Design for a Nameserver Control Protocol
draft-dickinson-dnsop-nameserver-control-00.txt

Status of this Memo

By submitting this Internet-Draft, each author represents that any
applicable patent or other IPR claims of which he or she is aware
have been or will be disclosed, and any of which he or she becomes
aware will be disclosed, in accordance with Section 6 of BCP 79.

Internet-Drafts are working documents of the Internet Engineering
Task Force (IETF), its areas, and its working groups. Note that
other groups may also distribute working documents as Internet-
Drafts.

Internet-Drafts are draft documents valid for a maximum of six months
and may be updated, replaced, or obsoleted by other documents at any
time. It is inappropriate to use Internet-Drafts as reference
material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at
http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at

This Internet-Draft will expire on April 30, 2009.
Abstract

This document presents a design for a nameserver control protocol (NSCP).

A common data model for describing the configuration and operation of a basic, but usable, generic name server is defined. This is expressed in a formal modeling language (YANG) and can be used as the basis of a set of NETCONF operations and capabilities.

The data model described is extensible and will allow for the creation of additional capabilities, ensuring that the protocol can support all the features of a name server.

Table of Contents

1. Introduction ........................................... 4
   1.1. Background ........................................ 4
   1.2. NSCP ................................................ 4
   1.3. Use of NETCONF .................................... 4
       1.3.1. NETCONF Overview .............................. 4
       1.3.2. Why NETCONF? ................................ 5
   1.4. Requirements notation .............................. 6
2. Object Models ........................................ 7
   2.1. NSCP Base Capability .............................. 7
       2.1.1. Server .......................................... 8
       2.1.2. Statistics ...................................... 8
       2.1.3. DNSSEC Policy .................................. 9
       2.1.4. Peer ............................................ 10
       2.1.5. Peers ........................................... 11
       2.1.6. Panorama ....................................... 11
       2.1.7. View ............................................ 12
       2.1.8. Access Control List ............................ 13
       2.1.9. Zone ............................................ 15
   2.2. NSCP Basic Control Capability .................... 16
       2.2.1. Methods .......................................... 16
   2.3. NSCP Start Control Capability ..................... 16
       2.3.1. Methods .......................................... 16
3. Examples ............................................... 17
   3.1. Obtaining Configuration Information .............. 17
   3.2. Modifying Configuration Information ............... 18
   3.3. Controlling the Name Server ........................ 20
4. IANA Considerations .................................. 21
5. Security Considerations ................................ 22
6. Normative References .................................. 23
Appendix A. Meeting the Requirements .................. 24
   A.1. Management Architecture Requirements ............ 24
A.1. Expected Deployment Scenarios .......................... 24
A.1.2. Name Server Types .................................. 25
A.2. Management Operation Types .............................. 25
A.2.1. Control Requirements ................................. 25
A.2.2. Configuration requirements ............................ 25
A.2.3. Monitoring Requirements ............................... 26
A.2.4. Alarm and Event Requirements .......................... 26
A.2.5. Security Requirements ................................. 26
A.2.6. Other Requirements .................................. 27
Appendix B. NSCP Capabilities ................................ 28
B.1. The NSCP Base Capability ................................ 28
B.1.1. Capability Name ..................................... 28
B.1.2. Overview ............................................. 28
B.1.3. Dependencies ......................................... 28
B.1.4. Capability Identifier ................................. 28
B.1.5. New Operations ....................................... 28
B.2. The NSCP Basic Control Capability ......................... 28
B.2.1. Capability Name ..................................... 28
B.2.2. Overview ............................................. 28
B.2.3. Dependencies ......................................... 29
B.2.4. Capability Identifier ................................. 29
B.2.5. New Operations ....................................... 29
B.3. The NSCP Start Control Capability ......................... 29
B.3.1. Capability Name ..................................... 29
B.3.2. Overview ............................................. 29
B.3.3. Dependencies ......................................... 29
B.3.4. Capability Identifier ................................. 29
B.3.5. New Operations ....................................... 29
Appendix C. YANG Data Model for Base NSCP capability ............ 30
Appendix D. YANG Data Model for the NSCP Basic Control Capability .......... 36
Appendix E. YANG Data Model for the NSCP Start Control Capability .......... 37
Authors’ Addresses ........................................... 38
Intellectual Property and Copyright Statements ................... 39
1. Introduction

1.1. Background

Management of DNS name servers is currently carried out via vendor-specific control, configuration and monitoring methods. Organizations run multiple name server implementations from a variety of vendors. A common method of name server management can simplify administration and reduce cost.

