Network Configuration Extensions: Access Control Model

draft-bierman-ncx-acm-00

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 http://www.ietf.org/shadow.html.

This Internet-Draft will expire on February 6, 2008.

Copyright Notice

Copyright (C) The IETF Trust (2007).
Abstract

The standardization of network configuration interfaces for use with the NETCONF protocol requires a structured data modeling environment which promotes human usability and multi-vendor interoperability. The Network Configuration Extensions (NCX) are a set of specifications intended to address NETCONF data modeling issues. This document defines an Access Control Model (ACM) suitable for use with the NETCONF protocol, intended to promote a more consistent, manageable, and interoperable infra-structure for ongoing development of NETCONF-based management interfaces.

Table of Contents

1. Introduction ........................................... 5
   1.1. Terminology ....................................... 6
   1.1.1. Requirements Notation .......................... 6
   1.1.2. NETCONF Terms .................................. 6
   1.1.3. NCX-SMI Terms .................................. 6
   1.1.4. Terms .......................................... 7
   1.2. Problem Statement .................................. 7
   1.3. Solution Requirements .............................. 8
2. Model Overview ........................................ 10
   2.1. External Dependencies .............................. 10
   2.2. Message Processing Model .......................... 10
3. Model Components ...................................... 14
   3.1. Users ............................................. 14
   3.2. Groups .......................................... 14
   3.3. Administrative Roles .............................. 15
   3.4. Sessions ......................................... 16
   3.5. Security Levels .................................. 16
   3.6. Access Permissions ................................ 17
      3.6.1. RPC Method Invocation Access .................. 17
      3.6.2. Configuration Data Access ...................... 17
      3.6.3. RPC Response Access .......................... 19
      3.6.4. Notification Access ............................ 19
   3.7. Controls .......................................... 20
      3.7.1. Global Controls .............................. 20
      3.7.2. Administrative Role Controls .................. 20
      3.7.3. RPC Operation Access Controls ............... 20
      3.7.4. Configuration Data Access Controls .......... 21
      3.7.5. Notification Access Controls ................ 21
   3.8. Access Violation Events ........................... 21
4. Access Control Procedures ............................ 23
   4.1. Initial Operation ................................. 23
   4.2. Session Establishment .............................. 23
   4.3. access-denied Error Handling ...................... 24
4.3.1. access-denied Report for RPC Access ............... 24
4.3.2. access-denied Report for Data Access ............. 25
4.4. Incoming Message Validation .......................... 25
4.5. Configuration Database Access Validation ............ 27
4.6. Outgoing Message Validation ........................... 28
5. Data Model Definition .................................. 31
  5.1. High Level Procedures ............................. 31
  5.2. Using Access Control Rules .......................... 31
  5.3. Data Organization .................................. 32
  5.4. configCapabilities Object .......................... 33
    5.4.1. globalConfig Field ................................ 33
    5.4.2. groupConfig Field ................................ 34
    5.4.3. rpcAccessConfig Field ............................ 34
    5.4.4. rpcTypeAccessConfig Field ......................... 34
    5.4.5. rpcAccessConfig Field ............................ 34
    5.4.6. notificationAccessConfig Field ................... 34
    5.4.7. configCapabilities Example ....................... 34
  5.5. profile Object ...................................... 35
    5.5.1. accessControlMode Field .......................... 36
    5.5.2. accessDeniedEventEnabled Field ................... 36
    5.5.3. profile Example .................................. 36
  5.6. groups Object ....................................... 37
    5.6.1. name Key ......................................... 37
    5.6.2. role Field ....................................... 38
    5.6.3. users Object ..................................... 38
    5.6.4. hosts Object ..................................... 38
    5.6.5. groups Example .................................. 38
  5.7. rpcRules Object ..................................... 40
    5.7.1. ruleNumber Key .................................. 40
    5.7.2. ruleType Field .................................. 40
    5.7.3. groupList Field .................................. 41
    5.7.4. rpcTarget Object ................................ 41
      5.7.4.1. rpcTypeList Field ............................ 41
      5.7.4.2. rpcMethodList Object ......................... 41
      5.7.4.3. xpathExpr Field ................................ 41
      5.7.4.4. all Field .................................... 42
    5.7.5. rpcRules Example ................................ 42
  5.8. dataRules Object .................................... 44
    5.8.1. ruleNumber Key .................................. 44
    5.8.2. access Field ..................................... 44
    5.8.3. groupList Field .................................. 44
    5.8.4. dataTarget Object ................................ 44
      5.8.4.1. dataNodeList Object ........................... 45
      5.8.4.2. xpathExpr Field ................................ 45
      5.8.4.3. all Field .................................... 45
    5.8.5. dataRules Example ................................ 45
  5.9. notificationRules Object ............................. 48
    5.9.1. ruleNumber Key .................................. 48
5.9.2. ruleType Field ........................................... 48
5.9.3. groupList Field ........................................ 48
5.9.4. eventTarget Object ................................. 48
  5.9.4.1. eventNodeList Object ............................. 49
5.9.4.2. xpathExpr Field .................................... 49
  5.9.4.3. all Field ........................................... 49
5.9.5. notificationRules Example ............................. 49
5.10. accessDeniedEvent Notification .......................... 51
  5.10.1. eventClass Field ..................................... 52
5.10.2. eventSeverity Field ................................. 52
5.10.3. eventTime Field ....................................... 52
5.10.4. eventSessionId Field ................................. 52
5.10.5. eventSessionAddress Field ......................... 52
5.10.6. eventGroupName Field ................................. 52
5.10.7. eventAccessRequested Field ......................... 52
5.10.8. eventErrorPath Field .................................. 53
  5.10.9. accessDeniedEvent Example .......................... 53
6.  Data Model XSD ........................................... 54
7.  IANA Considerations ..................................... 67
8.  Security Considerations ................................. 68
9.  References ............................................... 70
  9.1.  Normative References .................................. 70
  9.2.  Informative References ............................... 70
Author’s Address ............................................. 71
Intellectual Property and Copyright Statements .............. 72
1. Introduction

The standardization of network configuration interfaces for use with the NETCONF [RFC4741] protocol requires a structured data modeling environment which promotes human usability, and multi-vendor interoperability. This document defines an access control model suitable for use with the NETCONF protocol.

The Network Configuration Protocol utilizes the Extensible Markup Language [W3C.REC-xml] for encoding protocol messages used to manage network device configuration databases. However, there is no specification for the inter-operable management of the controlled access to potions of a conceptual NETCONF configuration database. Access control is currently not part of the NETCONF standard, so a NETCONF session is granted access based entirely on the session characteristics in an undocumented proprietary manner, if any access control is available at all.

This document discusses the following concepts:

- Access Control Model
- Access Control Procedures
- Access Control Data Model

The NETCONF stack can be conceptually partitioned into four layers.

```
<table>
<thead>
<tr>
<th>Layer</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>(4) Content</td>
<td></td>
</tr>
<tr>
<td></td>
<td>+-------------------+ +-------------------+</td>
</tr>
<tr>
<td>(3) Operations</td>
<td></td>
</tr>
<tr>
<td></td>
<td>+-------------------+ +-------------------+</td>
</tr>
<tr>
<td>(2) RPC</td>
<td></td>
</tr>
<tr>
<td></td>
<td>+-------------------+ +-------------------+</td>
</tr>
<tr>
<td>(1) Transport</td>
<td></td>
</tr>
<tr>
<td></td>
<td>+-------------------+ +-------------------+</td>
</tr>
</tbody>
</table>
```
This document addresses NETCONF protocol access control for the RPC, Operations, and Content layers, as defined in [RFC4741], and [I-D.ietf-netconf-notification].

1.1. Terminology

1.1.1. 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].

1.1.2. NETCONF Terms

The following terms are defined in RFC 4741 and are not redefined here:

- agent
- client
- manager
- operation
- RPC
- RPC method
- server
- session
- user

1.1.3. NCX-SMI Terms

The following terms are defined in the NCX-SMI [I-D.bierman-ncx-smi] and are not redefined here:

- application
- data model
- object
1.1.4. Terms

The following terms are used throughout this documentation:

access control: A security feature provided by the NETCONF agent, which allows an operator to restrict access to a subset of all NETCONF protocol operations and configuration data, based on various criteria.

access control model (ACM): A conceptual model used to configure and monitor the access control procedures desired by the operator to enforce a particular access control policy.

access control rule: The conceptual criteria used to determine if a particular NETCONF protocol operation should be permitted or denied.

