Certificate Path Processing Implementation Guideline

Version 1.0

JKST IWG 2002

Table of Contents

1 Objective & Scope

The objective of this *Path Processing Implementation Guideline*(for short PPIG) is to help the PKI developers to implement the complicated Certificate Path Processing algorithms well, which are essential to make the multi PKI domains interoperable each other.
Because the certificate path validation algorithms in RFC 3280 are so complex and difficult that it is possible for PKI developers to make mistakes in implementing its algorithms.
In IWG member's hope that all PKI S/Ws output one same result on the certain certificate paths in interoperable PKI domains, PPIG is developed to minimize any possible mistakes and errors by the PKI developers at implementation levels.

The procedures of validating certificate paths, so called *"Certificate Path Processing"*, consists of two main parts. One is certificate path validation and the other is certificate path construction. In this guideline, we will describe those two main parts of certificate path processing and some additional considerations.
For the first part, the certificate path validation algorithm based on the algorithm described in RFC3280[1] will be covered. According to the algorithms, the conformant implementation must output the same result for the same inputs. To help such a conformant implementation, this guideline will give detailed explanations about the algorithm. And this guideline will derive some interoperability requirements between different PKI domains and will describe some considerations to support the IWG recommended profiles[2][3].

For the second part, this guideline will include the restricted certificate path construction algorithm with some environmental assumptions. The certificate path construction procedure depends on many environmental aspects, such as interoperability model, repository and CRL distribution model. For those reasons above, it is very difficult to derive a general algorithm for certificate path construction. So, we have made some assumptions on which the restricted certificate path construction algorithm have been created.

- Certificate Path Validation Algorithm
  In this part, we will describe the algorithm that gives the validity of given certificate path. This algorithm is fully based on RFC3280. To assist understanding of the algorithm, sample

diagrams and corresponding explanations are added. This guideline will derive some requirements for the interoperability between different PKI domains and will include IWG profile considerations based on "*Recommendations on Technical Certificate Profile*".

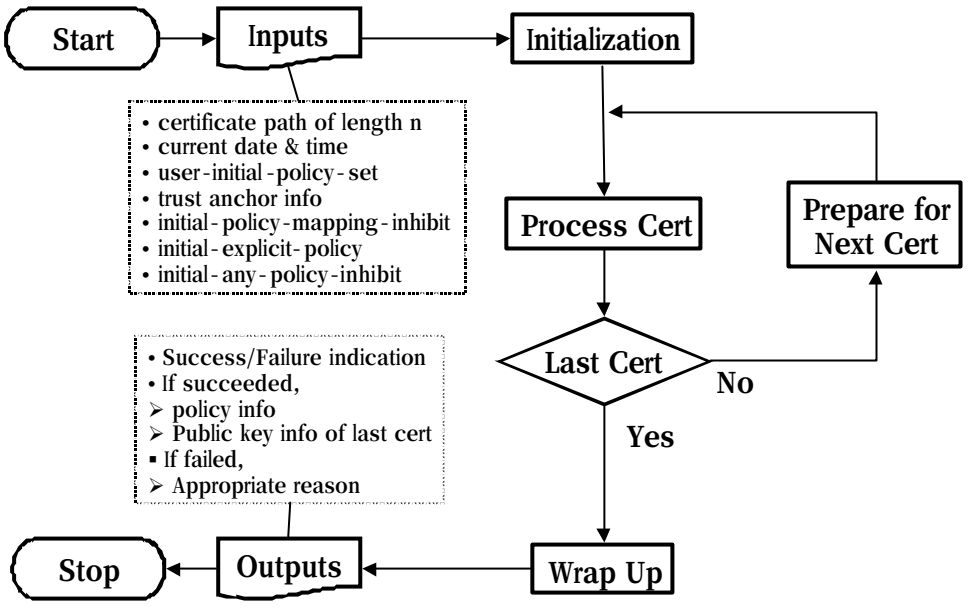
- Certificate Path Construction Algorithm
  In this part, we will describe the algorithm of constructing certificate path for a given certificate. Constructed certificate path will be the input of above certificate path validation algorithm It is difficult to design a general path construction algorithm because path construction depends on the PKI environment such as interoperability model, repository and CRL distribution method. The restricted Path Construction algorithm is described with some assumptions of those factors that could affect the path constructions.

At the end of this guideline, Some considerations for the certificate validation software will be covered. In this version of the guideline, the considerations for OCSP services will be made.

2 Contents

2.1 Certificate Path Validation Algorithm

The primary goal of path validation is to verify the binding between a subject distinguished name or a subject alternative name and subject public key, as represented in the end entity certificate, based on the public key of trust anchor information. The certificate path validation algorithm is a procedure to achieve above goal for a given certificate path. The following diagram shows the overall procedure of the path validation algorithm.



[Figure 1] Certificate Path Validation Flowchart

The diagram shows that the path validation algorithm is composed of four basic steps. At first, "Initialization" step is performed using "Inputs". "Basic Certificate Processing" step will be done for each certificate in the path. For each intermediate certificate "Preparation for Next Certificate" step is performed and for the final certificate "Wrap-up" step is performed after the "Basic Certificate Processing". As a result of four steps, "Outputs" will be given to the relying party (or the application using the certificate).

To make this guideline more organized, concise and meaningful to implementers, each main phase of RFC3280 is described briefly and then three types of explanations are attached. One is for explanation of concerned rfc3280 phase and another is for the interoperability requirements. And the last one is for description from the point of IWG Recommended Profile.

2.1.1 Inputs

The algorithm uses seven input values of which they are the certificate path of length n, current date/time, user-initial-policy-set, trust anchor information and policy related values. Three police related input values, initial-policy-mapping-inhibit, initial-explicit-policy, initial-any-policy-inhibit, are used to constrain policy processing.

(a) Additional explanations

(1) The general implementation SHOULD give the user interface with which the relying party

can set the input values. However, the input values can be set in advance according to the local policy.

(b) Interoperability requirements

(1) For the cross recognition models, the relying parties can have multiple trust anchor information. And the trust anchor information is delivered as a self-signed certificate. So, the implementation SHOULD support the methods to import and manage multiple trust anchor information in a self-signed certificate format.