The requirements for the management of name servers have been established and documented [I-D.hardaker-dnsops-name-server-management-reqs]. In essence, the document describes a set of common operations that name servers are known to implement.

1.2. NSCP

NSCP is a name server control protocol that meets the requirements set out in [I-D.hardaker-dnsops-name-server-management-reqs]. Based around NETCONF [RFC4741], NSCP consists of a common data model for describing the configuration and operation of a basic, generic name server that is comparable with a subset of the features of existing name server implementations. This data model, expressed in a formal modeling language (YANG [I-D.ietf-netmod-yang]), is used as the basis for a set of NETCONF operations and capabilities.

The basic NSCP data model is extensible and allows for the creation of additional NETCONF capabilities that will ensure the protocol can support all the features of a name server.

Using NSCP, a suitable client should be able to communicate with and manage any name server implementing the protocol.

It is the intention of NSCP that the NSCP data model will be transformable into a data model suitable for use with a particular name server implementation. In the longer term it is hoped that name servers will implement the NSCP data model directly.

1.3. Use of NETCONF

1.3.1. NETCONF Overview

NETCONF is a protocol for the management and configuration of network devices, where operations are layered on top of a remote procedure call interface. A client establishes a session with a server via a secure, connection-oriented transport mechanism (such as SSH).
The operations sent to the server, the replies from it and the configuration data itself are encoded in XML.

Within NETCONF, capabilities identify sets of functionality that a client or server may implement. During the setup of the session, client and server exchange a list of capabilities. As each system ignores capabilities that it does not require or understand, the two systems can settle on a common set understood by both. These capabilities allow the client and server to agree on

- New operations
- Modifications to existing operations
- The data model(s) being used.

NETCONF provides a set of simple operations that allow management of configuration data and retrieval of state information.

1.3.2. Why NETCONF?

There are a number of reasons for using NETCONF as the basis of NSCP:

- It is an established application protocol, allowing reuse of existing tools and code.
- It is connection-oriented, running over secure links. This matches anticipated use of NSCP.
- Use of XML allows established transformation methods such as XSLT to be used to transform the NSCP configuration into a vendor specific configuration format.
- Configuration information is opaque to NETCONF, allowing any data to be transported over it.
- It supports seamless extension of functionality via its capabilities feature.

It is capabilities that make NETCONF particularly suitable for use as the basis of NSCP. In particular, three requirements of [I-D.hardaker-dnsops-name-server-management-reqs] (section 5.1) are:

- The management solution must be flexible and be able to accommodate new future operations.
- It must be possible for vendors to extend the standardized management model with vendor-specific extensions.
It must be possible for a management station to understand which parts of returned data are specific to a given vendor or other standardized extension.

Capabilities clearly fit these requirements; by defining extensions to NSCP – either vendor-specific or standard ones defined at a later date – as NETCONF capabilities, additional features can be added to NSCP whilst maintaining backwards compatibility with existing systems.

1.4. Requirements notation

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].
2. Object Models

NSCP is built up from a set of capabilities. These provide different parts of the object model as described below.

2.1. NSCP Base Capability

[I-D.hardaker-dnsops-name-server-management-reqs] (section 2.1.5) states that a common data model MUST be defined. This is a very necessary requirement, since only by establishing a common definition about what is being managed can management clients and name servers exchange meaningful information.

During the design of NSCP, several existing name server implementations were examined and the common details abstracted into a common model. Inevitably, many detailed features of individual name servers are not included. However, the capability mechanism of NETCONF will allow them to be added as vendor-specific extensions.

The following figure shows the overall structure of the object model within the "basic" capability.

```
+------------+---Server----+-------------+
|1           |1            |1            |1
|            |             |             |
|1           |1            |1            |1
Peers       Panorama    Statistics   DNSSECPolicy
|1           |1            |1
|*           |*             |
Peer   +-----View-----+
|1             |1
|*             |
Zone           ACL
```

(The numbers indicate the cardinality of the associations.) The following sections describe each object in more detail. For every object, the following information is provided:

- A discussion of the object – what it represents and its purpose.
- A list of the object’s elements – the name of each element and the contents.
2.1.1. Server

The server object is the root of the configuration tree and serves as the focus for server-wide operations and repository for server-wide information.

2.1.2. Statistics

In the base capability it is assumed that the server is capable of generating the subset of Bind 8 style statistics currently supported by both NSD and Bind 9.

Additional statistics will be the subject of future capabilities.