NCX-ACM: The access control model defined in this document.

administrative role: The type of administrative role for any session associated with a particular group. This is typically used within an ACM to implicitly enforce a particular set of access control rules, associated with the role.

root: The special administrative role, associated with the 'root' user identity for a NETCONF session, with unlimited access, and it is exempt from all access control enforcement. The special user name 'root' is used to identify the this entity on an NCX agent. Also called the 'super-user'.

1.2. Problem Statement

Network operators and network equipment (NE) software developers need a lot of expertise, and spend a lot of time, to configure, control, and monitor network devices.

Network configuration has traditionally been very proprietary in nature, especially security configuration. By design, the NETCONF protocol specification does not address access control behavior because it is considered to be part of the content layer, and out of
scope.

An access control model is needed that is simple to understand, implement, and deploy for the NETCONF protocol, to promote secure and interoperable usage of the NETCONF protocol.

The NETCONF protocol uses a procedural interface model, and set of standard protocol operations. There is no standard access control model defined for NETCONF, and a specialized model is needed, which fully secures all mechanisms within the protocol.

A standard access control model for the NETCONF protocol will allow a manager to configure and monitor the protocol and data access permissions for any NETCONF session, based on various criteria.

Suitable control, monitoring, and notification mechanisms are needed to allow an operator to easily manage all aspects of the ACM behavior. An XML data model, suitable for use with the <edit-config> operation must be available for this purpose.

1.3. Solution Requirements

There is a need for a comprehensive access control model for the NETCONF protocol which provides the following features:

- An extensible and interoperable conceptual model of the correct access control behavior of a NETCONF agent must be defined.
- A data model suitable for use with the <edit-config> and other standard protocol operations is needed to implement the ACM.
- For the mandatory transport (SSH), the agent must provide a user identification mechanism, utilizing the user identity provided by the SSH subsystem to the NETCONF agent.
- The data model should support the concept of administrative roles, to support the well-established distinction between a root account and other types of (less-privileged) user accounts.
- The data model must allow significant access control mechanisms which do not require any Xpath usage. For example, the ability to disable write access, or restrict certain RPC method access such as <edit-config>, must not rely on Xpath.
- Access control rules to restrict access to specific sub-trees within the configuration database should be supported. If so, an Xpath expression must be used to identify the sub-tree(s) for this purpose.
Access control XPath expressions for configuration data should be restricted to static terms only, such as object identifiers and instance identifiers. These identifiers are defined in the NCX-SMI [I-D.bierman-ncx-smi]. Other static conceptual objects, such as a device type, a physical port number, or a statically allocated MAC or IP address, should also be accepted by the agent. An agent may choose to reject access control rules which represent conceptual objects which are dynamic, such as counters, status objects, and user-supplied description strings.

This requirement is needed to reduce the threat of granting access unintentionally due to rapidly changing dynamic terms within the XPath expression, or usage of complex XPath expressions that grant more access than intended.
2. Model Overview

This section provides a high-level overview of the access control model structure. It describes the NETCONF protocol message processing model, and the conceptual access control requirements within that model.

2.1. External Dependencies

The NETCONF [RFC4741] protocol is used for all management purposes within this document. The agent must support the features identified by the ’NETCONF-base’ capability. It is expected that the mandatory transport mapping NETCONF Over SSH [RFC4742] is also supported by the agent, and that the agent has access to the user name associated with each session.

The XML Path Specification [W3C.xpath] is used within the NETCONF protocol to define filters for configuration data retrieval operations and filters for notification suppression. Support for XPath is optional, and only fully supported if the ’#xpath’ capability is advertised by the agent. Use of XPath within the data model defined in this document is also optional, based on the ’#xpath’ capability.

The NETCONF Notifications draft [I-D.ietf-netconf-notification] defines notification delivery mechanisms for the NETCONF protocol. An agent must advertise the ’#notification’ capability for notification delivery to be supported. A notification event type for an access violation event is defined in this document, which may be enabled by the administrator to generate a notification message when an access violation is detected by the agent.

The NCX-SMI [I-D.bierman-ncx-smi] specification is used for some data modeling concepts, such as the ’NCX Name’ data type, object identifier syntax, and instance identifier syntax.

2.2. Message Processing Model

The following diagram shows the NETCONF message flow model, including the points at which access control is applied, during NETCONF message processing.
The follow sequence of conceptual processing steps is executed for each received message:

- Access control is applied to all messages received by the agent, individually, for each active session, unless the administrative role for the session is ‘root’, or the <accessControlMode> field within the <profile> object is set to ‘off’.
A session is bound to an administrative role and a group identity for the lifetime of the session.

An incoming message is dispatched, based on its namespace and element name.

Incoming `<rpc>` requests are received by the RPC access control enforcer.

If the session is authorized to execute the RPC method, then processing continues, otherwise the following steps are taken by the agent:

1. If the agent is configured to generate the ‘accessDeniedEvent’ notification, then one is generated for this access violation event. The `<eventAccessRequested>` field in the notification is set to ‘rpcRequest’.

2. The RPC request is rejected with an ‘access-denied’ error.

If access is granted, the incoming RPC request is processed.

If the configuration database is accessed by the RPC method, then the configuration access control enforcer must authorize the requested operation.

If the requested access is some sort of read operation, and the `<accessControlMode>` field within the `<profile>` object is set to ‘loose’, then configuration data access control enforcement is skipped.

If the session is authorized to perform the requested operation on the requested data, then processing continues, otherwise the following steps are taken by the agent:

1. If the agent is configured to generate the ‘accessDeniedEvent’ notification, then one is generated for this access violation event. The `<eventAccessRequested>` field in the notification is set to ‘dataRead’ or ‘dataWrite’, depending on the nature of the requested configuration database operation.

2. The RPC request is rejected with an ‘access-denied’ error.

If the `<accessControlMode>` field within the `<profile>` object is set to ‘loose’, then RPC reply access control enforcement is skipped. Otherwise, the RPC reply access control enforcer checks the `<data>` element (if it exists) within the conceptual reply. It will then prune any data for which the session is not authorized.
to read. There is no additional configuration for this enforcement.

- The agent should verify that no restricted data (as configured for the configuration database) is contained in the RPC reply. This is especially important for non-standard RPC methods which may return data without properly accessing the configuration database.

- The <rpc-reply> is generated and sent to the manager.

The follow sequence of conceptual processing steps is executed for each generated event notification:

- Agent instrumentation generates a conceptual notification, for a particular session.

- If the <accessControlMode> field within the <profile> object is set to 'loose', then notification access control enforcement is skipped, otherwise, processing continues.

- The notification access control enforcer checks the notification, and if it contains any data for which the session is not authorized to read, then the notification with the access violation is dropped for that session only.

- If there is no access violation, then the <notification> element is generated and sent to the manager.
3. Model Components

This section defines the conceptual components related to access control model.

3.1. Users

A ‘user’ is the conceptual identity, which is associated with the access permissions granted to a particular session. A user is identified by an NCX Name string, defined in NCX-SMI, which must be unique within the agent.

The user name string is usually derived from the transport layer during session establishment. An agent is required to have an authenticated user name for a session before <rpc> requests will be accepted. Otherwise all requests must be rejected with an ‘access-denied’ error-tag value.

If groups are not supported by an agent (in the data model), then the user name will be used to automatically create a group with the same group name.

An agent may choose not to provide NETCONF-based configuration of the users permitted access to the system, via RPC method or configuration database access.

3.2. Groups

Access to a specific NETCONF protocol mechanism is granted to a session, associated with a group, not a user.

A group is identified by its group name, which is an NCX Name string. All group names must be unique within the agent.

A group member can be identified by either a user name (NCX Name), or an Internet Address (InetAddress). An agent may choose not to support Internet Addresses as group members however.

An agent may choose not to fully support groups. At a minimum, all groups will be created by the agent, and there will be one group corresponding to each user, using the same name.

An agent may choose not to support user configuration of the groups used within the system. In this case, the groups are configured in some manner outside the scope of this document.
3.3. Administrative Roles

An administrative role identifies the type of network operations activity that is expected and permitted for a sessions associated with a particular group.

An administrative role is identified by a fixed-value string. There should be very few roles used within a system, in order to reduce complexity.

Each group definition within the agent must be associated with a single administrative role, which should not be changed while any sessions associated with that group are active.

The following administrative roles are defined within the access control model:

root: The ‘root’ role, associated with the NETCONF session, is given unlimited access permissions. It is exempt from all access control enforcement.

admin: The ‘admin’ role, associated with the NETCONF session, is given specific access permissions, as configured in the agent. It is subject to all access control enforcement.

guest: The optional ‘guest’ role, associated with the NETCONF session, is given limited access permissions, as configured in the agent. It is subject to all access control enforcement, and must not be given any write access to any configuration database.

