/2011	5.1	None	No changes to the meaning, language, or formatting of the technical content.
12/16/2011	6.0	Major	Updated and revised the technical content.
3/30/2012	6.0	None	No changes to the meaning, language, or formatting of the technical content.
7/12/2012	6.0	None	No changes to the meaning, language, or formatting of the technical content.
10/25/2012	6.0	None	No changes to the meaning, language, or formatting of the technical content.
1/31/2013	6.0	None	No changes to the meaning, language, or formatting of the technical content.
8/8/2013	7.0	Major	Updated and revised the technical content.
11/14/2013	7.0	None	No changes to the meaning, language, or formatting of the technical content.
2/13/2014	7.0	None	No changes to the meaning, language, or formatting of the technical content.
5/15/2014	7.0	None	No changes to the meaning, language, or formatting of the technical content.
6/30/2015	8.0	Major	Significantly changed the technical content.
7/14/2016	9.0	Major	Significantly changed the technical content.

Table of Contents

1 Introduction

This document specifies the Microsoft OCSP Administration Protocol. The protocol consists of a set of **Distributed Component Object Model (DCOM)** interfaces that allow administrative tools to configure the properties of the **Online Responder**.

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

1.1 Glossary

This document uses the following terms:

access control entry (ACE): An entry in an **access control list (ACL)** that contains a set of user rights and a security identifier (SID) that identifies a principal for whom the rights are allowed, denied, or audited.

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

certificate: A certificate is a collection of attributes (1) and extensions that can be stored persistently. The set of attributes in a certificate can vary depending on the intended usage of the certificate. A certificate securely binds a public key to the entity that holds the corresponding private key. A certificate is commonly used for authentication (2) and secure exchange of information on open networks, such as the Internet, extranets, and intranets. Certificates are digitally signed by the issuing **certification authority (CA)** and can be issued for a user, a computer, or a service. The most widely accepted format for certificates is defined by the ITU-T X.509 version 3 international standards. For more information about attributes and extensions, see [RFC3280] and [X509] sections 7 and 8.

certificate revocation list (CRL): A list of **certificates** that have been revoked by the **certification authority (CA)** that issued them (that have not yet expired of their own accord). The list must be cryptographically signed by the **CA** that issues it. Typically, the certificates are identified by serial number. In addition to the serial number for the revoked certificates, the CRL contains the revocation reason for each certificate and the time the certificate was revoked. As described in [RFC3280], two types of CRLs commonly exist in the industry. Base CRLs keep a complete list of revoked certificates, while delta CRLs maintain only those certificates that have been revoked since the last issuance of a base CRL. For more information, see [X509] section 7.3, [MSFT-CRL], and [RFC3280] section 5.

certificate template: A list of attributes that define a blueprint for creating an X.509 **certificate**. It is often referred to in non-Microsoft documentation as a "certificate profile". A **certificate template** is used to define the content and purpose of a digital certificate, including issuance requirements (certificate policies), implemented X.509 extensions such as application policies, key usage, or extended key usage as specified in [X509], and enrollment permissions. Enrollment permissions define the rules by which a **certification authority (CA)** will issue or deny certificate requests. In Windows environments, **certificate templates** are stored as objects in the Active Directory and used by Microsoft enterprise **CAs**.

certification authority (CA): A third party that issues public key **certificates**. Certificates serve to bind public keys to a user identity. Each user and certification authority (CA) can decide whether to trust another user or CA for a specific purpose, and whether this trust should be transitive. For more information, see [RFC3280].

class identifier (CLSID): A GUID that identifies a software component; for instance, a DCOM object class (4) or a COM class.

cryptographic service provider (CSP): A software module that implements cryptographic functions for calling applications that generates digital signatures. Multiple **CSPs** may be installed. A **CSP** is identified by a name represented by a NULL-terminated Unicode string.

Distributed Component Object Model (DCOM): The Microsoft Component Object Model (COM) specification that defines how components communicate over networks, as specified in [MS-DCOM].

fully qualified domain name (FQDN): An unambiguous domain name (2) that gives an absolute location in the Domain Name System's (DNS) hierarchy tree, as defined in [RFC1035] section 3.1 and [RFC2181] section 11.

Interface Definition Language (IDL): The International Standards Organization (ISO) standard language for specifying the interface for remote procedure calls. For more information, see [C706] section 4.

interface identifier (IID): A GUID that identifies an interface.

Online Certificate Status Protocol (OCSP): The protocol specified in [RFC2560] that enables applications to determine the (revocation) state of an identified **certificate**.

Online Responder: Same meaning as **Online Responder Service**.

Online Responder Role: A list of administrator-defined rights or **ACLs** that define the capability of a given principal on an Online Responder. Online Responder Roles are specified in [CIMC-PP] section 5.2 and include administrator and enrollee.

Online Responder Service: The Microsoft implementation of an OCSP server. The **Online Responder Service** receives and processes OCSP requests from clients and has components for managing the online responder.

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

responder properties: The set of configuration information that specifies Online Responder request processing behavior across all revocation configurations.

revocation configuration: The set of configuration information specific to each CA for which the Online Responder is authorized to issue OCSP responses. It includes how the Online Responder obtains an OCSP response signing key and how it obtains revocation information. See section 3.2.1.1 for details on all the properties of a revocation configuration.

revocation provider: The set of configuration information, within the revocation configuration, that enables the **Online Responder Service** to determine the revocation status of a certificate.

security descriptor: A data structure containing the security information associated with a securable object. A **security descriptor** identifies an object's owner by its security identifier (SID). If access control is configured for the object, its **security descriptor** contains a discretionary access control list (DACL) with SIDs for the security principals who are allowed or denied access. Applications use this structure to set and query an object's security status. The **security descriptor** is used to guard access to an object as well as to control which type of auditing takes place when the object is accessed. The **security descriptor** format is specified in [MS-DTYP] section 2.4.6; a string representation of **security descriptors**, called SDDL, is specified in [MS-DTYP] section 2.5.1.

security principal name (SPN): The name that identifies a security principal (for example, machinename$@domainname for a machine joined to a domain or username@domainname for a user). Domainname is resolved using the Domain Name System (DNS).

signing certificates: The **certificate** that represents the identity of an entity (for example, a **certification authority (CA)**, a web server or an S/MIME mail author) and is used to verify signatures made by the **private key** of that entity. For more information, see [RFC3280].

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

1.2 References

Links to a document in the Microsoft Open Specifications library point to the correct section in the most recently published version of the referenced document. However, because individual documents in the library are not updated at the same time, the section numbers in the documents may not match. You can confirm the correct section numbering by checking the Errata.

1.2.1 Normative References

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

[C706] The Open Group, "DCE 1.1: Remote Procedure Call", C706, August 1997, https://www2.opengroup.org/ogsys/catalog/c706

[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

[FIPS140] FIPS PUBS, "Security Requirements for Cryptographic Modules", FIPS PUB 140, December 2002, http://csrc.nist.gov/publications/fips/fips140-2/fips1402.pdf

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

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

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

[MS-OCSP] Microsoft Corporation, "Online Certificate Status Protocol (OCSP) Extensions".

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

[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

[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

[RFC2560] Myers, M., Ankney, R., Malpani, A., Glaperin, S., and Adams, C., "X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP", RFC 2560, June 1999, http://www.ietf.org/rfc/rfc2560.txt

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

[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.rfc-editor.org/rfc/rfc4120.txt

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/

[HOWARD] Howard, M., "Writing Secure Code", Microsoft Press, 2002, ISBN: 0735617228.

1.3 Overview

The Microsoft OCSP Administration Protocol consists of a set of DCOM interfaces [MS-DCOM] that allows administrative tools to configure the properties of a responder.

A responder is a server implementation of the **Online Certificate Status Protocol (OCSP)**. A responder can be configured to provide revocation information for **certificates** issued by one or more **certificate authorities (CAs)** by creating a **revocation configuration** for each CA key. A responder also has properties that apply generically across all revocation configurations. These properties are sometimes referenced as "responder-wide" properties or simply **responder properties**.