- RR
  Count of responses received.
- RNXD
  Count of NXDomain received
- RDupR
  Count of duplicate responses
- RFail
  Count of SERVFAIL responses received
- RFErr
  Count of FORMERR responses received
- RErr
  Count of other errors received
- RAXFR
  Count of AXFR initiated
- RLame
  Count of lame delegations received
- SSysQ
  Count of NS address fetches
- SAns
  Count of answers sent
- SFwdQ
  Count of queries sent
2.1.3. DNSSEC Policy

The DNSSEC policy defines a policy for the DNSSEC validation and signing operations performed by the name server. Any signing policy will be held in a key and signing policy database (KASP). The details of KASP are still being developed; for now the data model will just specify the location of the database as a string. Trusted keys used for validation will be stored in a manner defined by the implementation, therefore there will be no need for NSCP to specify a trusted keys file. It is also assumed that all name servers managed using NSCP will be DNSSEC-capable.

2.1.3.1. Elements

secure
The name of a domain that must validate as secure. There may be more than one of these.

nonsecure
The name of a domain that may validate as insecure. There may be more than one of these.

KASP-database
The name of the KASP database.
2.1.4. Peer

2.1.4.1. Discussion

A Peer is an object representing an external system or systems (since there is no way to tell if a key or IP address refers to a single system). A Peer is either a system that participates in the nameserver service by being a master or a slave, or is a system that uses the nameserver service. It is identified by a name and must contain either a key element or an address element, or both. All references to external systems in the rest of the NSCP object model refer to a Peer.

2.1.4.2. Elements

  o  name
    A name for this Peer. This name is unique, and is used elsewhere in the model as a reference to this object.

  o  address
    Describes an IP address. There can be zero or more address elements in the Peer although, if there are no address elements, a key element MUST be present. The contents of the address element are:

    *  ip
      This is mandatory and holds the IP address of the Peer. The IP address can be either IPV4 or IPV6. The address/network is represented in standard form (e.g. 192.0.2.0/24 or 2001:0DB8::/32).

    *  port
      Optional port number.

  o  key
    A TSIG key to be used when talking to this Peer. This is optional. The contents of a key element are one of:

    *  hmac-md5
      This holds the secret for HMAC-MD5

    *  hmac-sha1
      This holds the secret for HMAC-SHA1

    Use of an element per algorithm will allow easy augmentation with future algorithms.

A Peer allows for the identification of an external systems. Where
only an address element is present, the identification is clear. Should only a key element be present an external system corresponds to the Peer if it presents a suitable signature. The combination of address and key elements allows TSIG transactions to be more discriminating; an external system corresponds to the Peer if and only if it presents correct signatures and its address is correct.

2.1.5. Peers

A container for Peer objects. This may be useful as it provides a point at which a partial lock can be placed [I-D.ietf-netconf-partial-lock] to lock a group of Peers without affecting the rest of the server.

It is thought that there may be some use for named groups of peers to be allowed. This will be considered more in future versions of this draft.

2.1.5.1. Example

The following is an example of a group of Peers.

```xml
<peers>
  <peer>
    <name>Server123</name>
    <address>
      <ip>192.0.2.0/24</ip>
      <port>53</port>
    </address>
    <address>
      <ip>2001:0DB8::08:EF</ip>
      <port>53</port>
    </address>
  </peer>
  <peer>
    <name>OtherServer</name>
    <key>
      <hmac-sha1>hQdEr7iwB8ATMuZAj1YFQ==</hmac-sha1>
    </key>
  </peer>
</peers>
```

2.1.6. Panorama
2.1.6.1. Discussion

The Panorama is a collection of views, it provides a point at which a partial lock can be placed [I-D.ietf-netconf-partial-lock] to lock all views without affecting the rest of the server.

2.1.7. View

2.1.7.1. Discussion

The view object can be considered as a virtual server. It groups zones with similar access characteristics. It is more than just the association of a zone and an ACL: a view allows the server to send different replies to clients according to the following criteria:

- Source address (and port).
- Destination address (and port).
- Whether or not the query requests recursion.

The replies can differ in terms of:

- Which zones are available to a given client.
- The contents of those zones.
- Is recursion available?
- Is validation performed?
- Any configuration parameters of the server or zone.

To aid in the definition of a common object model, a view will always exist in a NSCP name server configuration, even if the name server concerned does not implement views. All implementations of NSCP MUST implement a view named "_default".

In this data model all zones in a given view will have the same masters and slaves.

2.1.7.2. Elements

- name
  A name for the view.

- listen-on
  What IP address and port to bind this view to. This can occur
zero or more times.