The agent should support a ‘root’ administrative role, and the user name for this identity should be named ‘root’. This role (and user identity) should be built into the agent and not be configurable on the device.

The agent must support the ‘admin’ session role, besides the ‘root’ role, which is subject to all access control enforcement.

The agent may choose to support additional types administrative roles, besides the ‘admin’ role, which are subject to all access control enforcement.

The agent may choose to support the ‘guest’ role, which is subject to all access control enforcement, and never allowed to write to any configuration data.
3.4. Sessions

A session is simply a NETCONF session, which is the entity which is granted access to specific NETCONF protocol mechanisms.

A session is always associated with a specific administrative role, for the lifetime of the session.

A session is associated with a single group name for the lifetime of the session. (The user name may be the group name in some cases). It will be used during all access control rule evaluation procedures, performed by the agent.

3.5. Security Levels

There are four security levels built into the access control model, which affect the way the agent will behave. These security levels will override individual access control rules configured on the system. They are used to assist in access control deployment and debugging, and provide a simple mechanism to configure common security profiles with a single data model object.

An agent should support the following security levels:

off: All access control is disabled.

warn: All access control is checked, but instead of enforcing an access control rule, the agent will generate a log entry and/or an event notification, and allow the requested operation to proceed.

loose: During configuration database access control enforcement, only write access is checked.

strict: During configuration database access control enforcement, all access is checked.

An agent implementer must choose a default security level that is in effect when the device is shipped. It is suggested that the ‘loose’ or ‘strict’ security level be selected as the default.

An agent should require that the ‘root’ session role be used in order to modify the global security level.

Once access control configuration has been tested and fully functional, an administrator should change the security level to ‘strict’, and operate in this mode during normal network operations.
3.6. Access Permissions

The access permissions are the NETCONF protocol specific set of permissions that have been assigned to a particular session role or group. The same access permissions should stay in affect for the lifetime of a session.

The access control model treats RPC method execution separately from configuration database access and outgoing messages:

1. Permission to invoke the <rpc> request.
2. Permission to access the configuration database.
3. Permission to send the <rpc-reply> response or <notification> message.

3.6.1. RPC Method Invocation Access

Permission to invoke an RPC method is treated separately from all other access mechanisms, within the access control model and the data model. Instead of multiple types of access, an access control rule for controlling RPC method access uses a simple boolean, determining whether permission to invoke the RPC method is granted or not.

3.6.2. Configuration Data Access

There are three access permissions associated with configuration database data structures that are defined. These coincide with the SMIv2 [RFC2578] MAX-ACCESS clause values in name and concept. They also support the ‘operation’ attribute used with the <edit-config> protocol operation, which makes a clear distinction between create and merge operations.

The following fixed access permissions apply only to configuration data. These permissions are associated statically via the data model definition (e.g., MAX-ACCESS clause) or dynamically via an access control configuration data model.

none:
   * No access is permitted at all.
read-only:
   * Configuration database retrieval operations are permitted.
* Notification generation is permitted.

read-write:

* Configuration database edit operations are permitted.
* Configuration database retrieval operations are permitted.
* Notification generation is permitted.

read-create:

* Configuration database create operations are permitted.
* Configuration database delete operations are permitted.
* Configuration database edit operations are permitted.
* Configuration database retrieval operations are permitted.
* Notification generation is permitted.

These permissions need to be explicitly mapped to the NETCONF operation attribute, used in the <edit-config> protocol operation. The ‘read-only’ access permission does not apply to the ‘operation’ attribute. If a manager has read-only access to some node <a> in the configuration database, it is possible that it may have higher access for a child node of <a>. This is typically the case if a user-configurable optional object is contained within a fixed node (e.g., <interface> within the <interfaces> container).

The ‘read-write’ permission applies to the following <edit-config> operation, and indicates access permission in the following situations:

- merge with existing node
- replace an existing node

The ‘read-write’ permission can only be granted for writable objects which cannot be created by the manager, either directly, or as a side effect of the protocol operation. In other words, all the nodes (from root to the target nodes) must already exist in the agent for write access to be granted.

If an object has a agent-supplied default value, the object is considered to be present, even if the <get-config> operation does not indicate the object is present, due to suppression of default values.
In this case, write permission is granted as if the object was present with the agent supplied default value.

The ‘read-create’ permission applies to all possible <edit-config> operation attribute situations:

- merge
- replace
- create
- delete

This write permission can only be granted for writable objects which can be created and deleted by the manager, via direct action within a protocol operation.

3.6.3. RPC Response Access

A session must have read access to all contents of the <data> element, if it is present. The agent must prune any data which the session is not authorized to receive.

If some object within the data corresponds to an object in the configuration database, then the agent should use the access control configuration for that object to determine if the session should be granted access. If the data does not correspond to objects within the configuration database, then the agent should use mechanisms (outside the scope of this document) to determine if read access should be granted.

3.6.4. Notification Access

A session must have read access to all contents of the <notification> element, if any data is present. The agent must drop the notification (for that session only) if it contains any data which the session is not authorized to receive.

If an access control rules exist, which pertain to the namespace URI and/or notification event type element, then access must denied if the rules indicate the group is not authorized to receive that notification event type. If the session is authorized to receive the event type, it must also be authorized to receive any data within the notification as well.

If some object within the data corresponds to an object in the configuration database, then the agent should use the access control
configuration for that object to determine if the session should be granted access. If the data does not correspond to objects within the configuration database, then the agent should use mechanisms (outside the scope of this document) to determine if access should be granted.

3.7. Controls

3.7.1. Global Controls

An agent may choose to allow a manager to configure the global security level used to control the overall operation of the access control system. If so, then the ‘root’ administrative role should be required to modify this parameter.

If notifications are supported then a manager should be able to configure whether or not the ‘accessDeniedEvent’ notification should be generated. If so, then the ‘root’ administrative role should be required to modify this parameter.

An agent may choose to allow a manager to configure the groups used to control which user names or InetAddresses (associated with each session) should be given access to the system. If so, then the ‘root’ administrative role should be required to modify this configuration data.

3.7.2. Administrative Role Controls

The administrative role assigned to a group can be used to control course-grained access to configuration data, e.g., no restrictions for ‘root’, or no write access for ‘guest’.

The administrative role is associated with a group within the data model in Section 5.

3.7.3. RPC Operation Access Controls

A manager can specify the scope of the access control rule for an RPC method in four ways:

1. for all RPC methods defined with the specified RPC method type(s), as defined in NCX-SMI (e.g., ‘config’, ‘monitor’, ‘exec’, ‘debug’).

2. for all RPC methods defined within a specific namespace.

3. for one RPC method defined within a specific namespace.
4. for all RPC methods indicated by a particular boolean Xpath expression.

An agent may choose not to support the RPC method type form of access control rule definition.

An agent must support the Xpath expression form of access control rule if it advertises the ‘#xpath’ capability.

3.7.4. Configuration Data Access Controls

A manager can specify the scope of the access control rule for a portion of the configuration database in three ways:

1. for all configuration data within a specific namespace
2. for all configuration data within a specific XML sub-tree
3. for all configuration indicated by a particular node set Xpath expression.

An agent must support the Xpath expression form of access control rule if it advertises the ‘#xpath’ capability.

3.7.5. Notification Access Controls

A manager can specify the scope of the access control rule for a notification in three ways:

1. for all notification event types within a specific namespace
2. for a specific event types within a specific namespace
3. for all notification content indicated by a particular boolean Xpath expression.

The ‘event type’ corresponds to the one and only child node of the <notification> element. The namespace URI and element name for this node are used to define the notification type.

An agent must support the Xpath expression form of access control rule if it advertises the ‘#xpath’ capability.

3.8. Access Violation Events

If the agent supports the ‘#notification’ capability, then it should support generation of the ‘accessDeniedEvent’ notification type, defined within the data model.
The 'accessDeniedEvent' event type is controlled by a global parameter, used to enable or disable notification generation for this event type.

If generation of the <accessDeniedEvent> notification is enabled, and the global security level is not 'off', then one <accessDeniedEvent> notification is generated for each received message that contains any access violations. Logging and delivery of this notification are outside the scope of this document.
4. Access Control Procedures

NCX access control procedures (within the NETCONF agent) are specific to the NETCONF message processing model, and are based on the specific vulnerabilities that can occur during each type of message processing procedure.