[IWG Profile Considerations]

- At this time, the special OID anyPolicy and the inhibitAnyPolicy extension is not used in IWG certificate profile. So, it's not necessary to support initial-any-policy-inhibit input value in the implementations. But, considering any enhancement of IWG Profiles to RFC3280, anyPolicy can be used in the near future so that we recommend the implementation to have the ability to process anyPolicy related items.

2.1.2 Initialization

This step establishes eleven state variables based upon the seven input values. valid_policy_tree is the variable that represents the policy information constrained by CAs and the policy mapping information. Two name-related variables and three policy-related variables are used to constrain name scope and policy processing respectively. max_path_length variable is used to check the path length constraint and four more variables are used to verify the chaining of certificate path.

(a) Additional explanations

(1) valid_policy_tree is the variable which represents the policy information constrained by CAs. So, it indicates what certificate policies can be used by a subordinate CA, what certificate policies can be considered equivalent and what qualifiers are given to a certificate policy.

(2) Two name-related variables, permitted_subtrees and excluded_subtrees, are used to constrain what names can be used by subordinate CAs. This is important to make the subject name unique to the subscriber and the naming rules organized.

(b) Interoperability requirements

(1) Not assigned yet

[IWG Profile Considerations]
Not assigned yet

2.1.3 Basic Certificate Processing

This step is performed for each certificate in the path and it includes chain verifications, name constraint checks and policy processing.

(a) Additional explanations

(1) Name constraint checks for self-issued certificate: A certificate is called the self-issued if the DNs that appear in the subject and issuer fields are identical and are not empty. Three types of self-issued certificates can be used for different purposes.

- Distribution of the trust anchor information: Root CA can use self-issued certificates to distribute the trust anchor information. This is the special case of self-issued certificate where the private key used to sign the certificate corresponds to the public key, which is certified within that certificate.
- For the certification of another usage(ex. Timestamp): CA can issue a certificate to itself for other usage certification. For example, CA can issue a certificate for timestamp to itself. This certificate can appear as the final certificate of the path and it is handled as an end entity certificate.
- Key Rollover: Root CA can issue a certificate to itself for key rollover operations. This certificate can appear as the first (non-final) certificate in the path and it is temporarily used for Root CA key change. So, it is not a target certificate to validate.

If a Non-final self-issued certificate is the third type of self-issued certificates that are used for key rollover operations, it is not necessary to check the name scopes for it. But, if a final self-issued certificate is the second type of self-issued certificates that is used for other usage, it is necessary to check the name scope for it.

A self-issued certificate for key rollover should not affect the path validation process. And the certificate policy extensions should have valid policies (anyPolicy or all policies used by the CA) to make valid_policy_tree not NULL.

(b) Interoperability requirements

(1) GeneralName type used to constrain name scopes SHOULD not use otherName type for the interoperability. And it is necessary to make a full consideration, when to use

x400Address, ediPartyName and registeredID.

[IWG Profile Considerations]
Inconsistent application of name comparison rules may result in acceptance of invalid targeted certificates, or rejection of valid ones. The X.500 series of specifications defines rules for comparing distinguished names. These rules require comparison of strings without regard to case, character set, multi-character white space substrings, or leading and trailing white space. The DN String matching rules should follow those matching rules of X.520 and the RDN String Representation rules should follow those representation rules of rfc2253

2.1.4 Preparation for Next Certificate

For all the intermediate certificates, this step will be performed after "Basic Certificate Processing" steps. This step includes the procedures such as policy mapping, updating state variables, key usage checks and path length constraint checks. And any other critical extensions and recognized non-critical extensions SHOULD be processed.

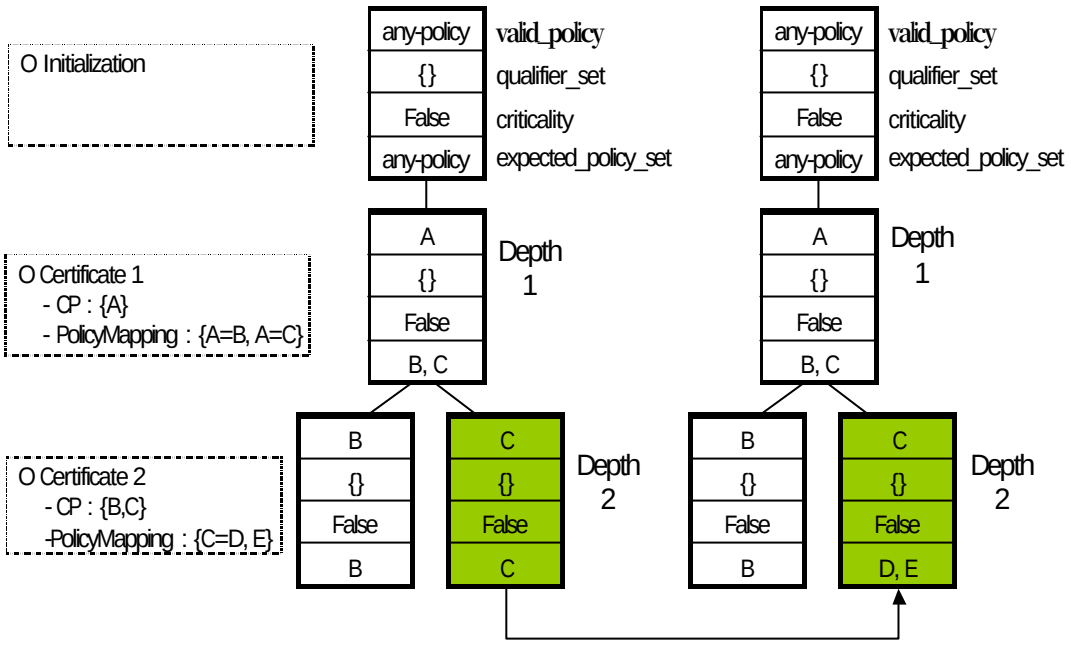
(a) Additional explanations

(1) Processing policyMapping extensions: If the policy mapping extensions are present in the certificate i, policy mapping processing is performed according to the types described in [Table 1]. We will give the procedures for processing policyMapping extensions for each type using sample certificate paths.

