Troubleshooting Certificate Status and Revocation

Published: October 1, 2001 | Updated: October 26, 2001
By Brian Komar

Abstract

Microsoft Windows 2000 and Microsoft Windows XP offer significant features in the areas of X.509 support, PKI as well as certificate status checking and revocation. This White paper details the basics of certificate status, chain building, and how they work in Windows operating systems to assist administrators in troubleshooting a PKI implementation.

On This Page

Acknowledgements

David Cross, Program Manager, Microsoft Corporation.

Trevor Freeman, Program Manager, Microsoft Corporation

Sergio Dutra, Software Design Engineer, Microsoft Corporation

Carsten Kinder, Senior Consultant, Microsoft Consulting Services

Top of page

Introduction

For certificate status to be determined, a Public Key Infrastructure (PKI), certificate revocation information must be made available to individuals, computers, and applications attempting to verify the validity of certificates. Traditionally a PKI uses a distributed method of verification so that the clients do not have to contact the Certification Authority (CA) directly to validate the credentials presented. Without checking certificates for revocation, the possibility exists that a security principal will accept credentials that have been revoked by a CA administrator.

Certificates are issued with a planned lifetime and explicit expiration date. A certificate may be issued for one minute, thirty years or even more. Once issued, a certificate becomes valid once its validity time has been reached, and it is considered valid until its expiration date. However, various circumstances may cause a certificate to become invalid prior to the expiration of the validity period. Such circumstances include change of name, change of association between subject and CA (for example, when an employee terminates employment with an organization), and compromise or suspected compromise of the corresponding private key. Under such circumstances, the CA needs to revoke the certificate.

There are several mechanisms to represent revocation information; RFC 2459 defines one such method. This method involves each CA periodically issuing a signed data structure called a certificate revocation list (CRL). A CRL is a time stamped list identifying revoked certificates, which is signed by a CA and made freely available in a public repository. Each revoked certificate is identified in a CRL by its certificate serial number. When a certificate aware system uses a certificate (for example, for verifying a remote user's digital signature), that system should not only check the certificate signature and time validity, but it should also acquire a suitably recent certificate status to ensure the certificate being presented is not revoked. In the case of CRLs, Microsoft defines as suitably recent a CRL that is not past the next update time of the CRL. A CA issues a new CRL on either a configured regular periodic basis (for example, hourly, daily, or weekly) or on an event basis; for example, if an important certificate is deemed compromised, the CA may issue a new CRL to expedite notification of that fact.

There are several types of CRLs: full CRLs (also known as base CRLs), delta CRLs, and CRL Distribution Points (CDPs). Full CRLs contain the status of all certificates. Delta CRLs contain only the status of all certificates that have changed status between

the issuance the last Base CRL. CRL Distribution Points are used to anchor a well-known location for Base, Delta, and even partitioned CRLs..

An entry is added to the CRL as part of the next update following notification of revocation. An entry may be removed from the CRL after appearing on one regularly scheduled CRL issued beyond the revoked certificate's validity period

Note: The ability to remove an entry from the CRL is only available if the certificate was revoked with the reason "Certificate Hold."

Windows 2000 and Microsoft Windows .NET allow you to implement a Public Key Infrastructure (PKI) using Certificate Services. Certificate Services incorporate industry-standard X509 v3 CRLs to distribute information about certificate revocation status. The CRLs can be published to Web servers, FTP servers or to Active Directory.

Scope

The scope and audience of this White paper is to assist organizational system architects and administrators in understanding how certificate chaining and revocation work in Windows 2000 and Windows XP to allow the administrators to troubleshoot problems related to certificate chaining and revocation. For an introduction to PKI and Certificate Services, please refer to http://www.microsoft.com/technet/security/topics/crypto/default.mspx

Terms Used in this White paper

The following terms are used in this white paper:

Authority Information Access (AIA). A certificate extension that contains information useful for verifying the trust status of a certificate. This information potentially includes URL locations where the issuing CA's certificate can be retrieved, as well as a location of an OCSP Responder configured to provide status for the certificate in question. The AIA extension can potentially contain HTTP, FTP, LDAP or FILE URLs.

Authority Key Identifier (AKI). This certificate extension is used by the certificate chaining engine to determine what certificate was used to sign a presented certificate. The AKI can contain the issuer name and serial number, public key information, or no information at all. By matching the information in a certificate's AKI extension to a CA certificate's Subject Key Identifier (SKI) extension a certificate chain can be built.

CRL Distribution Point (CDP). A certificate extension that indicates where the certificate revocation list for a CA can be retrieved. This extension can contain multiple HTTP, FTP, File or LDAP URLs for the retrieval of the CRL.

Certificate Trust List (CTL). A method of restricting certificates chaining to a designated CA for limited time periods or usages. Used more prevalently in a Windows 2000 network. In a Windows Server 2003 network, qualified subordination is the preferred method for restricting certificate usage between organizations.

Certificate Revocation List (CRL). A digitally signed list issued by a Certification Authority (CA) that contains a list of certificates issued by the CA that have been revoked. The listing includes the serial number of the certificate, the date that the certificate was revoked, and the revocation reason. Applications can perform CRL checking to determine a presented certificate's revocation status.

Online Certificate Status Protocol (OCSP). A protocol that allows real-time validation of a certificate's status by having the CryptoAPI make a call to an OCSP responder and the OCSP responder providing an immediate validation of the revocation status for the presented certificate. Typically, the OCSP responder uses CRLs for retrieving certificate status information.

Public Key Infrastructure (PKI). A PKI provides an organization with the ability to securely exchange data over a public network using public key cryptography. A PKI consists of Certification Authorities (CA) that issue digital certificates, directories that store the certificates (including Active Directory in Windows 2000 and Windows Server 2003), and X.509 certificates that are issued to security entities on the network. The PKI provides validation of certificate-based credentials and ensures that the credentials are not revoked, corrupted, or modified.

Subject Key Identifier (SKI). A certificate extension included in CA certificates that contains a hash of the CA certificate's public key. This hash is placed in the Authority Key Identifier (AKI) extension of all issued certificates to facilitate chain building.

Certificate Chaining. Certificate chaining is defined as the trust validation of an x.509 certificate as it is compared to a trust anchor such as a root certificate.

Understanding Revocation and Status Checking

The best way to start a discussion of certificate revocation and status checking is to look at how an end user sees the effects of certificate revocation and status checking in the Windows XP and Windows 2000 user interfaces. This section will look at scenarios where a certificate chain is both valid and invalid.

Certificate Chaining

One of the most common scenarios where a user sees the use of the certificate chaining engine occurs when a user validates a digital signature in email. Upon receipt of a digitally signed email message, the user will notice an icon next to the mail message indicating that the message is digitally signed (see Figure 1).

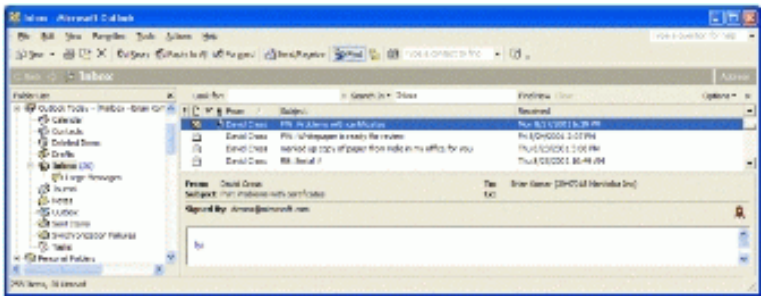

Figure 1: A Digitally signed message is indicated by a certificate icon
See full-sized image.

To verify that the content has not been modified in transit, the ribbon icon in the details pane in Figure 1 can be clicked to reveal the status of the certificate as shown in Figure 2.

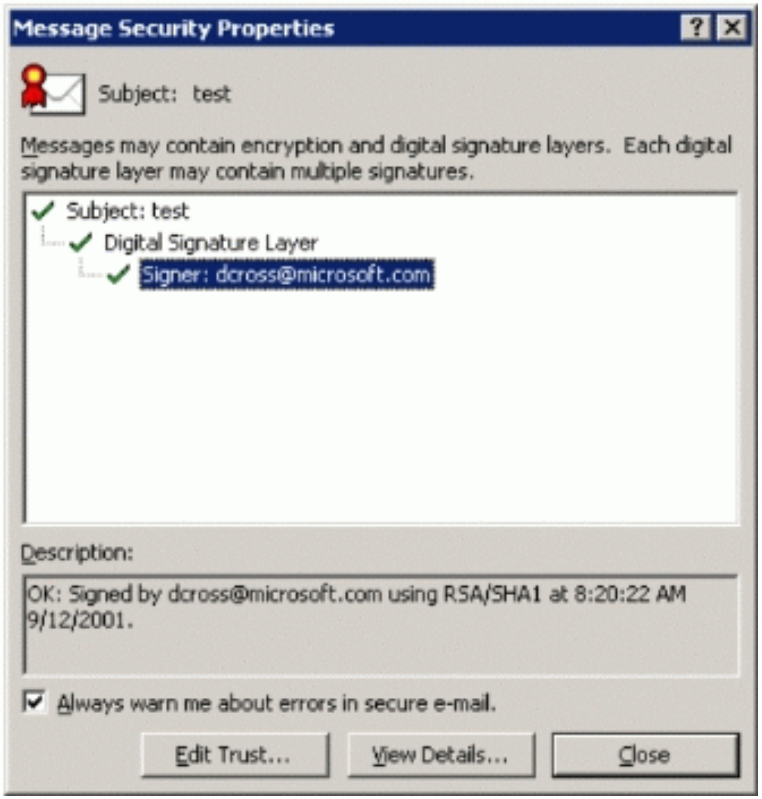

Figure 2: Verifying the digital signature

See full-sized image.

Not only does this dialog box show that the message was not altered after the digital signature was applied to the message, but it also includes information from the certificate chaining engine. Specifically, the certificate chaining engine reveals that:

- The certificate was not revoked. This statement includes all certificates in the certificate chain. From the certificate associated with the private key used to sign the email message, and a Certification Authority (CA) certificates from the end certificate to a trusted root CA certificate.
- The certificate has not expired. This statement indicates that all certificates in the certificate chain are time valid and are not expired.
- The certificate is trusted. The certificate chaining engine was able to establish a certificate chain for the certificate used to sign the email message to a trusted root CA.

Note: A certificate is "trusted" when it successfully chains (without revocation failure) to a trust anchor such as a root certificate, Certificate Trust List (CTL), and so forth. For additional information on trust, please refer to the following article: http://www.microsoft.com/technet/security/topics/identity/corepki.mspx

Further details about the certificate chain can be analyzed by clicking the Details button, as shown in Figure 3. The details button allows us to investigate the full certificate chain, from the end-certificate to the root CA certificate, validating each certificate in the certificate chain manually if necessary (see Figure 3). Note that the entire certificate chain can be visually validated in the first dialogue.

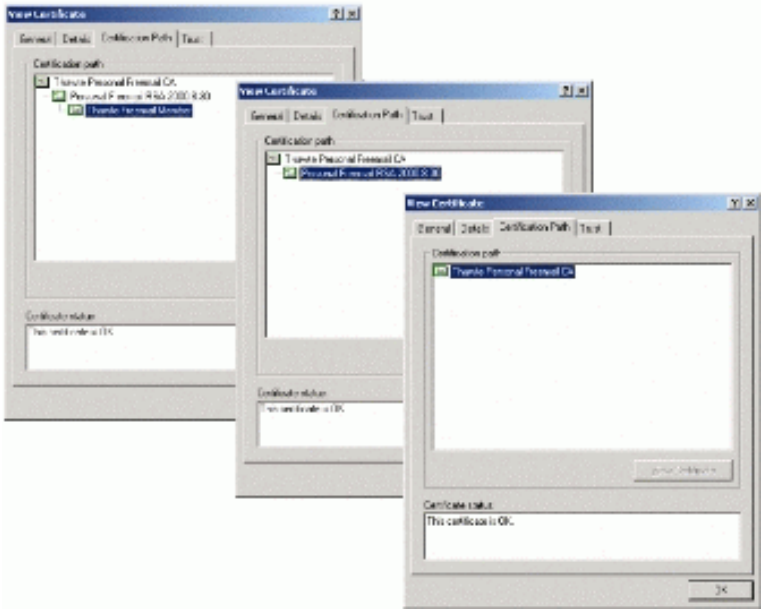

Figure 3: Each certificate in the certificate chain is validated
See full-sized image.

Troubleshooting Problems

There are instances where the digital signature is not valid. As mentioned earlier, this can be due to a certificate in the certificate chain being revoked, expired, or not chaining to a trusted root.

In Outlook 2000 SR1 and greater, when a certificate does not pass the validity checks, a dialog box as shown in Figure 4 can appear.

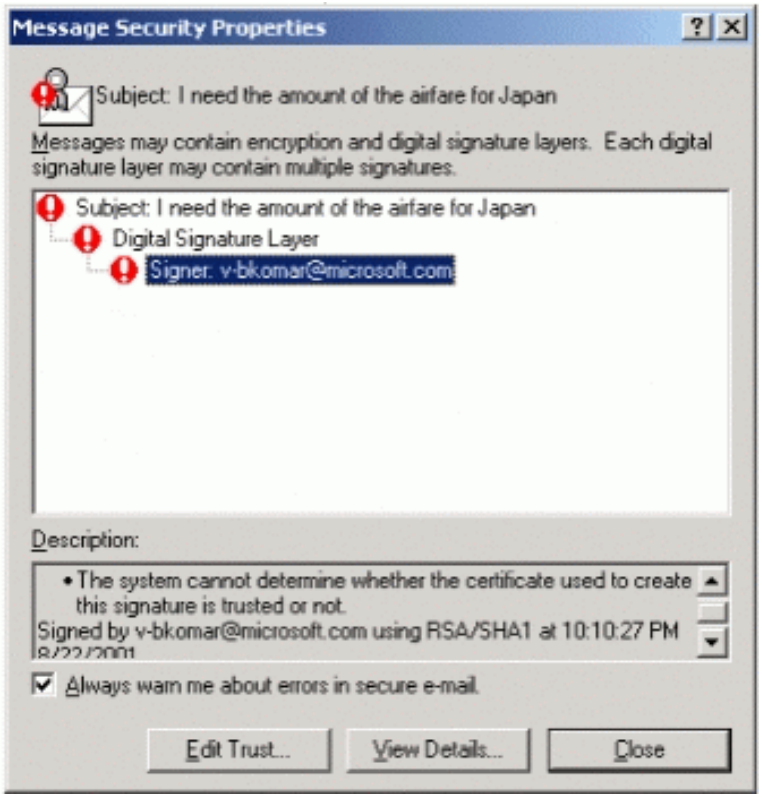

Figure 4: A warning indicating that the certificate used to create the digital signature is trusted
See full-sized image.

The dialog box shown in Figure 4 indicates that the reason the digital signature is not considered valid is that the certificate does

not chain to a trusted root CA. By clicking the **View Details** button, further details are shown, as indicated in Figure 5.

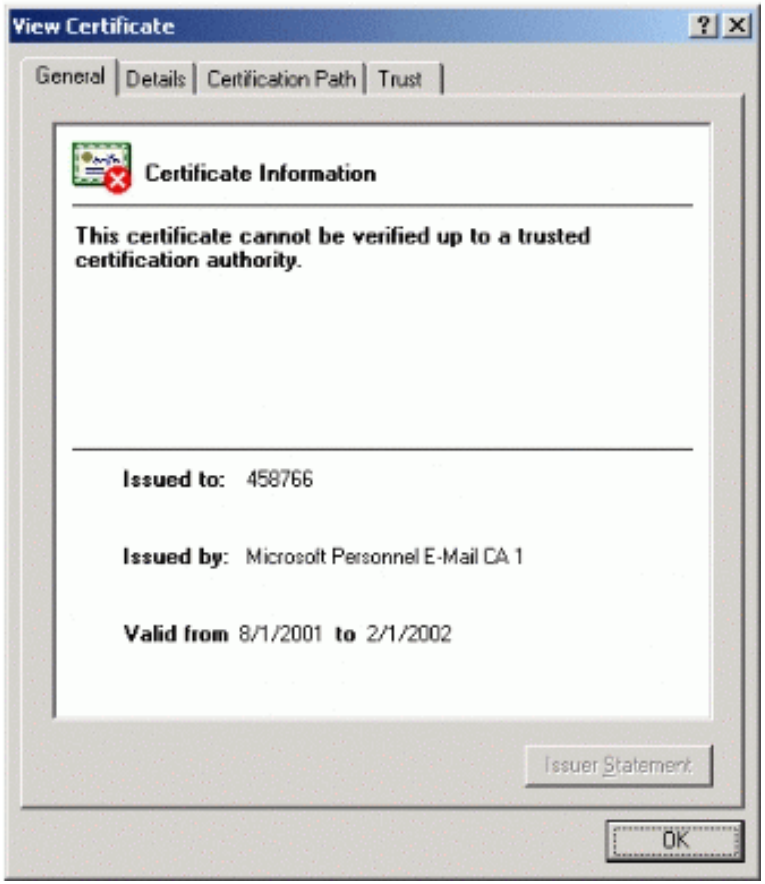

Figure 5: Details of an invalid digital signature
See full-sized image.

The General tab indicates further information about the certificate associated with the private key used to sign the email message. To view the path for the certificate, the **Certification Path** tab shows all CAs from the end certificate to the root CA, as shown in Figure 6.

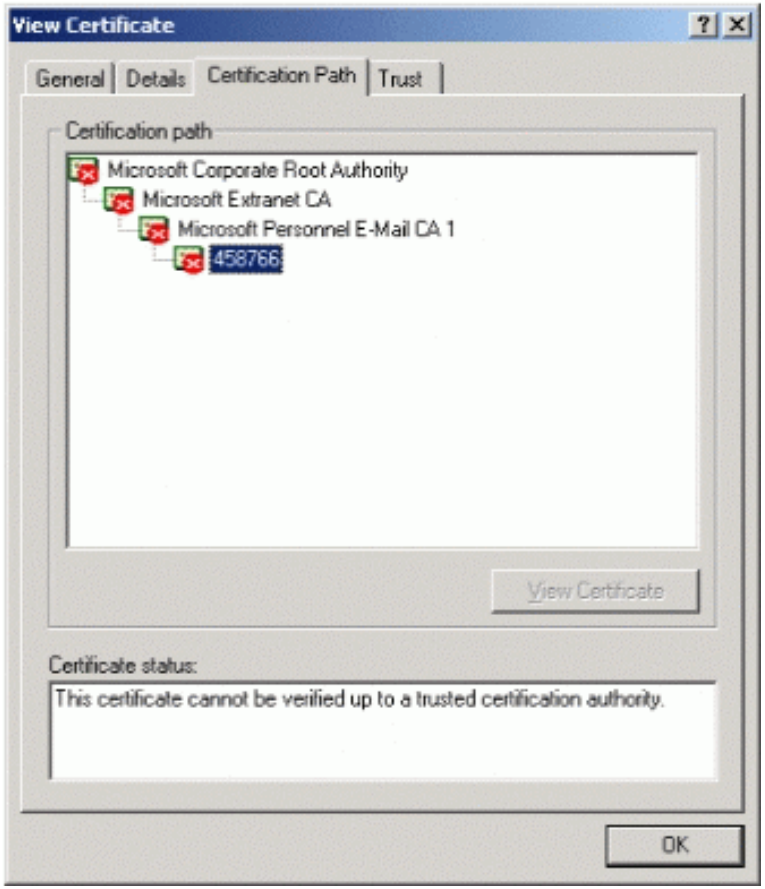

Figure 6: Viewing the entire certificate chain
See full-sized image.

To rectify the situation, an administrator must establish some means of either issuing certificates that are trusted by their organization, or establish trust with other organizations so that certificates used to sign email messages are trusted between organizations.

This white paper outlines several methods that are commonly employed to establish trust between organizations, and how the certificate chaining engine builds and evaluates chains to determine the status of a specific certificate chain. The next section discusses specifically how the Windows operating system validates certificates and their status. For additional information on troubleshooting issues, refer to the Troubleshooting section of this white paper.