Using this protocol, administrative tools can perform such functions as getting or setting responder properties, creating and removing revocation configurations, and retrieving **signing certificates** from a responder.

The participants in this protocol are as follows:

- Online Responder computer.
- Administrator computer: A client computer that performs remote configuration or administration tasks on the Online Responder computer.

The protocol uses the IOCSPAdminD DCOM interface, which offers the 10 methods documented in the following sections. These methods allow the administrator to set and retrieve properties, set and retrieve security information, and to test whether the service is responding.

1.4 Relationship to Other Protocols

The Microsoft OCSP Administration Protocol depends on the DCOM Remote Protocol [MS-DCOM]. Microsoft DCOM negotiates its authentication method using the Generic Security Services (GSS) API

[RFC2478]. Either NT LAN Manager (NTLM) [MS-NLMP], or Kerberos [RFC4120] and [MS-KILE], can be selected as the authentication method.

No other Microsoft Windows protocol directly depends on the Microsoft OCSP Administration Protocol. This protocol is designed to manage a responder that implements the OCSP [RFC2560].

1.5 Prerequisites/Preconditions

An Online Responder requires at least one CA certificate or delegated certificate, as defined in [RFC2560], to sign responses to certificate status requests from clients.

1.6 Applicability Statement

No higher-level protocol would benefit from using this protocol as a component. This protocol is designed to be used directly by applications and tools that require the capability to administer a responder.

1.7 Versioning and Capability Negotiation

None.

1.8 Vendor-Extensible Fields

This protocol uses responder properties as defined in section 3.2.1.2. Vendors can define their own custom responder properties using the SetOCSPProperty method as defined in section 3.2.4.1.2.

1.9 Standards Assignments

No standard assignments have been received for the Microsoft OCSP Administration Protocol described in this protocol specification.

This protocol uses the following RPC UUID.

Parameter	Value
RPC Interface UUID for IOCSPAdminD	{784b693d-95f3-420b-8126-365c098659f2}

2 Messages

The following sections specify how Microsoft OCSP Administration Protocol messages are transported and Microsoft OCSP Administration Protocol message syntax.

2.1 Transport

The Distributed Component Object Model (DCOM) Remote Protocol [MS-DCOM] is used as the transport protocol.

This protocol uses DCOM to create and use DCOM object references to server objects.

Microsoft OCSP Administration Protocol clients initialize a connection to the server by creating and executing a DCOM activation request. As a result of this DCOM activation, the Microsoft OCSP Administration Protocol 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 an application using DCOM passes to the DCOM client as part of the initial activation request. Below are the values that the Microsoft OCSP Administration Protocol sends to the DCOM layer.

General DCOM settings:

- Server name (the application-supplied server name as specified in [MS-DCOM] section 3.2.4.2). The Microsoft OCSP Administration Protocol client sends the name of the Microsoft OCSP Administration Protocol server.
- **Class identifier (CLSID)** of the object requested. This value is 6d5ad135-1730-4f19-a4eb-3f78e7c976bb.
- **Interface identifier(s) (IID)** of interface(s) requested.
  - IOCSPAdminD: 784b693d-95f3-420b-8126-365c098659f2 (see section 1.9). Security settings ([MS-DCOM] section 3.2.4.1.1.2)
- Security provider: RPC_C_AUTHN_GSS_NEGOTIATE (9) for a remote server and RPC_C_AUTHN_DEFAULT (0xFFFFFFFF) for a server on the local machine.
- Authentication level: RPC_C_AUTHN_LEVEL_PKT_PRIVACY (6).

  When the security provider has the value of RPC_C_AUTHN_GSS_NEGOTIATE, 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.

  When the security provider has the value of RPC_C_AUTHN_DEFAULT, DCOM will choose an authentication method as specified in [MS-DCOM] section 3.2.4.2.
- 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 security provider means that the ORPC call uses the identity and credentials of the higher-layer application.

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

2.2 Common Data Types

In addition to RPC base types and definitions specified in [C706] and [MS-RPCE], additional data types are defined in this section.

2.2.1 Common Structures and Data Types

This section defines the structures used by the Microsoft OCSP Administration Protocol. These structures are used when using interface methods to perform various operations on the server and as part of the server's response.

2.2.1.1 CERTTRANSBLOB

The CERTTRANSBLOB type is implemented as specified in [MS-WCCE] section 2.2.2.2.

2.2.1.1.1 CERTTRANSBLOB Marshaling

Within the Microsoft OCSP Administration Protocol, the CERTTRANSBLOB is used to send a **security descriptor** from the client to the responder and to retrieve same. The marshaling of security descriptors into a CERTTRANSBLOB is documented in [MS-DTYP] section 2.4.6.

2.2.1.2 BSTR

The BSTR type is implemented as specified in [MS-OAUT] section 2.2.23.2.

2.2.1.3 VARIANT

The VARIANT type is implemented as specified in [MS-OAUT] section 2.2.29.2.

3 Protocol Details

The Microsoft OCSP 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 code. The primary usage of this protocol is Online Responder management. Except where specified in the following section, the protocol is a single message followed by a single reply.

3.1 IOCSPAdminD Client Details

3.1.1 Abstract Data Model

None.

3.1.2 Timers

None.

3.1.3 Initialization

Microsoft OCSP Administration Protocol is initialized by instantiation of DCOM objects.

3.1.4 Message Processing Events and Sequencing Rules

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 can 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. Return codes are documented in [MS-ERREF].

3.1.5 Timer Events

None.

3.1.6 Other Local Events

None.

3.2 IOCSPAdminD Server Details

3.2.1 Abstract Data Model

3.2.1.1 RevocationConfigurationList

The server implementing the Microsoft OCSP Administration Protocol must maintain a list of revocation configurations, each of which represents a CA certificate for which the server can provide an OCSP response. This list will be referenced as RevocationConfigurationList.

Each revocation configuration in the list has a set of properties referenced as RevocationConfigurationProperties.

Within RevocationConfigurationProperties, there is a property called "Provider" that comprises a set of properties referenced as RevocationProviderProperties.

Unless a property is otherwise specified as optional in the table below, it is a required property.

Property name	Meaning
RevocationConfigurationId	A string value that uniquely identifies the revocation configuration.
CACertificate	Binary data that contains the certificate holding the CA public key corresponding to this revocation configuration. This is the key of the CA that issued the certificate whose status is being checked.
HashAlgorithmId	An optional string property whose value is the name of the hash algorithm that is used in signing the OCSP responses generated by the responder.
SigningFlags	An optional unsigned integer value that specifies flags to control the selection of the **private key** to be used for signing OCSP responses for this revocation configuration.
SigningCertificate	An optional binary value that contains the certificate corresponding to the private key used for signing OCSP responses for this revocation configuration.
ProviderCLSID	A string property whose value is the string representation of the class identifier (CLSID) for the **revocation provider** COM server that is used by this revocation configuration to check the status of a certificate.
Provider	List of revocation provider properties in section 3.2.1.1.1.
SigningCertificateTemplate	An optional string property whose value is the common name of the **certificate template** [MS-CRTD] to be used by the responder (if configured) to create a certificate request for the signing certificate.
CAConfig	An optional string property whose value is the configuration string for the CA [MS-WCCE] to which the responder (if configured) submits the certificate request for the signing certificate.
LocalRevocationInformation	Optional binary value that contains the **certificate revocation list (CRL)** to be used for revocation checking at the responder for a particular revocation configuration.
KeySpec	The key specification indicates whether the key bound to the configuration is used for encryption or for signing content. This property is not set directly using the SetCAConfigInformation method. Rather, a SigningCertificate is assigned, and the KeySpec value returned with the GetCAConfigInformation is based on the SigningCertificate. (Optional)
CSPName	The **cryptographic service provider (CSP)** used by the responder to sign OCSP responses for this revocation configuration. (Optional)
ErrorCode	The status of this revocation configuration. (Optional)
ReminderDuration	The reminder duration is expressed as a percentage of the signing certificate validity period and defines the time at which the responder will notify the administrator that the signing certificate is nearing the end of its lifetime. The default value is 90, but this value can be modified. (Optional)

3.2.1.1.1 RevocationProviderProperties

The following table describes the properties for the revocation provider referenced as "provider" in the following list and their meanings.