All aspects of data access via NETCONF protocol behavior, including side effects from <rpc> operations, and external protocols (such as SNMP or SSH/CLI), are considered within the responsibility of the NETCONF agent to secure against unauthorized access.

There are six separate phases that must be addressed, three of which are related to the NETCONF message processing model. In addition, the initial start-up mode for an NCX agent, session establishment, and ‘access-denied’ error handling must also be considered.

4.1. Initial Operation

Upon the very first start-up of the NCX agent, the access control configuration will probably not be present. If not, an agent should not allow any write access to any session role except ‘root’ in this state.

If the configured (or default) access control profile is ‘strict’, then the agent must not allow any write access to any session role except ‘root’ in this state.

If the configured (or default) access control profile is ‘strict’, then the agent must not allow any read access to any session role except ‘root’ in this state.

The agent should prompt or alert the operator somehow until the access control configuration has been set. This initial configuration mode continues until the agent deems that the access control configuration has been set.

4.2. Session Establishment

The access control model applies specifically to the well-formed XML content transferred between a manager and an agent, after session establishment has been completed, and after the <hello> exchange has been successfully completed.

An agent must not include any sensitive information in any <capability> elements within the <hello> exchange.

Once session establishment is completed, and a user identity has been
authenticated, a NETCONF agent will enforce the access control rules, based on the supplied user identity and the configuration data stored on the agent.

4.3. access-denied Error Handling

The 'access-denied' error report is generated when the access control system denies access to either a request to invoke an RPC method or a request to perform a particular operation on the configuration database.

An agent must not include any sensitive information in any <error-info> elements within the <rpc-error> response.

If the <accessControlMode> field within the <profile> object is equal to 'warn', then the agent may choose to generate ‘access-denied’ warning reports, in addition to generating an ‘accessDeniedEvent’ notification (if supported). In this case, the <error-severity> field is set to 'warning' instead of 'error'.

4.3.1. access-denied Report for RPC Access

If permission to invoke an RPC method is not granted, the following <rpc-error> element is sent within the RPC response.

The following example shows an 'access-denied' error report that might be returned if access to the <edit-config> operation is denied.

```xml
<rpc-reply message-id="107"
 xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <rpc-error>
   <error-type>rpc</error-type>
   <error-tag>access-denied</error-tag>
   <error-severity>error</error-severity>
   <error-path>/rpc/edit-config</error-path>
   <error-message>
     Access to RPC method denied
   </error-message>
 </rpc-error>
</rpc-reply>
```

Figure 3
4.3.2. access-denied Report for Data Access

If permission to invoke the requested protocol operation (e.g., read, write, create/delete) is not granted for the specified configuration data, the following <rpc-error> element is sent within the RPC response.

The following example shows an ‘access-denied’ error report that might be returned if some access to the interface named ‘eth1’ is denied.

```xml
<rpc-reply message-id="108"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <error-type>application</error-type>
  <error-tag>access-denied</error-tag>
  <error-severity>error</error-severity>
  <error-path
    xmlns:ex="http://example.com/interfaces/1.0">
    /rpc/edit-config/config/ex:interfaces/ex:interface[ex:name='eth1']
  </error-path>
  <error-message>
    Access to configuration data denied
  </error-message>
</rpc-error>
```

Figure 4

4.4. Incoming Message Validation

The diagram below shows the basic conceptual structure of the access control processing model for incoming NETCONF messages, within an agent.
Access control begins with the message dispatcher. Only well-formed XML messages should be processed by the access control system.

An agent should not allow access to configuration databases through any top level element except the <rpc> element in the NETCONF-base namespace. If it does, (e.g., <my-rpc>) the agent should still enforce access control to configuration data, even if the access is from some mechanism outside the standard protocol operations.

After the agent validates the <rpc> element, and determines the namespace URI and the element name of the RPC method being requested, the RPC access control enforcer verifies that the session is authorized to invoke the RPC method.
If the session is not authorized to invoke the RPC method with the
given set of parameters, then an <rpc-error> is generated with the
following information:

error-tag: access-denied

error-path: /rpc/method-QName, where ‘method-QName’ is a qualified
name identifying the actual RPC method name. For example, ‘/rpc/
edit-config’ represents the <edit-config> operation in the NETCONF
base namespace..

If the configuration database is accessed, either directly or as a
side effect of the RPC operation, then the configuration database
access enforcer must intercept the operation and make sure the
session is authorized to perform the requested operation on the
specified data.

If an ‘access-denied’ error is generated at this point, then no
information related to any parameters within the <rpc> request should
be returned in the <rpc-error> response. A <data> element must not
be included in the response. Only the name of the RPC method should
be returned in <error-path> element or the <error-message> element.
The <error-app-tag> should not reveal any additional information
about the access violation.

4.5. Configuration Database Access Validation

An agent must ensure that all access to the data within the
configuration database is authorized, no matter which RPC method,
top-level construct other than <rpc>, or even other protocol than
NETCONF, is used to access the database.

For access control purposes, there is only one conceptual
configuration database. The same access control rules are used for
all configuration database instances, such as the <candidate> or
<running> configurations.

Access to configuration data is controlled by assigning a access
control rule to a specific sub-tree in the database, for a conceptual
object, using the { group-name, max-access, object-identifier }
values for the sub-tree to be restricted.

Access to configuration data can also be controlled by assigning a
access control rule to a specific namespace URI in the database, for
all conceptual objects defined within the namespace, using the { group-name, max-access, namespace-URI } values for the namespace to
be restricted.
If the ‘#xpath’ capability is advertised by the agent, then an Xpath expression can be used to specify the configuration data sub-tree(s) associates with an access control rule.

Figure 6

4.6. Outgoing Message Validation

There are two outbound message types to consider: the <rpc-reply> and the <notification> messages.

The <rpc-reply> message should be checked by the agent to make sure no unauthorized data is contained within it. If so, the restricted data must be removed from the message before it is sent to the manager.

Configuration of access control rules exclusively for <rpc-reply>
messages is outside the scope of this document. It is suggested that the <dataRules> is used for this purpose.

The <notification> message should be checked by the agent to make sure no unauthorized data is contained within it. If so, the entire notification is dropped for that session. The <notificationRules> object is used to configure access control rules for notification messages.

The following figure shows the conceptual message processing model for outgoing messages.
Figure 7
5. Data Model Definition

This section defines the semantics of the conceptual data structures found in the data model in Section 6.

5.1. High Level Procedures

There are some high level management procedures that an administrator needs to consider before using this access control model:

1. Set the global security level.
2. Enable or disable the <accessDeniedEvent> notification.
3. Configure (or pre-configure) the <groups> object.
4. Configure zero or more access control rules for RPC method invocation control.
5. Configure zero or more access control rules for configuration database access.
6. Configure zero or more access control rules for notification event type access.

Notification delivery must be properly configured (outside the scope of this document) in order for the <accessDeniedEvent> notification to actually be generated.

5.2. Using Access Control Rules

There are some important considerations for proper configuration of the access control rules used within the data model.

- The start state for allowing access is considered to be undefined.
- The agent determines the correct set of access control rules to use, based on the stage of protocol message processing. For example, if is processing an RPC request, then the <rpcRules> object is used.
- Access control rules are processed by the agent in ascending order, starting with the lowest numbered rule (i.e., lowest value of the ruleNumber key).
- The agent determines the target of each rule. If the target defined within the rule includes the target being checked for access permissions, then the rule is considered a match, and the
specific access is either permitted or denied based on that rule.

- The special target rule called `<all/>` is used to match anything. It is customary to use this type of rule as the last rule within a sequence of access control rules. For example, in order to deny all access except a few RPC methods, an operator might set up an `<rpcRules>` object with a few rules to permit invocation of certain RPC methods, and the very last rule would be a ‘catchall’ rule, which denies access to all RPC methods.

- If no access rule is found associated with the requested operation, then the agent should consider the access to be denied. However, if the global security level is ‘loose’, the agent may choose to permit the requested operation instead.

### 5.3. Data Organization

The top-level element is called `<security>`, and it is defined the ‘NCX-security’ namespace. The second level container element is named `<accessControl>` and it is also defined in the NCX-security namespace.

There are six groups of parameters defined as child nodes of the `<accessControl>` element. Each group is defined in detail within this section. The data model also contains a notification event definition for reporting an access control violation.