Type	**Policy_mapping state variable**	**valid_policy of each node of depth i**
I	greater than 0	matches a CP of issuerDomainPoicy of i certificate
II	greater than 0	equals anyPolicy
III	0	-

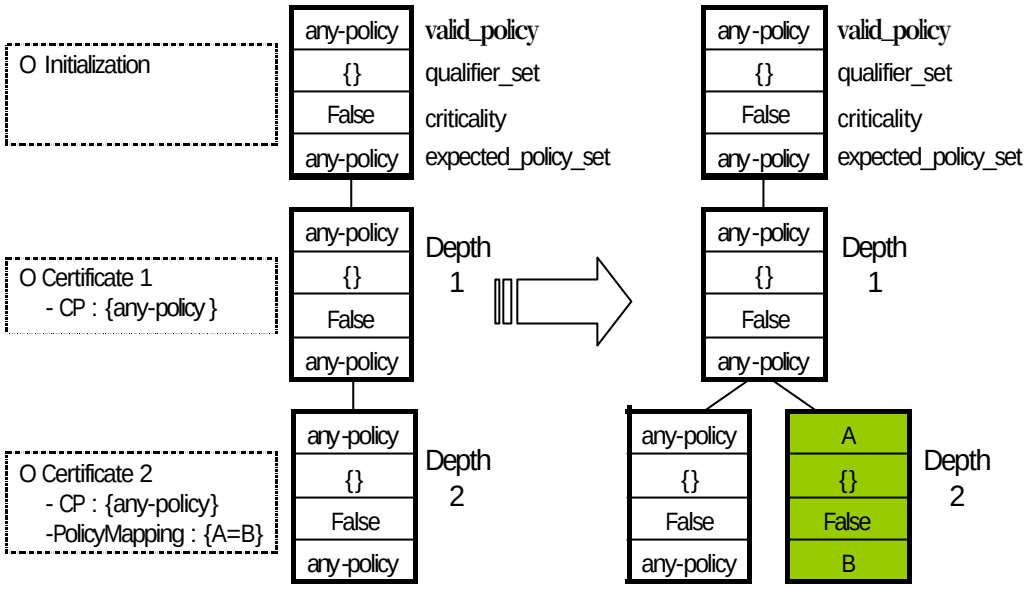
[Table 1] Types of processing policyMapping extension

- The following is the sample of processing policy mappings when there is a node of Type I. In processing the sample, the node having valid_policy of *C* is modified to have expected_policy_set as *D* and *E* according to the policyMapping extension of the certificate 2.



[Figure 2] Processing policy mapping extension (Type I)

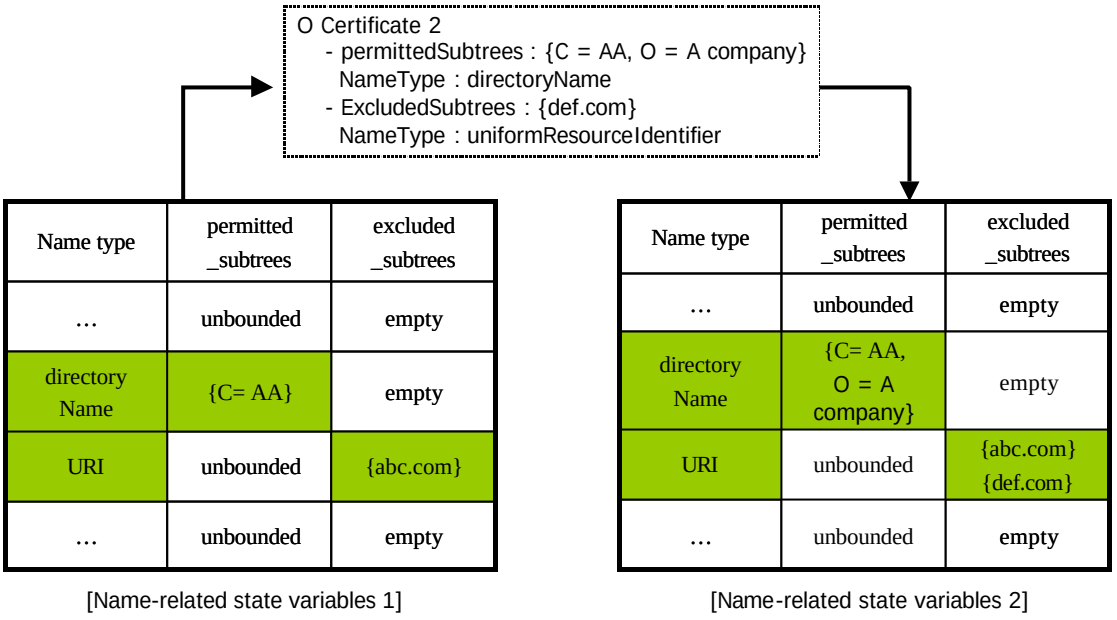
- The following is the sample of processing policy mapping when there is a node of Type II. In processing the sample, because no node of depth 2 has a valid_policy of *A* and there is a node of depth 2 with a valid_policy of *anyPolicy*, new node with the valid_policy of *A* will be generated with expected_policy_set as *B* referencing to policyMapping extension.



[Figure 3] Processing policy mapping extension (Type II)

- For Type III, delete each node of depth i that has valid_policy appearing in issuerDomainPoicy of policyMapping extension. After deleting such nodes, repeat deleting nodes of depth i-1 or less without child node.

(2) The samples of updating name-related state variables: The following is the sample of updating two state variables, the permitted_subtrees and excluded_subtrees. The scope of permitted_subtrees variable will be reduced after the intersections with permitted_subtrees appearing in nameConstraints extension. On the other hand, the scope of excluded_subtrees variable will be extended after union with excluded_subtrees appearing in nameConstraints extension. In the sample, the [Name-related state variables 1] is changed to the [Name-related state variables 2] after processing of nameConstraints extension of the certificate 2



[Figure 4] Updating name-related state variables

(b) Interoperability requirements

(1) The implementation SHOULD support the policy mapping procedures so that policy processing of certificate path including cross certificate could be processed properly.

(2) If it is necessary to use any private extensions in certificate, then such extensions SHOULD be marked as non-critical for the interoperability between different PKI domains.