Property name	Meaning
CrlUrlTimeOut	An optional value that represents the time-out in milliseconds that the revocation provider must wait before it times out while trying to retrieve the CRL for which it is configured.
BaseCrlUrls	An optional list of strings where each string value represents a Uniform Resource Identifier (URI) from where a base CRL can be obtained.
DeltaCrlUrls	An optional list of strings where each string value represents a URI from where a delta CRL can be obtained.

Property name	Meaning
BaseCRL	The Base CRL loaded by the revocation provider. (Optional)
DeltaCRL	The Delta CRL loaded by the revocation provider. (Optional)
RevocationErrorCode	The status of this revocation provider in terms of its ability to provide revocation information. (Optional)
IssuedSerialNumbersDirectories	An array of strings, each of which specifies a UNC or local file path. These directories are used by the CA to record the serial numbers of certificates that have been issued.

3.2.1.2 ResponderProperties

Apart from revocation configuration properties, the server must also maintain a list of properties pertaining to service operation. This list is called ResponderProperties. Each property is described in the following table.

Unless a property is otherwise specified as optional in the following table, it is a required property.

Property name	Meaning
AuditFilter	An optional set of flags that identify the responder events for which the security audit is performed.
ArrayController	An optional string containing the name of the machine designated as Array controller, among the list of machines that have the **Online Responder Service** running with the same configuration information.<1>
ArrayMembers	An optional list of strings containing names of the machines that have the Online Responder Service running with the same configuration information.<2>
NumOfThreads	An optional integer value that specifies the maximum number of simultaneous OCSP requests [MS-OCSP] that can be served by the Online Responder Service.
MaxNumOfCacheEntries	An optional integer value that specifies the maximum number of OCSP responses [MS-OCSP] cached by the responder.
LogLevel	An optional flag that specifies the level of information that is to be communicated to the system (application eventlog channel) as part of operations being performed on the service.
Debug	An optional value that specifies whether the tracing for errors on the responder is enabled or not.
EnrollPollInterval	An optional value that specifies the interval at which the responder is to attempt to enroll for a signing certificate (for signing OCSP responses).
RequestFlags	An optional set of flags that controls how the OCSP requests [MS-OCSP] are processed on the server.
MaxIncomingMessageSize	An optional integer value that specifies the maximum size of the OCSP request [MS-OCSP], in bytes, that is allowed to be processed on the server.
NumOfBackendConnections	An optional integer value that specifies the maximum number of connections that can be created by the web server to the responder.
RefreshRate	An optional value that specifies the interval at which the web server attempts to contact the responder to obtain the latest revocation configuration information.
MaxAge	An optional integer value that specifies the value for the HTTP max-age cache-control directive [RFC2616] as part of the OCSP response.
ISAPIDebug	An optional value that specifies whether or not the tracing for errors on the web server is enabled.
MaxNumOfRequestEntries	An optional value that specifies the maximum number of requests that can be included in the **requestList** field of the OCSPRequest structure ([RFC2560] section 4.1.1). The default value of MaxNumOfRequestEntries is 1.

3.2.1.3 Online Responder Permissions

The responder SHOULD store a list of permissions or access rights. The possible permissions values include Read and Administer. Read is defined as providing the caller with read access to the configuration information and properties of the responder. Administer access is defined as providing the caller with read and update access to the configuration information and properties of the responder.<3>

The responder security information is configured with the SetSecurity method and is retrieved with the GetSecurity method.

The responder SHOULD enforce that the caller of any method specified in section 3.2.4 possesses specific permissions.<4>

3.2.2 Timers

None.

3.2.3 Initialization

Interface initialization: On startup, the responder MUST ensure that remote clients have permissions to activate and call DCOM objects. Subsequently, DCOM object and interface initialization is performed by the DCOM object exporter in response to an activation request from the DCOM client. The OCSP 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 OCSP Administration Protocol client can use this client object reference to make calls to the OCSP 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.

Cryptographic initialization: The responder MUST have access to the signing certificate private keys for each revocation configuration.

3.2.4 Message Processing Events and Sequencing Rules

3.2.4.1 IOCSPAdminD

The IOCSPAdminD interface provides an RPC interface for a client to manage an Online Responder.

IOCSPAdminD interface inherits the IUnknown interface.

The version number for IUnknown is 1.0. The UUID for the IOCSPAdminD interface is: "784b693d-95f3-420b-8126-365c098659f2". Method opnum field values start with 3; opnum values 0 through 2 represent the IUnknown methods: QueryInterface, AddRef, and Release methods inherited from [MS-DCOM].

Each method MUST NOT throw exceptions.

Methods in RPC Opnum Order

Method	Description
GetOCSPProperty	Opnum: 3
SetOCSPProperty	Opnum: 4
GetCAConfigInformation	Opnum: 5

Method	Description
SetCAConfigInformation	Opnum: 6
GetSecurity	Opnum: 7
SetSecurity	Opnum: 8
GetSigningCertificates	Opnum: 9
GetHashAlgorithms	Opnum: 10
GetMyRoles	Opnum: 11
Ping	Opnum: 12

3.2.4.1.1 GetOCSPProperty (Opnum 3)

This method retrieves the value of a responder property from the Online Responder Service.

```
HRESULT GetOCSPProperty(
  [in, ref] const BSTR bstrEntryName,
  [out, ref] VARIANT* pEntryValue
);
```

bstrEntryName: A BSTR that specifies the name of the property to retrieve. The Unicode string value SHOULD be one of the values listed in ResponderProperties or one of the following values.

Property name	Meaning
CAEntries	A list of strings containing the RevocationConfigurationId corresponding to each configured revocation configuration in RevocationConfigurationList.
AllEntries	A list of all the configured properties in the list ResponderProperties and all the revocation configuration properties for all revocation configurations in RevocationConfigurationList.

pEntryValue: A pointer to a VARIANT. The data returned is the value of the property referenced by *bstrEntryName*. See the following table for the processing rules that apply to the *bstrEntryName* values. Other, vendor-defined *bstrEntryName* values, not defined in the following table, MAY be used, as described in the processing rules that follow the table.

Property name	Processing rule for data returned
AuditFilter	The **vt** member of the VARIANT referenced by *pEntryValue* MUST be set to VT_I4, and the **lVal** member MUST be either 0 or a bitwise OR of the following values. Flag value – Meaning 0x00000000 – Nothing is Audited. 0x00000001 – Audit start/stop of the service. 0x00000002 – Audit changes to the revocation configurations on the responder. 0x00000004 – Audit OCSP requests received by the responder. 0x00000008 – Audit changes to the security descriptor on the responder.
ArrayController	The **vt** member of the VARIANT referenced by *pEntryValue* SHOULD be set to VT_BSTR, and the **bstrVal** member SHOULD be BSTR for the Unicode string value of the Domain Name System (DNS) name of the machine designated as Array controller for the array of responder machines.
ArrayMembers	The **vt** member of the VARIANT referenced by *pEntryValue* SHOULD be set to VT_ARRAY | VT_BSTR, and the **pArray** member SHOULD reference a single dimension safearray. The number of elements of the safearray referenced by **pArray** SHOULD be equal to the number of machines running Online Responder Service with the same configuration information. For each machine, there SHOULD be an element in the safearray referenced by **pArray** containing the BSTR for Unicode string value of the **FQDN** of the machine.
NumOfThreads	The **vt** member of the VARIANT referenced by *pEntryValue* MUST be set to