- `configCapabilities`: Indicates access control configuration capabilities of the agent (read-only)
- `profile`: Configures global access control parameters.
- `groups`: Configures the groups used within the access control system.
- `rpcRules`: Configures the access control rules for RPC method invocation.
- `dataRules`: Configures the access control rules for configuration database access.
- `notificationRules`: Configures the access control rules for controlling delivery of `<notification>` messages.

The following figure shows the XML layering used within the data model.
5.4. configCapabilities Object

The <configCapabilities> is a read-only element, which contains some boolean fields indicating which configuration objects the agent will allow user configuration via the NETCONF protocol.

Agent implementation of this entire object is mandatory.

5.4.1. globalConfig Field

Boolean indicating whether the agent allows user configuration of the parameters within the <profile> object.
5.4.2. groupConfig Field

Boolean indicating whether the agent allows user configuration of the parameters within the <groups> object.

5.4.3. rpcAccessConfig Field

Boolean indicating whether the agent allows user configuration of access control rules within the <rpcRules> object.

5.4.4. rpcTypeAccessConfig Field

Boolean indicating whether the agent allows user configuration of access control rules bases on the conceptual type of an RPC method (as defined in NCX-SMI). The ’rpcAccessConfig’ field must be ’true’ if this field is ’true’.

5.4.5. rpcAccessConfig Field

Boolean indicating whether the agent allows user configuration of access control rules within the <dataRules> object.

5.4.6. notificationAccessConfig Field

Boolean indicating whether the agent allows user configuration of access control rules within the <notificationRules> object.

5.4.7. configCapabilities Example

The following example shows a <get> operation and response for an example <configCapabilities> object.
<rpc message-id="101"
    xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <get>
        <filter type="subtree">
            <security xmlns="http://netconfcentral.org/ncx/security">
                <accessControl>
                    <configCapabilities/>
                </accessControl>
            </security>
        </filter>
    </get>
</rpc>

<rpc-reply message-id="101"
    xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <data>
        <security xmlns="http://netconfcentral.org/ncx/security">
            <accessControl>
                <configCapabilities>
                    <globalConfig>true</globalConfig>
                    <groupConfig>true</groupConfig>
                    <rpcAccessConfig>true</rpcAccessConfig>
                    <rpcTypeAccessConfig>false</rpcTypeAccessConfig>
                    <databaseAccessConfig>true</databaseAccessConfig>
                    <notificationAccessConfig>true</notificationAccessConfig>
                </configCapabilities>
            </accessControl>
        </security>
    </data>
</rpc-reply>

Figure 9

5.5. profile Object

The <profile> object is a container with some read-write elements, which are enumerated strings indicating the global access control behavior for the agent.

Agent implementation of this object is mandatory. Write access is required if the 'globalConfig' field in the <configCapabilities> object is set to 'true'. However, write access to the 'accessDeniedEventEnabled' field is not required if the '#notification' capability is not supported.
5.5.1. accessControlMode Field

Enumerated string indicating the access control security level that should be applied by the agent.

- **off**: All access control is disabled.
- **warn**: All access control is checked, but instead of enforcing an access control rule, the agent will generate a log entry and/or an event notification, but allow the requested operation to proceed.
- **loose**: During configuration database access control enforcement, only write access is checked.
- **strict**: During configuration database access control enforcement, all access is checked.

5.5.2. accessDeniedEventEnabled Field

Boolean indicating whether the agent should generate the <accessDeniedEvent> notification when access violations are detected.

5.5.3. profile Example

The following example shows a <get> operation and response for an example <profile> object.
<rpc message-id="102"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get>
    <filter type="subtree">
      <security xmlns="http://netconfcentral.org/ncx/security">
        <accessControl>
          <profile/>
        </accessControl>
      </security>
    </filter>
  </get>
</rpc>

<rpc-reply message-id="102"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <data>
    <security xmlns="http://netconfcentral.org/ncx/security">
      <accessControl>
        <profile>
          <accessControlMode>strict</accessControlMode>
          <accessDeniedEventEnabled>true</accessDeniedEventEnabled>
        </profile>
      </accessControl>
    </security>
  </data>
</rpc-reply>

Figure 10

5.6. groups Object

The <groups> object is a container with one or more instances of the <group> object within it. Each <group> element instance is indexed by the value of its ‘name’ field.

Agent implementation of this object is mandatory. Write access is required if the ‘groupConfig’ field in the <configCapabilities> object is set to ‘true’.

5.6.1. name Key

NCX Name string identifying the name of the group. This value must be unique for all instances of the <group> element.
5.6.2. role Field

String defining the administrative role assigned to this group. This field should not be changed once the group is created.

5.6.3. users Object

Table of user name strings which are considered to be part of the group. This nested object is optional to use. If not used, then the <hosts> object may be used instead.

This object is a container, which contains indexed and ordered entries. It should contain one or more <user> elements. Each one contains an NCX Name string, which must be unique for all instances of the <user> element.

5.6.4. hosts Object

Table of InetAddress strings which represent management station addressed, which are considered to be part of the group. This nested object is optional to use. If not used, then the <users> object may be used instead.

This object is a container, which contains indexed and ordered entries. It should contain one or more <host> elements. Each one contains an InetAddress string, which must be unique for all instances of the <host> element.

5.6.5. groups Example

The following example shows a <get> operation and response for an example <groups> object.

```
<rpc message-id="103"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get>
    <filter type="subtree">
      <security xmlns="http://netconfcentral.org/ncx/security">
        <accessControl>
          <groups/>
        </accessControl>
      </security>
    </filter>
  </get>
</rpc>
```
<rpc-reply message-id="103"
    xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <data>
    <security xmlns="http://netconfcentral.org/ncx/security">
      <accessControl>
        <groups>
          <group>
            <name>superuser</name>
            <role>root</role>
            <users>
              <user>root</user>
            </users>
          </group>
          <group>
            <name>routers</name>
            <role>admin</role>
            <users>
              <user>fred</user>
            </users>
          </group>
          <group>
            <name>support</name>
            <role>admin</role>
            <users>
              <user>fred</user>
              <user>barney</user>
              <user>wilma</user>
            </users>
            <hosts>
              <host>192.168.1.1</host>
            </hosts>
          </group>
          <group>
            <name>interns</name>
            <role>guest</role>
            <users>
              <user>bam-bam</user>
              <user>pebbles</user>
            </users>
          </group>
        </groups>
      </accessControl>
    </security>
  </data>
</rpc-reply>
In this example, four groups are defined:

- A group called ‘superuser’, containing the user ‘root’, using the ‘root’ administrative role. No access rules are needed for this group, since access control is not applied to the ‘root’ administrative role.

- A group called ‘routers’, containing the user ‘fred’, using the ‘admin’ administrative role. This group is allowed full access to protocol features, depending on the access control rules configured for the system.

- A group called ‘support’, containing three users and one IPv4 address, using the ‘admin’ administrative role.

- A group called ‘interns’, containing two users, using the ‘guest’ administrative role. This group does not have any write permissions to any configuration data, regardless of the access control rules configured for the system.

5.7. rpcRules Object

The <rpcRules> object is a container with one or more instances of the <rpcRule> object within it. Each <rpcRule> element instance is indexed by the value of its ‘ruleNumber’ field. If the ‘rpcAccessConfig’ field in the <configCapabilities> object is ‘true’, then the agent supports user creation, modification, and deletion of this object.

Agent implementation of this object is mandatory. Write access is required if the ‘rpcAccessConfig’ field in the <configCapabilities> object is set to ‘true’.

5.7.1. ruleNumber Key

A non-negative integer identifying the relative order of this entry among all other instances of the <rpcRule> element. This value must be unique for all instances of the <rpcRule> element. The agent will evaluate RPC method access rules in ascending order. The ‘ruleNumber’ values do not have to be contiguous.

5.7.2. ruleType Field

Enumerated string defining the type of rule match for this access control rule:
permit: Rule match indicates that access permission is granted.

deny: Rule match indicates that access permission is denied.

5.7.3. groupList Field

List of one or more group names for which this rule applies.

5.7.4. rpcTarget Object

Represents the RPC method targets that are identified by this access control rule. Contains a choice between one <rpcTypeList> object, one <rpcMethodList> object, one <xpathExpr> object, or one <all/> element.

5.7.4.1. rpcTypeList Field

If the ‘rpcTypeAccessConfig’ field in the <configCapabilities> object, then the agent supports this field. This field represents a list of RPC method types, which are used to match against the type of RPC method being invoked. If the type of the requested RPC method is in the list, then the access control rule is considered a match.

5.7.4.2. rpcMethodList Object

This field identifies the RPC methods that are considered part of this access control rule.