Top of page

Certificate Status Checking

The status of a public key certificate is determined through three distinct, but inter-related processes implemented in the CryptoAPI:

•Certificate Discovery

The process of collecting CA certificates from Group Policy, Enterprise Policy, and Authority Information Access (AIA) pointers in issued certificates, and the certificate enrollment process.

•Path validation

The process by which public key certificates and their issuer certificates are processed in a hierarchical fashion until the certificate chain terminates at a trusted, self-signed certificate. Typically, this is a root CA certificate.

•Revocation checking

Each certificate in the certificate chain is verified to ensure that none of the certificates are revoked. The revocation checking can take place either in conjunction with the chain building process, or after the chain is built.

Chain Building

Chain building is the process of building a trust chain, or certification path, from the end certificate to a root CA that is trusted by the security principal. For example, Figure 7 shows a certification path that exists in a two-level CA hierarchy.

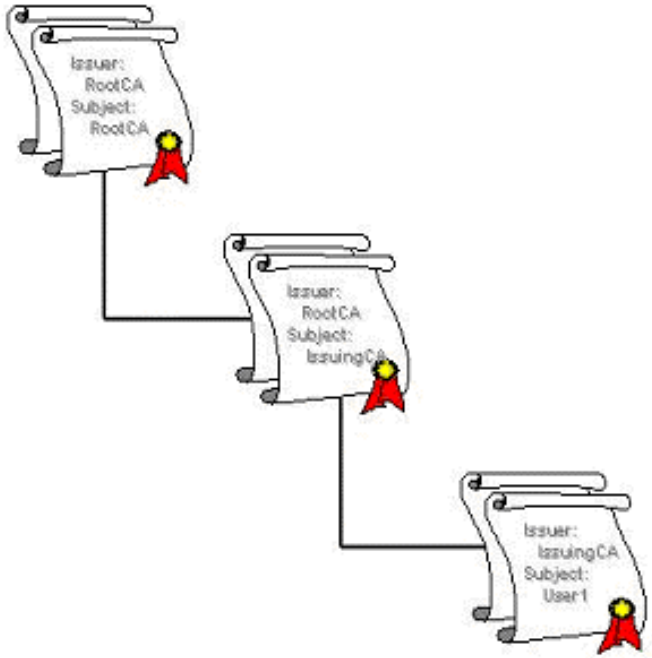

Figure 7: Certification Path

In this example, the User1 certificate was issued by the IssuingCA, and the IssuingCA's certificate was issued by the RootCA. This is considered a trusted chain, because the Root CA certificate is contained in the Trusted Root Certification Authorities store.

The chain building process will validate the certification path by checking each certificate in the certification path from the end certificate to the Root CA's certificate. The certificates are retrieved from the Intermediate Certification Authorities store, the Trusted Root Certification Authorities store, or from a URL specified in the Authority Information Access (AIA) attribute of the certificate. If the CryptoAPI discovers a problem with one of the certificates in the path, or if it cannot find a certificate, the certification path is discarded as a non-trusted certification path.

To improve performance, the CryptoAPI will store subordinate CA certificates in the Intermediate Certification Authorities store so that future requests for the certificate can be satisfied from the store, rather than accessing the certificate through a URL.

Certificate Storage

Windows 2000 and Windows XP store certificates locally on the computer or hardware device that requested the certificate, or in the case of a user, on the computer or device that the user used to request the certificate. The certificates are stored in a location known as a *certificate store*. There are separate stores known as the *machine store*, used by the computer, and the *user store* or

My store used by the currently logged on user. A certificate store will often contain numerous certificates, possibly issued from a number of different certification authorities.

Note: The currently logged on user will have access to read certificates contained in both the machine store and the My store, referred to as the Personal store in the Certificates MMC.

Note: Only if the currently logged on user is a member of the local Administrators group will the user be able to view the machine store in the Certificates MMC console.

Certificate chains are formed by looking at certificates available in multiple certificate stores. The certificate chain engine must determine what scope of certificate stores to search when building certificate chains. By default, the chain engine searches the ROOT, TRUST, CA and MY certificate stores (see Figure 8). The certificate stores may be viewed through the Certificates MMC snap-in.

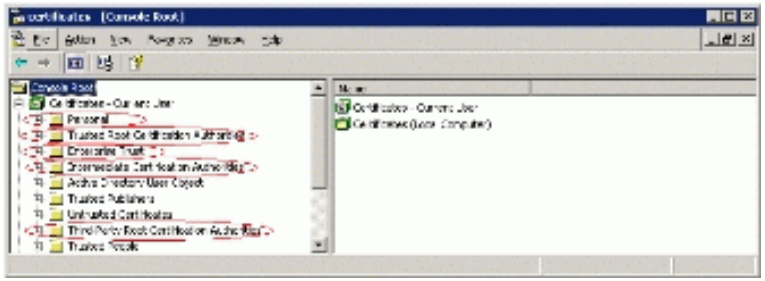

Figure 8: Stores searched by the Certificate Chain Engine
See full-sized image.

In addition to the default stores, the certificate chain engine can be configured to use different stores, such as restricted root, restricted trust, restricted other and additional stores. For example, the Trusted People store is used by both Outlook and EFS when searching for certificates. Alternatively, an application can create its own store for certificate storage or even call additional revocation providers registered with CryptoAPI. For additional information on programmatic settings that can be called for certificate chaining, refer to Appendix A of this white paper.

Purpose

The certificate chain engine builds all possible certificate chains. The entire graph of certificate chains is constructed and then ordered by the "quality" of the chain. The best quality chain for a given end certificate is returned to the calling application as the default chain.

Each chain is built using a combination of the certificates available in the certificate stores and certificates available from published URL locations. Each certificate in the chain is assigned a status code. The status code indicates whether the individual certificate is signature valid, time valid, expired, revoked, time nested, and so on. Each status code has a precedence assigned to it. For example, an expired certificate has a higher precedence than a revoked certificate. This is because an expired certificate should not be checked for revocation status.

If different status codes are assigned to the certificates in a certificate chain, the status code with the highest precedence is applied to the certificate chain and propagated into the certificate chain status. Certificate status codes are determined by the CERT_TRUST_STATUS structure defined in the Platform SDK. Appendix A provides detailed information regarding the various status codes and error codes that can be assigned to individual certificates and certificate chains by the chaining engine.

Note: The full status and error codes definitions can be found at http://msdn.microsoft.com/library by searching for CERT_TRUST_STATUS.

Selection

For each certificate in the chain, the certificate chain engine must select a certificate of the issuing CA. This process, known as path validation, is repeated until a self-signed certificate is reached, typically, a root CA certificate. CryptoAPI treats root certificates as the absolute trust anchor in trust decisions.

There are different processes that can be used to select the certificate for an issuing CA. The actual process that is used is based on whether the certificate currently being investigated has the Authority Key Identifier (AKI) extension defined. Inspection of the AKI extension will lead to one of three matching processes being implemented:

- **Exact match**. If the AKI extension contains the issuer's user name and issuer serial number, only certificates that match on user name and serial number will be chosen in the chain building process. As a further test, the issuer name on the issued certificate must match the subject name on the issuer certificate. Figure 9 shows a certificate where exact matching was used to find the issuer's certificate. Note that the subject and serial number in the AKI extension in the left hand certificate match the Serial number and Subject of the certificate on the right.

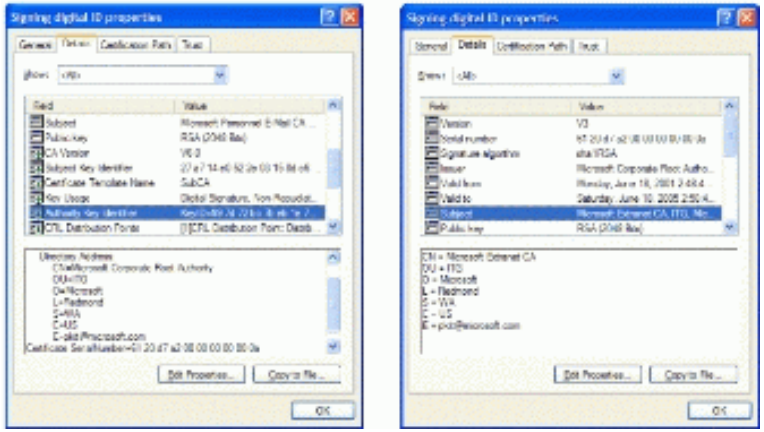

Figure 9: An Exact match
See full-sized image.

Important: In Windows 2000, exact match was assigned a much higher value for quality than a key match or name match. This resulted in the path validation process always selecting a certificate chain that was built using exact match over a certificate chain built using key match or name match, even if other factors should have lead to a different certificate chain being selected. In Windows XP, the weight assigned to an exact match was reduced so that other factors could result in a key match or name match-built chain being selected as the best quality chain.

- **Key match**. If the AKI extension only contains public key information, then only certificates that contain the indicated public key in the Subject Key Identifier (SKI) extension will be chosen as valid issuers. Figure 10 shows a scenario where key matching was used to find the issuing CA certificate. Note that the hash of the public key in the AKI extension matches for the certificate on the left matches the hash of the public key in the AKI extension of the certificate on the right.

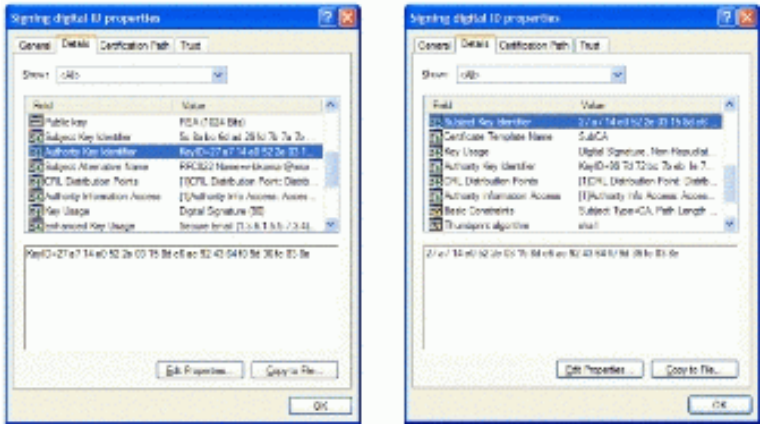

Figure 10: A Key Match

See full-sized image.

Note: The Public Key information in the AKI extension and in the SKI extension is the hash of the public key. Therefore, both hashes must have been calculated using the same algorithm. Otherwise, no key match will be determined even though the PK used for the hashes matches. The default hash algorithm used by the Microsoft CA and CryptoAPI is SHA-1 when no SKI exists in the certificate. However, when an SKI exists, both MD5 and SHA-1 algorithms are supported but the AKI and SKI must use the same algorithm.

- **Name Match**. If no information exists in the AKI, or if the AKI does not exist in the certificate, then the certificate will be marked as "name match" and name matching will occur. In name matching the subject name of a certificate must match the issuer name in the current certificate in order for the certificate to be chosen as a valid issuer. Because the data is stored in a binary format, the name matching process is case sensitive. Figure 11 shows name matching for a root CA certificate. The root certificate does not include an AKI extension, so name matching was used to match the issuer and subject attributes of the certificate (a self-signed certificate in this case).

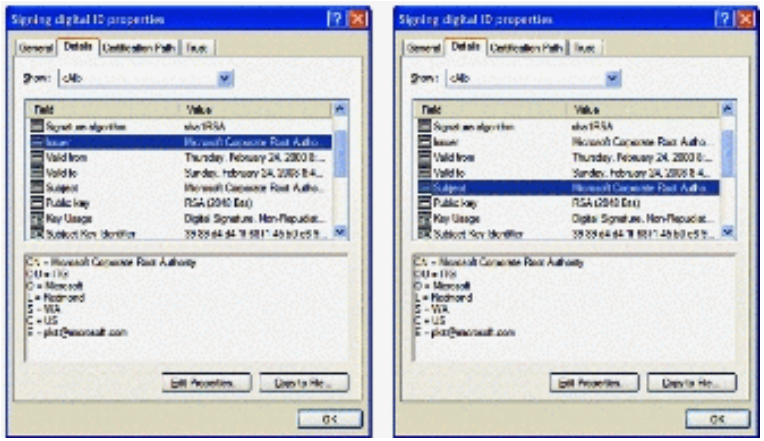

Figure 11: A Name Match
See full-sized image.

In all cases, even if a matching certificate is not found in the store, the current certificate will still be marked as "exact match", "key match" or "name match", since this is descriptive of the match attempted rather than the match attained.

Note: The exact path validation process used will be discussed in detail later in this paper.

Caching

To increase performance, the certificate chain engine uses a least-recently-used (LRU) caching scheme. This scheme creates a

cache entry for each certificate it encounters during its process of building the certificate chain. Each cache entry includes the status of the certificate so that the best certificate chain may be built from cached items on subsequent calls to the chaining API without having to re-determine the status of each certificate. Once a certificate has been added to the cache, it will not be removed until it expires or is revoked.

During the path validation process, valid cached certificates will always be selected. If valid cached certificates are not found, then a store search will be performed. For issuer certificates and CRLs, URL retrieval may be required to download the certificates and CRLS from the distribution point indicated in the URL.

All certificates are stored in cache when the certificates are selected from a store or from a URL. The only difference is the location where the cached certificates are stored. Certificates can be stored in:

- Memory. All retrieved certificates are cached in memory
- CA Store. All certificates retrieved from any WinInet-supported URLs (e.g. HTTP, FTP, LDAP, and FILE) via the Authority Information Access (AIA) extension are cached in the CA store. For more information on WinInet, refer to MSDN at http://msdn.microsoft.com/library
- Local File System. If the retrieval URL is LDAP://, FTP://, or HTTP://, then the certificate (or CRLs) is also cached by WinInet in the local file system. The actual location is the \Documents and Settings*username*\Local Settings\Temporary Internet Files folder.

Note: Caching settings cannot be modified or turned off.

In some cases, the certificate may be cached in all three locations. For example, a certificate retrieved from an http: URL will be cached in memory, the CA store and in the local file system by WinInet.

Revocation

The certificate chain engine performs basic revocation checking during chain building, but the process differs between Windows 2000 and Windows XP.

For Windows 2000, *post-processing revocation* is used. This means that the CRL checking is performed after the chain is built. The certificate chain engine will check each certificate in the chain to determine if a certificate is revoked and the reason for the revocation.

For Windows XP and Windows Server 2003, revocation checking is used as a determination option when building the certificate chain. Rather than waiting until the chain is built, each certificate is examined as the chain is build. The discovery of a revoked certificate in the chain will result in the chain getting assigned a lower quality value.

Note: The existence of a revoked certificate in a certificate chain does not preclude the chain from being presented to the calling application as the best quality certificate chain. The best quality chain may not necessarily be a trustworthy chain.

Certificate Path Validation

The path validation process ensures that a valid certification path can be established for a given end certificate. A valid certification path is defined as an end-entity (leaf) certificate that chains to a trusted root CA. Each node in the path must be discovered and subsequently validated until a trust anchor such as a root CA is obtained. During the validation process, a certificate can be deemed invalid, or not trusted, for many reasons. The reasons include:

- The certificate is not time valid. This occurs when the start and expiration dates are improper, have not occurred yet, or are expired.

  Note: An expired CA certificate in the certification path reduces the quality of the path; it does not invalidate the path. In the Windows Server 2003 PKI, a certification path can be valid as long as the CA certificate was valid at the time the certificate was issued. For example, a third-party CA might issue a certificate with a lifetime that extends past the CA certificate's expiration date. After the CA's certificate expires, the certification path for the certificate is still valid and the certificate is trusted as long as all other validation criteria are met. In Windows 2000 however, CryptoAPI enforces time nesting rules by default where a child certificate must have a validity period shorter than the parent. Signatures, however, may be considered valid past the date of the certificate lifetime. Signature validation and processing is, however, outside the scope of this white paper.

  Note: Windows 2000 and Windows Server 2003 CAs never issue certificates with a lifetime that extends past the CA certificate's expiration date. The issued certificate will be configured to expire on or before the CA's certificate expiration date.
- The certificate is unrecognized. If the certificate format is improper, does not conform to the X.509 v1 - v3 standard for digital certificates, the certificate is discarded.
- The information in certificate fields is improper or incomplete.
- The certificate's digital thumbprint and signature fail the integrity check, indicating that the certificate has been tampered with or corrupted.
- The certificate's status was determined to be revoked.
- The issuing CA is not in either a trusted certification hierarchy or a Certificate Trust List (CTL).
- The root CA for the certification path is not in the Trusted Root Certification Authorities store.
- The certificate is not permitted for the intended use as specified in a CTL.
- The certificate contains a critical extension not understood by the application.

Critical Extensions

The CryptoAPI engine does not enforce critical extensions in certificates, only Certificate Revocation Lists (CRLs). The

CryptoAPI engine and programming model places the burden of parsing critical extensions on to the calling application. Therefore, an application must understand and enforce a critical extension when evaluating a certificate. The CryptoAPI revocation provider, however, does evaluate critical extensions in CRLs and therefore would reject a CRL with a critical extension that is not known or understood.

Certificate Status Checking

All certificates in a certificate chain may be processed to verify that none of the certificates is revoked. Certificate chain validation is of course optional from an application standpoint and may not be enforced by CryptoAPI. The Windows operating system by default checks certificate revocation status via certificate revocation lists, as the CRL processing engine is the native revocation provider included with CryptoAPI. When this functionality has been invoked each certificate in the certificate chain is checked against the CRL published in the CRL Distribution Point (CDP) extension in the certificate. If the certificate is found to be included in the CRL, the certificate is then considered revoked. Additionally, third-party revocation providers can be registered with CryptoAPI to add support for additional revocation status checking mechanisms protocols including OCSP, SCVP and XKMS.

When a certificate's status is verified using a CRL, several steps must be performed by an application to check the status of the certificates in the certificate chain. These steps are performed against each certificate in the chain. These steps include:

•Verify the signature of the CRL. The CRL must be signed so that the application can determine whether it trusts the CRL issuer to issue CRLs.

Important: Windows 2000 and Windows XP can only verify a CRL that was signed by the same private key used to sign the issued certificate. Windows 2000 and Windows XP do not support CRLs signed by an entity other than the CA that signed the issued certificate.

•Verify that the CRL has not expired. A CRL is considered expired if the current data is after the date contained in the next update field of the CRL. If the certificate that is being checked has expired, the application must verify that the CRL's issue date follows the effective date of the certificate, but precedes the certificate's expiration date.
•Search the list of revoked certificates to determine that either the target certificate is not included or its revocation date is after the effective date.

When a third-party revocation provider supporting OCSP has been registered, an OCSP responder will be used for certificate status checking; in this case, the process is slightly modified.

•Verify the response indicates the certificate is valid. A valid response indicates that the certificate has not been revoked.
•Verify the signature on the OCSP response. This activity includes developing and processing a path that establishes that the certificate issuer or a trust point trusted the responder for the express purpose of issuing responses

Neither Windows XP nor Windows 2000 contains an OCSP client component by default. However, a third party OCSP client may be installed as a revocation provider to the CryptoAPI. OCSP responders may be located using the AIA extension in the certificate as defined by RFC 2459. The Windows .NET certificate authority supports the OCSP responder location to be included in the AIA extension of certificates. Multiple revocation providers may be added to CryptoAPI depending on revocation requirements. For additional information on revocation providers, refer the Platform SDK at http://msdn.microsoft.com/library/

Note: For more information on OCSP, please see RFC 2560 at http://www.ietf.org/rfc/rfc2560.txt.

No matter what process is used to verify certificate validity, if the status check fails any of the above checks for any certificate in the certificate chain, then the certificate chain shall be rejected.

Basic Constraint Validation

Basic constraints allow an application to determine if a presented certificate is a CA or end-certificate. The basic constraint includes the ability for a CA to designate whether an issued certificate is a CA certificate, or an end-certificate. A CA certificate can then be used by the certificate chain engine to build certification paths.