Property name	Processing rule for data returned
	VT_I4, and the **lVal** member MUST be set to the maximum number of simultaneous OCSP requests [MS-OCSP] that can be served by the Online Responder Service. <5>
MaxNumOfCacheEntries	The **vt** member of the VARIANT referenced by *pEntryValue* MUST be set to VT_I4, and the **lVal** member MUST be the maximum number of OCSP responses that can be cached by the responder.
CAEntries	The **vt** member of the VARIANT referenced by *pEntryValue* SHOULD be set to VT_ARRAY | VT_BSTR, and the **pArray** member SHOULD reference a single dimension safearray. The number of elements of the safearray reference by **pArray** SHOULD be equal to the number of entries in RevocationConfigurationList. For each revocation configuration in RevocationConfigurationList, there SHOULD be an element containing the BSTR for the Unicode string value of the RevocationConfigurationId.
LogLevel	The **vt** member of the VARIANT referenced by *pEntryValue* SHOULD be set to VT_I4, and the **lVal** member SHOULD be set to the integer value that specifies the level of information to be communicated to the system (application eventlog channel) as part of operations being performed on the service.<6>
Debug	The **vt** member of the VARIANT referenced by *pEntryValue* SHOULD be set to VT_I4, and the **lVal** member SHOULD be an integer value that specifies whether tracing for errors on the responder is enabled or not.<7>
EnrollPollInterval	The **vt** member of the VARIANT referenced by *pEntryValue* SHOULD be set to VT_I4, and the **lVal** member SHOULD be set to the integer value that specifies the frequency (in number of hours) with which the responder will attempt to enroll for a signing certificate (for signing OCSP responses).<8>
RequestFlags	The **vt** member of the VARIANT referenced by *pEntryValue* SHOULD be set to VT_I4, and the **lVal** member SHOULD be either 0 or the following value. Flag value – Meaning 0x00000001:Responder MUST reject OCSP requests that have signatures on them.
MaxIncomingMessageSize	The **vt** member of the VARIANT referenced by *pEntryValue* SHOULD be set to VT_I4, and the **lVal** member SHOULD be set to the integer value that specifies the maximum size of the OCSP request [MS-OCSP], in bytes, that is allowed to be processed on the server.
NumOfBackendConnections	The **vt** member of the VARIANT referenced by *pEntryValue* SHOULD be set to VT_I4, and the **lVal** member SHOULD be set to the integer value that specifies the maximum number of connections that can be created by the web server to the Online Responder Service. <9>
RefreshRate	The **vt** member of the VARIANT referenced by *pEntryValue* SHOULD be set to VT_I4, and the **lVal** member SHOULD be set to the integer value that specifies the frequency (in number of milliseconds) with which the web server will attempt to contact the Online Responder Service to obtain the latest revocation configuration information.
MaxAge	The **vt** member of the VARIANT referenced by *pEntryValue* SHOULD be set to VT_I4, and the **lVal** member SHOULD be set to the integer value that specifies the value for the HTTP max-age cache-control directive [RFC2616] as part of the OCSP response.
ISAPIDebug	The **vt** member of the VARIANT referenced by *pEntryValue* SHOULD be set to VT_I4, and the **lVal** member SHOULD be set to the integer value that specifies whether the tracing for errors on the web server is enabled or not.<10>
MaxNumOfRequestEntries	The **vt** member of the VARIANT referenced by *pEntryValue* SHOULD be set to VT_I4, and the **lVal** member SHOULD be set to the integer value that specifies the maximum number of requests that can be included in the **requestList** field of the OCSPRequest structure ([RFC2560] section 4.1.1).<11>
AllEntries	The **vt** member of the VARIANT MUST be set to VT_ARRAY | VT_VARIANT, and the **pArray** member MUST reference a two-dimensional safearray. The number of elements in the second dimension (signifying the number of columns) of the safearray referenced by **pArray** MUST be 2. The number of elements in the first dimension (signifying the number of rows) of the safearray referenced by **pArray** MUST be set to the sum of the number of entries in ResponderProperties

Property name	Processing rule for data returned
	and the number of entries in the RevocationConfigurationList. For each property in ResponderProperties, the first column of the row MUST be a VARIANT with **vt** member as VT_BSTR and the **bstrVal** member MUST be BSTR for the Unicode string value of the name of the property. The second column of the row MUST be a VARIANT with the value defined in this table, corresponding to the name of the property. For each revocation configuration in RevocationConfigurationList, the first column of the row MUST be a VARIANT with **vt** member as VT_BSTR and the **bstrVal** member MUST be BSTR for the Unicode string value of RevocationConfigurationId. The second column of the row MUST be a VARIANT with the value defined in section 3.2.4.1.3.

The following additional processing rules apply:

- If the value of *bstrEntryName* is not the same as one of the values specified in the preceding list or of a vendor-defined property, or if the property with the same name is not yet configured on the responder, the server MUST fail. The error code SHOULD be 0x80070002.
- If the value of *bstrEntryName* corresponds to a vendor-defined property, the server MAY return the value as a VARIANT containing data of the type integer, string, date, or binary object. Otherwise, for *bstrEntryName* values that do not correspond to the previous list, the server responds as if the property were not yet configured on the responder.<12>

3.2.4.1.2 SetOCSPProperty (Opnum 4)

This method configures the value of a responder property on the server.

```
HRESULT SetOCSPProperty(
  [in, ref] const BSTR bstrEntryName,
  [in, ref] const VARIANT* pEntryValue
);
```

bstrEntryName: A BSTR that specifies the name of the property to set. The Unicode string value SHOULD be one of the property name values listed in ResponderProperties.

pEntryValue: A pointer to VARIANT data.

The following processing rules apply:

1. The vt member of the VARIANT referenced by pEntryValue SHOULD match the type specified in section 3.2.4.1.1 for the property corresponding to the bstrEntryName value.<13>
2. If the **vt** member of the VARIANT referenced by *pEntryValue* is VT_EMPTY and the server has a property configured with the same name as the value of *bstrEntryName*, the server MUST delete the property identified by *bstrEntryName* and return success.
3. If the **vt** member of the VARIANT referenced by *pEntryValue* is VT_EMPTY and the server does not have a property configured with the same name as the value of *bstrEntryName*, the server MUST return an error. The error code SHOULD be 0x80070002.
4. If *bstrEntryName* matches one of the properties specified in section 3.2.1.2:
    1. If the server has a property configured with the same name, the server MUST replace the existing value with the value specified in the VARIANT data referenced by *pEntryValue*.
    2. If the server does not have a property configured with the same name, the server MUST store the property and value specified in the VARIANT data referenced by *pEntryValue* in its configuration.

5. If *bstrEntryName* is NULL or empty, the server SHOULD return the error E_INVALIDARG (0x80000003L).

6. If *bstrEntryName* is not empty and does not match one of the properties specified in section 3.2.1.2, the server SHOULD store the property and value specified in the VARIANT data referenced by *pEntryValue* in its configuration. The values "CAEntries" and "AllEntries" SHOULD NOT be used for bstrEntryName because of their special treatment by the GetOCSPProperty method.

3.2.4.1.3 GetCAConfigInformation (Opnum 5)

The GetCAConfigInformation method retrieves all the properties associated with a particular revocation configuration.

```
HRESULT GetCAConfigInformation(
  [in, ref] const BSTR bstrCAId,
  [out, ref] VARIANT* pEntryValue
);
```

bstrCAId: A BSTR that specifies the RevocationConfigurationId for the revocation configuration whose properties are to be retrieved.

pEntryValue: A pointer to a VARIANT data type that contains the names and values of all configured revocation configuration properties.

The following processing rules apply:

1. The server MUST look in the RevocationConfigurationList for the revocation configuration whose RevocationConfigurationId value is the same as the *bstrCAId* value. This is a case-insensitive lookup. If the revocation configuration is not found, then the responder MUST fail the request. The error code SHOULD be 0x800710d8.

2. Otherwise, for the revocation configuration identified by *bstrCAId*, the server MUST return the properties associated with it as the value of *pEntryValue*. The server MUST set the **vt** member of the VARIANT referenced by *pEntryValue* to VT_ARRAY | VT_VARIANT. The pArray member of the VARIANT referenced by *pEntryValue* MUST reference a two-dimensional safearray. The number of elements in the second dimension (signifying the number of columns) of the safearray referenced by pArray MUST be 2. The number of elements in the first dimension (signifying the number of rows) of the safearray referenced by pArray MUST be set to the number of properties for the revocation configuration. For each revocation configuration property, the first column of the row MUST be a VARIANT with **vt** member set to VT_BSTR, and the **bstrVal** member set to the BSTR for the Unicode string value of the name of the property. The second column of the row MUST be a VARIANT with value defined in the following table, in the row corresponding to the name of the property.