(3) When issuing cross certificate, pathLengthConstraint SHOULD be set carefully to avoid the establishment of unexpected trust relationship and to allow the establishment of unexpected trust relationship.

[IWG Profile Considerations]

- In the experiment, IWG did not impose the pathLengthConstraint. But when applying cross certification in real business, we need to pay enough attention to avoid unexpected extension of trust relationship.

2.1.5 Wrap-up

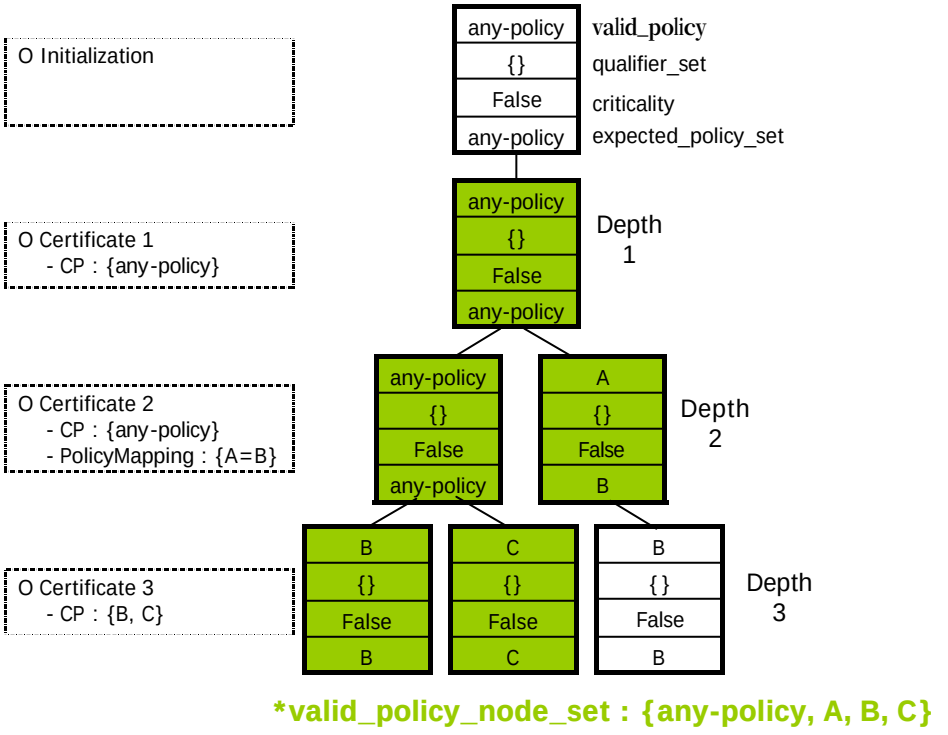
To complete the certificate path validation, this step is performed for the last certificate after "Basic Certificate Processing" step. This step includes the procedures such as updating state variables, calculating the intersection of valid_policy_tree and user-initial-policy-set and all critical extensions and recognized non-critical extensions will be processed. If explicit_policy

variable is greater than 0 or the intersection of valid_policy_tree and user-initial-policy-set is not NULL, the path validation process succeeds.

(a) Additional explanations

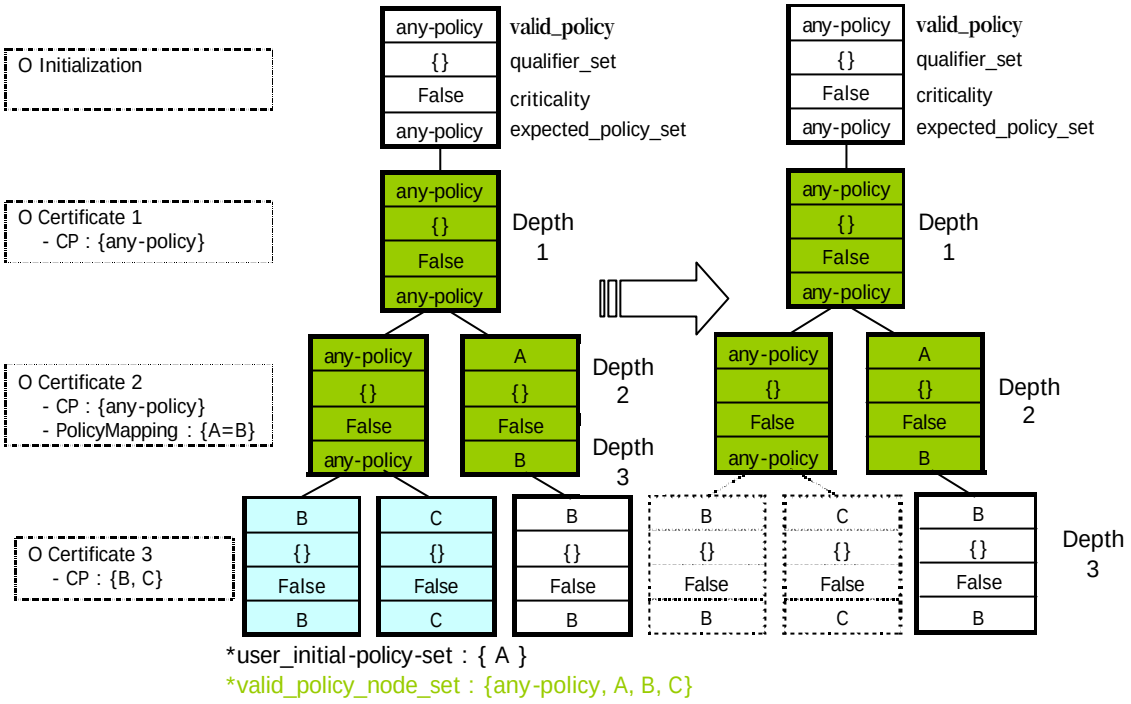
(1) The samples of calculating the intersections of valid_policy_tree and user-nitial-policy-set : When valid_policy_tree is not NULL and user-initial-policy-set is not anyPolicy, the procedure of calculating the intersection is as follows.

- The valid_policy_node_set are the set of policy nodes whose parent nodes have a valid_policy of anyPolicy. In the following sample, the valid_policy_node_set will be the set of *anyPolicy*, *A*, *B* and *C*.