- recursion
  Is recursion enabled in this view ("on" or "off").

- validation
  Is DNSSEC validation performed in this view ("on" or "off").

### 2.1.8. Access Control List

Access to services on the name server are controlled by Access Control Lists (ACL). Using common nomenclature, an ACL comprises one or more Access Control Entries (ACEs). Each ACE links an attribute of the accessor (an "identifier") to one or more access rights to the resource being controlled.

An ACL is attached to a view. However, it may be useful for it to be attached to a zone. This will be considered further in a future version of this draft.

An ACL contains the following elements:

- ace
  Zero or more access control entries, each entry defining a match between an identifier and access modes.

- default
  Zero or one default access control entries defining access rights should no other ACE match.

The order of ACEs within an ACL is important. When checking for access, the system MUST attempt to match each ACE in turn. If none match, the system MUST attempt to match a default ACE (a wildcard that matches any accessor). If there is no default accessor, access MUST be denied.

#### 2.1.8.1. ace

An ACE links an identifier with one or more access modes. An ACE has the following sub-elements:

- identifier
  This element - of which there must be one and only one - defines what is accessing the system. The element’s value is the name of a Peer defining one or more external systems.

- access
  Access elements define what access rights the accessor has to the
server, such as operations they are able to undertake and data they are able to see. There are one or more access elements in each ACE. The value of this element is one of the following:

- **none**  No access. This overrides any other type of access, and denies access to the accessor.
- **notify**  Causes the server to take notice of NOTIFY messages from the accessor.
- **nonrecursive**  Allows the accessor to make a non-recursive request to the server.
- **recursive**  Allows the accessor to make a recursive request to the server.
- **transfer**  Allows the accessor to transfer data from the server (either AXFR or IXFR).
- **update**  Causes the server to accept dynamic update messages from the accessor.

### 2.1.8.2. default

A default access control entry is consulted only if all explicit ACEs fail to match. It does not contain an "identifier" clause (being deemed to match all identifiers), only access elements as described above.

### 2.1.8.3. Example

The following ACL would allow any accessor to query the zones in this view. Any system in the Peer LocalGroup would also be able to AXFR and IXFR from zones in the view. Systems in the Peer MasterSystem (which would presumably contain a "key" element defining a TSIG/SIG0 key) could dynamically update the zone.
2.1.9. Zone

2.1.9.1. Discussion

The zone object represents a zone served by the name server and comprises a collection of resource records. In virtually all cases, the zone will not be created by an NSCP client, being defined instead by the contents of a zone file, an external database or by zone transfer. If it is necessary to define zone contents using NSCP then the zone contents can be defined using a Zone Data Capability which will be the subject of a separate draft. This means that with just the base capability it will only be possible to provision zones via AXFR (In other words, to create the zone configuration specifying a master server from which the name server will have to obtain the zone data).

In this data model it is expected that filenames and directory structure will be fixed by the implementation. Therefore, a zone does not have a filename element.

2.1.9.2. Elements

- **name**
  
  The name of the zone being served. The name of a zone MUST be unique within a view, although different views may have zones with the same name.

- **master**

  The identifier of a Peer that can act as a master server for this zones. This can occur zero or more times.
o slave
The identifier of a Peer that can act as slave server for this
zones. This can occur zero or more times. Note: This says
nothing about whether or not those servers should be sent
notifies. Both master and slave can appear for a single zone.

o notify
The identifier of a Peer to which notifies should be sent to for
this zone. This can occur zero or more times.

2.2. NSCP Basic Control Capability

This capability provides the basic control functions for the
operation of the name server. It adds the following methods

2.2.1. Methods

- server-stop
  Stops the server software.

- server-reload
  Reloads the zone data

- server-restart
  Stops and the restarts the server software.

2.3. NSCP Start Control Capability

This capability provides an additional control functions. It has
been added as a separate capability because they will only be
possible if the name server is being managed by an external NSCP
process. It adds the following methods

2.3.1. Methods

- server-start
  Starts the server software.
3. Examples

This section gives some examples of NSCP requests.

3.1. Obtaining Configuration Information

To obtain configuration information from a remote nameserver via NSCP, a NETCONF <get-config> operation is used. <get-config> is used to retrieve all or part of a configuration. By default it uses subtree filtering to select the part of the configuration tree to return. XPath filtering can also be used if the :xpath capability is supported.

A <get-config> is required to specify a source configuration data store. By default NETCONF only specifies the "running" configuration. However, NSCP implementations are free to add additional data stores and advertise their-presence via implementation-specific capabilities (e.g. a "candidate" configuration, see [RFC4741] section 8.3).