An additional use of the basic constraint extension is to limit the maximum number of CA certificates that can be included under the given CA. For example, a path length of zero only allows end-entity certificates issued by that specific CA. Likewise; a path length of two in the basic constraints extension will only allow three CA certificates in a certification path. With this example, any certification paths discovered with more than three CAs in the path will be discarded.

Name Constraint Validation

A CA certificate can contain name constraints that are applied to all certificate requests made to the CA. Each request is compared to the list of permitted and excluded constraints to determine whether the certificate should be considered permitted, not permitted, excluded, or not defined.

Important: Name constraint validation can only be performed by Windows XP and Windows Server 2003 clients. Name constraints are not evaluated by Windows 2000 clients. If you require name constraints be applied, you can indicate that the extensions is critical, which should result in the chain being discarded by an application conforming to RFC 2459.

For example, a permitted constraint could allow all DNS names that end in yz.com. This would include DNS names such as yz.com and xyz.com. If you only wanted DNS names from the yz.com DNS name space, you could use the permitted constraint .yz.com. This constraint would permit x.yz.com but exclude xyz.com.

When name constraints are present in a CA certificate, the following rules are applied to the subject name and alternate subject name entries.

- If the name constraints extension exists in a CA certificate, then all name constraints should be present in the extension. Any name constraints that are not included are considered wild cards that will match all possibilities. For example, if the DNS name constraint were absent, the entry would be treated as DNS="".
- All name constraints will be considered. There is no precedence applied to the listed name constraints. It is for this reason that name constraints that are not present are treated as wildcards.
- An exclude name constraint will take precedence over a permitted name constraint
- Name constraints are applied to the Subject name extension and any existing Subject Alternate Name extensions.
- Name constraints apply to all names contained in an end entity certificate. Each name in the subject or subject alternate name extensions should match at least one of the name constraints listed for that name type. A subject name or subject alternate name that does not match a listed name type will be rejected. Note that most client name spaces are not included in a CA certificate and generally do not apply.
- Name constraints are case sensitive if the names are stored in an ASCII or Unicode format.

Name restrictions must be enforced across the following alt name info entries in the subject name: Other Name (NT Principal Name only); RFC822 Name; DNS Name; URL; Directory Name and IP address.

Important: Microsoft does not currently support maximum path lengths for name constraints and policy constraints.

When the certificate chain engine validates an end certificate for name constraints, it will arrive at one of the following results:

- **Permitted**. The end certificate contains a name that is listed as permitted in an issuer's name constraints extension.
- **Not permitted**. The end certificate contains a name that is not listed as permitted in an issuer's name constraints extension.
- **Excluded**. The end certificate contains a name that is listed as excluded in an issuer's name constraints extension
- **Not Defined**. The issuer certificate does not list a constraint for a specific name type (such as Directory Name or IP Address)
- **Note**: When a constraint is defined, it is the resultant name space of all CA certificates in the chain and not just the issuer. Name constraint and other validation rules are defined in RFC 2459.

Policy Constraint Validation

A policy constraint allows a CA administrator to ensure that specific constraints are met when a certificate is issued or used by an application. The policy constraint ensures that all certificates issued by the CA implement the required policy constraints.

By definition, a root CA implements all policies. This applies to both Enterprise and Standalone CAs. At some point down the hierarchy, a CA can have one or more policies defined. Once a CA certificate is encountered with any policy OIDs, then all

certificates below that CA in the hierarchy must also have a subset of those policy OIDs. A certificate chain with no valid policy set will be considered invalid, whereas one with no policy OIDs at all will be considered valid and matching the "any policy" OID. The above is valid only for Application Policies and not Issuance Polices. For issuance policy, the absence of the certificatePolicies extension in a non-root certificate implies no issuance policy.

There are two policies currently used with a Windows .NET CA: Issuance policy and application policy. Differences apply in determining validity of a certificate chain when issuance and application policies are applied.

Issuance Policy

Issuance policies, referred to as the certificatePolicies extension in RFC2459, allow a CA administrator to define the circumstances and requirements for certificate issuance. The assurances required for certificate issuance can vary based on certificate templates. An issuance policy defines a set of administrative rules than must be followed when issuing a certificate. The rules are defined in the certificate by an object identifier (OID) defined at the CA. Each certificate issued by the CA will include the OID. The OID is not defined in the CA, but instead in the certificate templates.

Important: Issuance policy is only available to Windows Server 2003 and Windows XP clients. Issuance policy is not recognized by Windows 2000 clients.

When an entity presents a certificate to an application, the application can examine the certificate to verify the issuance policy and determine if the issuance policy is sufficient to perform the action requested by the end entity.

Currently, two types of constraints are defined: Require explicit policy and inhibit policy mapping.

- Require explicit policy specifies the number of certificates that can exist in the hierarchy below the current certificate before an explicit policy must exist.
- Inhibit policy mapping specifies the number of additional certificates that may appear in the path before policy mapping is no longer permitted.

Policies can also be mapped to other policies on a one-to-many basis. Policy mapping allows interoperability between two organizations that implement similar policies, but have deployed different OIDs. If one company's policy OID (e.g.; 1.2.3.4) stands to mean for some specific function, and another company's policy OID (e.g.; 11.22.33.44) stands to mean for the same function, then they could be mapped, so that 11.22.33.44 would be mapped to 1.2.3.4. For more information on this feature and qualified subordination, refer to the Planning and Deploying Qualified Subordination white paper.

Application Policy

Certificates provide key information that is not specific to an application. However, the ability to decide which certificates can be

used for certain functions is important. Application policy allows you to issue certificates widely and restrict their usage to only the intended purposes.

Application policies are settings that inform a target that the subject holds a certificate that can or cannot be used to perform a specific task. They are represented in a certificate by an object identifier (OID) that is defined at the certification authority. This OID is included in all issued certificates. When a subject presents its certificate, it can be examined by the target to verify the application policy and determine if it can perform the requested action.

Application policy is Microsoft specific and is treated much like Extended Key Usage.

•If a certificate has an extension containing an application policy and has an Extended Key Usage (EKU) extension, then the EKU extension is ignored.
•If there is only an EKU extension then that is treated like an application policy extension.

If there is an application policy extension (no EKU extension) and an EKU property on the certificate, then the intersection of the EKU property OIDs and the application policy OIDs is taken and the result is the effective policy for the certificate (just like for the EKU extension).

Note: If the application policy extension is absent, CryptoAPI will function like any other RFC 2459 compliant client. If it is present, CryptoAPI will implement the application policy rules. It is recommended that implementers make the extension non-critical extension so that its presence should be benign to other clients.

Important: Policy mapping can only be used with the application policy extension. If the Extended Key Usage (EKU) extension is used, then policy mapping is not possible, as EKU does not support policy mapping. Non-Windows clients and applications may not understand this extension or use as designed.

Chain Building in Different PKI Architectures

The CA architecture that you deploy will vary how the certification path is built by the chain building process. This section will look at chain building in different CA models. The models discussed include:

•Single CA
•Hierarchical CA
•Cross-Certification
•Bridge CA

Single CA

The single CA is the most basic of PKI architectures. In this scenario, a single CA provides all certificates and CRL information for an organization as shown in Figure 12.

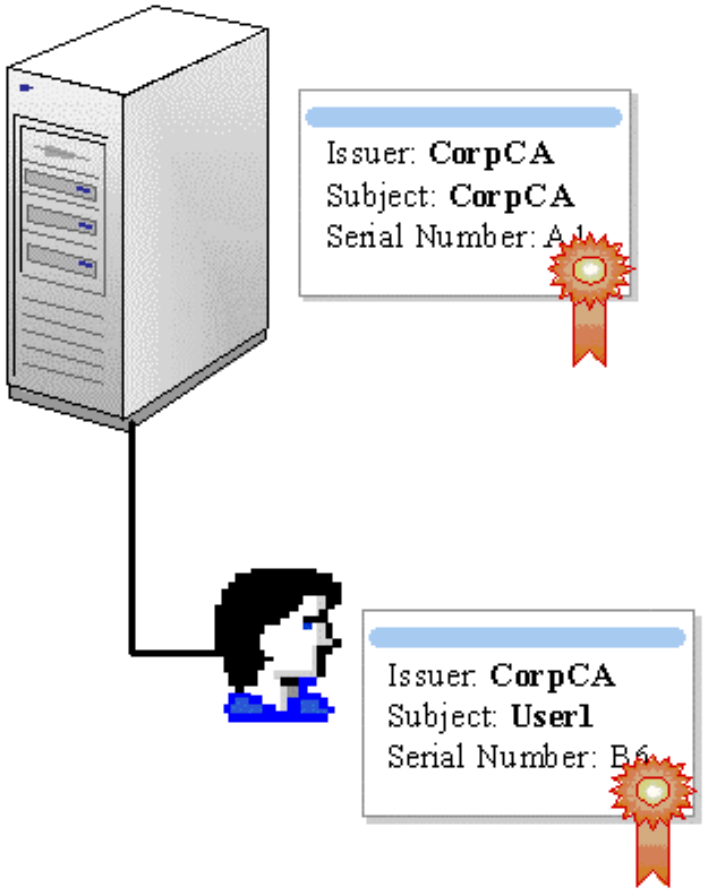

Figure 12: Certificate Chain in a Single CA structure

In a single CA architecture, all certificate chains will be two certificates deep in length. The root certificate for the CA will be the start of the chain, and the chain will terminate at the issued end certificate. Yet, it is still possible for multiple chains to exist for a single end certificate.

Multiple certificate chains are possible when a CA renews its certificate.

- If the CA's certificate is renewed using a new public/private key pair, a name match or a key match will link the end certificate to both the previous root CA certificate and the renewed CA certificate when name matching is used because the root CA's certificate will still have the same Subject name and match the issuer's name in the end certificate.
- If the CA's certificate is renewed using the same public/private key pair, the certificate chaining engine will produce two chains for an end certificate when both name matching and key matching are used by the certificate chaining engine to build certificate chains. Key matching will now produce two certificate chains because the public key material is the same on both versions of the CA's root certificate.

Important: An exact match will chain to the root CA certificate that was used to sign the end certificate in all cases because the serial number for a renewed CA certificate always changes from the previous serial number.

Hierarchical CA

In a hierarchical CA structure, two or more CA's are organized in a structure with a single root CA and one or more subordinate CA's as shown in Figure 13.

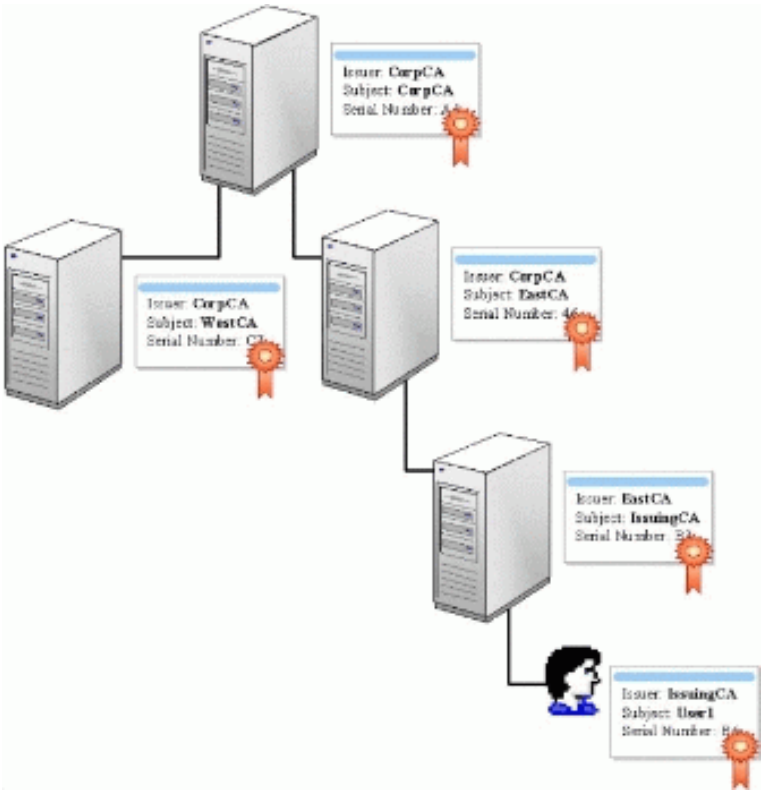

Figure 13: A Hierarchical CA structure

See full-sized image.

In this CA structure, it is easily seen that the certificate path from the Corp CA to the User1 certificate is CorpCA =>EastCA=>IssuingCA=>User1. Yet, it is still possible for multiple certificate chains to exist if any of the CAs in the certificate path renews their certificates.

In each case, the renewal of a CA certificate will result in more than one certificate path being generated for the end certificate. For example, if the EastCA certificate was renewed with a new serial number of 57 using the same public/private key and the IssuingCA certificate was renewed with a new serial number of B7 using a new public/private key pair, the following certificate chains could be generated for the User1 end certificate.

•CorpCA (serial #: A1) => EastCA (serial #: 46) =>IssuingCA (serial #: B3) => User1 (serial #: B6)
•CorpCA (serial #: A1) => EastCA (serial #: 57) =>IssuingCA (serial #: B3) => User1 (serial #: B6)
•CorpCA (serial #: A1) => EastCA (serial #: 46) =>IssuingCA (serial #: B7) => User1 (serial #: B6)
•CorpCA (serial #: A1) => EastCA (serial #: 57) =>IssuingCA (serial #: B7) => User1 (serial #: B6)

Only through the path validation process will the best chain be found for the User1 certificate.

Cross-Certification

Cross-certification allows two organizations to establish a trust relationship between their PKI infrastructures. There are several different ways that organizations can cross-certify their PKI hierarchies. For example, Figure 14 shows cross-certification between root CAs.

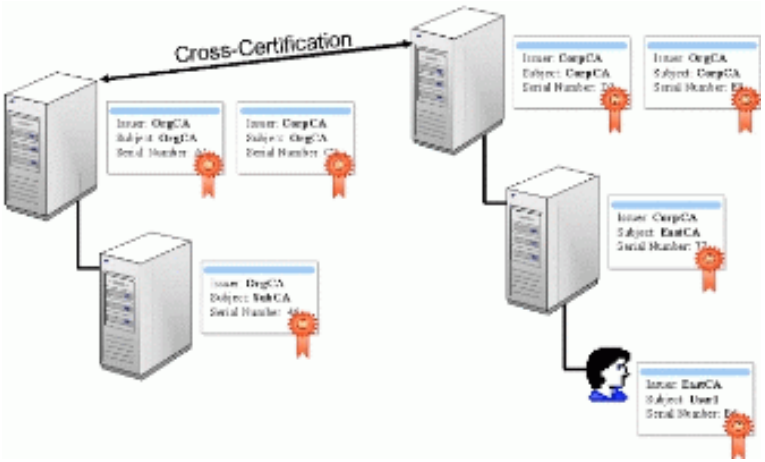

Figure 14: Cross-Certification Structure

See full-sized image.

In this example, as with the simple CA and hierarchical CA structures, multiple certification paths can be built if any of the CAs in the certification path has renewed their certificates. In addition to these certification paths, additional paths can now be built through the cross-certification. For example, the User1 certificate can be viewed with two different paths:

•CorpCA (Serial#: D3) => EastCA (Serial#: 77) =>User1 (Serial#: B6)
•OrgCA (Serial#: A1) => CorpCA (Serial#: E9) => EastCA (Serial#: 77) =>User1 (Serial#: B6)

The certificate chaining engine will always prefer chains that chain to a trusted root authority. Depending on whether the user or computer validating the certificate chain trusts the OrgCA root or the CorpCA root will determine which certificate chain will be selected by the certificate chain building engine. For additional information, refer to the Planning and Deploying Qualified Subordination white paper.

Important: The Windows 2000 and Windows Server 2003 certificate chaining engine is configured to not propose paths that contain the same certificate more than one time. This prevents the cross-certification path from being presented more than once in a certification path. For example, this configuration prevents the path CorpCA=>OrgCA=>CorpCA=>EastCA=>User1 from being proposed.

Alternatively, the cross-certification may take place between subordinate CAs, rather than between root CAs as shown in Figure 15. The actual design may vary depending on specific organizational or business requirements.

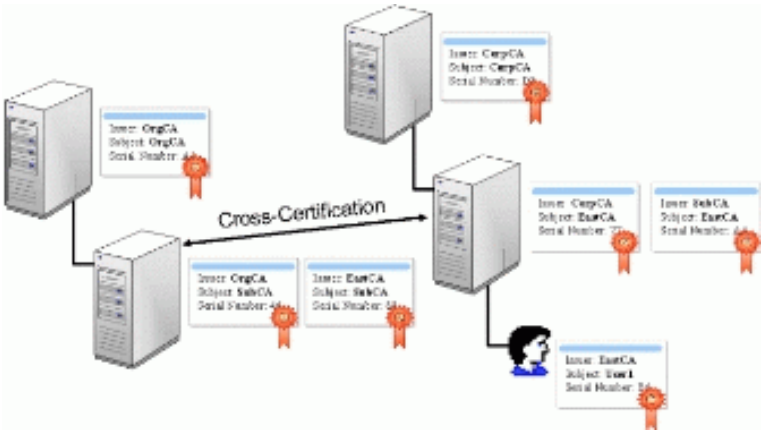

Figure 15: Cross-Certification between Subordinate CAs
See full-sized image.

In this cross-certification example, two different certification paths can be built for the User1 certificate:

•CorpCA (Serial #: D3)=>EastCA (Serial #: 77)=>User1 (Serial #: B6)
•OrgCA (Serial#: A1)=>SubCA (Serial #: 46)=>EastCA (Serial #: AA)=>User1 (Serial #: B6)

Important: In this scenario, additional certificate paths can be generated if any of the CAs in the certificate path has renewed their CA certificates.

Note: For more information on deciding where cross certification should be established between two CA hierarchies, please see the Windows Server 2003 Resource Kit Deployment Guide chapter, *Designing a Public Key Infrastructure*.

BridgeCA

A bridge CA structure takes cross-certification between two organizations and extends the model to allow multiple organizations to configure trust between their CA hierarchies. In a bridge CA structure, one CA becomes the hub or bridge for the trust between the CA hierarchies. Figure 16 shows a bridge CA that links three separate CA hierarchies.

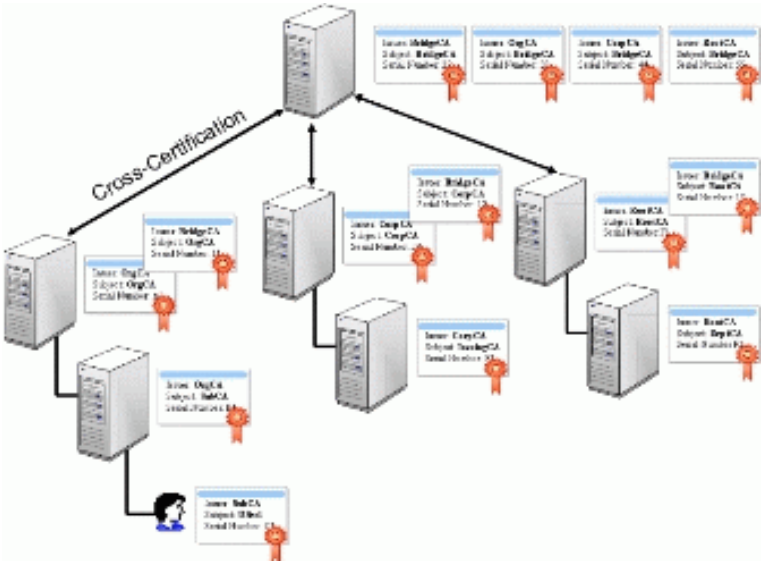

Figure 16: A Bridge CA Hierarchy
See full-sized image.

In this scenario, several different certificate paths can be built for the User1 certificate.