This nested object contains two fields:

namespaceUri: The namespace URI associated with the RPC method(s) identified by this access control rule.

elementNames: List of element names of the RPC methods within the indicated namespace that are identified by this access control rule. If this field is absent, then all RPC methods defined in the indicated namespace are identified by this access control rule.

5.7.4.3. xpathExpr Field

This field contains a boolean XPath expression which the agent will use to identify the requested RPC method or parameter data within the RPC method, to match this access control rule. An agent which advertises the ‘#xpath’ capability must support this object.
5.7.4.4. all Field

Empty element indicating that the access control rule applies to all RPC methods in all namespaces.

5.7.5. rpcRules Example

The following example shows a <get> operation and response for an example <rpcRules> object.

```xml
<rpc message-id="104"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <get>
   <filter type="subtree">
     <security xmlns="http://netconfcentral.org/ncx/security">
       <accessControl>
         <rpcRules/>
       </accessControl>
     </security>
   </filter>
 </get>
</rpc>

<rpc-reply message-id="104"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <data>
   <security xmlns="http://netconfcentral.org/ncx/security">
     <accessControl>
       <rpcRules>
         <rpcRule>
           <ruleNumber>1</ruleNumber>
           <ruleType>permit</ruleType>
           <groupList>routers</groupList>
           <rpcTarget>
             <rpcMethodList>
               <namespaceUri>
                 urn:ietf:params:xml:ns:netconf:base:1.0
               </namespaceUri>
               <elementNames>
                 edit-config copy-config delete-config commit discard-changes
               </elementNames>
             </rpcMethodList>
           </rpcTarget>
         </rpcRule>
       </rpcRules>
     </accessControl>
   </security>
 </data>
</rpc-reply>
```
<rpcRule>
  <ruleNumber>4</ruleNumber>
  <ruleType>permit</ruleType>
  <groupList>routers support</groupList>
</rpcTarget>
  <rpcMethodList>
    <namespaceUri>
      urn:ietf:params:xml:ns:netconf:base:1.0
    </namespaceUri>
    <elementNames>
      get-config get validate
    </elementNames>
  </rpcMethodList>
</rpcRule>

<rpcRule>
  <ruleNumber>6</ruleNumber>
  <ruleType>permit</ruleType>
  <groupList>interns</groupList>
</rpcTarget>
  <rpcMethodList>
    <namespaceUri>
      urn:ietf:params:xml:ns:netconf:base:1.0
    </namespaceUri>
    <elementNames>
      get-config get
    </elementNames>
  </rpcMethodList>
</rpcRule>

<rpcRule>
  <ruleNumber>99</ruleNumber>
  <ruleType>deny</ruleType>
  <groupList>support interns</groupList>
</rpcTarget>
</rpcRule>
</rpcRules>
</accessControl>
</security>
</data>
</rpc-reply>

Figure 12
Rule ‘1’ allows the ‘routers’ group permission to invoke the 
<edit-config>, <copy-config>, <delete-config>, <commit>, and 
<discard-changes> protocol operations.

Rule ‘4’ allows the ‘routers’ and ‘support’ groups permission to 
invoke the <get-config>, <get>, and <validate> protocol 
operations.

Rule ‘6’ allows the ‘interns’ group permission to invoke the <get-
config> and <get> protocol operations.

Rule ‘99’ denies access to the ‘support’ and ‘interns’ groups from 
invoking any RPC method (not explicitly mentioned in a previous 
rule).

5.8. dataRules Object

The <dataRules> object is a container with one or more instances of 
the <dataRule> object within it. Each <dataRule> element instance is 
indexed by the value of its ‘ruleNumber’ field.

Agent implementation of this object is mandatory. If the 
‘databaseAccessConfig’ field in the <configCapabilities> object is 
‘true’, then the agent supports user creation, modification, and 
deletion of this object.

5.8.1. ruleNumber Key

A non-negative integer identifying the relative order of this entry 
among all other instances of the <dataRule> element. This value must 
be unique for all instances of the <dataRule> element. The agent 
will evaluate configuration data access rules in ascending order. 
The ‘ruleNumber’ values do not have to be contiguous.

5.8.2. access Field

The level of access permissions associated with this access control 
rule. This object is encodes as an enumerated string. The values 
are defined in Section 3.6.2.

5.8.3. groupList Field

List of one or more group names for which this rule applies.

5.8.4. dataTarget Object

Represents the objects that are identified by this access control 
rule. Contains a choice between one <dataNodeList> object, one
<xpathExpr> object, and one <all> object.

5.8.4.1. dataNodeList Object

Contains a list of elements which are used to match against the configuration data elements within the agent, or data within an RPC response or notification message. If the element is identified within the node list, then the access control rule is considered a match.

This nested object contains two fields:

namespaceUri: The namespace URI associated with the data node(s) identified by this access control rule.

elementNames: List of element names of the data nodes within the indicated namespace that are identified by this access control rule. If this field is absent, then all data nodes defined in the indicated namespace are identified by this access control rule.

5.8.4.2. xpathExpr Field

This field contains a node set XPath expression which the agent will use to identify the data nodes which are considered to match this access control rule. An agent which advertises the '#xpath' capability must support this object.

5.8.4.3. all Field

Empty element indicating that the access control rule applies to all configuration data in all namespaces.

5.8.5. dataRules Example

The following example shows a <get> operation and response for an example <dataRules> object.

```
<rpc message-id="105"
    xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get>
    <filter type="subtree">
      <security xmlns="http://netconfcentral.org/ncx/security">
        <accessControl>
          <dataRules/>
          <all/>
        </accessControl>
      </security>
    </filter>
  </get>
</rpc>
```
<security>
</filter>
</get>
</rpc>

<rpc-reply message-id="105"
   xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <data>
      <security xmlns="http://netconfcentral.org/ncx/security">
         <accessControl>
            <dataRules>
               <dataRule>
                  <ruleNumber>1</ruleNumber>
                  <access>read-create</access>
                  <groupList>routers</groupList>
                  <dataTarget>
                     <dataNodeList>
                        <namespaceUri>
                           http://netconfcentral.org/ncx/security
                        </namespaceUri>
                        <dataNodeList>
                        </dataTarget>
                      </dataRule>
                      <dataRule>
                         <ruleNumber>3</ruleNumber>
                         <access>read-create</access>
                         <groupList>routers</groupList>
                         <dataTarget>
                            <xpathExpr xmlns:ex="http://example.com/router/config">
                                /ex:routers/ex:config
                            </xpathExpr>
                        </dataTarget>
                      </dataRule>
                      <dataRule>
                         <ruleNumber>4</ruleNumber>
                         <access>read-only</access>
                         <groupList>support</groupList>
                         <dataTarget>
                            <dataNodeList>
                               <namespaceUri>
                                  http://netconfcentral.org/ncx/security
                               </namespaceUri>
                               <dataNodeList>
                            </dataTarget>
                         </dataRule>
                      </dataRule>
                      <dataRule>
                         <ruleNumber>7</ruleNumber>
                      </dataRule>
                  </dataRules>
               </accessControl>
            </dataRules>
         </security>
      </data>
   </rpc-reply>
<access=read-write><groupList=support><dataTarget><xpathExpr xmlns:ex="http://example.com/interfaces/1.0">
  /ex:interfaces/ex:interface[ex:name='eth0']
</xpathExpr>
</dataTarget>
</dataRule>
<dataRule>
  <ruleNumber>10</ruleNumber>
  <access>none</access>
  <groupList=interns><dataTarget>
    <dataRule>
      <ruleNumber>99</ruleNumber>
      <access=read-only><groupList=routers support interns><dataTarget>
        <all/>
      </dataTarget>
      </dataRule>
    </dataRule>
  </groupList>
</dataRules>
</accessControl>
</security>
</data>
</rpc-reply>

Figure 13

- Rule '1' allows the 'routers' group permission to invoke any operation on all data defined within the 'NCX security' namespace.
- Rule '3' allows the 'routers' group permission to invoke any operation on the 'config' element within the 'example.com' router data model.
- Rule '4' allows the 'support' group permission to read any data defined in the 'NCX security' namespace.
o Rule ’7’ allows the ’support’ group permission to read and write the interface named ’eth0’, (in the example.com Interfaces data model), but not create or delete operations.

o Rule ’10’ denies the ’interns’ group permission to read or write any data defined in the ’NCX security’ namespace.

o Rule ’99’ permits read access to all the groups to any data not mentioned in a previous rule.