The following example request uses <get-config> (in an implementation of NSCP with the :xpath capability) to select a Peer (named "master") from the configuration.

```xml
<rpc message-id="101">
  <get-config>
    <source>
      <running/>
    </source>
    <filter type="xpath" select="server/peers/peer[name='master']"/>
  </get-config>
</rpc>
```
... and an example of a possible reply is:

```
<rpc-reply message-id="101"
    xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <data>
        <peer xmlns="urn:ietf:nscp:1.0">
            <name>master</name>
            <address>
                <ip>192.0.2.0/24</ip>
                <port>53</port>
            </address>
            <address>
                <ip>2001:0DB8::08:EF</ip>
                <port>53</port>
            </address>
        </peer>
    </data>
</rpc-reply>
```

3.2. Modifying Configuration Information

Modification of configuration information is achieved via `<edit-config>`, another standard operation defined by NETCONF [RFC4741]. It takes an operation attribute that allows the specification of how the target should be modified; elements can be added or removed, or the contents of the `<edit-config>` message can be merged into the target.

The next example updates the NSCP configuration to enables recursion in a view called `my_view`. The "replace" attribute ensures that recursion is enabled, whatever the current state.

```
<edit-config message-id="102">
    <target>
        <running/>
    </target>
    <config xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
        <server xmlns="urn:ietf:nscp:1.0">
            <panorama>
                <view>
                    <name>myview</name>
                    <recursion xc:operation="replace">on</recursion>
                </view>
            </panorama>
        </server>
    </config>
</edit-config>
```
On success, the reply looks like:

```xml
<rpc-reply message-id="102"
   xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <ok/>
</rpc-reply>
```

(This is the form of reply expected when the server is merely acknowledging the completion of an NSCP command and not returning data. In the following examples, such replies are omitted.)

In the next example, the NSCP configuration is modified to add a new zone to a secondary. The master is configured to allow AXFR. To configure a master server one would also use a zone data capability to upload or point NSCP at the zone data.

```xml
<rpc message-id="103">
   <edit-config>
      <target>
         <running/>
      </target>
      <config xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
         <server xmlns="urn:ietf:nscp:1.0">
            <panorama>
               <view>
                  <name>_default</name>
                  <zone xc:operation="create">
                     <name>example.org</name>
                     <master>peer1</master>
                     <notify>peer4</notify>
                  </zone>
               </view>
            </panorama>
         </server>
      </config>
   </edit-config>
</rpc>
```
The next example is the same as the previous one (i.e. the addition of a zone), but illustrates a server that also supports the :rollback-on-error capability. By advertising this capability, the server guarantees that if the requested change fails for some reason, the existing configuration will be left unaltered; there will be no partial alteration of the configuration. The :rollback-on-error capability is one of several capabilities defined in [RFC4741]

```
<rpc message-id="104">
  <edit-config>
    <target>
      <running/>
    </target>
    <error-option>rollback-on-error</error-option>
    <config xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
      <server xmlns="urn:ietf:nscp:1.0">
        <panorama>
          <view>
            <name>_default</name>
            <zone xc:operation="create">
              <name>example.org</name>
              <master>peer1</master>
              <notify>peer4</notify>
            </zone>
          </view>
        </panorama>
      </server>
    </config>
  </edit-config>
</rpc>
```

3.3. Controlling the Name Server

The following example illustrates how to stop a server if it has the Basic Control capability

```
<rpc message-id="105">
  <server-stop/>
</rpc>
```

With the Start Control capability, it is possible to start a stopped server

```
<rpc message-id="106">
  <server-start/>
</rpc>
```
4. IANA Considerations

This memo includes no request to IANA.
5. Security Considerations

To be completed.
6. Normative References

[I-D.hardaker-dnsops-name-server-management-reqs]
Hardaker, W., "Requirements for Management of Name Servers for the DNS",

[I-D.ietf-netconf-partial-lock]
Lengyel, B. and M. Bjorklund, "Partial Lock RPC for NETCONF",
draft-ietf-netconf-partial-lock-03 (work in progress), August 2008.

[I-D.ietf-netmod-yang]
Bjorklund, M., "YANG - A data modeling language for NETCONF",
draft-ietf-netmod-yang-01 (work in progress), August 2008.


Appendix A. Meeting the Requirements

This section discusses how NSCP meets the requirement described in [I-D.hardaker-dnsops-name-server-management-reqs]. The requirements are listed in the same order as those described in that document.