•OrgCA (Serial #:A4) =>SubCA (Serial #:B1)=>User1(Serial #:C1)
•BridgeCA (Serial #:22)=> OrgCA (serial #:11) =>SubCA (Serial #:B1)=>User1(Serial #:C1)
•CorpCA (Serial #:D1)=>BridgeCA (Serial #:44)=> OrgCA (serial #:11) =>SubCA (Serial #:B1)=>User1(Serial #:C1)
•RootCA (Serial #:F1)=>BridgeCA (Serial #:55)=> OrgCA (serial #:11) =>SubCA (Serial #:B1)=>User1(Serial #:C1)

In addition to the certificate paths, additional paths can exist if any of the CAs in the shown certificate paths have renewed their CA certificates.

Note: A bridge CA deployment requires a level of trust between the partner organizations and the organization hosting the bridge CA. A common scenario where this scenario is deployed is a large organization with largely independent subsidiaries. The bridge CA allows PKI interoperability between the subsidiaries, yet allowing private management of the individual CA hierarchies. For more information, refer to the Planning and Deploying Qualified Subordination white paper.

Certificate Discovery Process

One of the most difficult aspects of a PKI deployment is the management of certificate discovery for path validation purposes. The Windows operating system and Active Directory provide a level of integrated support to abstract the certificate discovery process from users and applications. The process of certificate discovery, however, differs between Windows 2000 and Windows XP. This section details the exact processes used by Windows 2000 and Windows XP to discover CA certificates for path validation.

Windows 2000

In a Windows 2000 domain, the certificate discovery process is completed as follows:

1.Certificates defined in Group Policy are applied and loaded into the Local Machine certificate store. The root CA certificates are published in the Computer Configuration\Windows Settings\Security Settings\Public Key Policies\Trusted Root Certification Authorities policy.

2. Certificates defined in the Enterprise policy are loaded into the Local Machine certificate store. The Enterprise policy includes certificates stored in the NTAuth store.

   Note: The NTAuth store is created and populated during the setup of Enterprise CAs and by using the DSSTORE command in Windows 2000 or the Certutil command in Windows XP. The NTAuth store designate CAs which are capable of issuing certificates for use in smart card logon and enroll on behalf of behavior .The NTAuth store is found at the following distinguished name:

   CN=NTAuth,CN=Public Key Services,CN=Services,CN=Configuration,DC=*ForestRootDomain*,DC=*tld*

   Note: By default, for performance reasons, every eight hours, the certificates defined in Group Policy and the NTAuth stores are downloaded and any new certificates are added to the machine's certificate cache.
3. If a certificate in the user's personal store does not have CA certificate from the same issuer and is not revoked or expired, the CA certificate will be retrieved using Authority Information Access (AIA) pointers in the end certificate,
4. During the process of certificate enrollment and certificate validation, all subordinate and the root CA certificate are cached in the user or computer store.

Windows XP and Windows Server 2003

In a Windows Server 2003 domain, a Windows XP client will use the following certificate discovery process:

1. Certificates defined in Group Policy are applied and loaded into the Local Machine certificate store. There is a change in behavior in that all CA certificates published in Active Directory, rather than just root CA certificates, are downloaded to the Machine store.
2. Certificates defined in the NTAuth store are loaded into the Local Machine certificate store. The Enterprise policy, stored in the Configuration naming context and available to all domains in a forest, can include both root CA and subordinate CA certificates.
3. If a certificate in the user's personal store does not have CA certificate from the same issuer, then the certificate will be retrieved using Authority Information Access (AIA) pointers in the end certificate.
4. During the process of certificate enrollment and certificate validation, all subordinate and the root CA certificate are downloaded to the user or computer store.
5. CA certificates stored in other directories referenced by cross certificates are downloaded every eight hours. The cross certificates act as certificate repository pointers.
6. If the Update Root Certificates component is installed, updated root certificates are downloaded from the Windows download site periodically.

Path Validation Process

When the chain building engine must determine which chain to use, the certificates currently in the machine store and/or user store are used to perform the process of path validation. Path validation is comprised of two phases. In the first phase, the certificate chains are assembled by finding the certificate of the CA that issued an end certificate. This process is repeated until all certificates available have been checked or each chain ends in a self-issued or root certificate. As with the certificate discovery process, the path validation process differs between Windows 2000 and Windows XP.

Windows 2000

In a Windows 2000 domain, a Windows 2000 client will use the following path validation process using all certificates acquired during certificate discovery:

1. If the AKI of the current certificate of being evaluated contains the issuer's user name and issuer serial number, only certificates that match on user name and serial number will be selected. This selection method is known as an *exact match*.

   Important: If an end certificate contains information in Subject Name and Serial Number information in its AKI field, then Windows 2000 will only perform an exact match. Even if the issuing CA's certificate can be found using a name match or a key match, the search will fail if an exact match is not possible.
2. If the AKI only contains public key information, then only certificates that contain the matching public key in the SKI extension will be chosen. This selection method is known as a *key match*.
3. If there is no information in the AKI, or the AKI does not exist in the certificate being evaluated, a certificate whose subject name matches the evaluated certificate's issuer name will be chosen. This selection method is known as a name match.

   Important: In Windows 2000, an exact match was given a larger weight than a key match or name match. This resulted in a certificate chain selected using an exact match to always be selected over any chains built using key matches or name matches.
4. Once all chains are built, the path validation process will perform revocation checking on all certificates in the possible chains. This is known as post-processing revocation checking
5. If application policy is defined, any defined Extended Key Usage (EKU) constraints are applied. The EKU constraints are applied from the root CA to the end certificate.

Windows XP and Windows Server 2003

A Windows XP client will use the following path validation process using all certificates acquired during certificate discovery:

1. If the AKI of the current certificate of being evaluated contains the issuer's user name and issuer serial number, only certificates that match on user name and serial number will be selected. This selection method is known as an *exact match*.
2. If the AKI only contains public key information, then only certificates that contain the matching public key in the SKI extension will be chosen. This selection method is known as a *key match*.

3. If there is no information in the AKI, or the AKI does not exist in the certificate being evaluated, a certificate whose subject name matches the evaluated certificate's issuer name will be chosen. This selection method is known as a name match.

   Important: In Windows XP, an exact match only applies a slightly larger weight than a key match or name match. This allows a certificate chains built using name matches or key matches to be selected over a chain built using an exact match, if the chains meet other criteria such as issuance policy, or revocation checks, better than the exact match chain.
4. Certificate status checking is performed during the path validation process, rather than after the chains are assembled. If a certificate in the chain is found to be revoked or expired, the chain is not discarded; the chain is only weighted less than a chain without a revoked or expired certificate.
5. If application policy is defined, application policy is applied using the application policy extension. The application policy extension allows the use of application policy mapping. If the application policy extension is does not exist in the certificate, then the Extended Key Usage (EKU) extension is used. If the EKU extension is used, application policy mapping cannot be used because the EKU extension does not support application policy mapping.
6. If issuance policy is defined, the issuance policy is evaluated starting at the root certificate to the end certificate.
7. If name constraints are defined, the name constraints are applied starting at the root certificate to the end certificate.

Once the path validation process is completed and a chain is identified as the best chain, if the best chain contains a revoked or expired certificate, the certificate discovery process is initiated to acquire any missing certificates. If no additional certificates are found, the path is not valid, and the certificate action fails. Revocation checking is, of course, the responsibility of the calling application and not CryptoAPI.

If at the end of the path validation process two certificate chains are found to be equivalent in weight, then the following process is used to select one certificate chain over the other:

1. A certificate chain with issuance policy defined is selected over a certificate chain without issuance policy.
2. A shorter chain will be selected over a longer chain. For example, when using cross-certification, you can end up with multiple, equivalently weighted chains of differing length, that chain to a root CA that you trust. In this case, the shortest chain would be selected.
3. The newest chain will be selected. Starting at the end certificates, the issuance date will be compared between the certificate chains, and the most recently issued certificate will be selected. If the end certificates were issued at the same time, the process is repeated at the issuing CA certificate of the end certificate, until one chain is determined to be newer than the other chain.

Certificate Trust Lists

A Certificate Trust List (CTL) allows a Web site to restrict from which CAs a Web site will accept a client certificate. The CTL is a predefined list of certificates that is signed by a trusted entity. The CTL includes either the hashes of certificates or a list of the actual certificate names. In most cases, the CTL is a list of hashed certificate contexts.

A CTL allows an administrator to limit the purposes that a certificate issued by an external CA can be used for, and limit the validity period of those certificates. Typically, CTLs are defined when an organization does not manage its own CAs or the company must trust external CAs for certificate services.

Important: While a CTL is commonly used in Windows 2000 to restrict what purposes an external CA's issued certificates can be used for, in Windows Server 2003 it is preferred to use Qualified Subordination to restrict external certificate usage. Qualified Subordination allows more complex rules to be defined for external certificate usage.

Once a CTL is defined, the CTL can be applied to client computers using Group Policy in Active Directory. Any computers located in the Group Policy container where the Group Policy Object is applied will use the CTL to limit certificate usage.

Top of page

Certificate Revocation Lists

A certificate revocation list (CRL) is a list, created and signed by a certificate authority (CA), which contains serial numbers of certificates that have been issued by that CA and are currently revoked. In addition to the serial number for the revoked certifications, the CRL also contains the reason for revocation for each certificate and the time the certificate was revoked.

The serial number for each revoked certificate is kept in the CA's database and published in the CRL until the certificate expires. After the revoked certificate is expired, the certificate's entry in the CRL is removed and the CA may remove the certificate from its database. Typically, the revoked certificate will remain in the CRL for one publication period after the certificate expires.

Windows 2000 and Windows Server 2003 only supports the practice of a CA signing a CRL. There is no support for the CA using a separate key for signing a CRL.

Revocation Reasons

When a certificate is revoked, it is possible for a certificate issuer to specify why the action was taken. This is done by specifying a revocation reason; these reasons are defined by RFC 2459 and include:

- KeyCompromise. The token or disk location where the private key associated with the certificate has been compromised and is in the possession of an unauthorized individual. This can include the case where a laptop is stolen, or a smart card is lost.
- CACompromise. The token or disk location where the CA's private key is stored has been compromised and is in the possession of an unauthorized individual. When a CA's private key is revoked, this results in all certificates issued by the CA that are signed using the private key associated with the revoked certificate being considered revoked.

- AffiliationChanged. The user has terminated his or her relationship with the organization indicated in the Distinguished Name attribute of the certificate. This revocation code is typically used when an individual is terminated or has resigned from an organization. You do not have to revoke a certificate when a user changes departments, unless your security policy requires different certificate be issued by a departmental CA.
- Superseded. A replacement certificate has been issued to a user, and the reason does not fall under the previous reasons. This revocation reason is typically used when a smart card fails, the password for a token is forgotten by a user, or the user has changed their legal name.
- CessationOfOperation. If a CA is decommissioned, no longer to be used, the CA's certificate should be revoked with this reason code. Do not revoke the CA's certificate if the CA no longer issues new certificates, yet still publishes CRLs for the currently issued certificates.
- CertificateHold. A temporary revocation that indicates that a CA will not vouch for a certificate at a specific point in time. Once a certificate is revoked with a CertificateHold reason code, the certificate can then be revoked with another Reason Code, or unrevoked and returned to use.

  Note: While CertificateHold allows a certificate to be "unrevoked", it is not recommended to place a hold on a certificate, as it becomes difficult to determine if a certificate was valid for a specific time.
- RemoveFromCRL. If a certificate is revoked with the CertificateHold reason code, it is possible to "unrevoke" a certificate. The unrevoking process still lists the certificate in the CRL, but with the reason code set to RemoveFromCRL.

  Note: This is specific to the CertificateHold reason and is only used in DeltaCRLs.
- Unspecified. It is possible to revoke a certificate without providing a specific reason code. While it is possible to revoke a certificate with the Unspecified reason code, this is not recommended, as it does not provide an audit trail as to why a certificate is revoked.

Revocation Checking

The process of revocation invalidates a certificate before its end validity date using one of the revocation codes mentioned in the previous section. A Microsoft CA publishes certificate status information in the form of CRLs. Third party products use this information to provide revocation information in other formats such as OCSP, SCVP and XKMS.

Prior to checking the status of a certificate, client must first checks a certificate to ensure that is trusted and it is time valid. Every issued certificate has a defined period in which the issuing CA will vouch for the validity of the certificate. The validity period is defined using two fields within the issued certificate.

- NotBefore. This field defines the date and time on which the certificate's validity period begins.
- NotAfter. This field defines the last date on which the certificate is considered valid by the issuing CA. The NotAfter date will never be set to a date later than the NotAfter date defined in the issuing CA's certificate.

If CRL based status checking is used and the certificate is found to have a valid time, the presented certificate is examined to see if it contains a CRL Distribution Point (CDP) extension. The CDP extension indicates both the protocol that must be used to retrieve the CRL (HTTP, FTP, LDAP, or FILE) and the location where the CRL is stored (represented as a URL).

Using this information, CryptoAPI first searches the local certificate stores and the local cache for any CRL signed by the issuer (Certification Authority) of the certificate being validated. A cached version of a current CRL will always be used (as long as it is valid), rather than downloading the same CRL again. The following logic is used to evaluate the CRL:

- If a CRL is found, and the certificate's serial number is listed in this CRL then the certificate will be considered revoked.
- If the CRL is expired and the certificate is listed in the CRL with any reason other than *certificate hold*, the certificate will be considered to be revoked and no attempt to retrieve a new CRL will be performed.
- If the certificate is not listed in the CRL, or the revocation reason is *certificate hold*, then a new CRL will be retrieved from the URLs listed in the certificate's CDP. The new CRL is fetched only if it is past the NextUpdate field in the currently held CRL. The new CRL is checked to determine if the certificate is revoked. If the original reason was certificate hold, the CRL is checked to determine if the certificate is unrevoked by looking for the remove from CRL revocation code.
- If the CRL cannot be obtained, the client will generate a "Server offline" error.

CRL Format

A version 2 CRL includes the following fields:

- Version. The version of formatting used for the CRL. Typically, this is version 2.
- Issuer. The issuing CA's Distinguished Name (DN) represented using an X.500 DN such as cn=TestCA, ou=PKI,dc=nwtraders, DC=com.
- This Update. The date that this CRL was issued. UTCTime format is used for date up to the year 2049. GeneralizedTime format is used for dates in the year 2050 and beyond.
- Next Update. The date at which the next CRL will be issued. The same date formatting rules are used as for This Update.

- Revoked Certificates. A sequence of zero or more revoked certificates with the following fields represented for each revoked certificate.

  - Certificate Serial Number. The serial number assigned by the issuing CA for each revoked certificate.
  - Revocation Date. The date at which the revocation took place.
  - CRL Entry Extension. An optional field that contains version 2 extensions for CRLs that provides additional information about a single CRL entry.
  - ReasonCode. Identifies the reason for the certificate revocation. Reasons can include: unspecified, keyCompromise, cACompromise, affiliationChanged, superseded, cessationOfOperation, certificateHold, and removeFromCRL.
  - HoldInstructionCode. This option, only defined in the PKIX format, indicates the action to be taken if the ReasonCode is defined as CertificateHold. This extension is not supported by the Windows .NET certificate authority.
  - InvalidityDate. The date on which it is known, or suspected, that the private key associated with the certificate was compromised.
  - CertificateIssuer. An optional field that indicates the CA that issued the certificate if indirect CRLs are implemented. The Windows .NET certificate authority does not use this extension, but it is supported by Windows XP clients.
  - AuthorityKeyIdentifier. A numeric representation of the issuer name and serial number from the CRL issuer's certificate that is used in the certification chain building process
  - IssuerAltName. An optional field that allows alternative identities to be associated with the issuer. This can contain a DNS name or a Uniform Resource Indicator (URI). The Windows .NET certificate authority does not use this extension, but it is supported by Windows XP clients.
  - cRLNumber. A monotonically increasing sequence number that indicates the version of the CRL since the first CRL was published.
  - IssuingDistributionPoint. Identifies the CRL distribution point if CRL partitioning is implemented. For example, the CRL may be partitioned so that end-entity, CA certificate, or specific reason code revocations are published at a specific distribution point. The Windows .NET certificate authority does not use this extension, but it is supported by Windows XP clients.

    Note: Windows Server 2003 does not support partitioning CRLs by reason code.
  - deltaCRLIndicator. Indicates that the CRL is a Delta CRL. This also specifies the minimum CRL number of BaseCRL with which the delta CRL can be combined to obtain the complete state of the CA.
  - Signature (Issuer Signature Algorithm). Identifies the algorithm used to certify the CRL.
  - SignatureValue. The actual digital signature computed using the indicated Issuer Signature Algorithm on the ASN.1 DER encoded CRL. The value is encoded as a BIT String and omits the signature algorithm and SignatureValue fields.

Note: Windows 2000 and Windows XP do not support the HoldInstructionCode, InvalidityDate, CertificateIssuer or IssuerAltName extensions in CRLs. If the extensions are not marked critical, they will be ignored. If these extensions are marked critical in the CRL, CryptoAPI will reject the CRL and invalidate the certificate being processed.

CDP Extensions

When any CA issues a cert, it may place a CDP extension in the new certificate that points to the relevant CRL for the certificate containing the extension only.

For example, the root CA issues certificates with a CDP that points only to the root CA's CRL. A first level subordinate CA issues certificates with a CDP that points only to the first level subordinate CA's CRL. Finally, a second level CA issues certificates with a CDP that points only to the second level CA's CRL.

When the default revocation provider is used, the full certificate chain must be built first, then each certificate in the chain is searched for a CDP extension to fetch a CRL that will prove the revocation status of that one certificate only. The Windows client does not support a certificate status model where one CA can supply information about revoked certificates that were issued by a different CA, because the key that was used to sign the certificate in question must be the same key used to sign the CRL being used to verify revocation state of the certificate

The CDP may point to both primary and alternative locations for CRL retrieval. When a client must acquire the latest CRL, the CDP extension in a certificate is checked and the CRL retrieval is attempted by contacting the URLs in the CDP. If the name resolution of a Fully Qualified Domain Name fails, or the CRL is unavailable, the client tries the next CRL CDP from the CDP extension field

Note: If the client is able to resolve the name of the CDP but the CRL is not physically available it may take a significant period of time before the timeout threshold is reached. Windows XP provides enhancements in reducing the timeout threshold as well as CRL unavailability detection.

If the CDP extension is not available in a certificate, then CryptoAPI will only check the local stores and cache for a CRL. If a local CRL is available, it will be checked despite the absence of a CDP extension. However, a specific application must make the decision whether or not to demand a revocation check on a certificate.

CRL Publication

A CDP URL can specify several protocols for CRL access. These include:

•HTTP – Hyper Text Transfer Protocol Web sites can store the updated CRLs.
•LDAP – A Lightweight Directory Access Protocol (LDAP) store can contain the updated CRLs.
•File – CRLs can be accessed using Server Message Blocks (SMBs)
•FTP – File Transfer Protocol servers can be used to distribute updated CRLs.

The CRL, by default, is published in two locations by a Microsoft Enterprise CA.

•http://*CAName*/certenroll/*CRLName*
•LDAP:///CN=*CAName*,CN=*CAComputerName*,CN=CDP,CN=Public Key Services,CN=Services,CN=Configuration,

DC=*ForestRootDomain*,DC=*TLD*

When a CRL is published, the CRL will take on different names, depending on whether the CA's certificate has been renewed, and whether the existing private keys are re-used when the certificate is renewed. Table 1 outlines some examples of the CRL name that is automatically generated for a Microsoft CA.

Table 1 Auto-generated CRL file names

Scenario	Name of CRL file	Name of Delta CRL file	CA Version
A certification authority named "IssuingCA" that has never had its CA certificate renewed	issuingca.crl	issuingca+.crl	0.0
A certification authority named "IssuingCA" that has been renewed once with the same key	issuingca.crl	issuingca+.crl	0.0
A certification authority named "IssuingCA" that has been renewed once with a new key	issuingca.crl issuingca(1).crl	issuingca+crl issuingca(1)+.crl	2.2
A certification authority named "IssuingCA" that has been renewed 3 times, twice with a new key	issuingca(3).crl	issuingca(3)+.crl	3.3

Changing the CRL Distribution Point during Installation

There are specific scenarios where it is required to modify the default CDP URLs for a CA. The two most common cases where the CDP must be modified are:

•The deployment of an offline CA. An offline CA is not available on the network. To enable CRL checking, the updated CRL must be published to a location that is online and available to network users.
•Allowing external access to CRL information. When a CA issues certificates that are to be used on the Internet, the CRL for the CAs in those certificates certificate chain must be available to external users. By modifying the default CDPs for the CAs, the CRLs can be made available to external users.