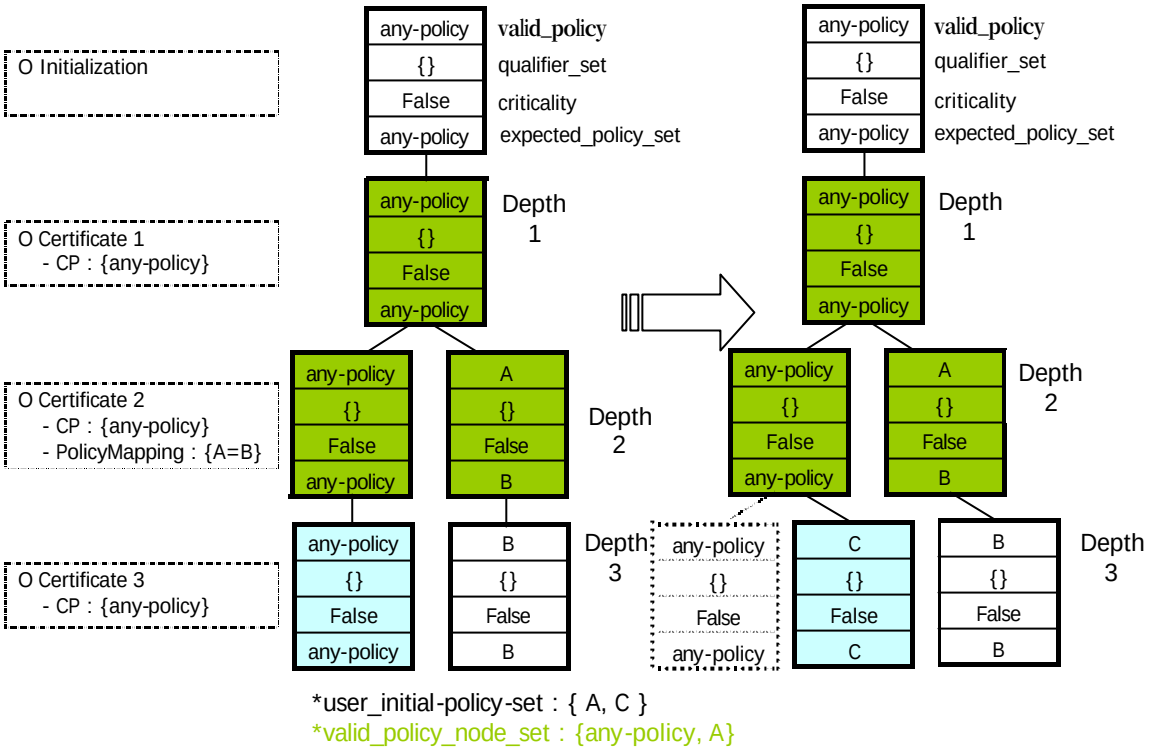
[Figure 5] Calculating the intersection (1)

- If the valid_policy of any node in the valid_policy_node_set is not among the user-initial-policy-set and is not anyPolicy, delete this node and all its children as seen in the following figure.



[Figure 6] Calculating the intersection (2)

- If the valid_policy_tree includes a node of depth n that has the valid_policy of *anyPolicy* and the user-initial-policy-set does not contain *anyPolicy*, generate new nodes with valid_policy of user-initial-policy-set among which they are not in valid_policy_node_set and delete the node with valid_policy of *anyPolicy*. In the following sample, because there is a node with valid_policy of *anyPolicy* of the depth 3, new node is generated with valid_policy of *C* and the node of *anyPolicy* is deleted.



[Figure 7] Calculating the intersection (3)

- If there is a node in the valid_policy_tree of depth n-1 or less without any child nodes, delete that node. Repeat this step until there are no nodes of depth n-1 or less without children.

(a) Interoperability requirements

(1) Not assigned yet

[IWG Profile Considerations] Not assigned yet

2.1.6 Output

If the path validation processes were successful, the procedures terminate returning a success indication together with the final value of the valid_policy_tree, the working_public_key, the working_public_key_algorithm and the working_public_key_parameters.

(a) Additional explanations

(1) It is not necessary to show the outputs to the user. The outputs will be provided to the application for the validations of certificates in use.

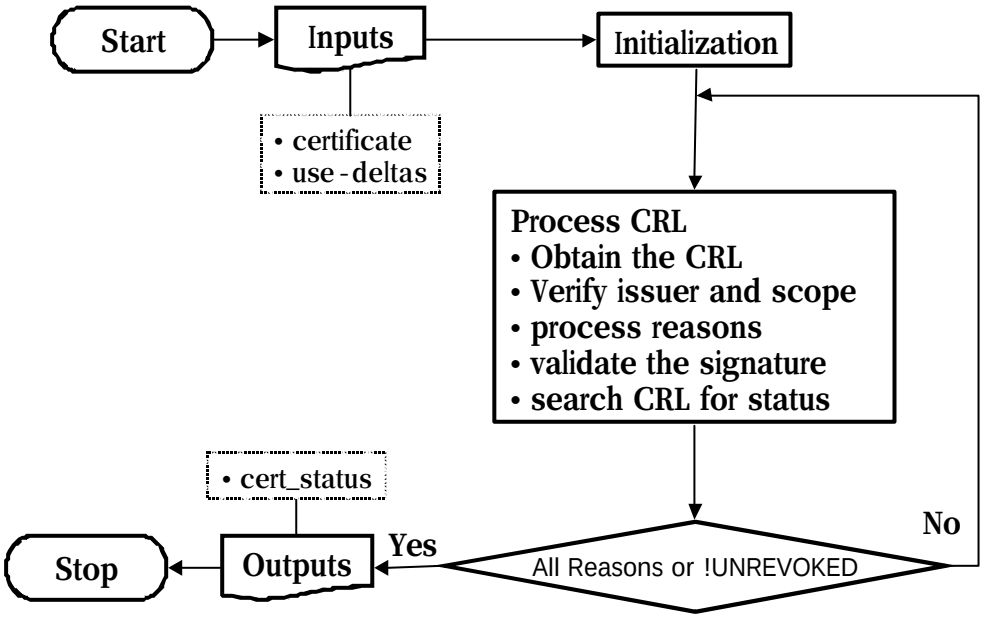
(b) Interoperability requirements

(1) Not assigned yet

[IWG Profile Considerations]
Not assigned yet

2.2 CRL Validation Algorithm

The CRL validation algorithms are the procedures to check whether the target certificates are revoked or on hold status when CRL is the revocation mechanism used by the certificate issuer. The following diagram shows the overall procedure of the CRL validation algorithm.