A.1. Management Architecture Requirements

A.1.1. Expected Deployment Scenarios

Nothing in NSCP restricts the range of deployment scenarios. Indeed, the very nature of NETCONF lends itself to supporting different scenarios through the use of capabilities.

A.1.1.1. Zone Size Constraints

Nothing in NSCP restricts the size or number of zones served by any NSCP managed name server.

A.1.1.2. Name Server Discovery

This is not supported in the base NSCP capability. However, nothing prevents this being added later, either as some kind of capability or via a different protocol.

A.1.1.3. Configuration Data Volatility

Nothing in NSCP restricts the frequency of changes to the name server configuration.

A.1.1.4. Protocol Selection

It is anticipated that NSCP, along with extensions built using capabilities, can meet the needs of a full management protocol. There may however, be occasions where there is an existing protocol better suited for carrying a particular task. For example, loading zone contents via AXFR/IXFR.

A.1.1.5. Common Data Model

A common data model is a core part of NSCP and is described in detail in this document.

A.1.1.6. Operational Impact

It is not anticipated that NSCP will add significant overhead to any DNS service.
A.1.2. Name Server Types

NSCP, along with extensions built using capabilities will support all the name server types.

A.2. Management Operation Types

A.2.1. Control Requirements

A.2.1.1. Needed Control Operations

NSCP along with the two control capabilities described in this document can provide the minimum set of control operations. Additional operations can be added via new capabilities.

A.2.1.2. Asynchronous Status Notifications

[RFC5277] describes an asynchronous message notification delivery service for NETCONF. This is an optional capability built on the base NETCONF definition. It is expected that this will form part of NSCP and will be considered further in future versions of this draft.

A.2.2. Configuration requirements

A.2.2.1. Served Zone Modification

The base NSCP protocol will be able to add, modify and delete the configuration about served zones. The ability to configure a zone with data will be the subject of additional capabilities described in other drafts. It is anticipated that there may be a variety of zone data modification capabilities to reflect the variety of methods possible for sending zone data to a name server. These may include capabilities that define methods to obtain zone data using DDNS, AXFR, HTTP, SCP, or even over NETCONF itself.

A.2.2.2. Trust Anchor Management

The base NSCP protocol will be able to add, modify and delete trust anchors.

A.2.2.3. Security Expectations

The base NSCP capability will be able to configure validation policies.
A.2.2.4. TSIG Key Management

The base NSCP protocol is able to add, modify and delete peers, which can be identified by TSIG keys.

A.2.2.5. DNS Protocol Authorization Management

NSCP contains a sophisticated access control mechanism that will allow control of

- Access to operations on zone data.
- Access to zone data from certain sources.
- Access to specific DNS protocol services.

A.2.3. Monitoring Requirements

It remains to be decided how much monitoring will form part of the base NSCP capability. Some minimal health monitoring and statistics gathering are included in the base NSCP capability. Future capabilities are expected to allow more state to be monitored.

A.2.4. Alarm and Event Requirements

As mentioned before, [RFC5277] describes an asynchronous message notification delivery service for NETCONF. It is expected that this will form part of NSCP and will be considered further in future versions of this draft.

A.2.5. Security Requirements

A.2.5.1. Authentication

Authentication will be provided by the NETCONF transport layer.

A.2.5.2. Integrity Protection

Integrity Protection will be provided by the NETCONF transport layer.

A.2.5.3. Confidentiality

Confidentiality will be provided by the NETCONF transport layer.

A.2.5.4. Authorization

Authorization will be provided by NETCONF or a capability.
A.2.5.5. Solution Impacts on Security

It is believed that NSCP does minimize any security risks introduced to the name server system. The security is provided by the NETCONF transport layer and a variety of suitable transports are available including ssh.

A.2.6. Other Requirements

A.2.6.1. Extensibility

NSCP is flexible and be able to accommodate new future management operations.

A.2.6.1.1. Vendor Extensions

NSCP does allow vendors to extend the standardized management model with vendor-specific extensions.

A.2.6.1.2. Extension Identification

In NETCONF capabilities are advertised in messages exchanged during session establishment. If multiple capability are used then the each define their own xml namespace.

A.2.6.1.3. Namespace Collision Protection

Again, in NETCONF capabilities are advertised in messages exchanged during session establishment. If multiple capability are used then the each define their own xml namespace.
Appendix B. NSCP Capabilities

This section defines the NSCP capabilities using the template in Appendix C of [RFC4741]. NSCP is broken into several capabilities to allow for maximum flexibility.

