GWD-I

CAOPS-WG

David Groep, Nikhef
Jens Jensen, STFC

Version 20100905287
Updated: May, 201099

Relying Party Defined Namespace Constraints Policies in a Policy Bridge PKI Environment

Status of This Document
This document provides information to the Grid community. It does not define any standards or technical recommendations. Distribution is unlimited.

Abstract

Relying Party Defined Namespace Constraints (RPDNC) are limitations on the subject namespace issued by X.509 certification authorities (CAs) that are defined and enforced by the end-point at the relying party side. As grid authentication based on X.509 credentials provides the subject DN as a handle that identifies the authenticated entity, the capability to ensure subject name uniqueness is of critical importance in ensuring overall integrity of the authentication system.
This document described the rationale and use cases for relying party defined name space constraints, and lists the set of desired features a policy language expressing such constraints should have.

Contents

1. Introduction

This document described the rationale and use cases for relying party defined name space constraints in X.509 Certificate Authorities, and lists the set of desired features a policy language expressing such constraints should have.
The background to this document can be found in the Trust Anchor Management group current requirements document, the most recent being http://www.ietf.org/internet-drafts/draft-ietf-pkix-ta-mgmt-reqs-03.txt [visited April 2009]:

A trust anchor represents an authoritative entity via a public key and associated data. The public key is used to verify digital signatures and the associated data is used to constrain the types of information for which the trust anchor is authoritative. A relying party uses trust anchors to determine if a digitally signed object is valid by verifying a digital signature using the trust anchor's public key, and by enforcing the constraints expressed in the associated data for the trust anchor.

This document describes the requirements for this "associated data" for managing subject namespaces for Grid PKIs.

2. Rationale for Relying Party Defined Namespace Constraints (RPDNC)

Relying Party Defined Namespace Constraints (RPDNC) are limitations on the subject namespace in which X.509 certificate authorities (CAs) issue certificates. These constraints are in principle defined by the Relying Party (RP) and enforced by the end-point at the relying party side. In grid authentication based on X.509 credentials, the subject distinguished name (DN) provides a handle that identifies the authenticated entity[1]. The capability to ensure subject name uniqueness is thus of critical importance in ensuring overall integrity of the authentication system. To some extent, RPDNC help enforce this constraint. Moreover, in certain types of security incidents, RPDNC help limit the scope of the incident. Finally, RPDNC give the RP some additional control over the range of certificates they accept.

In practice, the RPDNC are often provided by the IGTF [IGTF] as part of a trust anchor distribution, or by a coordinated-deployment project, or provided by the CA upon IGTF ~~[IGTF]~~ accreditation, but can be defined, replaced or augmented by individual relying parties. In principle, RPs ultimately decide which CAs and which certificates issued by those CAs to trust, but the RPNDC is normally used to enable the following use cases:

- Enforce non-overlapping CA name spaces. RPDNC allow relying parties to ensure that within the ensemble of PKIs which they trust there are no inadvertent overlaps in the subject names issued by the diverse CAs.
- Allow CAs to sub-divide their subject name space and apply different policies to different branches of this namespace in absence of any other mechanisms. For example, a specific part of the namespace may be reserved for end-entity certificates or subordinate CA certificates that comply with specific additional requirements requested by relying parties, and these relying parties can opt to accept only the part of the namespace where such requests are honoured[2].

[1] There are multiple handles that identify the authenticated entity, but the subject distinguished name is used most frequently as the primary handle, since it is persistent and uniquely assigned to the entity. This handle can then be used directly, but is also frequently used in an indirect manner when obtaining other attributes that are associated to this 'handle' of the authenticated entity. For example, an attribute issuance service such as VOMS relies on the subject distinguished name to provide attributes associated with the authenticated entity.

[2] For example, in absence of an RPDNC mechanism a root CA can issue any number of subordinate CAs, and credentials issued by these subordinates in the absence of other methods of enforcement would automatically be trusted since the root is part of the trust anchor repository. See example 6.5.

Authority-defined namespace constraints policies are common in PKI Bridging architectures that use a Bridge Certification Authority [RFC4158] to express trust relationships between the participating authorities. In a *policy bridge* architecture, this technical means of expressing relationships and coordinating the namespace for the subject directory names does not exist. With a policy bridge, it is up to the relying parties to enforce limitations on the subject namespace of each of the participating authorities in order to guarantee subject name uniqueness across the PKI as seen from that specific relying party.