[Figure 8] CRL Validation Flowchart

The diagram shows that the CRL validation algorithm is composed of two basic steps. “Initialization of Revocation State Variables” step makes use of the variables of “Inputs”. And the revocation status of target certificates are checked in “CRL Processing” steps. The algorithm output is the revocation status of

the target certificates.

2.2.1 Revocation Inputs

The CRL Validation algorithms take two input values, one is target certificate and the other is use-deltas. use-deltas indicates whether delta CRLs are applied to CRLs.

(a) Additional explanations

Not assigned yet

(b) Interoperability requirements

(1) Even though delta CRL scheme is used, a full CRL must be generated and published for the interoperability.

[IWG Profile Considerations]
● At this time, IWG does not use delta CRLs. It is not necessary to support delta CRLs. And the CAs conformant to IWG profile SHOULD provide full CRLs regardless of using delta CRLs.

2.2.2 Initialization and Revocation State Variables

This step establishes three state variables. Two reason-related variables contain the set of revocation reasons supported by the CRLs and delta-CRLs processed so far or currently. And cert_status variable contains the status of the certificate.

(a) Additional explanations

(1) Not assigned yet

(b) Interoperability requirements

(1) To agree on reason codes to support : It is necessary to agree on reason codes to support between PKI Domains in the interoperability environments because it's practical not to support all reason codes in the rfc documents.

Need to specify which reason codes to support in the interoperability models

[IWG Profile Considerations]

Not assigned yet

2.2.3 CRL Processing

For the purposes to check the status of the target certificates, this phase includes the procedures such as updating CRLs and delta-CRLs, verifying the issuer and scope of CRLs and dalta-CRLs, comp uting the set of revocation reasons and determining the status of target certificate.

(a) Additional explanations

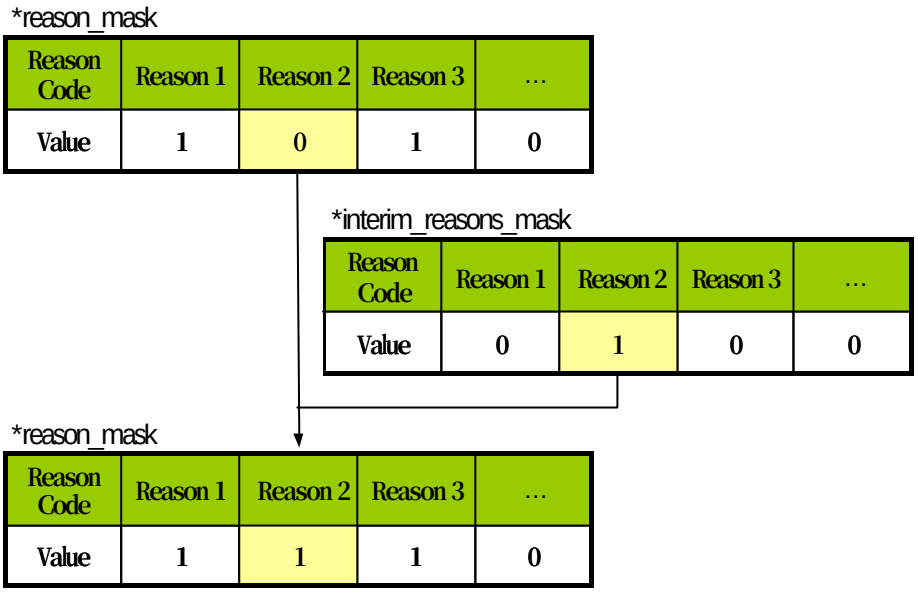
(1) Processing reasons: the variable reasons_mask contains the set of revocation reasons supported by the CRLs and delta-CRLs processed so far. This variable is updated using interim_reasons_mask that includes the intersections of the reasons in the DP and onlySomeReasons in IDP CRL extension.

- To compute the interim_reasons_mask.[4]

[Table 2] Compute the interim_reasons_mask

CRL DP-> distributionPoint->reasons	**IDP->distributionPoint ->onlySomereasons**	**interim-reasons-mask**
Present	Present	Intersection
Present	Omitted	CRL DP-> distributionPoint->reasons
Omitted	Present	IDP->distributionPoint ->onlySomereasons
Omitted	Omitted	All reasons

- The following sample shows that the reasons_mask is updated with the interim_reasons_mask : If the interim_reasons_mask includes one or more reasons that is not included in the reasons_mask, these reasons are included in the reasons_mask.



[Figure 9] Compute the reasons_mask

(2) To verify the scopes of the CRL :

- If CA issues a full CRL and ARL together, the application must be able to distinguish them. If a full CRL and a full ARL are substituted, perhaps applications will regard revoked certificate as a valid certificate. So, the applications must check the IDP->onlyContainsUserCerts or onlyConstainsCACerts against CRL/ARL substitution-attack. It is verified by using the basic constraints extension with cA boolen of the target certificate.

- Also, if CA issues CRLs or ARLs which are partitioned with various separation mechanisms(reason codes and serial number range of the certificate, etc.), the relying party S/Ws must confirm that one of the names in the IDP matches one of the names in the DP to prevent CRL or ARL substitution.

(3) To determine the status of target certificate : According to the reasons_mask and cert_status state variables, the status of target certificate is determined.

[Table 3] Determine the status of target certificate

Reasons-mask	Cert-status	State of certificate
All-reasons	UNREVOKED	Valid
All-reasons	Not-UNREVOKED	Revoked
Not-All-reason	UNREVOKED	The application must retrieve other ARL/CRL because state of target certificate is not determined
Not-All-reason	Not-UNREVOKED	Revoked

(b) Interoperability requirements

(1) Not assigned yet

[IWG Profile Considerations]

- According to the Asia Recommendation Profile, indirect-CRL and delta-CRL is not used. So, the implementation to support IWG profiles doesn't have to implement procedures related to indirect-CRL and delta-CRL
- The applications don't have to implement procedure related to the reasons_mask and interim_reasons_mask because CRL or ARL is not separated by the revocation reason according to the Asia Recommendation Profile.