B.1. The NSCP Base Capability

B.1.1. Capability Name

NSCP Base

B.1.2. Overview

Exchange of this capability at session set up time indicates that the client and server both understand the base NSCP data model described in Section 2.1.

B.1.3. Dependencies

None.

B.1.4. Capability Identifier

urn:ietf:dnsop:nscp:1.0

B.1.5. New Operations

None.

B.2. The NSCP Basic Control Capability

B.2.1. Capability Name

NSCP Basic Control

B.2.2. Overview

Exchange of this capability at session set up time indicates that the client and server both understand the basic NSCP control operations described in Section 2.2.

There is no reconfig operation. It is assumed that this will happen automatically whenever the configuration is updated or that a form of commit-confirmed capability such as the one in [RFC4741] (section 8.4) will be used.
B.2.3. Dependencies

NSCP Base

B.2.4. Capability Identifier

urn:ietf:dnsop:nscp-control:1.0

B.2.5. New Operations

Note: Startup of a name server is a separate capability. This is to account for name servers that have a built in NSCP agent.

<stop>
Signals the name server to cleanly shutdown.

<reload>
Signals the name server reload a specified zone or all zones.

<restart>
Signals the name server to re-start.

B.3. The NSCP Start Control Capability

B.3.1. Capability Name

NSCP Start Control

B.3.2. Overview

Exchange of this capability at session set up time indicates that the client and server both understand the Start NSCP control operation described in Section 2.3

B.3.3. Dependencies

NSCP Start Control

B.3.4. Capability Identifier

urn:ietf:dnsop:nscp-start-control:1.0

B.3.5. New Operations

<start>
Signals the name server to start up.
Appendix C.  YANG Data Model for Base NSCP capability

YANG Data Model for Base NSCP capability

module nscp {
  namespace "urn:ietf:dnsop:nscp:1.0";
  prefix nscp;

  import yang-types { prefix yang; }
  import inet-types { prefix inet; }
  import ieee-types { prefix ieee; }

  organization
    "IETF";
  description
    "Base data model for NSCP.";

  container server {
    description "root of configuration and data";
    container statistics {
      description "container to hold statistics elements";
      leaf rr {
        description "Count of responses received.";
        config false;
        type yang:counter64;
      }
      leaf rnxd {
        description "Count of NXDomain received";
        config false;
        type yang:counter64;
      }
      leaf rdup {
        description "Count of duplicate responses";
        config false;
        type yang:counter64;
      }
      leaf rfail {
        description "Count of SERVFAIL responses received";
        config false;
        type yang:counter64;
      }
      leaf rferr {
        description "Count of FORMERR responses received";
        config false;
        type yang:counter64;
      }
      leaf rerr {
        description "Count of other errors received";
      }
  }
}
leaf raxfr {
    description "Count of AXFR initiated";
    config false;
    type yang:counter64;
}

leaf rlame {
    description "Count of lame delegations received";
    config false;
    type yang:counter64;
}

leaf ssysq {
    description "Count of NS address fetches";
    config false;
    type yang:counter64;
}

leaf sans {
    description "Count of answers sent";
    config false;
    type yang:counter64;
}

leaf sfwdq {
    description "Count of queries sent";
    config false;
    type yang:counter64;
}

leaf sdupq {
    description "Count of queries retried";
    config false;
    type yang:counter64;
}

leaf rq {
    description "Count of requests received";
    config false;
    type yang:counter64;
}

leaf snxd {
    description "Count of NXDomain sent";
    config false;
    type yang:counter64;
}

leaf ruq {
    description "Count of non-recursive queries rejected";
    config false;
    type yang:counter64;
}
leaf rurq {
    description "Count of recursive queries rejected";
    config false;
    type yang:counter64;
}
leaf ruxfr {
    description "Count of XFR rejected";
    config false;
    type yang:counter64;
}
leaf ruudp {
    description "Count of updates rejected";
    config false;
    type yang:counter64;
}

container dnssec-policy {
    description "DNSSEC policy";
    leaf-list secure {
        description
        "List of domains that are expected to be secure";
        type inet:domain-name;
    }
    leaf-list nonsecure {
        description
        "List of domains that are expected to be insecure";
        type inet:domain-name;
    }
    leaf KASP-database {
        description
        "Path or URL (To be decided) to allow the policy
database to be located.";
        type string;
    }
}

container peers {
    description
    "Container for peer elements. May be used for partial
locking";
    list peer {
        key "name";
        leaf name {
            description
            "name - used to refer to this peer elsewhere.";
            type string;
        }
    }
}
list address {
  description "Address and port that will identify this peer";
  key "ip port";
  leaf ip {
    type inet:ip-prefix;
  }
  leaf port {
    type inet:port-number;
  }
}