Property name	Processing rule
CACertificate	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 MUST be equal to the length in bytes of the ASN.1 DER encoding of the CA certificate for which this revocation configuration is configured.
HashAlgorithmId	The **vt** member of the VARIANT MUST be set to VT_BSTR, and the **bstrVal** member MUST be the BSTR for the Unicode string value of the hash algorithm used by the responder to sign OCSP responses for this revocation configuration.<14>
SigningFlags	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. 0x00000002 - The server is configured to use the CA certificate for this

Property name	Processing rule
	revocation configuration to sign the OCSP responses. 0x00000010 - The revocation configuration is configured to look for an OCSP certificate that has the designated OCSP signing certificate enhanced key usage in its extension (see OCSP [RFC2560]). 0x00000020 - The revocation configuration is configured to require that an OCSP signing certificate be designated manually by setting the "SigningCertificate" property. 0x00000004 - When a delegated signing certificate (as defined in [RFC2560]) is used for generating responses, the revocation configuration is configured to use the renewed signing certificate automatically, whenever such a certificate becomes available at the responder machine. Renewal is described in [MS-WCCE]. 0x00000040 - The responder is configured to include the signing public key hash in responses for this revocation configuration. (See [RFC2560] for OCSP server identifier details.) 0x00000080 - The responder is configured to include the signing certificate subject in responses for this revocation configuration. (See [RFC2560] for OCSP server identifier details.) 0x00000100 - The responder is configured to accept a nonce in the request. (See [RFC2560] nonce extensions in the OCSP request.) 0x00000001 – For this revocation configuration, the responder is configured to silently acquire the private key associated with the signing certificate identified by the "SigningCertificate" property. 0x00000008 - If this value is set, the responder is configured only to use signing certificates issued by the same cryptographic key as the CA for which this revocation configuration is configured. 0x00000200 – For this revocation configuration, the responder is configured to enroll for a signing certificate from the CA defined by the property "CAConfig" using the certificate template defined by the property "SigningCertificateTemplate".
SigningCertificate	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 referenced by pArray MUST be equal to the length in bytes of the ASN.1 DER encoding of the signing certificate used by the responder to sign OCSP responses for this revocation configuration.
ErrorCode	The **vt** member of the VARIANT MUST be set to VT_I4, and the **lVal** member MUST be the HRESULT DWORD value denoting the status of this revocation configuration. A value of 0 implies that this revocation configuration is properly configured with an OCSP signing certificate and that the cryptographic key pair associated with the signing certificate is accessible and ready to use for signing OCSP responses. See [MS-ERREF] for a list of the possible error codes.
CAConfig	The **vt** member of the VARIANT MUST be set to VT_BSTR, and the **bstrVal** member MUST be the BSTR for the Unicode string for the CA configuration string [MS-WCCE] to which the responder submits the certificate request for the signing certificate.
SigningCertificateTemplate	The **vt** member of the VARIANT MUST be set to VT_BSTR, and the **bstrVal** member MUST be the BSTR for the Unicode string for the common name of the certificate template [MS-CRTD] to be used by the responder to create a certificate request for signing certificate.
LocalRevocationInformation	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 MUST be equal to the length in bytes of the ASN.1 encoded CRL to be used for local revocation checking at the responder for a particular revocation configuration.
CSPName	The **vt** member of the VARIANT MUST be set to VT_BSTR, and the **bstrVal** member MUST be the BSTR for the Unicode string value of the cryptographic service provider (CSP) used by the responder to sign OCSP responses for this revocation configuration.
KeySpec	The **vt** member of the VARIANT MUST be set to VT_I4, and the **lVal** member MUST be the key Specification. The possible values are based on the

Property name	Processing rule
	SigningCertificate and the following rules: Flag value – Meaning 0x00000000 - The asymmetric key-pair associated with the SigningCertificate and used for signing could be either an encryption key or a signing key type. 0x00000001 - The asymmetric key-pair associated with the SigningCertificate and used for signing is an encryption key type. 0x00000002 - The asymmetric key-pair associated with the SigningCertificate and used for signing is a signing key type.
ProviderCLSID	The **vt** member of the VARIANT MUST be set to VT_BSTR, and the **bstrVal** member must be the BSTR for the Unicode string representation of CLSID for the revocation provider COM server that is used by this revocation configuration to check the status of a certificate.<15>
Provider	This is a VARIANT data type that contains the value of the properties of the revocation provider. The **vt** member of the VARIANT MUST be set either to VT_EMPTY or VT_ARRAY | VT_VARIANT. If the **vt** member is set to VT_ARRAY | VT_VARIANT, then the pArray member MUST reference a two-dimensional safearray. The number of elements in the second dimension (signifying the number of columns) of the safearray referenced by pArray MUST be 2. The number of elements in the first dimension (signifying the number of rows) of the safearray referenced by pArray MUST be set to the number of properties for the revocation provider. For each revocation provider property, the first column of the row MUST be a VARIANT with **vt** member set to VT_BSTR, and the **bstrVal** member set to the BSTR for the Unicode string value of the revocation provider property name. The second column of the row MUST be a VARIANT with the value defined in the following table (corresponding to the name of the revocation provider property).

The following table gives the processing rules for the revocation provider.

Property name	Processing rules
CrlUrlTimeOut	The **vt** member of the VARIANT MUST be set to VT_I4, and the **lVal** member MUST be the time-out in milliseconds that the revocation provider must wait before it times out while trying to retrieve the CRL for which it is configured.
BaseCrlUrls	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 URIs from where a base CRL [RFC3280] can be retrieved. For each URI, there MUST be an element in safearray referenced by pArray containing the BSTR for the Unicode string value of the URI.
DeltaCrlUrls	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 URIs from where a delta CRL [RFC3280] can be retrieved. For each URI, there MUST be an element in safearray referenced by pArray containing the BSTR for the Unicode string value of the URI.
BaseCrl	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 referenced by pArray MUST be equal to the length in bytes of the ASN.1 encoded binary representation of the Base CRL loaded by the revocation provider.
DeltaCrl	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 referenced by pArray MUST be equal to the length in bytes of the ASN.1 encoded binary representation of the Delta CRL loaded by the revocation provider.
RevocationErrorCode	The **vt** member of the VARIANT MUST be set to VT_I4, and the **lVal** member MUST be the HRESULT DWORD value denoting the status of this

Property name	Processing rules
	revocation provider. A value of 0 means that the revocation provider can provide certificate revocation status for certificates issued by the certificate authority configured for the revocation configuration. See [MS-ERREF] for a list of the possible error codes.
IssuedSerialNumbersDirectories	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 UNC or local file paths that are being used by the CA to store the serial numbers of certificates.<16>

3.2.4.1.4 SetCAConfigInformation (Opnum 6)

This method sets all the properties for a particular revocation configuration.

```
HRESULT SetCAConfigInformation(
  [in, ref] const BSTR bstrCAId,
  [in, ref] const VARIANT* pEntryValue
);
```

bstrCAId: This is a BSTR that specifies the RevocationConfigurationId for the revocation configuration whose properties are to be set.

pEntryValue: This is a pointer to a VARIANT data type that contains the names and values of the properties to set.