2.3 Restricted Certificate Path Construction Algorithm

It is difficult to specify a general Certificate Path Construction Algorithm for the various models of interoperability and Repository. So, this guideline will make assumptions about PKI environment and describe the Restricted Certificate Path Construction Algorithms based on it.

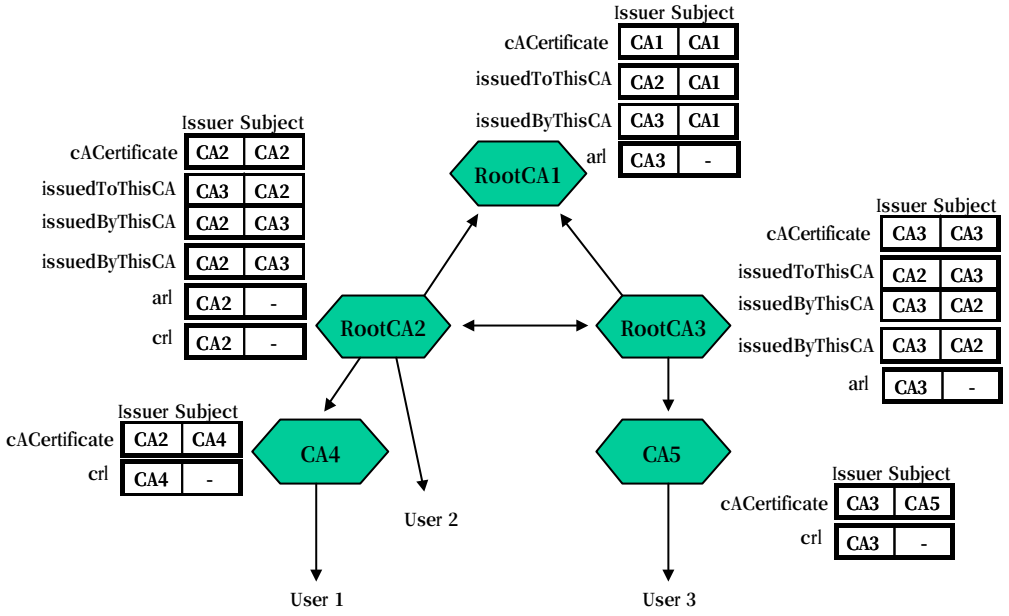
2.3.1Assumptions

2.3.1.1 Interoperability Model

(a) Each domain has strict hierarchy

(b) Cross Certification between top level CA's

2.3.1.2 Repository



[Figure 10] Structure of Directory

For the distribution of certificates and ARL/CRLs, the CA can use any repository mechanisms among which are Directory, HTTP, Mail and FTP, etc. But, this guideline made the assumptions on using the LDAPv3 Directory.

(a) Schema: This guideline assumes that the following attributes are used to convey certificates and ARL/CRL and the distinguished name of an entry equals to the subject of a certificate stored in that entry.

(1) Root CA entry

- cACertificate : self-sign certificate of this Root CA
- crossCertificatePair : If there are cross certificates related to this Root CA, this attribute MUST appear and contain those cross certificates.
- authorityRevocationList : If this Root CA issues CA certificates, this attribute MUST appear and contain an ARL.
- certificateRevocationList : If this Root CA issues user certificates, this attribute MUST appear and contain a CRL.

(2) Subordinate CA entry

- cACertificate : a certificate of this CA
- authorityRevocationList : If this Root CA issues CA certificates, this attribute MUST appear and contain an ARL.
- certificateRevocationList : If this Root CA issues user certificates, this attribute MUST appear and contain a CRL.

2.3.1.3 Certificate/CRL Profiles

(a) Directory Access Information: In order to retrieve certificates and ARL/CRLs, it is assumed that the relying party accesses its directory. The relying party can obtain the directory access information from the followings:

(1) AIA extension field in certificates
(2) Directory Access Information managed by the relying party

If the AIA extension is present in a certificate, the relying party can obtain Directory Access Information from it. If not, the relying party should use the Directory Access Information locally managed by himself/herself. When a referral or chaining method is used for the directory, the local directory information will be sufficient to get information from various directories.

The Directory Access Information for retrieving ARL/CRL can be obtained from the CRL DP extension. The DP name must be the directoryName or the LDAP URI. When directoryName used, the relying party is able to get directory server information in other way. When LDAP URI used without attribute value, the relying party is able to determine the attribute name using basicConstraints extension.

(b) Key Identifier for Certificate Chain: To facilitate path construction certificates and CRLs should use AKI and SKI containing Key Identifier.

2.3.2 Certificate Chain Construction

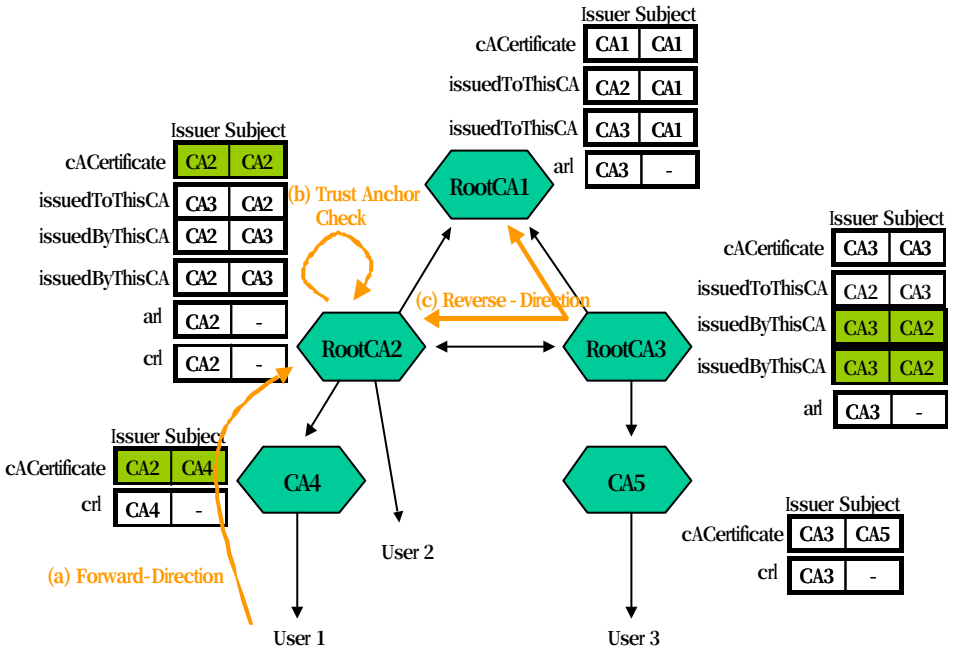
Followings are three methods of how to construct the certificate chains. (The selection is a local policy)