3. Namespaces

For the purposes of this document, a Namespace is a non-empty set of Distinguished Names (DNs) as used in RFC 5280, containing the subject Distinguished Names that are or can be issued by a single CA (as constrained by its policies). With each full DN being an ordered sequence of sets of attribute-value pairs (referred to as relative distinguished names, RDNs[3]), the set of distinguished names will have a fixed (common) part and a naming (variable) part. The fixed, 'common' part that the longest initial sequence[4] of RDNs that all DNs have in common. The naming, 'variable' part consists of all remaining RDNs.

For example, the two DNs that, in RFC2253 format, are expressed as:

CN=John Doe, OU=pdp, O=rl, DC=example, DC=org
CN=John Doe, O=nikhef, DC=example, DC=org

have a fixed part: the common initial set of RDNs (namely DC=org and DC=example), and a remaining naming part: the 'variable' part of the DN (the O, OU and CN RDNs)

The two DNs

CN=John Doe, O=rl, DC=example, DC=org
CN=John Doe, O=rl, DC=example, DC=com

have an empty (null) fixed part, since the first RDN in the sequence is different.

The *Namespace* of a CA is the set of all DNs that the CA can issue by its policy. For convenience, the Namespace is often summarised by quoting just the fixed part (which is by definition the same for all subject DNs). This simplification allows the variable part to contain any sequence of any RDNs, subject only to technical restrictions such as the total length of the DN.

We define a Relying Party Defined Namespace Constraint (RPDNC) as a set of restrictions enforced by the relying party (as opposed to, e.g., *nameConstraints* set by a certification authority) that defines the set of fixed parts that are accepted, rejected, or for which no explicit decision is made ~~(i.e. accept, reject, or *unknown*)~~ by the Relying Party. The RPDNC set may be different from the Namespace of the CA.

Formatted: Font: Italic

Again occasionally for simplicity reasons, we may liken the RPDNC to the *positive* set, the Namespace for which the policy accepts DNs can be called the RPDNC. In turn, this can be

[3] An RDN is defined as a SET of AttributeTypeAndValue [RFC5280] but we require these sets to contain exactly one element; thus the type and value are both well defined.

[4] "Initial sequence" means a (possibly empty) subsequence starting with the first RDN of the ASN.1 encoding of the DN (RDNSequence in RFC5280), irrespective of how this is represented as a string. For example, in the LDAP representation (e.g. RFC2253 section 2.1), "DC=org, DC=example,DC=org", and "DC=host,DC=example,DC=org" are all initial sequences of the DN "DC=host,DC=example,DC=org".

summarised by its fixed part for further simplification.
Finally, we shall refer to the engine which accepts a DN and parses it against a policy and returns a decision as a Policy Decision Point (PDP).

4. RPDNC Policy Language and Expression Requirements

A quick-scan in the community of Relying Parties, e-Science grid deployment projects and Grid Certification Authorities, indicated the following features to be important for expressing a Relying Party Defined Namespace Constraints Policy.

4.1 Co-existence of authorities with and without RPDNC policies

It must be possible to have issuers with and without namespace constraints policies co-exist within the same trust anchor repository.

4.2 Independence of RPDNC policies

a. Association
It must be possible to associate a RPDNC with each individual trust anchor[5].

b. Distribution, verification
It must be possible to distribute and verify (cf. Requirement 4.6) RPDNC policies in conjunction with each individual trust anchor, independent of any other trust anchors present in the trust anchor repository.

c. Validation
It must be possible to validate certificates against the RPDNC independently of any other trust anchor that is not used to build a trust path.

d. Combination
There are use cases where it is useful to associate multiple RPDNC to a single trust anchor (s. This makes sense if the single RPDNC returns unknown, and PDPs are able to combine policies in simple sequence when this happens. See Example 6.8). The order in the sequence is significant. See also requirement 4.11.

4.3 Support for dynamic hierarchies

It must be possible to support the concept of "subordinate" issuers in a hierarchical chain of issuers, such that a single namespace constraints policy collection (file) supports the expression of namespace constraints on any subordinate issuer.