The following processing rules apply:

1. If *bstrCAId* is NULL or empty, the server SHOULD return the error E_INVALIDARG (0x80000003L).
2. The type of the variant as identified by *pEntryValue* MUST be either VT_EMPTY or VT_ARRAY|VT_VARIANT. If *pEntryValue* is a variant of type VT_ARRAY|VT_VARIANT, each element in this variant array SHOULD be a property identified in section 3.2.1.1, with the exception of the KeySpec property as noted in the table in that section. If the variant is of any other type, the responder SHOULD fail the request with an error code of 0x8000FFFF.
3. If there is a revocation configuration in the RevocationConfigurationList with the same RevocationConfigurationId as the value of *bstrCAId*, then:
    1. If *pEntryValue* is of type VT_EMPTY, the responder MUST delete the revocation configuration identified by *bstrCAId* from the RevocationConfigurationList, delete all its associated properties, and return success.
    2. If *pEntryValue* is of type VT_ARRAY|VT_VARIANT, the responder MUST delete the original revocation configuration from RevocationConfigurationList, add a new entry with the same RevocationConfigurationId, store the properties and values in the VARIANT reference by *pEntryValue* parameter, and return success.
4. If there is not a revocation configuration in the RevocationConfigurationList with the same RevocationConfigurationId as the value of *bstrCAId*, then:
    1. If *pEntryValue* is of type VT_ARRAY|VT_VARIANT, the responder MUST create a new revocation configuration in the RevocationConfigurationList, and the responder MUST store all the properties present in the *pEntryValue* parameter for the new revocation configuration. The responder MUST successfully process the request.

2. If *pEntryValue* is of type VT_EMPTY, the responder MUST fail the request. The error code SHOULD be 0x800710d8.

3.2.4.1.5 GetSecurity (Opnum 7)

The GetSecurity method is used to retrieve the security descriptor associated with the responder.

```
HRESULT GetSecurity(
  [out, ref] CERTTRANSBLOB* pctbSD
);
```

pctbSD: This is a pointer to a CERTTRANSBLOB structure that contains the marshaled Security Descriptor. Information on Security Descriptors is documented in [MS-DTYP] section 2.4.6.<17>

3.2.4.1.6 SetSecurity (Opnum 8)

The SetSecurity method is used to set the Online Responder Security, as defined in the Abstract Data Model.

```
HRESULT SetSecurity(
  [in, ref] CERTTRANSBLOB* pctbSD
);
```

pctbSD: A pointer to the CERTTRANSBLOB structure that contains the marshaled security descriptor. Information on security descriptors is documented in [MS-DTYP] section 2.4.6.

The responder SHOULD use the permissions set in *pctbSD* to deny and allow operations on the responder.

3.2.4.1.7 GetSigningCertificates (Opnum 9)

The GetSigningCertficates method retrieves a list of certificates available at the responder machine that can be used to sign responses to OCSP requests regarding certificates issued by the CA certificate specified.

```
HRESULT GetSigningCertificates(
  [in, ref] const VARIANT* pCAVar,
  [out, ref] VARIANT* pSigningCertificates
);
```

pCAVar: A pointer to a VARIANT data type containing the CA certificate. The **vt** member of VARIANT SHOULD be set to VT_ARRAY | VT_UI1, and the pArray member SHOULD reference a safearray that contains the ASN.1 DER encoded X.509 certificate data type containing the CA certificate.

pSigningCertificates: A pointer to VARIANT data type containing the list of certificates. On successful return, the server SHOULD set the vt member of the VARIANT to VT_ARRAY|VT_UI1, and the pArray member SHOULD reference a safearray that contains the ASN.1 DER encoded degenerate PKCS#7 [RFC2315] containing the certificates.

The following processing rules apply:

1. If *pCAVar* or *pSigningCertificates* is NULL, the responder MUST fail the request. The error code SHOULD be 0x800706f4.

2. The VARIANT referenced by *pCAVar* SHOULD have **vt** member set to VT_ARRAY | VT_UI1; otherwise, the server MUST fail the request. The error code SHOULD be 0x80070057.

3. The pArray member of the VARIANT referenced by *pCAVar* SHOULD reference a safearray that contains the ASN.1 DER encoded X509 certificate; otherwise, the server MUST fail the request. The error code SHOULD be 0x80070057.

4. Each certificate returned in the PKCS#7 contained in *pSigningCertificate* SHOULD meet the following criteria:

   - The certificate MUST include the extension id-kp-OCSPSigning in an extendedKeyUsage defined in [RFC2560] section 4.2.2.2.
   - The certificate MUST be signed by the CA certificate passed in *pCAVar* to this method.
   - The responder MUST have access to the private key corresponding to the public key in the certificate.

5. If there are no OCSP signing certificates that match the criteria specified in processing rule (4), the server SHOULD return an empty list of signing certificates as an ASN.1 DER encoded degenerate PKCS#7 [RFC2315] containing no certificates.

3.2.4.1.8 GetHashAlgorithms (Opnum 10)

The GetHashAlgorithms method retrieves the list of hash algorithms available at the responder that could be used along with the signing certificate associated with a revocation configuration to sign OCSP responses.

```
HRESULT GetHashAlgorithms(
  [in, ref] const BSTR bstrCAId,
  [out, ref] VARIANT* pHashAlgorithms
);
```

bstrCAId: A BSTR that specifies the RevocationConfigurationId.

pHashAlgorithms: A pointer to a VARIANT that is of type VT_ARRAY | VT_BSTR. Each element in the array is the name of a hash algorithm that could be used along with the signing certificate associated with a revocation configuration identified by *bstrCAId* to sign OCSP responses.

The following processing rules apply:

1. If *bstrCAId* or *pHashAlgorithms* has a NULL value, the responder MUST fail the request. The error code SHOULD be 0x800706f4.

2. If a revocation configuration with same RevocationConfigurationId as the value of *bstrCAId* does not exist in the RevocationConfigurationList, the responder MUST fail the request. The error code SHOULD be 0x800710d8.

3. Otherwise, if the revocation configuration (identified) has a signing certificate property associated with it, the server SHOULD return the list of hash algorithms that can be used with the public key associated with the signing certificate to sign OCSP responses.

4. If the revocation configuration does not have a signing certificate property associated with it, the server SHOULD return a default list of hash algorithms.<18>

3.2.4.1.9 GetMyRoles (Opnum 11)

The GetMyRoles method retrieves the **Online Responder Roles** [CIMC-PP] assigned to the user that calls the method.

```
HRESULT GetMyRoles(
  [out] LONG* pdwRoles
```

```
);
```

pdwRoles: Reference to an unsigned integer value that represents the retrieved Online Responder Role for the caller. This can be a bitwise OR of the following values.

Value	Meaning
CA_ACCESS_READ 0x00000100	The caller can read the configuration information at the responder.
CA_ACCESS_ENROLL 0x00000200	The caller can request the response status for a particular certificate from the responder.
CA_ACCESS_ADMIN 0x00000001	The caller can update the configuration information at the responder.
0x00000000	The caller has no roles.

3.2.4.1.10 Ping (Opnum 12)

This method queries the Online Responder Service to find out whether it is running.

```
HRESULT Ping();
```

This method has no parameters.

If the Online Responder Service is running, the server MUST return success (0) when this method is invoked.

3.2.5 Timer Events

None.

3.2.6 Other Local Events

None.

4 Protocol Examples

An administrator needs to enable the responder to create OCSP responses for clients requesting revocation status for certificates issued by a CA called "CA1", which runs on a machine called "Server1".

CA1 is configured to issue certificates based on a certificate template called "OCSPResponseSigning".

This means that the administrator needs to create a new revocation configuration on the responder, configured for the CA certificate of CA1.

1. The client needs to query all the revocation configuration IDs currently configured on the responder to ensure that it does not overwrite an existing entry. The client queries the responder revocation configuration list using the GetOCSPProperty method, with *bstrEntryName* of "CAEntries".

2. The server returns the list of revocation configuration IDs currently on the server as the variant referenced by parameter *pEntryValue* (for each revocation configuration in RevocationConfigurationList, there is an element in the safearray referenced by pArray that contains the BSTR for the Unicode string value of the RevocationConfigurationId).

3. The client reads the list of revocation configuration IDs returned by the server and creates a unique RevocationConfigurationId (that is, not a duplicate of any existing entry).

4. The client then constructs a variant of type VT_ARRAY|VT_VARIANT whose pArray member points to a two-dimensional array. The two-dimensional array will have one element of the first dimension (that is, one row) for each revocation configuration property the administrator wishes to set. The array will have two elements of the second dimension (that is, columns) for each element of the first dimension: one containing a variant of type VT_BSTR whose **bstrVal** contains the name of a revocation configuration property, and one containing a variant with the value for that property. The property value variants are constructed as follows:

    1. CA Certificate property:

        1. **vt** member is VT_ARRAY |VT_VARIANT.