5.9. notificationRules Object

The <notificationRules> object is a container with one or more instances of the <notificationRule> object within it. Each <notificationRule> element instance is indexed by the value of its ’ruleNumber’ field.

Implementation of this object is mandatory. If the ’notificationAccessConfig’ field in the <configCapabilities> object is ’true’, then the agent supports user creation, modification, and deletion of this object.

5.9.1. ruleNumber Key

A non-negative integer identifying the relative order of this entry among all other instances of the <notificationRule> element. This value must be unique for all instances of the <notificationRule> element. The agent will evaluate notification data access rules in ascending order. The ’ruleNumber’ values do not have to be contiguous.

5.9.2. ruleType Field

Enumerated string defining the type of rule match for this access control rule:

permit: Rule match indicates that access is granted.

deny: Rule match indicates that access is denied.

5.9.3. groupList Field

List of one or more group names for which this rule applies.

5.9.4. eventTarget Object

Represents the notification event types that are identified by this access control rule. Contains a choice between one <eventNodeList>
object, one <xpathExpr> object, and one <all/> object.

Agent implementation of this object is mandatory. Write access is required if the ‘rpcAccessConfig’ field in the <configCapabilities> object is set to ‘true’.

5.9.4.1.  eventNodeList Object

Contains a list of elements which are used to match against the configuration data elements within the agent, or data within an RPC response or notification message. If the element is identified within the node list, then the access control rule is considered a match.

This nested object contains two fields:

namespaceUri: The namespace URI associated with the notification data node(s) identified by this access control rule.

elementNames: List of element names of the notification data nodes within the indicated namespace that are identified by this access control rule. If this field is absent, then all notification data nodes defined in the indicated namespace are identified by this access control rule.

5.9.4.2.  xpathExpr Field

This field identifies a boolean Xpath expression which the agent will use to identify the notification data nodes which are considered to match this access control rule. An agent which advertises the ‘#xpath’ capability must support this object.

5.9.4.3.  all Field

Empty element indicating that the access control rule applies to all notification messages in all namespaces.

5.9.5.  notificationRules Example

The following example shows a <get> operation and response for an example <notificationRules> object.

```xml
<rpc message-id="106"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get>
```

Bierman                 Expires February 6, 2008               
[Page 49]
<filter type="subtree">
  <security xmlns="http://netconfcentral.org/ncx/security">
    <accessControl>
      <notificationRules/>
    </accessControl>
  </security>
</filter>

<rpc-reply message-id="105"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <data>
    <security xmlns="http://netconfcentral.org/ncx/security">
      <accessControl>
        <notificationRules>
          <notificationRule>
            <ruleNumber>1</ruleNumber>
            <ruleType>permit</ruleType>
            <groupList>routers support</groupList>
            <eventTarget>
              <all/>
            </eventTarget>
          </notificationRule>
          <notificationRule>
            <ruleNumber>2</ruleNumber>
            <ruleType>permit</ruleType>
            <groupList>interns</groupList>
            <eventTarget>
              <eventNameList>
                <namespaceUri>
                  urn:ietf:params:xml:ns:netconf:base:1.0
                </namespaceUri>
              </eventNameList>
            </eventTarget>
          </notificationRule>
          <notificationRule>
            <ruleNumber>3</ruleNumber>
            <ruleType>deny</ruleType>
            <groupList>interns</groupList>
            <eventTarget>
              <all/>
            </eventTarget>
          </notificationRule>
        </notificationRules>
      </accessControl>
    </security>
  </data>
</rpc-reply>
Figure 14

- Rule '1' permits the 'routers' and 'support' groups to receive all notification messages in all namespaces.
- Rule '2' permits the 'interns' group to receive all notification messages in the 'NETCONF base' namespace.
- Rule '3' prevents the 'interns' group from receiving any other notification messages in any other namespace.

5.10. accessDeniedEvent Notification

The access denied event is generated once for each received message which would have caused one or more potential access violations.

If the '#notification' capability is supported by the NETCONF agent, and the 'accessDeniedEventEnabled' field within the <profile> object is set to 'true', then the agent will generate notification messages for the <accessDeniedEvent> event type, as needed.

The event type namespace is the NCX-ACM data model target namespace. The event type element name is 'accessDeniedEvent'.

This notification message contains eight data elements:

- eventClass
- eventSeverity
- eventTime
- eventSessionId
- eventSessionAddress
- eventGroupName
- eventAccessRequested
- eventErrorPath
5.10.1.  eventClass Field

The ‘eventClass’ field will probably be defined as a standard outside the scope of this document. This field is static, and must contain the string ‘security’.

5.10.2.  eventSeverity Field

The ‘eventSeverity’ field will be defined as a standard outside the scope of this document. This field is static, and should contain the string ‘major’.

5.10.3.  eventTime Field

The ‘eventTime’ field contains the agent system time when the access violation occurred. This field contains a string in ‘dateTime’ format.

5.10.4.  eventSessionId Field

The ‘eventSessionId’ field contains the session ID number for the session which had the access violation. This field contains a positive integer string.

5.10.5.  eventSessionAddress Field

The ‘eventSessionAddress’ field contains the Internet address for the session which had the access violation. This field contains an InetAddress string.

5.10.6.  eventGroupName Field

The ‘eventGroupName’ field identifies the group name associated with the session This field contains a string in NCX Name format.

5.10.7.  eventAccessRequested Field

The ‘eventAccessRequested’ field contains a string identifying the type of operation that was requested, which was denied access. This field contains an enumerated string with one of the following values:

rpcRequest:  An RPC method invocation request was denied.
dataRead:  A configuration data read request was denied.
dataWrite: A configuration data write request was denied.

5.10.8. eventErrorPath Field

The ‘eventErrorPath’ object contains an absolute Xpath expression identifying the RPC method, configuration data, RPC response field, or notification field that caused the access violation. This field contains a string conforming to the Xpath syntax.

5.10.9. accessDeniedEvent Example

The following example shows a <notification> message for an example <accessDeniedEvent> event.

```xml
<nc:notification
 xmlns:nc="urn:ietf:params:xml:ns:netconf:notification:1.0">
<accessDeniedEvent
 xmlns="http://netconfcentral.org/ncx/security">
<eventClass>security</eventClass>
<eventSeverity>major</eventSeverity>
<eventTime>2007-08-06T13:20:00.000-08:00</eventTime>
<eventSessionId>42</eventSessionId>
<eventSessionAddress>192.168.1.12</eventSessionAddress>
<eventGroupName>interns</eventGroupName>
<eventAccessRequested>rpcRequest</eventAccessRequested>
<eventErrorPath>/nc:rpc/nc:edit-config</eventErrorPath>
</accessDeniedEvent>
</nc:notification>
```

Figure 15

This example shows an <accessDeniedEvent> notification for an attempt (by a session for a user in the 'interns' group) to invoke the <edit-config> operation.
6. Data Model XSD

This section contains an XML Schema Definition [W3C.REC-xmlschema-2] which defines the XML syntax associated with the conceptual data structure semantics found in Section 5.

BEGIN

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns="http://netconfcentral.org/ncx/security"
  targetNamespace="http://netconfcentral.org/ncx/security"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified"
  xml:lang="en" version="0.6"
  xmlns:ncx="http://netconfcentral.org/ncx/1.0"
  xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
  xmlns:n8="http://netconfcentral.org/ietf/types">
  <xs:annotation>
    <xs:documentation>
      Module: ncx-acm
      Owner: ncx
      Application: security
      Version: 0.6
      Contact Info: Send comments to ietf@andybierman.com.
      Description:
      NETCONF Access Control Model Parameters
    </xs:documentation>
  </xs:annotation>

  <!-- import standard XML definitions -->
    schemaLocation="http://www.w3.org/2001/xml.xsd"/>