4.4 Support for static hierarchies

It must be possible to exhaustively list namespaces. A RPDNC may limit the number of DNs that can appear in such an explicit list, but in this case the upper limit should not be less than 32 per installed trust anchor.

4.5 Expression of subject DN namespaces as strings

The string rendering identifier naming of directoryNames and X.500 DNs in the policy expression must comply with RFC4514[6].

[5] A single CA may operate with more than one trust anchor, e.g., by signing EE end-entity (EE) certificates with more than one CA certificate (with distinct names), or by having a separate certificate to sign CRLs.

[6] For purposes of the RPDNC, we propose to require that any RDN component contains one and only one Attribute-Value assertion, and that the name of the attribute be represented by the full name of the attribute as defined in the X.400/X.500 document series, or alternatively as defined in

4.6 Usability and human readability of the policy

The format used to express RPDNC policies must be human readable in order for relying parties to visibly inspect, edit, and assess the namespace constraint policy.

4.7 Name sub-tree support and the use of wild cards in names

The policy expression must support pattern matching[7] with at least a match-all wildcard and branch exclusions. The wild-carding should work on the full string representation of the DN

4.8 Sub-tree specific policies and policy-file precedence

It must be possible to explicitly set a namespace constraints policy for a subordinate issuer, without modifying the policy collection (file) for the up-stream issuer(s). Such a policy on a subordinate issuer must not be able to broaden the namespace constraints defined by higher-level CAs.

4.9 Independence of non-namespace trust anchor characteristics

A subordinate authority trust anchor must be able to change (i.e. a subordinate could be compromised and re-keyed) without having to change the namespace constraints policy in any end-system configuration, provided it does not change its DN.

4.10 Policy collision

The probability for collisions in the policy expression format must be vanishingly small[8].

4.11 Policy combination

It should be possible to selectively override parts of a name space definition received by agents of the relying party by 'downstream' redistributors or RPs, without modifying distributed RPDNC files.

~~*4.11 Dynamic Policy Extension*~~

~~If no specific RPDNC is not defined for a particular part of the Namespace, it can be extended to the full namespace either by a default-deny policy, or a default-unknown policy. A default-deny will be fail-safe, but limits the possibility for a relying party to combine RPDNCs from various providers. A default-unknown policy will allow combination or RPDNCs from multiple providers. A default-deny should then be applied only if none of the policies applicable to a trust anchor failed to render a decision. A default-unknown policy, with ultimate deny, is considered to be the most practical.~~

Some of the desired features correspond to similar namespace constraints requirements in the X.509. It is advised for a RPDNC policy language to follow closely the X.509 namespace constraints where possible.

---

the RFCs. When a unique short name has been defined in these documents, this short name must be used. Where no short name has been defined in these documents, a short name must not be used.

[7] Although it was requested to support wildcard matching anywhere in the pattern, in order to accommodate distinguished names where the most-variable part of the DN was not at the end of the string. However, this request conflicts with the request to align closely with the SubTree namespace constraints as defined in X.500

[8] Meaning that the file names of the files for different trust anchors derived from the subject DN should not collide, e.g., ~~the~~ non-colliding hash names ~~should~~ MAY be used. The hash length should be commensurate to the number of installed trust anchors (expecting 1000+ CAs is expected to be reasonable, and a 64-bit hash considered sufficient) .

Formatted: Normal

Formatted: Heading 2, Outline numbered + Level: 2 + Numbering Style: 1, 2, 3, … + Start at: 1 + Alignment: Left + Aligned at: 0 cm + Tab after: 1.02 cm + Indent at: 1.02 cm

Formatted: Normal

Formatted: English (United States)

5. RPDNC distribution and life cycle

The RPDNC is associated with a trust anchor or set of trust anchors, and should be managed together with the trust anchor. If a trust anchor distribution is used that provides RPDNCs alongside the trust anchor, and if those RPDNCs are not modified by the individual relying party, they should be installed and decommissioned together with the trust anchors involved.

If an RPDNC is modified locally, or – in a chain of trust anchors – an RPDNC of a higher-level trust anchor has been modified locally, any changes in the trust anchor chain at the same or a subordinate level should be updated to reflect the intended policy of the relying party.