Two locations may be edited to change CDP locations. For an offline root CA, you must modify the CDP for the root certificate

by implementing a capolicy.inf file at installation time. The capolicy.inf file allows you to modify the URLs contained in the CDP of the root CA's certificate. Remember that this certificate is created during the installation process of the standalone CA and it is important to note that a change will require renewal of the CA.

Modifying the Capolicy.inf configuration file

The capolicy.inf file is used during the setup of a root CA. It must be created prior to the setup of the CA. The capolicy.inf file determines the CDP location before the self-signed root CA certificate is generated. This means that the CDP defined in the policy file affects the extensions in the root certificate. When creating the capolicy.inf file to define AIA and CDP extensions, use the following configuration examples:

- Ensure that the modified capolicy.inf file is stored in the %systemroot% folder. The installation of the CA only looks in this folder for the existence of the capolicy.inf file.
- In the capolicy.inf file, add all required CDPs to the [CRLDistributionPoint] section as shown below:

```
[CRLDistributionPoint]
 URL = "ldap:///CN=%%7%%8,CN=%%2,CN=CDP,CN=Public Key Services,
       CN=Services,CN=Configuration,DC=northwindtraders,DC=com"
URL = "http://www.microsoft.com/Public/%%3%%8%%9.crl"
 URL = "D:\WINNT\System32\CertSrv\CertEnroll\%%3%%8%%9.crl"
 CriticAL = true
```

- In the capolicy.inf file, add all required CDPs to the [AuthorityInformationAccess] section as shown below:

```
[AuthorityInformationAccess]
 URL = http://%1/Public/My CA.crt
 URL = ftp://northwindtraders.com/Public/MyCA.crt
 URL = file://\\%1\Public\My CA.crt
```

Important: The order in which you list the URLs in the CRLDistributionPoint and AuthorityInformationAccess sections is the order in which the URLs will appear in the CDP extension of the root certificate.

- Do not delete the URL = "D:\WINNT\System32\CertSrv\CertEnroll\%%3%%8%%9.crl" URL line from the capolicy.inf file. This line is required as it defines where the CRL will be published locally on the offline CA. Remember that this is the version of the CRL that you will manually post to the other location referenced in the [CRLDistributionPoint] section if the CA is to be an offline CA.
- Ensure that once you have modified the CDP extension that you copy the updated CRL to each URL location so that it is available on demand.

Changing the CRL Distribution Point After CA Installation

If a *capolicy.inf* files was not used to setup the CA the CRL CDP can be modified through the CA MMC user-interface or the *certutil.exe* utility.

If the CDP is modified, the CA has to be restarted and all certificates issued after the CDP change will now incorporate the new CDP information. This does not mean that the old CRL CDP location becomes invalid because older certificates may have the previous CDP explicitly defined.

The following commands may be used to modify the CA configuration in a way to adjust the CDP and the AIA locations. Remember that the %-sign has to be replaced with a double-%-sign if used in a batch-script.

```
: [1] CRL CDP
:
certutil -setreg policy\RevocationCRLURL "http://cdp1.mydomain.com/%3%8.crl"
certutil -setreg policy\LDAPRevocationCRLURL "ldap://cdp2.mydomain.com/CN=%7%8,CN=%2,
CN=CDP,
CN=Public Key Services,CN=Services,CN=Configuration,Domain Controller=mydomain,
DC=COM?certificateRevocationList?base?objectclass=cRLDistributionPoint"
certutil -setreg policy\FileRevocationCRLURL ""
:
: [2] AIA
:
certutil -setreg policy\IssuercertURL "http://cdp1.mydomain.com/%1_%3%4.crt"
certutil -setreg policy\LDAPIssuercertURL "ldap://cdp2.mydomain.com/CN=%7,
CN=AIA,CN=Public Key Services,CN=Services,CN=Configuration,DC=mydomain,
DC=de?cACertificate?base?objectclass=certificationAuthority"
certutil -setreg policy\FileIssuercertURL ""
```

Important: The order in which you list the URLs in the CRLDistributionPoint section is the order in which the URLs will appear in the CDP extension of the root certificate.

Path recommendations for CRL CDP's

It is recommended to use abstract hostnames when building the FQDN for the CDP. Furthermore, a separate FQDN should be used for every CDP even if both CDP's as kept on the same machine currently. This makes it much easier to physically change the CRL location in the future. The CRL is not bound to a dedicated machine.

A suggestion would be to define a FQDN like cdp1.mydomain.com that points to the http-location of the CRL and a FQDN *cdp2.mydomain.com* that points to the LDAP CDP. Another though would be to put the type of CRL access protocol into the name.

Using Variables in the Capolicy.inf file

The Capolicy.inf file allows variables to be used as place holders when defining CDP and AIA extensions. In a capolicy.inf file, the following variables can be used:

· **%1. SERVERDNSNAME**	· **%8. CRLFILENAMESUFFIX**
· %2. SERVERSHORTNAME	· %9. CRLDELTAFILENAMESUFFIX
· %3. SANITIZEDCANAME	· %10. DSCRLATTRIBUTE
· %4. CERTFILENAMESUFFIX	· %11. DSCACERTATTRIBUTE
· %5. DOMAINDN	· %12. DSUSERCERTATTRIBUTE
· %6. CONFIGDN	· %13. DSKRACERTATTRIBUTE
· %7. SANITIZEDCANAMEHASH	· %14. DSCROSSCERTPAIRATTRIBUTE

Note: The variable definitions are different between Windows 2000 and Windows .NET. For a list of variable definitions in Windows 2000, please refer to Knowledge Base article 283119.

Modifying the CDP Extensions for Issued Certificates

In addition to modifying the CDP extension for the root CA's certificate, you must also ensure that offline CAs use a modified CDP for all issued certificates. This includes both offline root CAs and offline subordinate CAs. When modifying the CDP extension for issued certificates, keep the following best practices in mind.

•You cannot edit a URL once it is created. Be sure to verify the URL for a new CDP before you approve the addition.
•Do not delete the URL that references C:\WINDOWS\System32\CertSrv\CertEnroll \<CRLName><CRLNameSuffix><DeltaCRLAllowed>crl. This URL is required as it defines where the CRL will be published locally on the offline CA.
•Enter the new CDP URLs in the order that you want searches to take place. At this time, you cannot re-order the URLs in the interface.
•For an offline Windows Server 2003 CA, ensure that the CA only publishes Base CRLs, and does not publish Delta CRLs. This is accomplished by changing the properties of the CA so that Delta CRLs are not published, as shown in Figure 17

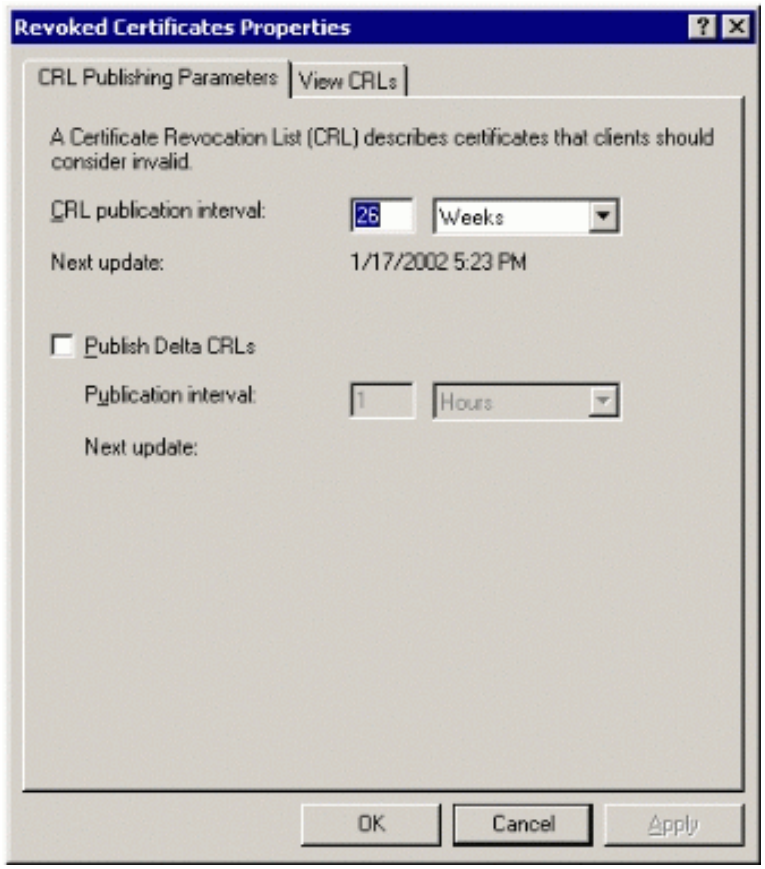

Figure 17: Preventing the Publication of Delta CRLs
See full-sized image.

Scheduling certificate revocation list (CRL) publication

Each CA is configured with a CRL publication setting. This setting defines when a CA will automatically publish an updated CRL known as the CRL publish period. When a CA is first installed, the publish period is set to one week, but can be manually configured.

A CRL is valid for a period that differs from this publish period. The validity period is the period of time that a CRL is considered authoritative for verifying an issued certificate. The validity period is extended to a length of time greater than the publication period to allow for Active Directory replication. By default, the validity period is defined to be 10% greater than the publication period, up to a maximum of 12 hours difference. For example, if your CRL publish period is set to 10 days, and then the validity period is set to 11 days. In addition, the validity period must be at least 1.5 times the skew value. Therefore, if the

skew value is defined to be 10 minutes, then the validity period must be a minimum of at least 15 minutes.

You can alter the default settings by modifying the CRLOverlapPeriod and CRLOverlapUnits values located in the registry in the HKLM\ SYSTEM\CurrentControlSet\Services\CertSvc\Configuration\<CA Name>\ hive. For example, to define validity period to be extended by two days, you would set CRLOverlapPeriod to be a value of "days" and CRLOverlapUnits to be a value of "2".

Note: It is recommended to modify these registry values using Certutil –setreg, rather than directly modifying the registry. The following command(s) are provided as examples:

certutil -setreg ca\CRLOverlapPeriod days

certutil -setreg ca\CRLOverlapUnits 2

Finally, there is a clock skew of an additional 10 minutes added to the validity period on either side of the publish period, so a CRL will be valid 10 minutes before the beginning of its publish period to account for variances in computer clock settings. You can modify this setting by changing the value of ClockSkewMinutes in the same registry location.

CRL Caching

The CRL is not retrieved every time that a client must validate a certificate issued by a CA. Once a client retrieves a CA's CRL from the CDP, the CRL is then stored in the client's local cache. The cached version of the CRL will be used until the CRL's validity period expires. This is true, even if a new CRL is manually published.

The only way to delete cached CRLs is by deleting all temporary Internet files history in Internet Explorer.

Note: This behavior changes if Delta CRLs, discussed later in this paper, are implemented.

Manually Publishing CRLs

It is possible for an administrator to publish a CRL before the next scheduled publish period. The manual publication does not alter the previously defined CRL publication period. The manual publication does replace the existing CRL with an updated CRL , but the CRL will still be published at the end of the current publish period. CRLs may be published using the certification authority MMC snap-in on the CA.

Remember that if a client has a cached copy of the previously published CRL, the client will continue using the cached version until the CRL's validity period expires. The client will only download the updated CRL if the client does not have a cached copy

of a valid CRL.

Enabling Netscape-compatible Web-based revocation checking

Netscape Internet browsers may use specific revocation check extensions to be included in an issued certificate. Although it is not recommended, to enable the Netscape-compatible Web-based revocation check extensions to be added to every certificate, you must run the certutil command at the issuing CA with the following flags:

```
certutil -SetReg Policy\RevocationType +AspEnable
```

Once the parameter is set, you must then stop and start the Certification Authority service. All certificates issued by the certification authority after the parameter is set will contain the extension.

Disadvantages of CRLs

While CRLs provide a method of determining certificate validity and are an ideal mechanism for wide distribution of revocation information to clients and additional services, several disadvantages must be considered when determining the publication interval for a CA's CRL.

1. Many applications require up to date revocation status information. This requires the CA to frequently publish a new CRL. The longer the publication interval, the more likely that a client will use a cached version of an out-of-date CRL.
2. A CRL is the entire list of the serial numbers of revoked certificates, so for CAs with a large population of users this could become a very large list (30 thousand revocations might reach 1 MB in size). If the publication interval is set to a short period of time, then the entire CRL must be downloaded to the client at the publication interval.
3. Even if there are no changes, a CA has to republish the entire list so that applications have the latest information, which involves a lot of repetition.
4. Frequent publication of large objects will in turn generate a large amount of directory replication traffic.

Top of page

Delta CRLs

One of the biggest decisions faced by a CA administrator is determining the publication schedule for CRLs. If a CA publishes a complete CRL frequently, then clients are aware of a newly revoked certificate very quickly. However, this causes higher amounts of network traffic due to the more frequent downloading of the updated CRL to all clients. If a CRL publishes CRLs less often, this reduces the amount of network traffic, but increases the latency before a client is aware of a newly revoked certificate.

If a CA revokes a large number of certificates, the size of the base CRL can grow to be larger than 1 MB in size if large numbers of certificates are revoked. If the CRL is published at frequent intervals, this can result in problems for clients connecting over slow connections. Alternatively, if the base CRL is published at longer intervals, this can result in the CRL information being out of date and reducing the validity of the CRL information.

Delta CRLS, defined in RFC 2459, address these problems, by publishing changes to a Base CRL (bCRL), in a smaller file known as a Delta CRL (sCRL). When Delta CRLs are implemented, a client can download a Base CRL at longer intervals, and then download smaller Delta CRLS at shorter intervals to validate any presented certificates. The Delta CRLs can be published at very short intervals, such as once an hour, to increase the confidence in the certificates being validated. All of the time information stored in CRLs is stored as UTC.

Note: This does not eliminate the requirement to download the larger Base CRLs. The Base CRL must be downloaded initially and when the previous Base CRL expires. The Delta CRL can force the client to retrieve a more recent Base CRL even though the current Base CRL is still time valid. This is achieved by having the Delta CRL point to a higher number Base CRL.

When Delta CRLs are implemented, only changes from a Base CRL are published in a Delta CRL, resulting in a reduction in the size of the CRLs downloaded to the clients. This reduction in size allows for more frequent publishing of the CRL with both a minimal impact on the network infrastructure, and an improvement on the up-to-datedness of CRL information.

Figure 18 shows how Delta CRLs are implemented.

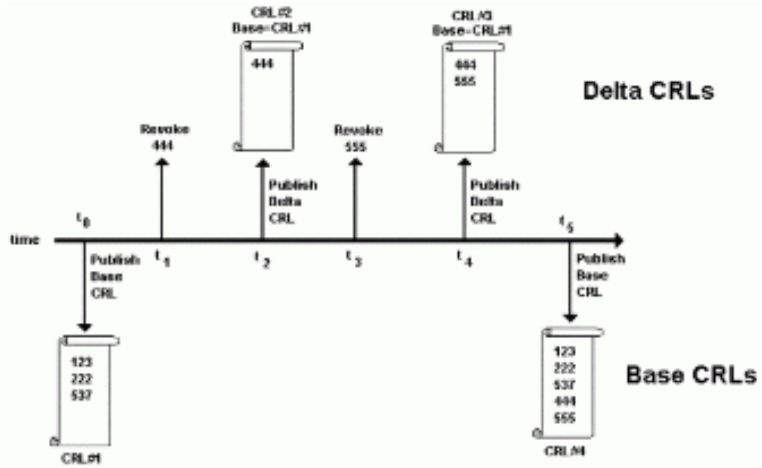

Figure 18: Delta CRL Processing
See full-sized image.

In this example the Initial Base CRL (CRL#1) is published at time t_0 with three revoked certificates. At time t_1 , the certificate with serial number 444 is revoked. When the Delta CRL (CRL#2) is published at time t_2, the Delta CRL indicates that it is to be used with Base CRL CRL#1. At time t_3 , a second certificate with serial number 555 is revoked. When the updated Delta CRL (CRL#3) is published at time t_4, the Delta CRL now contains the serial numbers for both 444 and 555.

Finally, when the Base CRL is updated at time t_5, the Base CRL (CRL#4) will now include the serial numbers for certificates 444 and 555. Any new Delta CRLs will now only include certificates that have been revoked since Base CRL CRL#2 was issued at time t_3.

Note: The Delta CRL process is very similar to a differential backup strategy. As a differential backup will include all files that have changed since the last full backup, a Delta CRL contains all revoked certificates since the last Base CRL was issued. The delta CRL will reference the base CRL according to the algorithm below:

1.The primary algorithm for delta publication will refer to the latest fully propagated Base CRL
2.If such a Base CRL is not available, the delta CRL will refer to the oldest, non-expired Base CRL.

CRL Changes

In order to support Delta CRLs, some new optional extensions defined in RFC 2459 have been added to the CRL. The extensions include:

- **CRL Number**. This non-critical extension is a monotonically increasing sequence number for each CRL issued by a CA. As shown in Figure 18, the CRL Number is increased for each issued CRL, no matter whether the CRL is a Base CRL or a Delta CRL. The extension is added to both the Base CRL and the Delta CRL.
- **Delta CRL Indicator**. This critical extension, only found in a Delta CRL, contains the minimum Base CRL that can be used in conjunction with the Delta CRL. If the only Base CRL available is less that the Base CRL indicated in the Delta CRL Indicator, then the Base CRL cannot be used with the Delta CRL. If this scenario takes place, the chaining engine will then attempt to retrieve an updated Base CRL.
- **Freshest CRL**. This non-critical extension lists the issuers and locations from which to retrieve the Delta CRLs. If the Freshest CRL is not present in both the CRL and the presented certificate, then the Base CRL will be treated as a regular CRL, not as part of a Base CRL/Delta CRL pair.
- **CRL Publication Period**. This non-critical extension provides information to the client about how frequently a CRL is published. This extension indicates to the client how frequently they should look for a new CRL. This extension will contain the number of minutes before the CA is scheduled to publish an updated CRL. If the Base CRL or the Delta CRL has a CRL publication period extension, this extension will govern when a new Base CRL or Delta CRL will be retrieved.

Note: Presently, a Microsoft Object ID (OID) is used for this extension. It is known as the nextPublish extension. If a standard OID is defined in the IETF, this extension will be changed to the standard OID.

In addition to additional the extensions added to CRLs to support Delta CRLs, some additional changes were made to CRLs to support the implementation of Delta CRLs. These changes include:

- **Certificate Hold Processing**. If a certificate is revoked with the status *CertificateHold*, it can be subsequently be un-revoked. With Delta CRL's this requires a new entry on the CRL *RemovedFromCRL* to indicate that the entry exists on the Base CRL, but has subsequently been un-revoked.
- **distributionPoint**. Any names in this field are enumerated and compared to any name in the subject certificate's CRL Distribution Point. If any one match is found, the CRL is considered valid for the subject.
- **"Inline" Revocation**. Window XP and Windows Server 2003 use "inline" revocation as the certificate chain is being built. This is a change from Windows 2000, where revocation checking was done after the best quality chain was built (known as post-process revocation checking). This change eliminates situations where revoked chains could be picked over available non-revoked chains.
- **Freshness Period**. Application running under Windows XP or Windows Server 2003 will be able to pass a freshness period when requesting a CRL. The freshness period specifies, in seconds, the maximum permissible age of the base or delta crl (this age is determined by taking the difference between the current time and the ThisUpdate field in the crl). If the crl is older than the freshness period, then a new crl *must* be retrieved.