(a) Utilize the certificates provided by the signer
(b) Utilize the certificates already stored locally

(c) Utilize certificates retrieved from the repository

In any PKI domains to which the target certificate belongs, the Forward-Direction method is more efficient to construct the Certificate Chains than the Reverse-Direction method if each PKI domain is based on the Strict Hierarchy model. However, it is better to construct the certificate chains in inter-PKI domains, because the CA can issue a number of cross-certificates to other domain Root CAs.

The Certificate Chain Constructions from a target certificates are as follows.[5][6]



[Figure 11] Certificate Chain Construction Algorithm

(a) Forward-Direction Method: The relying parties construct the Certificate Chains from target certificates to self-signed certificate by using Forward-Direction Method. In the case that the User3 verifies User1's certificate, the certificate chains to be constructed are like "RootCA2 -> CA4 -> User1". It is general to store the Root CA certificates as the attribute of *cACertificate* in the RootCA directory. However, the attribute *issuedToThisCA* field of crossCertificatePair Attribute may be used to construct Certificate Chain.

(b) Trust Anchor Checks: The relying parties verify that Self-Signed certificate(RootCA2) constructed through step(a) is included in the trust anchor information of the relying party. If included, the Certificate Chain Construction Algorithm is closed. If not included, the step(c) is

processed.

(c) Reverse-Direction Method: The relying parties construct the Certificate Chains from cross-certificate issued by own trust anchor to RootCA2, by using the Reverse-Direction Method. The attribute of issuedByThisCA field of the crossCertificatePair may be used in Reverse-Direction Method.

2.3.3 Retrieval of ARL/CRL

Followings are three methods of retrieving ARL/CRL. (The selection is a local policy)

(a) Utilize ARL/ CRL provided by the signer
(b) Utilize ARL/CRL already stored locally
(c) Utilize ARL/CRL retrieved from the repository

In each method, the validity checks, issuer checks and key identifier checks for ARL/CRL SHOULD be processed in advance for the efficiency of certificate path processing. If CRL DP extension is present in certificate, the relying party can obtain ARL/CRL based on the CRL DP extension. If not, the relying party can obtain ARL/CRL from the entry of issuer DN.

2.4 Considerations

In this chapter, some additional considerations for the implementation of certificate path processing that was not covered above will be covered.

2.4.1 Using VA(Validation Authority)

The some parts of certificate path processing defined in this specification can be delegated to a trusted VA. Protocols for accessing a VA server must stick to the relevant specification or standards. If the VA server provides not the whole of the certificate path processing described in this specification, the relying parties should process the rest of the certificate path processing. Also, the interfaces should be provided, through which the relying parties can determine whether to use VA or not.

(a) Interoperability requirements
    (1) We recommend using the OCSP protocol for the services of VA.

(2) Though a CA uses only VA as the validation mechanism, the CA MUST support both VA and CRL mechanisms for the interoperability between PKI domains.

(3) Some requirements using OCSP for VA services:

1. All extensions of OCSP messages SHOULD be non-critical for the interoperability.
2. The format of nonce extension is not specified clearly in RFC2560, so it MUST be specified. The implementation using the extension SHOULD handle all the formats of value appearing in that extension, even though the value is not DER encoding of some ASN.1 structure.
3. To inform the relying party the location of OCSP service, it is recommended to use AIA extension(id-ad-ocsp) in the certificate.

*Additional issues for OCSP interoperability

We have the various VA mechanisms such as OCSP, SCVP and DVCS, however, as of today the OCSP has been used generally to provide the VA service. The OCSP has been implemented to verify the status of certificates on the 2001 IWG and Korea has the accredited CAs operate the OCSP Server for the interoperability of financial sections.

This phase explains the additional issues related to the interoperability between different OCSPs

(a) The IETF RFC 2560 OCSP protocols can be customized to meet the various PKI application services. The private extensions can be added or some particular extensions can be marked as critical or non-critical, as opposed to the standard. For this reasons, if the OCSPv1 has additional extension fields for various services, these MUST be non-critical for interoperability.

(b) The followings are issues raised when OCSP interoperability test was performed between accredited CAs in Korea

(1) To prevent replay attacks, the nonce is are generally used. But, the RFC 2560 doesn't specify the length and the value type of nonce. So, accredited CAs of Korea agreed as follows

- The value type of nonce is OCTET STRING
- The length of nonce is not limited.

(2) The Relying parties must obtain the access information of the OCSP Server to receive the OCSP services. It is obtained by the AIA extension of the target certificate according to the RFC 2560. If

the AIA extension in the target certificate is not present, the relying parties have to use the local DB. In Korea, the AIA extension as well as the local DB are being used and the AIA extension has priority over the local DB.

3 Reference

[1] W.Ford and D.Sole, "Internet X.509 Public Key Infrastructure Certificate and CRL Profile", IETF PKIX RFC3280, April 2002

[2] “Achieving PKI Interoperability”, JKS-IWG, April 2002

[3] “Achieving PKI Interoperabiltiy- Recommendations on Technical Certificate Profile”, JKS-IWG, April 2002

[4] Russ Housley and Tim Polk, "Planning for PKI", WILEY, 2001

[5] ITU-T Recommendation X.509, “Information Technology-Open Systems Interconnection-The Directory : Public Key and Attribute Certificate Frameworks", March 2000

[6] Steve Llody, "Understanding Certification Path Construction", PKI Forum White Paper, September 2002