RPDNC definitions for trust anchors that are not present in the trust anchor store are not evaluated, but should be removed in order to prevent accidental use in case the associated trust anchor is installed at a later date.

Formatted: nobreak, Tab stops: Not at 0.63 cm

~~5.~~6. Current RPDNC Policy Languages

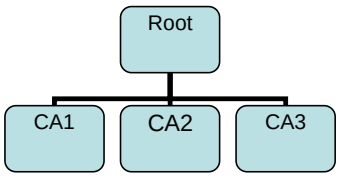
The first RPDNC Policy language was introduced in the Globus Toolkit [GT] in 1997, based on the EACL Extended ACL language format [EACL]. In this policy, commonly referred to as the "signing policy", specific restrictions can be based on the subject namespace on a per-authority basis. For all Globus Toolkit releases version 2.0 and higher, this policy is stored in a single~~gle~~ file associated with each CA certificate. The implementation allows for a list of allowed namespaces to be expressed, within certain limitations.

An alternative "namespaces" policy language [NS96] has been experimentally distributed since 2005 as part of the Common Trust Anchor distribution of the International Grid Trust Federation.

~~6.~~7. Examples

This section describes examples, most of which are taken directly from real-life cases. For convenience, the examples do not distinguish between a CA and its CA certificate unless stated otherwise.
The lines in the graphs indicate a directed signing relationship (from the top downwards, the higher level CA signs the subordinate CAs) and a bidirectional 'has common management' relationship.

6.1 Simple hierarchy



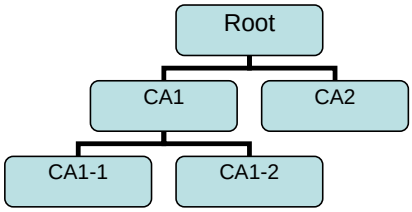
[Requirements 4.2, 4.4, 4.8]
This example shows a simple hierarchy. We assume CA1 issues Grid EE certificates. In this case, the RPDNC are defined to permit all end entities issued by CA1, and obviously none from CA2 and CA3. In Globus-based Grids, the Root will also have to be installed. For the Root, the RPDNC is then defined to accept CA1 only and not CA2 or CA3[9].

[9] In some versions of the Globus Toolkit (specifically versions 1.0 – 1.1.4, as well as version 4.0.8), the signing policy RPDNC must permit the Root to sign itself.

6.2 Namespace slicing

[Requirements 4.5, 4.6, 4.7]
A CA issues EE certificates: ..., OU=Koalas, O=Grid, C=XX and ..., OU=Wombats, O=Grid, C=XX.
However, this CA is not trusted on the Grid to authenticate Wombats. The Relying Party (RP) will wish to implement an RPDNC which enables Koalas to access the Grid and not Wombats[10],[11].

6.3 Multi-tier hierarchy



[Requirements 4.2, 4.4]
In this example, only CA1-1 issues EE certificates which are trusted on the Grid. In traditional Globus-based Grids, this means the path CA1-1 → CA1 → Root is installed on the server.
There are two use cases: one is to manage the namespaces for secured Grid resources which require each certificate in the path to be installed, and the other is to manage the namespace for those that do not. For the latter, the problem is the client may send CA1-2 and CA1 or CA2 along with its EE certificate (if issued by CA1-2, resp., CA2), and these certificates will be accepted because the server uses the supplied intermediate certificates to build a valid chain to the Root. In this case, RPDNC for CA1 shall ensure that only CA1-1 can be used, and likewise, RPDNC for the Root shall ensure that only CA1 will be relied on to build validation chains.
In the absence of RPDNC, the traditional approach to the trust repository is to install only the trusted CA, CA1-1 in this example. However, we still need to build a validation chain to the Root because we require that the CRLs of CA1 and the Root be checked during the validation of an EE certificate issued by CA1-1. Moreover, the repository has to serve resources from both worlds: both those that require that the chain be installed, and those that don't.
For the former case, where all intermediate certificates also have to be installed in the trust repository on the server side, the client's other intermediate CAs will not be trusted, and there is less need for the RPDNC for the CAs.