How Delta CRLs Work

If certificate revocation checking is invoked, CryptoAPI will in the case of the default revocation provider examine a presented certificate for a CDP that indicates where the Base CRL is published. If Delta CRLs are implemented, the Base CRL will contain the Freshest CRL extension, causing that the chain engine to attempt to retrieve the Delta CRL indicated by the Freshest CRL.

Note: When calling the chain-building engine, the calling application can specify the policy or target for the revocation freshness information. The policy can, for example, specify that revocation information may be as old as 8 hours, so if a BaseCRL (or Delta CRL) is found that was published only 6 hours previously, then the chain-building engine will not attempt to retrieve a new Delta CRL or Base CRL.

The retrieved Delta CRL has its own identifying number, plus the Delta CRL Indicator extension indicates the number of the Base CRL that must be used with the Delta CRL.

The revocation provider may look for an updated Delta CRL once the publication period has elapsed. This is determined by comparing the current time to the CRL This Update and CRL Publishing Interval. If the Current time is greater than or equal to the time indicated by adding the CRL Publishing Interval to the CRL This Update, then a new Delta CRL should be retrieved.

When Delta CRLs are implemented, the following results can occur during revocation checking:

- If the Delta CRL cannot be retrieved for some reason, the revocation result returned to the certificate chaining engine will be *revocation offline*.
- If freshness policy is implemented and the Base CRL falls within the period defined for freshness, then no Delta CRL retrieval is attempted.
- If the Delta CRL's Delta CRL indicator is less than the current Base CRL, or the Delta CRL cannot be reached, or the Delta CRL is time-invalid, a warning of *no delta available* should be returned to the certificate chaining engine.
- If the Delta CRL's CRL indicator is less than the Base CRL number, the chain engine should attempt to retrieve a new Base CRL
- If there is more than one sCRL in the cache, and the revocation information in the bCRL is older than what is specified by policy, then the first sCRL that is in the cache that meets the freshness policy will be used, even if there are more recent sCRLs.

Active Directory Considerations

The Windows Server 2003, by default, will store a copy each CRLs in Active Directory in the following location:

```
CN=CATruncatedName,CN=CAComputerName,CN=CDP,CN=Public Key
Services,CN=Services,CN=Configuration,DC=domain,DC=tld
```

While the publishing of the CRLs to Active Directory allows propagation of the CRL using Active Directory replication and ensures that a client can connect to a local copy of the CRL using site information, there are circumstances where the combination of CRL publishing schedules and Active Directory replication can result in issues arising.

Base CRL Publication Planning

The basic underlying premise with Delta CRLs is that the current Base CRL number will always be greater than or equal to the Base CRL number found in the Delta CRL Indicator extension of a Delta CRL.

This means that before a Delta CRL can reference a Base CRL, the Base CRL must be available to all clients. If the referenced Base CRL is not available, then the client will have to fall back to standard CRL processing.

To prevent this from occurring, the CA must be configured with a best estimate of the Directory Service replication time. The CA should not reference the updated CRL until

$$t^{Base}_{ThisUpdate} + t^{DS}_{Replication}$$

In addition to being concerned about a Base CRL not being available when referenced by a Delta CRL, you also must consider clients that are not running Windows XP or Windows Server 2003. These clients do not implement Delta CRL checking and will only reference Base CRLs for revocation checking. If you define the Base CRL publication period to be too long in this scenario, your older clients will have latency issues when determining revocation status.

Delta CRL Publication Planning

Typically, Delta CRLs will be published at more frequent intervals than Base CRLs. The limiting factor for Delta CRLs is the replication interval for Active Directory or the rate at which you want clients to be able to cache revocation information (if not using AD for deltas). Using the Active Directory for delta publication is not recommended because of the replication latency.

You cannot have a Delta CRL publication period that is less than the replication interval configured for remote sites in Active Directory. If this situation were to exist, clients would be unable to acquire an up-to-date CRL due to Active Directory replication latency.

You can get around this by implementing one the following changes in your Delta CRL publication configuration:

- Change the Delta CRL publication interval to a period that is greater than or equal to the Active Directory replication interval.
- Prevent the inclusion of the LDAP CDP in the Freshest CRL. By not including the LDAP path in the Freshest CRL, a client will not look to the Active Directory when acquiring a Delta CRL.
- Ensure that that the HTTP or FTP server reference in the Freshnest CRL is highly available. Consider publishing to a Web farm running Network Load Balancing Services or multiple Web servers that are referenced using DNS Round Robin addressing.

Removing LDAP URL from the Freshest CRL Extension

The Freshest CRL extension is used to identify how Delta CRLs are found. The same syntax is used for this extension as the CDP extension.

The following procedure can be used to remove the LDAP URL referencing the CRL in Active Directory from the Freshest CRL.

1. In Administrative Tools, open Certification Authority.
2. In the Certification Authority console, right-click the *CAName* (where *CAName* is the name of your CA), and then click Properties.
3. In the *CAName* Properties dialog box, click the Extensions tab.

4.In the Extensions tab, select the existing LDAP path in the list of CRL locations, and then clear the check box titled **Include in CRLS. Clients use this to find Delta CRL locations** as shown in Figure 19.

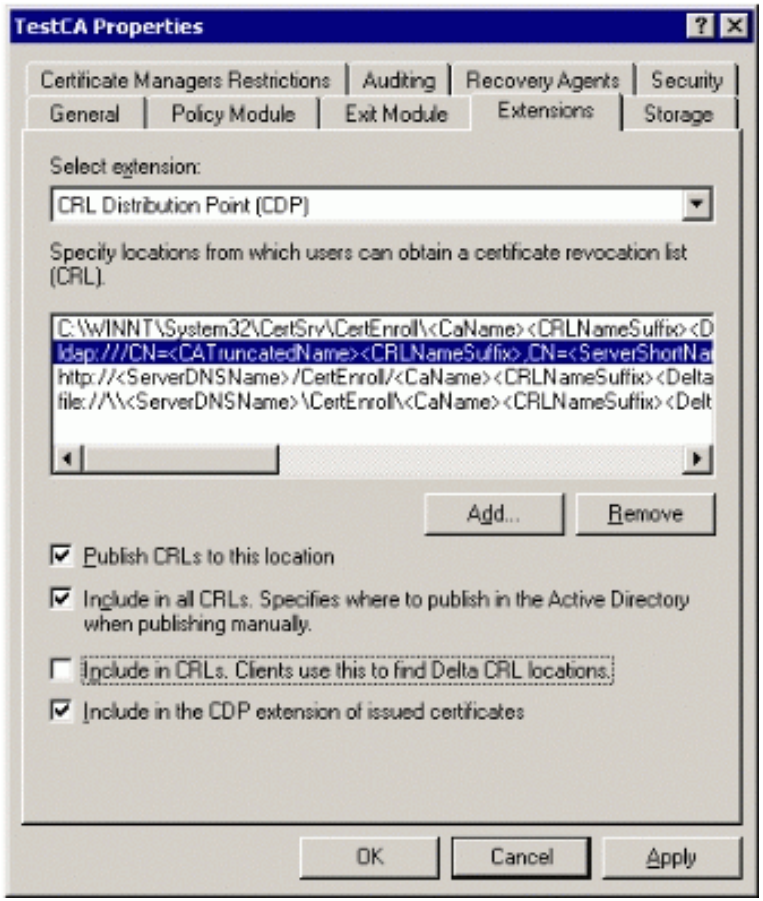

Figure 19: Removing the LDAP CDP from the Freshest CRL field.
See full-sized image.

5.In the CAName Properties dialog box, click OK.
6.In the Certification Authority dialog box, click Yes to restart Certificate Services.

Verifying the Removal of the LDAP URL from the Freshest CRL

Once the change has been made, the removal of the LDAP URL from the Freshest CRL can be verified by using the following procedure:

1. In the Certification Authority console, in the console tree, right click Revoked Certificates, click All Tasks, and then click Publish.
2. In the Publish CRL dialog box, click New CRL, and then click OK.

   Remember The Freshest CRL is found only in the Base CRL. The Freshest CRL indicates where the associated Delta CRLs for a Base CRL are found.
3. In the console tree, right click Revoked Certificates, and then click Properties.
4. In the Revoked Certificates Properties dialog box, click the View CRLs tab, and then click View CRL.
5. In the Certificate Revocation List dialog box, in the list of available Fields, select Freshest CRL.
6. Ensure that in the Value window that an LDAP URL does not exist. There should only be an HTTP URL.
7. In the Certificate Revocation List dialog box, click OK.
8. In the Revoked Certificates Properties dialog box, click OK.
9. Close the Certification Authority console.

Optimizing Delta CRLs

While in itself, Delta CRLs optimize the revocation checking process, you can further optimize the Delta CRL process by reducing the number of Base CRL fetches. This means that any client who has that oldest Base CRL will not be forced to download a new Base CRL until it expires. This minimizes the number of times a Base CRL is retrieved by the client, but increases the size of the Delta CRL. The Windows .NET Certificate Authority is primarily configured to ensure that the smallest Delta CRL sizes are used. If it is desired to optimize Base CRL usage, longer lifetimes should be applied to the BaseCRL publication period.

Publication Schedules

Both Windows 2000 and Windows Server 2003 allow the CA administrator to define the publication schedules for CRLs. The only difference between Windows 2000 and Windows Server 2003 is the support for Delta CRL publication in Windows Server 2003.

The publication schedule for CRLs is defined in the Certification Authority console using the following procedure:

1. Logon with an account having manage permissions on the CA.
2. From the **Start** menu, open **Administrative Tools**, then open **Certification Authority**.
3. In the console tree, right click **Revoked Certificates**, and then click **Properties**.

4. In the Revoked Certificates Properties dialog box, see Figure 20, you can define the publication intervals for both Base CRLs and Delta CRLs.

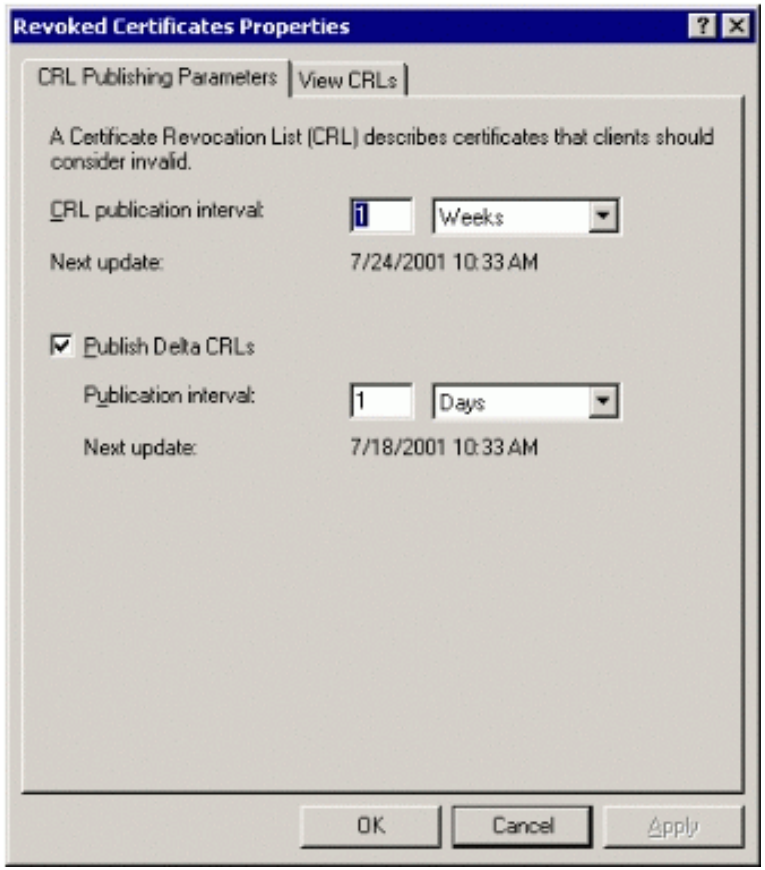

Figure 20: Defining the CRL publication schedule
See full-sized image.

Note: In Windows 2000, you can only define the CRL publication period for the Base CRL, as Windows 2000 does not support Delta CRLs.

5. In the Revoked Certificates Properties dialog box, click OK.

Manual Publication

There are circumstances that may require you to publish a CRL manually, rather than waiting for the next scheduled publication period. In these circumstances, such as wanting to immediately roll all Delta CRL information into a new Base CRL, you can

manually publish the Base CRL using the Certification Authority console.

Important: Remember that the manual publication of a new CRL will not cause clients to automatically download the new CRL. If the client has a cached version of the previous CRL, they will not download the updated version of the CRL until the cached version expires.

To manually publish a Base CRL or Delta CRL, use the following procedure:

1.From Administrative Tools, open Certification Authority
2.In the console tree, right click Revoked Certificates, click All Tasks, and then click Publish.
3.In the Publish CRL dialog box (see Figure 21), select New CRL to publish an updated Base CRL, or select Delta CRL Only to publish a new Delta CRL, and then click OK.

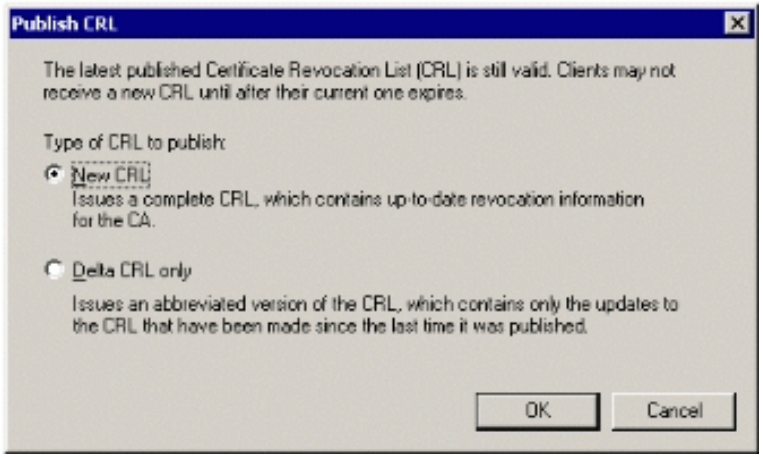

Figure 21: Manually Publishing Base and Delta CRLs
See full-sized image.

4.Close the Certification Authority console.

CDP Management & Publication Locations

In addition to managing when CRLs are published, CRL management can include changing where CRLs are published. Cases where you may wish to modify where CRLs are published include:

•Supporting an offline CA that will not be available for CRL publication
•Adding additional CRL publication points for CRLs that must be available to external clients

It is necessary to change the CDP locations for an offline root CA because the CA is not available on the network to distribute CRLs to requesting clients. In the case of an offline root CA, two modifications must be made:

- Before the Offline Root CA is installed, a capolicy.inf file must be configured that indicates the CDP and AIA information for the CA's self-signed CA certificate. The modified Capolicy.inf file must be placed in the %systemroot% folder before installing Certificate Services on the Root CA.
- Once the CA is installed, the CDP information must be changed in the Certification Authority console before any additional certificates are issued by the CA. For publishing CAs and Issuing CAs, this modification must also be completed if additional CDPs are required. This change must be performed before any certificates are issued so that all issued certificates have the corrected CDP information.

Note: The actual steps required to modify the CDP for an offline root CA are found in the Walk Through section of this White Paper.

Client Processing

Windows XP introduces several changes to how clients will interact with Delta CRL publication. Specifically, the following changes have been made:

- **Active Revocation Freshness Policy.** When calling the CertGetCertificatechain, the calling application can specify the policy or target for the revocation freshness information. The API will then make best efforts to meet this policy. If it succeeds, then it returns success, if not it returns an error as appropriate. If the code encounters information that meets the policy, it can terminate revocation checking at that point. For example, if the policy asks for CRL information to be considered valid for 8 hours and the code finds a Base CRL that was published 6 hours ago, then there is no need to check for Delta CRL's.
- **Revocation Freshness Property**. In the light of no policy as to the revocation freshness, the code will make best efforts to establish revocation freshness. On all calls, the value of the revocation freshness information shall be available to the calling application. Moreover, it shall be the latest "this update" value from the CRL.

- **Inclusion of the Issuing Distribution Point (IDP) extension**. The IDP extension allows *partitioned CRLs* to be deployed when using third-party CAs. Partitioned CRLs allow a third-party CA to publish CRLs with only specific certificate types within each CRL. For example, you can have separate CRLs for end certificates versus CA certificates. Specifically, the following options can be set in the IDP:

  - **onlyContainUserCerts**. This option in the IDP allows only certificates that do not have the values cA in the Basic Constraints extension. If the certificate does not contain a Basic Constraints extension, it is assumed it is not a CA.
  - **onlyContainsCACerts**. This option in the IDP allows only certificates having a Basic Constraints extension with cA set to be included in the CRL.

  Note: In Windows .XP, the onlySomeReasons and indirectCRL fields of the IDP are not supported. This prevents CRL partitioning by reason code.

  Note: Windows Server 2003 does not support partitioning CRLs by reason code or certificate type. If the IDP extension is present, CryptoAPI compares the names in the IDP and CDP extensions. If a successful match is made on a single name form, the CRL will be considered as valid for the certificate being validated.
- **Establishing if the CA Published Delta CRL's**. The client should check for the presence of the FreshnessCRL extension in both the certificate and CRL. If the extension is absent from both locations then current CRL processing based on CA publishing full CRL's is followed.
- **Delta CRL processing**. If the FreshestCRL extension is found, then it should be used to retrieve the CRL. The client should then examine both CRL's for the presence of the certificate serial number being checked.
- **Caching Delta CRL's**. As with Base CRLs, all retrieved Delta CRLs are cached using a Least Recently Used algorithm.
- **Looking for new Delta CRL's**. The client can start looking for a new Delta CRL once the publication period has elapsed from the *this update time* in the CRL i.e. Time now => CRL this update + CRL publish interval. The Windows XP client can use the CRLNextPublish field in the CRL to fetch the next Base or delta CRL accordingly.
- **New Delta CRL Error Conditions**. Delta CRLs introduce a new set of possible errors. These include the following:

  - If a Valid Base CRL exists and is available, but no delta, no delta for CRLNumber, or no time valid delta available, the certificate chaining engine should return a warning that no Delta CRL is available.
  - If the CRL Number in the Base CRL is less than the CRL Number in the available Delta CRLs, an error should be returned indicating that an updated Base CRL should be retrieved.
- **CertificateHold Processing**. The Windows XP client must be able to handle the condition of un-revoking a certificate. The processing will be dependent on the sequence that the CRL's are examined. If the Delta CRL is first checked, then RemovedFromCRL is equivalent to not appearing on the CRL and should return a response that the certificate is valid. If the Base CRL is checked first, then the Delta CRL must be checked to verify if the entry has been removed from the CRL.

Top of page

CryptoAPI Functions

On the client side, the Windows operating system and application utilize the CryptoAPI to perform certificate status checking and chain building. All applications that call CryptoAPI can ensure that proper chain building and certificate status is returned. This eliminates the need for application developers to write or create their own methods for performing these functions.

Note: At this time, only the Windows XP client supports Delta CRLs.

Fetching CRLs

The CryptoAPI fetches CRLs by using the CRLDistributionPoint extension in an X.509v3 certificate. The CRLDistrubtionPoint extension contains a sequence of one or more locations at which CRLs relevant to the presented certificate may be found. Typically, the locations are URLs that use one of the following protocols: LDAP, HTTP, FTP, or FILE.

Note: A FILE URL indicates that Server Message Blocks are to be used to access the CDP. This often has additional implications if the file(s) are protected by Access Control Lists (ACLs). The client machine running under the system account must have access to the location. Usually this means that anonymous access must be permitted on the file ACLs.