container key {
  description "key that will be used to identify this peer. A choice is used to allow augmentation in the future";
  choice type {
    case a {
      leaf hmac-md5 {
        type string;
      }
    }
    case b {
      leaf hmac-sha1 {
        type string;
      }
    }
  }
}

container panorama {
  description "A collection of views."
  list view {
    key "name";
    leaf name {
      type string;
    }
  }
  list listen-on {
    description "Address and port that the nameserver will listen on.";
    key "ip port";
    leaf ip {
      type inet:ip-address;
    }
    leaf port {
      type inet:port-number;
    }
  }
}
leaf recursion {
    description "Is recursion enabled for this view?";
    type enumeration {
        enum on;
        enum off;
    }
}

leaf validation {
    description "Is validation enabled in this view?";
    type enumeration {
        enum on;
        enum off;
    }
}

list zone {
    description "a zone";
    key "zone-name";
    leaf zone-name {
        description "the name of the zone.";
        type string;
    }
    leaf-list master {
        description
        "A reference to a peer that will act as a master for this zone";
        type keyref {
            path "/server/peers/peer/name";
        }
    }
    leaf-list slave {
        description
        "A reference to a peer that will act as a slave for this zone";
        type keyref {
            path "/server/peers/peer/name";
        }
    }
    leaf-list notify {
        description
        "A reference to a peer that notifies should be sent to for this zone";
        type keyref {
            path "/server/peers/peer/name";
        }
    }
}
container acl {
    description "The acl for this view";
    list ace {
        description "An access control entry.";
        key "identifier";
        leaf identifier {
            description
            "The name of a peer that will get
            this access.";
            type keyref {
                path "/server/peers/peer/name";
            }
        }
    }
    leaf-list access {
        description
        "The type of access that this peer
        will receive.";
        type enumeration {
            enum none;
            enum notify;
            enum query;
            enum recursion;
            enum transfer;
            enum update;
        }
    }
}
 Appendix D. YANG Data Model for the NSCP Basic Control Capability

module nscp-basic-control {
  namespace "urn:ietf:dnsop:nscp-basic-control:1.0";
  prefix nscp-basic-control;

  organization "IETF";

  description "Adds basic control to NSCP."

  rpc server-stop {
    /* Will cause server to stop cleanly. No inputs */
    output {
      leaf status {
        /* Will return something like "server stopping" */
        type string;
      }
    }
  }

  rpc server-reload {
    /* Will cause server to reload the
     * running configuration. No inputs */
    output {
      leaf status {
        /* Will return something like "server reloading" */
        type string;
      }
    }
  }

  rpc server-restart {
    /* Will cause server to reload the
     * running configuration. No inputs */
    output {
      leaf status {
        /* Will return something like "server restarting" */
        type string;
      }
    }
  }
}
Appendix E. YANG Data Model for the NSCP Start Control Capability

```yang
module nscp-start-control {
    namespace "urn:ietf:dnsop:nscp-start-control:1.0";
    prefix nscp-start-control;

    import yang-types { prefix yang; }
    import inet-types { prefix inet; }
    import ieee-types { prefix ieee; }

    organization
        "IETF";

    description
        "Adds start control to the base data model for NSCP."
;
    rpc server-start {
        /* Will cause server to stop cleanly. No inputs */
        output {
            leaf status {
                /* Will return something like "server starting" */
                type string;
            }
        }
    }
}
```
Authors’ Addresses

John A Dickinson
6 Nelson Close
Wallingford, Oxfordshire  OX10 0LG
UK

Email: jad@jadickinson.co.uk

Stephen Morris
Nominet UK
Minerva House
Edmund Halley Road
Oxford Science Park
Oxford, Oxfordshire  OX4 4DQ
UK

Email: stephen.morris@nominet.org.uk

Roy Arends
Nominet UK
Minerva House
Edmund Halley Road
Oxford Science Park
Oxford, Oxfordshire  OX4 4DQ
UK

Email: roy.arends@nominet.org.uk
Full Copyright Statement

Copyright (C) The IETF Trust (2008).

This document is subject to the rights, licenses and restrictions contained in BCP 78, and except as set forth therein, the authors retain all their rights.

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY, THE IETF TRUST AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM 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.

Intellectual Property

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights 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; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat 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 IETF on-line IPR repository at http://www.ietf.org/ipr.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.