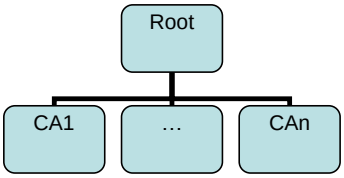
6.4 Multi-tier issuing hierarchy

[Requirements 4.2, 4.5, 4.6, 4.8]
This Example continues Example 6.3. Suppose in addition to the validation chain CA1-1 → CA1 → Root, all these CAs also issue end entity certificates[12], and that CA1-1 only is trusted to issue EE certificates for the Grid. (If CA1-1 also issues CA certificates, then, as in Example 6.3, RPDNC for CA1-1 must now prevent untrusted subordinates under CA1-1 from being inserted

[10] Some CAs do not permit slicing their namespaces like this. In this example, if the OU is not validated by the RA, or no special meaning is associated with the OU, it makes sense that the CA shall not permit RPs to distinguish EEs by OU.
[11] One might argue this should be done by the CA giving different policy OIDs to Koalas and Wombats, and RPs should be able to check this. Support for this in middleware is beginning to appear as of this writing but is not ubiquitous. Also, there are many cases where CAs have different classes of certificates (e.g., personal/host/robot) in different namespaces but give the same OIDs to all certificates because they are signed under the same policy.
[12] It may seem far fetched to have CAs that issue both certificates to subordinates and to end entities, but this example is built on a real case.

into validation chains for applications that do not need the whole path in the trust repository.)
We now need to exclude EE certificates from CA1 and Root from being trusted on the Grid. Currently, this is best done by restricting the namespaces with RPDNC.

6.5 Policy Root



[Requirements 4.3]
Suppose the Root is defines its policy so each of the subordinates can be accredited without being reviewed individually but the subordinate hierarchy is dynamic, e.g., because they are short lived (naturally, this would only work for applications that do not require the trust chain installed on the server side). In this case, we want the RPDNC for the Root to be able to accommodate a range of subordinates whose names are not necessarily known in advance. A related use case is where a CA operates with more than one CA certificate (with different names) which issue in the same namespace.

6.6 Improving Grid security through namespaces

[Requirements 4.4, 4.9]
RPDNC can limit the impact of security incidents. Suppose a CA (certificate) X has signed Y and X is compromised – not necessarily the certificate itself but more likely the CA's processes. An attacker obtaining Y_1 signed by X will not be able to use Y_1 unless the name of Y_1 is accepted by X's RPDNC.

Since Grid resources must have Y installed along with X in their trusted repositories, they will not a priori trust Y_1. To make use of Y_1 on the Grid, the attacker must make Y_1 an end entity certificate, but the name of Y_1 must still be accepted by the RPDNC. It thus limits the scope of the incident.

6.7 Distinguishing types of clients

[Requirements 4.2, 4.6]
A Grid CA can currently issue certificates to users (individual persons), hosts, and robots (automated clients). These certificates are typically issued under a single policy, with a single set of OIDs common to them. If an RP wants to distinguish between these, they have to be distinguished by DN (there is no single rule that will work for all CAs, but for each individual CA it is usually possible to derive rules that will match accordingly).
A RP will then wish to match against not just the fixed part of the DN, but also the variable part, to ensure that only the CN (and potentially other parts of the variable space) match the clients the RP wishes to let in. For example, current implementations have a CN containing "Robot:" for robot certificates (in a scheme designed not collide with the DN of a person who might be named "Robot").

6.8 Combining and the unknown

[Requirement 4.2, ~~4.11~~4.11]
We refer again to the diagram in Example 6.3. If we assume CA2 is IGTF-accredited and therefore trusted internationally, but CA1 issues certificates to national subordinate CAs which are trusted only by the national Grid.
In this case, the Root has an RPDNC which permits only CA2, but the national RP will want the

branch under CA1 accepted as well. However, the national RPs will refresh IGTF CAs directly from a suitable repository, but install the national CAs manually. The IGTF repository will include a ~~PMA~~IGTF-approved RPDNC permitting the Root to sign only CA2. If the Root has a single RPDNC, then this will conflict with (or overwrite) the national Root RPDNC which permits also CA1.
Suppose instead the IGTF Root RPDNC returns 'unknown'[13] for CA1, and the RPDNC PDP permits multiple RPDNC in a well defined sequence. In this case, the national Grid can have a second RPDNC which accepts CA2. On the national Grids, the default IGTF RPDNC for Root will return unknown for CA1, which ~~the PDP~~ a policy decision point (PDP) then passes to the second RPDNC for Root, which then is able to take a decision and accept CA1. Naturally, a PDP must deny DNs that ultimately end up being unknown, with no further RPDNC to process.
The advantage of this scheme is that both international and national RPs can use the same IGTF release, and the national modifications need only be installed once.