The CryptoAPI treats each URL in the CRLDistribtionPoint extension as a possible location for finding valid CRL information. For each URL in the CRLDistributionPoint extension, the CryptoAPI will perform the following tasks:

- Determine whether the content of the URL are cached locally using the WinInet cache. For more information on WinInet, refer to MSDN at http://msdn.microsoft.com/library
- If a cached version is found, the cached version is validated to ensure that it is time-valid, name-valid and signature-valid. If a valid CRL is found, the CRL is designated to be the definite source of revocation information.
- If none of the CRLDistributionPoint URLs is found cached locally, or if no valid cached CRLs are found, then the CryptoAPI will start looking for a valid CRL in the logical "CA" certificate store. For all name-valid CRLs found in the "CA" store, the first one that also passes time- and signature-validity checks will be accepted as the definitive source of revocation information for the certificate.
- In the event that a valid CRL for the certificate was not found in the "CA" store, the CryptoAPI will then proceed out onto the network to locate a valid CRL using the contents of the CRLDistributionPoint extension as hints to CRL locations.
- If a valid CRL was not found at any of the CRLDistributionPoint URLs (including network-related problems such as timeout, offline status, etc.) then processing terminates with an error denoting that current revocation information could not be found.

When programming with the CryptoAPI, CRL retrieval can be initiated in several ways. These include:

- Calling CertGetCertificateChain on a certificate with revocation flags set.
- Calling CertVerifyRevocation specifying a chain or a set of certificates

•Calling CryptRetrieveObjectByUrl to retrieve a CRL from a specific URI

Cached CRLs

Once the Crypto API has retrieved a CRL or Delta CRL for a Certification Authority, it will always use the WinInet (web) engine to cache a copy of that CRL locally. The cache remains valid for as long as the CRL is valid (Next Update). The cached copy of the CRL is stored in \Documents and Settings*UserName*\Local Settings\Temporary Internet Files.

The benefit of caching CRLs locally is that CryptoAPI will always look for a cached copy first to avoid traversing the network and introducing latency in the revocation status checking.

The disadvantage of local caching is that the client will not look for a new CRL or Delta CRL until the CRL has expired. So if a revocation has occurred on a Certification Authority and a new CRL published, the client may not use the updated CRL due to a locally cached copy.

Organizations may write logon scripts to delete local caches of CRLs depending on the requirements of the organization. As always, an increase in security may have effects on the infrastructure such as additional network traffic, client latency, etc so decisions should be made based on the total requirements and capabilities of the infrastructure.

Note: Base and Delta CRLs will be cached in memory for each application that calls the certificate status checking APIs . This may require an application to be restarted before the application will determine that a locally cached CRL no longer exists and must be fetched from the CDP location in the certificate.

For example, the following VBScript can be run as a logon script to delete the local cache on a client machine:

```
on error resume next
Set WshShell = WScript.CreateObject("WScript.Shell")
Dim regpath
Dim cachepath
regpath = "HKEY_CURRENT_USER \Software \Microsoft \Windows \CurrentVersion \Explorer
\Shell Folders\Cache"
WScript.Echo "Reading cache location from " & regpath
cachepath = WshShell.RegRead ( regpath )
If Len ( cachepath ) = 0 Then
   WScript.Echo "Cache value could not be found."
Else
   Dim ie5cache
   Dim fso
```

```
    Dim f
    Dim fc
    Dim file
    Dim subfolder
     ie5cache = cachepath & "\Content.IE5"
    WScript.Echo "Scanning " & ie5cache & " for CRL's"
    Set fso = CreateObject("Scripting.FileSystemObject")
    Set f = fso.GetFolder(ie5cache)
    Set fc = f.SubFolders
    For each subfolder in fc
        WScript.Echo "Searching subfolder " & subfolder.Path
        For each file in subfolder.Files
            if strcomp(right(file.Path, 4), ".crl", 1)=0 _
            Or instr(1, file.Path, "CN=", 1) > 0 Then
                WScript.Echo " " & file.Name
                file.Delete (True)
            End If
        Next
    Next
End If
```

Note: If you comment out the line **file.delete (True)**, the logon script will list all CRLs that are found, rather than delete all CRLs.

Top of page

Best Practices

CRL Deployment

The key to a successful deployment of a PKI infrastructure is planning. An organization must first understand the capabilities and characteristics of its existing network infrastructure and then develop guidelines and security practices for PKI that fit within the limitations and capabilities of the existing infrastructure. Key items to consider include:

•Replication latency of Active Directory
•Available network bandwidth for CRL download
•User requirements for latency due to status checking

An organization needs to develop a balance between the security goals of the enterprise and the capabilities of the overall IT infrastructure. For example, it is unrealistic to set certificate revocation latency goals of 8 hours if the replication latency of the Active Directory is 24 hours.

CRL Overlap

To configure fault tolerance against failures such as CRL publication failures, network outages, or replication latency, an organization may choose to create significant overlap in the validity period of published CRLs. CRL overlap can be set in the registry using the following registry values located in

HKLM\ SYSTEM\CurrentControlSet\Services\CertSvc\Configuration\<CA Name>\ hive.

•CRLOverlapPeriod. Defines the overlap period time frame that will be implemented for CRL publication.
•CRLOverlapUnits. Defines the actual units of CRLOverLap period that is implemented for CRL publication
•Clockskewminutes. Defines the period length to be added for overlap periods to allow for clock skews between clients.

The three registry settings work in conjunction to allow you to configure CRL publication to allow for Active Directory replication delays. For example, if you have to deal with an Active Directory replication delay of 3 hours, it would be wise to set the overlap period to be 4 hours. To allow the overlap period to be 4 hours, the following registry settings would be defined:

· CRLPeriod	REG_SZ = Weeks
· CRLPeriodUnits	REG_DWORD = 1
· CRLOverlapPeriod	REG_SZ = Hours
· CRLOverlapUnits	REG_DWORD = 4
· ClockSkewMinutes	REG_DWORD = a

The CRLPeriod and CRLPeriodUnits are defined in the Certification Authority console, while the CRLOverlapPeriod, CRLOverlapUnits, and ClockSkewMinutes are all defined by editing the registry. In this configuration, the resulting CRL lifetime would be 1 week, 4 hours, and ten minutes. This result is the sum of CRLPeriod, CRLOverlapUnits, and ClockSkewMinutes.

Algorithm to Compute Base CRL Publication

The following algorithm is used to calculate the Base CRL publication period:

```
If Base registry overlap period specified:
{
Start with Base registry setting rounded down to nearest minute multiple
}
else
{
Start with 10% of Base CRL period (1/10 period) rounded down to nearest minute
multiple
Maximum 12 hours
}
Minimum 1.5 times clock skew (usually 1.5 * 10 minutes)
Maximum 100% of Base CRL period
```

Algorithm to Compute Delta CRL Publication

The following algorithm is used to calculate the Delta CRL publication period:

```
If Delta registry overlap period specified:
{
Start with Delta registry setting rounded down to nearest minute multiple
}
else
{
Start with 100% of Delta CRL period (full period) rounded down to nearest minute
multiple
Maximum 12 hours
}
Minimum 1.5 times clock skew (usually 1.5 * 10 minutes)
Maximum 100% of Delta CRL period
```

CRL Partitioning

There are scenarios where the size of the Base CRL may grow to be too large and cause performance problems for clients performing revocation checking. You can partition the CRL for future certificates by renewing the CA's certificate using a new key pair. By using a new key pair, all certificates issued by the CA after the CA's certificate was renewed will be included in a separate CRL signed by the CAs new public key. The Microsoft CA manages certificate issuance to never exceed the lifetime of the issuing CA certificate. Once a certificate is expired, it is removed from the CRL as per RFC 2459, and it is no longer

considered valid or checked for revocation.

Note: The renewal of the CA certificate does not preclude the CA from creating CRLs for the certificates signed by the previous public key. The previous key pair is kept around and the CRL is kept up-to-date until the last certificate for the key pair expires. This is a distinct side-effect of CA renewal with a new key. Remember that when the CAs certificate expires, all certificates issued by the CA are considered expired.

Offline Certificate Authorities

In CA hierarchies, all trust flows from the root CA. If the root CA is compromised, this results in a certificates issued in the hierarchy being compromised as well. To increase the security of the root CA, it is recommended to keep the root CA disconnected, or offline, from the network and use subordinate CAs to issue certificates to end entities and other CAs.

To allow the root CA to be configured as an offline CA, you must install Certificate Services as a stand-alone root CA. This is required because an enterprise root CA requires access to Active Directory, which is not possible if the CA is removed from the network.

Note: In higher security networks, it is recommended to remove the second level of CA from the network as well as the root CA. This provides additional protection against CA compromise.

When configuring an offline CA, remember to include the following tasks in your deployment plan:

- For an offline root CA, modify CAPolicy.inf to change the CDP and AIA locations for the root CA certificate before installation to reference URLs that are always available.
- Ensure that the CDP and AIA information is changed in the CA's property pages to ensure that issued certificates reference available locations for AIA and CRL information.
- If certificates chaining to the offline CA are used externally, the CDP must include references to externally available locations.
- Ensure that the CA's certificate and CRL are manually copied to an available location referenced by the CDP and AIA information in issued certificates.
- Change the publication interval for Base CRLs to be a longer period. There is no prescribed interval, but remember that every time a new Base CRL is issued, the offline CA must be accessed and the updated CRL published
- Disable Delta CRL publication for all offline CAs.
- Ensure that the offline CA certificate's lifetime is a longer amount. Consider lifetimes of five years or more for offline CAs.
- Publish the offline root CA's certificate to active directory using the following commands:

  - Windows 2000: dsstore <DN of the root domain> -addroot <.crt file> <CA Name>
  - Windows .NET: certutil –dspublish –f <.crt file> <CA Name>

•Publish the offline CA's CRLcertificate to active directory using the following commands:

•Windows 2000: dsstore <DN of the root domain> -addcrl <.crl file> <CA Name>
•Windows .NET: certutil –dspublish –f <.crl file>

Fault Tolerance

Fault tolerance for CRL and certificate information can be increased by using the Active Directory for certificate storage. CRL and certificates are stored in the Configuration naming context, which is replicated between all domain controllers in the forest, no matter which domain they are located in.

By replicating the CRL and CA in the configuration naming context, this allows a client to query any domain controller for CRL and certificate retrieval.

If CRL or certificates must be published externally, consider publishing the CRLs and certificates to a Network Load Balancing Service (NLBS) cluster. This ensures the availability of the CRL and certificate information in the event that a single server in the cluster fails. In this scenario, the individual nodes in the cluster will share a common IP address that can be referenced by an externally available URL for external access.

Re-Signing a CRL

Another method for obtaining fault tolerance for a certificate authority in disaster recovery scenarios, is the CRL re-sign. A CRL may be re-signed using certutil.exe with the –sign parameter. Additions and deletions may also be made to the CRL using this command. The utility requires access to the CA's private key to re-sign the CRL and must be made available prior to the CA going offline. This utility is most useful in scenarios where a CA may not be restored to an operation state prior to the CRL expiration.

CRL Checking

It is the client application's responsibility to validate a presented certificate against the CRL. Depending on the client application, CRL checking may or may not be enabled by default. For example, smart card logon enables and requires CRL checking by default, while EFS encryption in Windows 2000 does not enable CRL checking at all.

In most cases, it is recommended to enable CRL checking so that the certificate-based authentication is validated for revocation status. Only in cases where the CRL may not be available should you consider not enabling CRL checking for an application.

Internet Explorer

By default, Internet Explorer does not check the revocation status of SSL certificates presented when connecting to an SSL-protected Web site. To enable CRL checking for Internet Explorer, follow these steps:

The default configuration settings of IE do not check the validity of SSL certificates. To enable CRL for IE follow these steps:

1.Open Internet Explorer.
2.From the Tools menu, click Internet Options.
3.In the Internet Options dialog box, select the Advanced tab
4.In the Settings box, scroll down to the list of Security settings. As shown in Figure 22, enable both "Check for publisher's certificate revocation" and "Check for server certificate revocation (requires restart)" and then click OK.

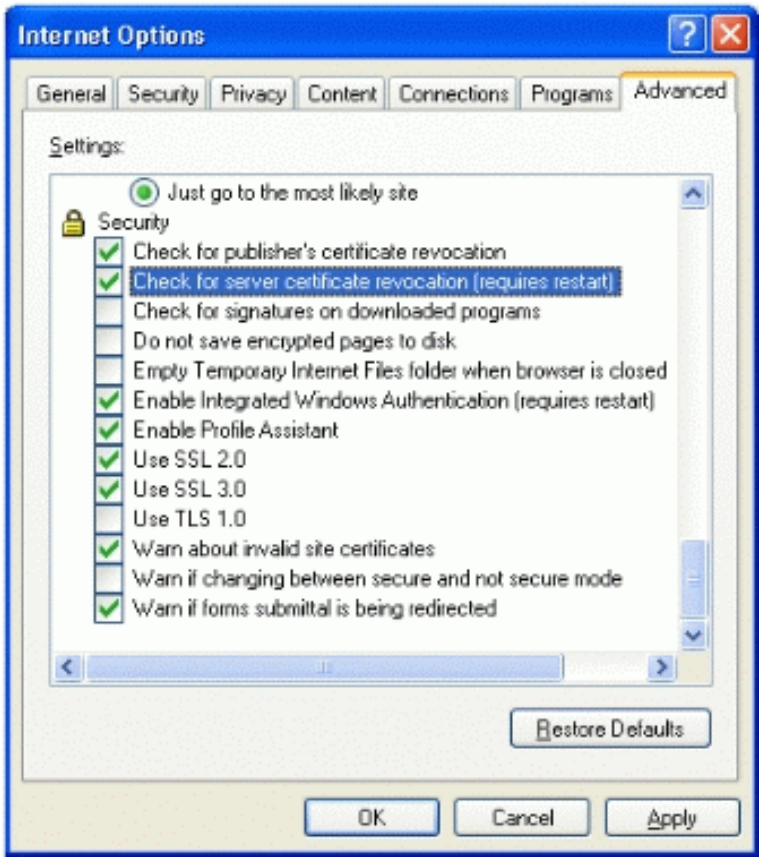

Figure 22: Enabling CRL Checking in Internet Explorer
See full-sized image.

5.Close Internet Explorer and restart for the changes to take effect.

IIS

IIS 5.0 included with Windows 2000 performs revocation checking by default.

Encrypting File System

EFS in Windows 2000 does not perform revocation checking, however in Windows XP, revocation checking is enabled by default when other users added to encrypted files.

Outlook

CRL checking settings vary depending on the version of Outlook that you are using. For example, Outlook 2002 has CRL checking enabled by default, while Outlook 2000 SR1 may or may not have CRL checking enabled. To determine whether CRL checking is enabled in Outlook 2000 SR1, check for the existence of the following registry entry:

```
[HKEY_LOCAL_MACHINE \SOFTWARE \Microsoft \Cryptography{7801ebd0-cf4b-11d0-851f-
0060979387ea}]
"PolicyFlags"=dword:00010000
```

Internet Protocol Security (IPSec)

CRL checking is not enabled by default in Windows 2000. With the release of Windows 2000 SP2, an additional registry key was added that can enable CRL checking for IPSec certificate-based authentication.

The registry key is:

```
HKEY_LOCAL_MACHINE \System \CurrentControlSet \Services \PolicyAgent \Oakley
\StrongCrlCheck
```

This Reg_DWORD data type can be assigned values from 0-2, with the following meanings:

- 0 – Disables CRL checking for certificate-based IPSec authentication
- 1 – Enables CRL checking and fails the validation process only if the CRL explicitly indicates that the certificate is revoked. All other failures, including when the CDP URL is unavailable, will be ignored.
- 2 – Enables CRL checking and fails certificate validation on any CRL check errors.

Note: This registry key does not exist by default. You must manually create the registry key as outlined at http://www.microsoft.com/windows2000/techinfo/planning/security/ipsecsteps.asp

Top of page

Walkthroughs

Using CAPolicy.inf

The CAPolicy.inf file is used to configure an offline root CA's CDP and AIA information for the root CA certificate. The CAPolicy.inf file must be modified before the installation of the offline root CA and saved in the %systemroot% folder.

Note: The CAPolicy.inf file can be used for more purposes that just defining AIA and CDP locations. Please refer to the Building a Windows .NET Certificate Hierarchy white paper for full documentation on the use of CAPolicy.inf

Using Variables in CAPolicy.inf

To ease the configuration of the CAPolicy.inf file, consider using variables to automatically input CAName and configuration information into URL paths for the CRL Distribution Point and Authority Information Access sections of the configuration file.

The following variables are available:

- %1 – SERVERDNSNAME – The CA's DNS Name used in the generation of an HTTP URL.
- %2 – SERVERSHORTNAME – The NetBIOS name of the CA
- %3 – SANITIZEDCANAME – The XXXXXX name of the CA
- %4 – CERTFILENAMESUFFIX – The suffix used by the certificate file. Typically, the .crt extension is used.
- %5 – DOMAINDN – Used to reference the distinguished name of the current domain
- %6 – CONFIGDN – Used to reference the distinguished name of the configuration naming context stored in the forest root domain
- %7 – SANITIZEDCANAMEHASH – The CAName used in the generation of CRL and CRT names.
- %8 - CRLFILENAMESUFFIX – The extension used for the generated CRL file. Typically, this is the .crl extension
- %9 – CRLDELTAFILENAMESUFFIX – The extension used for the generated Delta CRL file. Typically, this is the "+" character.
- %10 – DSCRLATTRIBUTE – Defines that the attribute is related to a CRL.
- %11 – DSCACERTATTRIBUTE – Defines that the attribute is related to a CA certificate
- %12 – DSUSERCERTATTRIBUTE – Defines that the attribute is related to a user certificate.

•%13 – DSKRACERTATTRIBUTE – Defines that the attribute is related to a Key Recovery Agent certificate.
•%14 – DSCROSSCERTPAIRATTRIBUTE – Defines that the attribute is related to a Cross certificate pair.

Important: If the URL path contains spaces, the URL path must be surrounded by the " characters. In addition, if the URL path is quoted, then each variable must have an additional % prefixed.

Modifying the CRL Distribution Point (CDP)

The CRL Distribution point is used by the root CA certificate to find the CDP. The default locations for

```
[CRLDistributionPoint]
D:\WINDOWS\System32\CertSrv\CertEnroll\%3%8%9.crl
ldap:///CN=%7%8,CN=%2,CN=CDP,CN=Public Key Services,CN=Services,%6%10
http://%1/CertEnroll/%3%8%9.crl
file://\\%1\CertEnroll\%3%8%9.crl
Critical=False
```

Note: When defining the CRL distribution point, be sure not to delete the default CDP path pointing to the local file system such as d:\windows. You must keep this path to ensure that the CRL is published directly to the local disk and can be manually copied to external locations.

Modifying the AIA

The AIA is used by the root CA certificate to indicate the locations where the root CA certificate can be retrieved.

```
[AuthorityInformationAccess]
URL = "ldap:///CN=%%7,CN=AIA,CN=Public Key Services,CN=Services,%%6%%10"
URL = "D:\WINNT\System32\CertSrv\CertEnroll\%%1_%%3%%4.crt"
URL = http://%1/Public/%7.crt
URL = ftp://northwindtraders.com/Public/%7.crt
URL = file://%1\Public\%7.crt
CriticAL = False
```

Note: A root CA may not point to an OCSP responder for its own self-signed certificate.

Publishing Root Certificates for an Offline Root CA

Offline Root CA's certificate must be published to Active Directory to ensure that domain members can retrieve the CA's certificate when required.

Using DSSTORE in Windows 2000

The following command can be used in Windows 2000 to publish an offline root CA's certificate to the Active Directory:. Dsstore is found in the Windows 2000 Resource Kit.

```
dsstore DC=MyForestRoot,DC=tld -addroot MyCAsCRTFile.crt MyCAsName
```

Note: The preceding DN is the root domain of the forest, even if the CA is installed on a server in a child domain. In addition, the DN component identifiers (DC=) must be capitalized. Finally, if any of the parameters contains spaces, enclose them in quotes.