        2. pArray references a single dimension safearray with one element for each byte of the ASN1 DER encoded CA certificate for CA1.

    2. SigningFlags property:

        1. vt member is VT_I4.

        2. **lVal** contains the value whose hex representation is 0x0000025d. This means that the following Signing Flags are set (defined in section 3.2.4.1.3):

            1. 0x01 – silently acquire private key

            2. 0x04 – auto-use renewed signing certificate

            3. 0x08 – signing certificate signed by CA certificate key

            4. 0x10 – automatically look for an OCSP cert

            5. 0x40 – responses include key hash of signing certificate

            6. 0x200 – enroll for a signing certificate from CAConfig, using SigningCertificateTemplate

    3. ProviderCLSID property:

1. **vt** member is VT_BSTR.

2. bstrVal contains the BSTR representation of the Unicode string representation of "{4956d17f-88fd-4198-b287-1e6e65883b19}".

4. Provider property:

    1. vt member is VT_ARRAY|VT_VARIANT.

    2. pArray points to another two-dimensional array with property name and property value pairs like those described for the containing variant array.

        1. BaseCrlUrls property:

            1. vt member is VT_ARRAY|VT_BSTR.

            2. pArray points to a single dimension safearray, in which each element is a BSTR representation of the Unicode representation of a URI where the responder can contain a CRL published by CA1. In this example, the value "http://CA1.Server1.contoso.com/CRL/CA1.crl" is used.

5. SigningCertificateTemplate:

    1. **vt** member is VT_BSTR.

    2. **bstrVal** is the BSTR representation of the Unicode string representation of "OCSPResponseSigning" (the template name).

6. CAConfig:

    1. **vt** member is VT_BSTR.

    2. **bstrVal** is the BSTR representation of the Unicode string representation of "Server1\CA1", the CA configuration string.

5. The client calls the SetCAConfigInformation method on the server, passing the newly generated RevocationConfigurationId as the *bstrCAId* parameter, and a *pEntryValue* pointing to the variant constructed in step 4, containing configuration values for the new revocation configuration.

6. The server creates the new revocation configuration and returns S_OK.

5 Security

5.1 Security Considerations for Implementers

5.1.1 Strong Administrator Authentication

An administrator of the responder must authenticate strongly. This could be via a high-entropy password or some multiple-factor authentication method (such as a smart card). It is recommended that the administrator use a login account that functions only for responder administration and not for any other function. Use of the same credentials on a vulnerable computer while performing some other task exposes the credentials to capture and misuse.

5.1.2 KDC Security

Because authentication of the administrator is by Kerberos, in this protocol, the key distribution center (KDC) is itself to be kept secure—free from tampering and free from vulnerabilities that would allow privilege-elevation penetrations.

5.1.3 Administrator Console Security

The administrator's console (the applications used by the administrator to run the client side of this protocol and the operating system in which that functionality runs) is to be kept secure from penetration that would allow an attacker to act as the administrator.

5.1.4 Administrator Credential Issuance

Because the administrator is identified as some name in a Kerberos domain, for the purpose of access control by the responder, the human procedures for assigning a name to the administrator, adding that name to some named group of administrators and adding that group name to the **access control list (ACL)** used by the responder, is to be kept free from either penetration (for example, social engineering) or human mistake via common misspelling or unwarranted assumptions.

5.1.5 Practices when Using Cryptography

Any cryptographic protocol has security considerations dealing with key handling during cryptographic operations and key distribution. Although a public-key certificate is not a protocol by itself, it has most of the same security considerations of a cryptographic protocol in the sense that a public key certificate is a message from the CA to the responder a message addressed, in effect, "to whom it may concern". A cryptographic protocol that deals with the transmission or issuance or other use of a public key certificate therefore has security considerations in two areas: around the protocol itself and around the certificate and its use.

In addition, a certificate binds two or more pieces of information together. In the most common case, that would be a public key and a name. The name in such a certificate has security relevance, and there are security considerations around the use and provisioning of those names. In some certificate forms, there are attributes bound to either a name or a key, and there are security considerations around the use and provisioning of those attributes.

5.1.5.1 Keeping Information Secret

Any cryptographic key has to be kept secret. One also keeps secret any function of a secret (such as a key schedule), because knowing such functions would reduce an attacker's work in cryptanalyzing the secret.

When a secret has to be in the normal memory of a general purpose computer in order to be used, that secret should be erased (for example, replaced with a constant value, such as 0) as soon as possible after use.

A secret can be kept in a specially protected memory where it can be used without being erased. Typically, one finds such memory in a Hardware Security Module (HSM). If an HSM is used, it has to be compliant with [FIPS140], or the equivalent at a level consistent with the security requirements of the customer deploying the cryptographic protocol or the CA that uses the HSM.

5.1.5.2 Coding Practices

Any implementation of a protocol exposes code to inputs from attackers. Such code has to be developed according to secure coding and development practices in order to avoid buffer overflows, denial of service attacks, escalation of privilege, and disclosure of information. For an introduction to these concepts, as well as secure development best practices and common errors, see [HOWARD].

5.1.5.3 Security Consideration Citations

Implementers of this protocol should consider the following security considerations:

- A client or server has to follow generally accepted principles of secure key management. For more information, see section 9 of [RFC3280]. For an introduction to these generally accepted principles, see [CRYPTO] and [HOWARD].
- Clients and servers should validate cryptographic parameters prior to issuing or accepting certificates. For more information, see section 9 of [RFC2797].
- A client and server should validate and verify certificate path information identified in section 6 of [RFC3280]. See section 9 of [RFC3280] for more information on the requirement for certificate path validation.
- A client and server should validate and verify the freshness of revocation information of all digital certificates prior to usage, trust, or encryption as identified in section 6.3 of [RFC3280]. See section 9 of [RFC3280] for more information on the requirement for revocation freshness.
- A client or server should follow all security considerations in section 5 of [RFC2560].
- A client or server should follow all security considerations discussed throughout [RFC2315] and [RFC2986], because neither normative reference has a specific security section.
- A client and server should use an authenticated HTTP session between client and server to mitigate denial of service attacks. For more information on generic denial-of-service (DoS) mitigation techniques, see [HOWARD].

5.2 Index of Security Parameters

None.

6 Appendix A: Full IDL

For ease of implementation, the full **Interface Definition Language (IDL)** is provided, where "ms-oaut.idl" is the IDL found in [MS-OAUT] Appendix A.<19>

```
import "ms-oaut.idl";


typedef struct _CERTTRANSBLOB
{
  unsigned long        cb;
  [size_is(cb), unique] BYTE *pb;
} CERTTRANSBLOB;
// Interface IOCSPAdminD

[

object,
uuid(784b693d-95f3-420b-8126-365c098659f2),
helpstring("IOCSPAdminD DCOM Interface"),
pointer_default(unique)
]

interface IOCSPAdminD: IUnknown
{

HRESULT GetOCSPProperty(
[in, ref] const BSTR bstrEntryName,
[out, ref] VARIANT* pEntryValue);

HRESULT SetOCSPProperty(
[in, ref] const BSTR bstrEntryName,
[in, ref] const VARIANT *pEntryValue);

HRESULT GetCAConfigInformation(
[in, ref] const BSTR bstrCAId,
[out, ref] VARIANT* pEntryValue);

HRESULT SetCAConfigInformation(
[in, ref] const BSTR bstrCAId,
[in, ref] const VARIANT *pEntryValue);

HRESULT GetSecurity(
[out, ref] CERTTRANSBLOB *pctbSD);

HRESULT SetSecurity(
[in, ref] CERTTRANSBLOB *pctbSD);

HRESULT GetSigningCertificates(
[in, ref] const VARIANT *pCAVar,
[out, ref] VARIANT* pSigningCertificates);

HRESULT GetHashAlgorithms(
[in, ref] const BSTR bstrCAId,
[out, ref] VARIANT* pHashAlgorithms);

HRESULT GetMyRoles(
[out]LONG *pdwRoles);

HRESULT Ping();

};
```

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 Server 2008 operating system
- Windows Server 2008 R2 operating system
- Windows Server 2012 operating system
- Windows Server 2012 R2 operating system
- Windows Server 2016 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 3.2.1.2: The name of the machine used in the Microsoft implementation is the fully qualified domain name (FQDN) of the machine.