Formatted: Font: Not Italic

~~7.~~8. Security Considerations

The namespace policy is an integral part of the security and protection mechanisms of a relying party, and as such should be protected from tampering at all times. Inadvertent or malicious modification of a RPDNC policy can lead to namespace collisions, resulting in incorrect subject being authorized, or may expose a relying party to credentials issues under policies that are inappropriate or unacceptable, or to denial-of-service.

In case the namespace constraints policy is distributed to the relying party by a third party, this distribution mechanism must be integrity-~~secured~~protected and protected from denial-of-service. Once obtained by the relying party, it should be adequately protected from tampering. It is acknowledged that these requirements are the same as those for the distribution of trust anchors, and are affected by similar boot-strap issues.

A different between the RPDNC and the actual namespace used by the CA is not indicative of any lack of trust or lack of trustworthiness of the CA, but merely reflects the decision of a relying party or their agent(s). The reasons for an RPDNC may be local, e.g., to prevent name space overlaps between the CAs accepted by an RP, or to select identities that comply with specific policies.

~~8.~~9. Contributors

The document is a work of the OGF CA Operations Working Group with contributions by the members of the International Grid Trust Federation (IGTF, see www.gridpma.org)

The editors,
David L. Groep, Nikhef, davidg@nikhef.nl
Jens Jensen, STFC, j.jensen@rl.ac.uk

~~9.~~10. Intellectual Property Statement

The OGF takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the

[13] Depending on an eventual implementation, see appendix A.

Formatted: English (United States)

OGF Secretariat.

The OGF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights which may cover technology that may be required to practice this recommendation. Please address the information to the OGF Executive Director.

~~10.~~11. Disclaimer

This document and the information contained herein is provided on an "As Is" basis and the OGF disclaims all warranties, express or implied, including but not limited to any warranty that the use of the information herein will not infringe any rights or any implied warranties of merchantability or fitness for a particular purpose.

~~11.~~12. Full Copyright Notice

~~12.~~13. References

[EACL]
Anonymous (the Globus Toolkit Authors) Extended Access Control List (EACL) Format Specification (as stored on http://www.eugridpma.org/documentation/eacl-signing-policy-format.txt)

[GT]
Globus Alliance *The Globus Toolkit* http://www.globus.org/

[IGTF]
The International Grid Trust Federation http://www.igtf.net/

[NS96]
D.L. Groep *Namespaces Format Specification* EUGridPMA Technical Documentation Series, 2006 (http://www.eugridpma.org/documentation/)

[RFC4158]
M. Cooper et al. *Internet X.509 Public Key Infrastructure: Certificate Path Building: RFC 4158*. September 1995.

Formatted: Heading 1, Indent: Left: 0 cm, Hanging: 0.63 cm, No bullets or numbering

Appendix A

This appendix describes additional suggestions for implementing or drafting an RPDNC policy language, but is explicitly non-normative.

The dynamic policy extension requirement 4.11 can be implemented in various ways, with the example in this appendix being a possible implementation.

A.1 Dynamic Policy Extension

If no specific RPDNC is not defined for a particular part of the Namespace, it can be extended to the full namespace either by a default-deny policy, or a default-unknown policy. A default deny will be fail-safe, but limits the possibility for a relying party to combine RPDNCs from various providers. A default-unknown policy will allow combination or RPDNCs from multiple providers. A default-deny should then be applied only if none of the policies applicable to a trust anchor failed to render a decision. A default-unknown policy, with ultimate deny, is considered to be the most practical.
The order of evaluation of the sequence is then significant.

Formatted: Normal, No bullets or numbering

Formatted: Indent: Left: 0 cm, Hanging: 0.63 cm, No bullets or numbering