Using Certutil in Windows Server 2003

The following command can be used in Windows Server 2003 to publish an offline root CA's certificate to the Active Directory.

```
Certutil.exe –dspublish –f MyCAsCRTFile.crt MyCAsName
Usage:
  CertUtil -dsPublish [Options] CertFile [NTAuthCA | RootCA | SubCA | CrossCA | KRA |
User | Machine
]
  CertUtil -dsPublish [Options] CRLFile [DSCDPContainer [DSCDPCN]]
  Publish Certificate or CRL to DS
    CertFile -- certificate file to publish
    NTAuthCA -- Publish cert to DS Enterprise store
    RootCA -- Publish cert to DS Trusted Root store
    SubCA -- Publish CA cert to DS CA object
    CrossCA -- Publish cross cert to DS CA object
    KRA -- Publish cert to DS Key Recovery Agent object
    User -- Publish cert to User DS object
    Machine -- Publish cert to Machine DS object
    CRLFile -- CRL file to publish
    DSCDPContainer -- DS CDP container CN, usually the CA machine name
    DSCDPCN -- DS CDP object CN, usually based on the sanitized CA short name and key
index
    Use -f to create DS object.
```

Normally, offline root CAs are published to the RootCA location

Publishing CRLs for an Offline Root CA

Offline Root CA's CRL must be published to Active Directory to ensure that domain members can retrieve the CA's CRL when performing revocation checking.

Using DSSTORE in Windows 2000

The following command can be used in Windows 2000 to publish an updated version of an offline CA's CRL into Active Directory.

```
dsstore DC=MyForestRoot,DC=tld -addcrl MyCAsCRLFile.crl MyCAsName CAServerName
```

Using Certutil in Windows Server 2003

The following command can be used in Windows Server 2003 to publish an updated version of an offline CA's CRL into Active Directory.

```
Certutil –dspublish –f MyCAsCRLFile.crl CAServerName
```

Creating Certificate Trust Lists

The following procedures create a signed CTL and save it to a certificate store.

To Programmatically create and sign a CTL

1. Create an array of items to be stored in the CTL. In the case of trusted certificates, this must be the SHA1 or MD5 hashes of the trusted certificates.
2. Initialize a CTL_INFO structure that includes the array of items just created.
3. Initialize a CMSG_SIGNED_ENCODE_INFO structure.
4. Call CryptMsgEncodeAndSignCTL. This function call returns a pointer to a signed, encoded CTL (in PKCS #7 format) that contains the list of items created in step 1.

To Programmatically Add a CTL to a certificate store

1.Get a pointer to a signed and encoded CTL.
2.Open the target certificate store with a call to CertOpenStore.
3.Call CertAddEncodedCTLToStore.

To Manually Create and Sign a CTL

The following procedure can be used to manually create and sign a CTL.

1.From Administrative Tools, open Domain Security Policy.

Note: The CTL can be defined at any OU in a domain, this example assumes that a CTL will be defined for an entire domain, not just for a specific OU structure.
2.In the console tree, expand Default Domain Policy, expand Computer Configuration, expand Windows Settings, expand Security Settings, expand Public Key Policies, and then select Enterprise Trust.

Note: In Windows Server 2003, the container is renamed from Enterprise Trust, to Certificate Trust Lists.
3.From the menu bar, click Action, then point to New, , and then click Certificate Trust List.
4.In the Certificate Trust List Wizard (as shown in Figure 23), click Next.

Figure 23: Starting the Certificate Trust List Wizard
See full-sized image.

5. In the Certificate Trust List Purpose page, enter the identifying prefix, duration, and designated purposes for the CTL as shown in Figure 24, and then click Next.

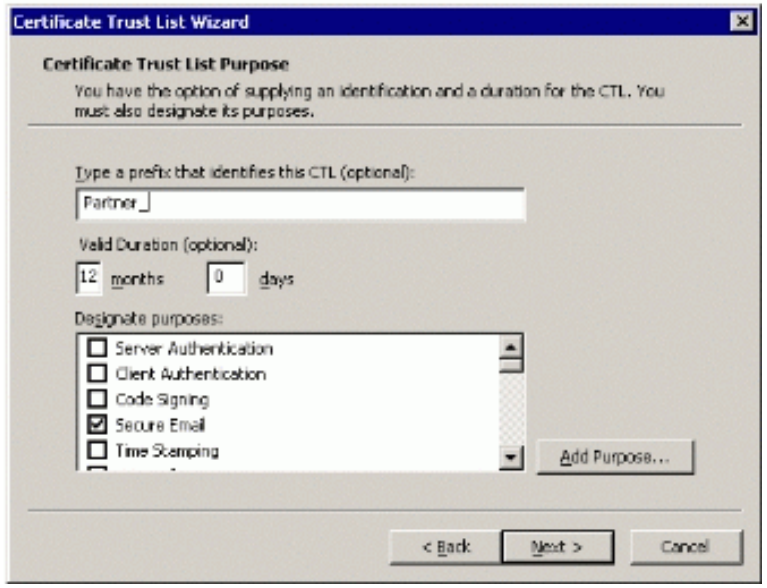

Figure 24: Defining the purposes and duration of a CTL
See full-sized image.

6. In the Certificates in the CTL page (see Figure 25), add certificates by clicking either the Add from Store button, or the Add from File button, and then click Next.

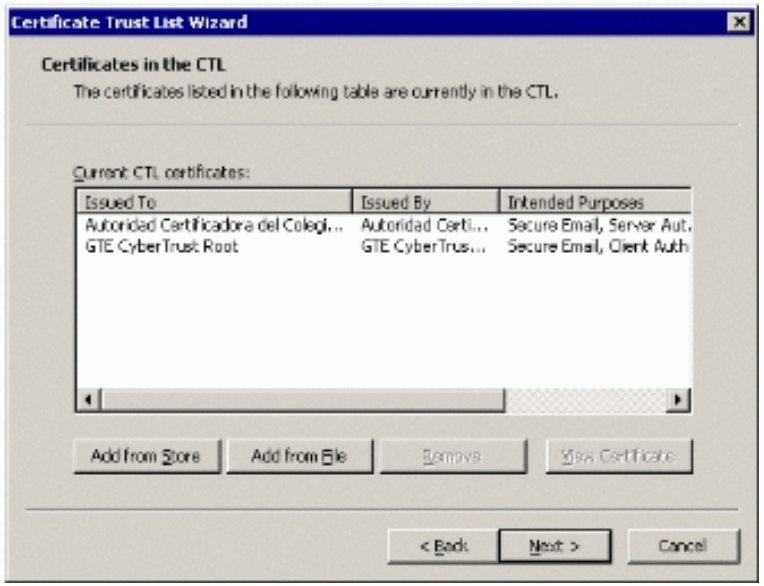

Figure 25: Starting the Certificate Trust List Wizard
See full-sized image.

7. In the Signing Certificate page, you can select a digital signing certificate from your user store to sign the certificate trust list to ensure that the CTL is not modified after creation. A certificate that has the Trust List Signing purpose and its associated private key must exist in your personal store to perform the digital signing. The Administrator certificate template includes this extended key usage setting. The selected certificate's properties will appear in the Signature Certificate window as shown in Figure 26. Verify the certificate properties and then click Next.

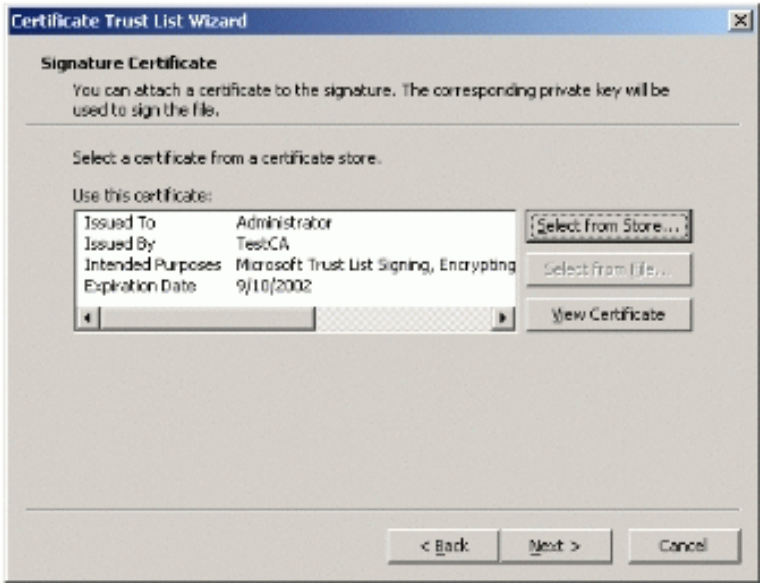

Figure 26: Select the certificate to use to sign the Certificate Trust List
See full-sized image.

8. In the Timestamping window (see Figure 27), you can optionally choose to add a timestamp to the data using a timestamp service to guarantee when the CTL was defined. You must provide an accessible timestamp service URL to read the current time. Once the timestamp server has been (optionally) defined, click **Next** to continue.

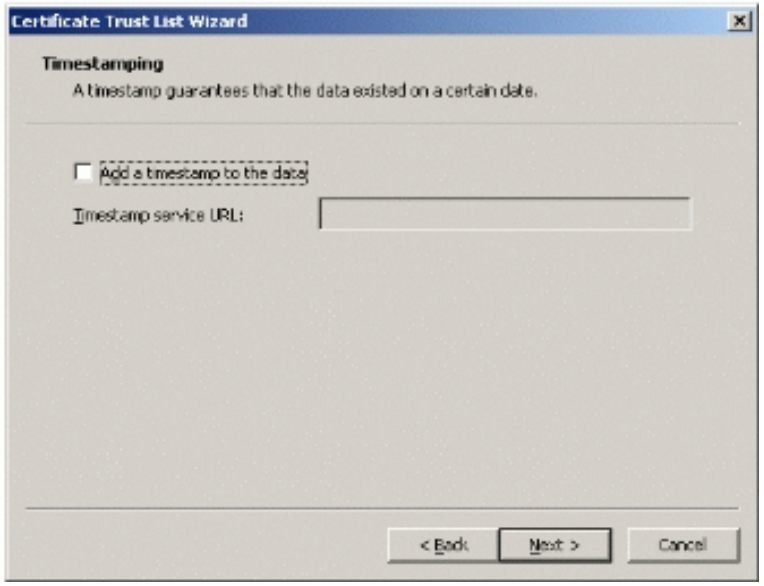

Figure 27: Applying a time stamp to the Certificate Trust List
See full-sized image.

9. In the Name and Description window (see Figure 28), enter a friendly name for the CTL and optionally, provide a description for the CTL, and then click Next.

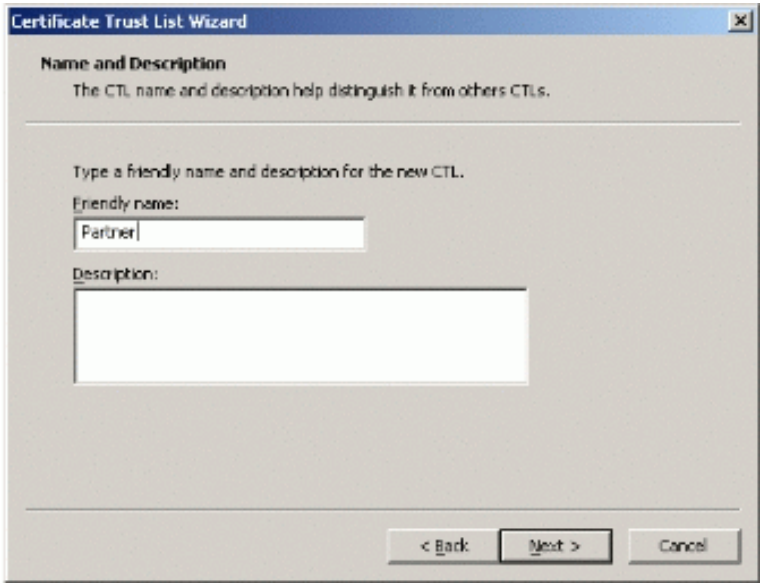

Figure 28: Setting a friendly name for the Certificate Trust List
See full-sized image.

10. Click Finish to complete the Certificate Trust List Wizard.
11. In the Certificate Trust List Wizard dialog box, click OK.

The newly created Certificate Trust List will appear in the details pane, indicating that the Certificate Trust List was successfully created.

Top of page

Troubleshooting

Verifying the Validity of a certificate

The validity of a certificate can be validated by using Certutil.exe with the following options:

```
Certutil.exe –isvalid Serialnumber
```

Where *SerialNumber* is the serial number of the certificate that you wish to verify in hexadecimal format.

Note: The certutil.exe utility is only available on Windows 2000 certificate authorities and in Windows .NET through the Administration Tools Pack.

Verifying that a Certificate was issued by a Specific CA

You can determine whether a certificate was issued by a specific CA by using Certutil.exe. To verify the certificate, you will require the certificate you wish to verify and the CA certificate you wish to verify against as parameters. The syntax of the command is:

```
Certutil.exe –verify CertFile CaCertFile
```

This command requires that both the CA certificate and the issued certificate be a PKCS#10 export file, not a PKCS#7 certificate chain. When the command is run, it also verifies the revocation status of the end certificate. An error will occur if the certfile does not contain CDP information, or if the URLs indicated in the CDP extension are unavailable.

Note: If you do not include the *CaCertFile* as a parameter, certutil will construct a certificate chain using all available certificates installed on the computer.

Validating AIA and CDP extensions in a Certificate

You can manually validate the AIA and CDP extensions for a specific certificate by using the following Certutil syntax:

```
Certutil.exe –url CertFile.crt
```

To run this command, you must have an exported version of the certificate in a DER encoded format. Certutil will only verify the basic certificate location pointer and the CRL(s) for the AIA and CDP locations respectively.

Retrieving Certificate Chaining

To retrieve the CA signing certificate and chain and save it into a PKCS#7 file, use certutil with the following syntax at the CA

whose CA chain you wish to export:

```
Certutil.exe –ca.chain CaCertChainOutputFile
```

Where *CaCertChainOutputFile* is the name of the output file you wish.

Top of page

For More Information

For the latest information and additional white papers on PKI technologies, check out our Web site at http://www.microsoft.com/technet/security/topics/crypto/default.mspx

Top of page

Appendix A – Certificate and Certificate Chain Status Codes

In addition to which certificate stores are searched during a chain validation call , the certificate chain engine can also be configured with the following parameters programmatically:

•Maximum number of cached certificates
•Which store to use. The certificate chain can be built using either the current user or local machine stores.
•URL retrieval settings. The certificate chain engine can be set to bypass Uniform Resource Location (URL) retrieval and use only cached certificates.
•End certificate caching. Allows the end certificate in the certificate chain to be cached.
•Cache synchronization settings. Allows the cache to be resynchronized when objects are added to the engine stores.
•Cyclic chain settings. At what number of certificates should a chain be checked for being a cyclic chain.
•Timeout settings. What timeout period should be used when accessing a CRL URL.
•Thread settings. Configures whether a separate thread of execution is used for store synchronization.

These parameters are set by the calling application using various application programming interfaces (APIs), including CertGetCertificateChain and CertVerifyRevocation.

Note: You can find detailed documentation about these APIs at http://msdn.microsoft.com/library and searching for CertGetCertificateChain and CertVerifyRevocation

The following information status codes can be assigned for each individual certificate in a certificate chain:

- **CERT_TRUST_HAS_EXACT_MATCH_ISSUER**. An exact match issuer certificate has been found for this certificate.
- **CERT_TRUST_HAS_KEY_MATCH_ISSUER**. A key match issuer certificate has been found for this certificate.
- **CERT_TRUST_HAS_NAME_MATCH_ISSUER**. A name match issuer certificate has been found for this certificate.
- **CERT_TRUST_IS_SELF_SIGNED**. This certificate is self-signed.

Certificate chains can be assigned the following information status codes:

- **CERT_TRUST_IS_COMPLEX_CHAIN**. The certificate chain created is a complex chain.
- **CERT_TRUST_HAS_PREFERRED_ISSUER**. The certificate chain is issued by a preferred issuer
- **CERT_TRUST_HAS_ISSUANCE_CHAIN_POLICY**. Issuance policy is present in the certificate chain
- **CERT_TRUST_HAS_VALID_NAME_CONSTRAINTS**. .The certificate chain is valid for name constraints.

In addition to information status codes, CERT_TRUST_STATUS will also return error codes. The following error status codes are defined for certificates and chains.

- **CERT_TRUST_NO_ERROR**. No error found for this certificate or chain.
- **CERT_TRUST_IS_NOT_TIME_VALID**. This certificate or one of the certificates in the certificate chain is not time valid.
- **CERT_TRUST_IS_NOT_TIME_NESTED**. Certificates in the chain are not properly time nested.
- **CERT_TRUST_IS_REVOKED**. Trust for this certificate or one of the certificates in the certificate chain has been revoked.
- **CERT_TRUST_IS_NOT_SIGNATURE_VALID**. The certificate or one of the certificates in the certificate chain does not have a valid signature.
- **CERT_TRUST_IS_NOT_VALID_FOR_USAGE**. The certificate or certificate chain is not valid for its proposed usage.
- **CERT_TRUST_IS_UNTRUSTED_ROOT**. The certificate or certificate chain is based on an un-trusted root.
- **CERT_TRUST_REVOCATION_STATUS_UNKNOWN**. The revocation status of the certificate or one of the certificates in the certificate chain is unknown.
- **CERT_TRUST_IS_CYCLIC**. One of the certificates in the chain was issued by a certification authority that the original certificate had certified.
- **CERT_TRUST_INVALID_EXTENSION**. One of the certificates has an invalid extension.
- **CERT_TRUST_INVALID_POLICY_CONSTRAINTS**. The certificate or one of the certificates in the certificate chain has a policy constraints extension, and one of the issued certificates has a disallowed policy mapping extension or does not have a required issuance policies extension.
- **CERT_TRUST_INVALID_BASIC_CONSTRAINTS**. The certificate or one of the certificates in the certificate chain has a basic constraints extension and either the certificate cannot be used to issue other certificates or the chain path length has been exceeded.
- **CERT_TRUST_INVALID_NAME_CONSTRAINTS**. The certificate or one of the certificates in the certificate chain has an invalid name constraints extension.

- **CERT_TRUST_HAS_NOT_SUPPORTED_NAME_CONSTRAINT**. The certificate or one of the certificates in the certificate chain has a name constraints extension containing unsupported fields. The minimum and maximum fields are not supported. Thus minimum must always be zero and maximum must always be absent. Only UPN is supported for an Other Name. The following alternative name choices are not supported:

  X400 Address
  EDI Party Name
  Registered Id.
- **CERT_TRUST_HAS_NOT_DEFINED_NAME_CONSTRAINT**. The certificate or one of the certificates in the certificate chain has a name constraints extension and a name constraint is missing for one of the name choices in the end certificate.
- **CERT_TRUST_HAS_NOT_PERMITTED_NAME_CONSTRAINT**. The certificate or one of the certificates in the certificate chain has a name constraints extension and there is not a permitted name constraint for one of the name choices in the end certificate.
- **CERT_TRUST_HAS_EXCLUDED_NAME_CONSTRAINT**. The certificate or one of the certificates in the certificate chain has a name constraints extension and one of the name choices in the end certificate is explicitly excluded.
- **CERT_TRUST_IS_OFFLINE_REVOCATION** . The revocation status of the certificate or one of the certificates in the certificate chain is either off-line or stale.
- **CERT_TRUST_NO_ISSUANCE_CHAIN_POLICY**. The end certificate doesn't have any resultant issuance policies, and one of the issuing CA certificates has a policy constraints extension requiring it.

In addition to the error status codes defined for certificates and chains, the following error status codes are defined for chains only:

- **CERT_TRUST_IS_PARTIAL_CHAIN**. The certificate chain is not complete.
- **CERT_TRUST_CTL_IS_NOT_TIME_VALID**. A CTL used to create this chain was not time valid.
- **CERT_TRUST_CTL_IS_NOT_SIGNATURE_VALID**. A CTL used to create this chain did not have a valid signature.
- **CERT_TRUST_CTL_IS_NOT_VALID_FOR_USAGE**. A CTL used to create this chain is not valid for this usage.

Information contained in this document represents the current view of Microsoft Corporation on the issues discussed as of the date of publication. Because Microsoft must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information presented after the date of publication.

This white paper is for informational purposes only. MICROSOFT MAKES NO WARRANTIES, EXPRESS OR IMPLIED, IN THIS DOCUMENT.

Top of page