<2> Section 3.2.1.2: The names of the machines used in the Microsoft implementation are the FQDNs of the machines.

<3> Section 3.2.1.3: A Microsoft Online Responder defines two permissions: Read and Administer. For responder security methods GetSecurity, SetSecurity, and GetMyRoles, the Microsoft Online Responder assigns permissions to principals (identified by the **access control entry (ACE)** in the following manner.

Permission	Bit value	Meaning
Read	0x00000100	The caller can read the configuration information and properties of the responder.
Administer	0x00000001	The caller can update the configuration information and properties of the responder.

If a principal has Administer permission, Read permission is implied (does not need to be explicitly set).

The responder can enforce Online Responder security for each of the following methods by checking for the permissions identified in the following table.

Method name	Acceptable permissions
GetOCSPProperty	Read
SetOCSPProperty	Administrator
GetCAConfigInformation	Read
SetCAConfigInformation	Administrator
GetSecurity	Read
SetSecurity	Administrator

The security descriptor on the responder controls which security principal can manage or read configuration information or request certificate status from the responder. Whenever a Read method on the responder is invoked, the responder checks this security descriptor to ensure that the calling

entity has read access; if the entity doesn't have read access, the responder returns 0x80070005 as the error code. Whenever any Write method is invoked, the responder checks this security descriptor to ensure that the calling entity has manage access on the responder; if it does not, 0x80070005 is returned by the responder.

These methods require read access:

- GetOCSPProperty
- GetCAConfigInformation
- GetSecurity
- GetSigningCertificates
- GetHashAlgorithms
- Ping

These methods require manage access:

- SetOCSPProperty
- SetCAConfigInformation
- SetSecurity

The following method can be invoked by any caller:

- GetMyRoles

<4> Section 3.2.1.3: A Microsoft Online Responder defines two permissions: Read and Administer. For responder security methods GetSecurity, SetSecurity, and GetMyRoles, the Microsoft Online Responder assigns permissions to principals (identified by the ACE) in the following manner.

Permission	Bit value	Meaning
Read	0x00000100	The caller can read the configuration information and properties of the responder.
Administer	0x00000001	The caller can update the configuration information and properties of the responder.

If a principal has Administer permission, Read permission is implied (does not need to be explicitly set).

The responder can enforce Online Responder security for each of the following methods by checking for the permissions identified in the following table.

Method name	Acceptable permissions
GetOCSPProperty	Read
SetOCSPProperty	Administrator
GetCAConfigInformation	Read
SetCAConfigInformation	Administrator
GetSecurity	Read
SetSecurity	Administrator

The security descriptor on the responder controls which security principal can manage or read configuration information or request certificate status from the responder. Whenever a read method on the responder is invoked, the responder checks this security descriptor to ensure that the calling entity has read access; if the entity does not have read access, the responder returns 0x80070005 as the error code. Whenever any write method is invoked, the responder checks this security descriptor

to ensure that the calling entity has manage access on the responder; if it does not, 0x80070005 is returned by the responder.

These methods require read access:

- GetOCSPProperty
- GetCAConfigInformation
- GetSecurity
- GetSigningCertificates
- GetHashAlgorithms
- Ping

These methods require manage access:

- SetOCSPProperty
- SetCAConfigInformation
- SetSecurity

The following method can be invoked by any caller:

- GetMyRoles

<5> Section 3.2.4.1.1: For the Microsoft responder, this property has values between 5 and 9999.

<6> Section 3.2.4.1.1: The Microsoft responder uses integer values between 0 and 6.

Value	Meaning
CERTLOG_MINIMAL 0x00000000	Log events for errors and warnings that occur on the responder.
CERTLOG_TERSE 0x00000001 — 0x00000003	Log errors, warnings, and informational events.
CERTLOG_VERBOSE 0x00000004	Log extended events.
CERTLOG_EXHAUSTIVE 0x00000005 — 0x00000006	Throttling is removed for events that can be generated quickly, such as MSG_E_POSSIBLE_DENIAL_OF_SERVICE_ATTACK.

<7> Section 3.2.4.1.1: The Microsoft responder uses a value of 0xfffffffe3 to indicate that debug tracing is enabled and 0 to indicate that it is not.

<8> Section 3.2.4.1.1: The Microsoft responder uses values between 1 and 24.

<9> Section 3.2.4.1.1: The Microsoft responder uses a default value of 20.

<10> Section 3.2.4.1.1: The Microsoft responder uses a value of 0xfffffffe3 to indicate that debug tracing is enabled and 0 to indicate that it is not.

<11> Section 3.2.4.1.1: The MaxNumOfRequestEntries property is not supported in Windows Server 2008, Windows Server 2008 R2, Windows Server 2012, or Windows Server 2012 R2.

<12> Section 3.2.4.1.1: Windows does not return any vendor defined properties.

<13> Section 3.2.4.1.2: The type must match the value specified in section 3.2.4.1.1 if the server is a Windows responder. Otherwise, the responder might not function correctly.

<14> Section 3.2.4.1.3: The Microsoft responder uses the hash algorithms supported by the cryptographic provider specified in the CSPName property.

<15> Section 3.2.4.1.3: The Microsoft Online Responder returns a value of {4956d17f-88fd-4198-b287-1e6e65883b19} for this property.

<16> Section 3.2.4.1.3: The IssuedSerialNumbersDirectories property is not supported in Windows Server 2008, Windows Server 2008 R2, Windows Server 2012 operating system, or Windows Server 2012 R2.

<17> Section 3.2.4.1.5: By default the Responder SD is as follows:

- Owner: SID for Builtin\Administrators (S-1-5-32-544)
- Group: SID for Builtin\Administrators (S-1-5-32-544)

2 ACE's with ACE_TYPE ACCESS_ALLOWED_ACE_TYPE (0x00):

- Allow Builtin Admins to read and manage the responder.
- Allow Network Service account to proxy requests.

2 ACE's with ACE_TYPE SYSTEM_AUDIT_ACE_TYPE (0x02):

- Audit Success and Failure for everyone when they try to access for the 0xffff (any) access rights.
- Audit Success and Failure for anonymous users when they try to access for the 0xffff access rights.

Within the ACCESS_MASK, the bit values have the following meanings:

Permission	Bit Value	Meaning
Read	0x00000100	Read the configuration information and properties of the responder.
Administer	0x00000001	Update the configuration information and properties of the responder.
Proxy requests	0x00000300	Proxy requests (if the responder is split into a front end and back end service).

<18> Section 3.2.4.1.8: The Microsoft Online Responder returns the hash algorithms supported by the "Microsoft Strong Cryptographic Provider" CSP in the default list of hash algorithms.

<19> Section 6: The Microsoft implementation of the OCSP admin interface has a CLSID whose value is { 0x6d5ad135, 0x1730, 0x4f19, { 0xa4, 0xeb, 0x3f, 0x78, 0xe7, 0xc9, 0x76, 0xbb}}.

8 Change Tracking

This section identifies changes that were made to this document since the last release. 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.
- The removal of a document from the documentation set.

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 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 changes were introduced. Minor editorial and formatting changes may have been made, but the technical content of the document is identical to the last released version.

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.
- 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 dochelp@microsoft.com.

Section	Tracking number (if applicable) and description	Major change (Y or N)	Change type
3.2.1.1.1 RevocationProviderProperties	Added content for this version of Windows Server.	Y	Content update.
3.2.1.2 ResponderProperties	Added content for this version of Windows Server.	Y	Content update.
3.2.4.1.1 GetOCSPProperty (Opnum 3)	Added content for this version of Windows Server.	Y	Content update.
3.2.4.1.3 GetCAConfigInformation (Opnum 5)	Added content for this version of Windows Server.	Y	Content update.

9 Index

T

V