  <!-- import inetAddress definition -->
  <xs:import namespace="http://netconfcentral.org/ietf/types"
    schemaLocation="http://netconfcentral.org/ietf/types
    http://www.netconfcentral.org/xsd/ietf/inetAddress.xsd"/>
</xs:schema>
```

---

Bierman                 Expires February 6, 2008               [Page 54]
<!--
Global Access Control Mode Type
-->
<xs:simpleType name="NcxAccessControlMode">
  <xs:annotation>
    <xs:documentation>
      NCX System access control mode.
      off == no access control checking enforced
      warn == warn only if access violation detected
      loose == any RPC method in the netconf namespace can
      be invoked; read-only data allowed for all
      strict == ncxacl RPC entry must be present to invoke
      an RPC method; ncxacl Data entry must be present
      to access any data. (Except for user == ‘root’.)
    </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string">
    <xs:enumeration value="off"/>
    <xs:enumeration value="warn"/>
    <xs:enumeration value="loose"/>
    <xs:enumeration value="strict"/>
  </xs:restriction>
</xs:simpleType>

<!--
Administrative Role Type
-->
<xs:simpleType name="NcxAccessAdminRole">
  <xs:annotation>
    <xs:documentation>
      Type of administrative role assigned to the session
      root == root account, no access control applied
      admin == administrative account, access control applied
      guest == guest account, read-only and configured
      access control applied.
    </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string">
    <xs:enumeration value="root"/>
    <xs:enumeration value="admin"/>
    <xs:enumeration value="guest"/>
  </xs:restriction>
</xs:simpleType>

<!--
Request Classification Type
-->
<xs:simpleType name="NcxAccessRequestType"/>
Type of request causing an access violation.

rpcRequest:
   An RPC method invocation request was denied.

dataRead:
   A configuration data read request was denied.

dataWrite:
   A configuration data write request was denied.

Name Type

NCX Access Name string.
(Should really be an extension of the 'NcxName' data type.)

List of Names Type

List of NcxAccessName strings.
<xs:simpleType>
<!--
RPC Classification Type
--> 
<xs:simpleType name="NcxAccessRpcType">
<xs:annotation>
<xs:documentation>
NCX RPC Type Classifications
</xs:documentation>
</xs:annotation>
<xs:restriction base="xs:string">
<xs:enumeration value="other"/>
<xs:enumeration value="config"/>
<xs:enumeration value="exec"/>
<xs:enumeration value="monitor"/>
<xs:enumeration value="debug"/>
</xs:restriction>
</xs:simpleType>

<!--
List of RPC Classifications Type
--> 
<xs:simpleType name="NcxAccessRpcTypeList">
<xs:annotation>
<xs:documentation>
List NCX RPC Type Classifications
</xs:documentation>
<xs:appinfo>
<ncx:no-duplicates/>
<ncx:merge-type>sort</ncx:merge-type>
</xs:appinfo>
</xs:annotation>
<xs:list itemType="NcxAccessRpcType"/>
</xs:simpleType>

<!--
Data Access Type
--> 
<xs:simpleType name="NcxAccessType">
<xs:annotation>
<xs:documentation>
NETCONF Data Access Types:
  none == no access allowed at all.
  This value is used in ACLs, not max-access clauses.
  read-only == read access allowed in all operations.
  This value implies notify access as well.
  read-write == all but create/delete allowed.
</xs:documentation>
</xs:annotation>
</xs:simpleType>
Values can be read and edited. Merge and replace operations will be permitted, even (e.g., create the instance if it does not exist. This allows a manager to save static or agent-created entries in copy-config operations, and then reload them later.

read-create == all access allowed

Instances can be read, created, edited, and deleted.

<!--
 Access Rule Type (not for data access)
 -->
<xs:simpleType name="NcxAccessRuleType">
  <xs:annotation>
    <xs:documentation>
      NCX Access Control Rule Type
    </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string">
    <xs:enumeration value="permit"/>
    <xs:enumeration value="deny"/>
  </xs:restriction>
</xs:simpleType>

<!--
 Access Node List Type
 -->
<xs:complexType name="NcxAccessNodeList">
  <xs:annotation>
    <xs:documentation>
      NCX Access Control Node Identifier Type
    </xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:element name="namespaceUri" type="xs:anyURI"/>
    <xs:element name="elementNames" minOccurs="0">
      <xs:simpleType>
        <xs:list itemType="NcxAccessName"/>
      </xs:simpleType>
    </xs:element>
  </xs:sequence>
</xs:complexType>
</xs:complexType>

<!--
accessDeniedEvent Notification Contents Type
-->
<xs:complexType name="NcxAccessDeniedEvent">
  <xs:annotation>
    <xs:documentation>
      Access Denied Event Notification Contents
    </xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:element name="eventClass" type="xs:string"/>
    <xs:element name="eventSeverity" type="xs:string"/>
    <xs:element name="eventTime" type="xs:dateTime"/>
    <xs:element name="eventSessionId" type="xs:unsignedInt"/>
    <xs:element name="eventSessionAddress" type="n8:InetAddress"/>
    <xs:element name="eventGroupName" type="NcxAccessName"/>
    <xs:element name="eventAccessRequested" type="NcxAccessRequestType"/>
    <xs:element name="eventErrorPath" type="xs:string"/>
  </xs:sequence>
</xs:complexType>

<!--
configCapabilities Type
-->
<xs:complexType name="NcxAccessConfigCapabilities">
  <xs:annotation>
    <xs:documentation>
      Read-only parameters indicating some agent access control
capabilities.
    </xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:element name="globalConfig" type="xs:boolean"/>
    <xs:element name="groupConfig" type="xs:boolean"/>
    <xs:element name="rpcAccessConfig" type="xs:boolean"/>
    <xs:element name="rpcTypeAccessConfig" type="xs:boolean"/>
    <xs:element name="databaseAccessConfig" type="xs:boolean"/>
    <xs:element name="notificationAccessConfig" type="xs:boolean"/>
  </xs:sequence>
</xs:complexType>

<!--
profile Type
-->

<xs:complexType name="NcxAccessGlobalProfile">
  <xs:annotation>
    <xs:documentation>
      Read-write parameters controlling some global agent access control behavior.
    </xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:element name="accessControlMode" type="NcxAccessControlMode"/>
    <xs:element name="accessDeniedEventEnabled" type="xs:boolean"/>
  </xs:sequence>
</xs:complexType>

<!--
  group Type
 -->
<xs:complexType name="NcxAccessGroupRec">
  <xs:annotation>
    <xs:documentation>
      One NCX Group Membership Record
      The 'role' parameter should not be changed after it is set.
      
      If a user name is present in the 'users' table then it is a member of the group identified by 'name'.
      
      If a host address is present in the 'hosts' table then it is a member of the group identified by 'name'.
      
      If both 'users' and 'hosts' are missing then the group will be considered empty.
    </xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:element name="name" type="NcxAccessName"/>
    <xs:element name="role" type="NcxAccessAdminRole"/>
    <xs:element name="users" minOccurs="0">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="user" type="NcxAccessName"/>
        </xs:sequence>
      </xs:complexType>
      <xs:key name="usersKey">
        <xs:selector xpath="user"/>
        <xs:field xpath="user"/>
      </xs:key>
    </xs:element>
    <xs:element name="hosts" minOccurs="0" minOccurs="0">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="user" type="NcxAccessName"/>
        </xs:sequence>
      </xs:complexType>
      <xs:key name="usersKey">
        <xs:selector xpath="user"/>
        <xs:field xpath="user"/>
      </xs:key>
    </xs:element>
  </xs:sequence>
</xs:complexType>
<xs:complexType>
  <xs:sequence>
    <xs:element name="host" type="n8:InetAddress" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="NcxAccessGroupTable">
  <xs:annotation>
    <xs:documentation>
      NCX Group Membership Table
      Users and hosts can belong to any number of groups.
      The group name must be unique.
    </xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:element name="group" type="NcxAccessGroupRec" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="NcxAccessRpcRule">
  <xs:annotation>
    <xs:documentation>
      One NCX RPC Method Access Control Rule
    </xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:element name="ruleNumber" type="xs:unsignedInt"/>
    <xs:element name="ruleType" type="NcxAccessRuleType"/>
    <xs:element name="groupList" type="NcxAccessNameList"/>
    <xs:element name="rpcTarget">
      <xs:complexType>
        <xs:choice>
          <xs:element name="rpcTypeList" type="NcxAccessRpcType"/>
        </xs:choice>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:complexType>
<xs:element name="rpcMethodList" type="NcxAccessNodeList"/>
<xs:element name="xpathExpr" type="xs:string"/>
<xs:element name="all"/>
</xs:choice>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>

<!--
rpcRules Type
-->
<xs:complexType name="NcxAccessRpcTable">
<xs:annotation>
<xs:documentation>
NCX Access Control Rules for RPC Methods.
</xs:documentation>
</xs:annotation>
<xs:sequence>
<xs:element name="rpcRule" type="NcxAccessRpcRule" minOccurs="0" maxOccurs="unbounded"/>
</xs:sequence>
</xs:complexType>

<!--
dataRule Type
-->
<xs:complexType name="NcxAccessDataRule">
<xs:annotation>
<xs:documentation>
One NCX Data Access Control Rule
</xs:documentation>
</xs:annotation>
<xs:sequence>
<xs:element name="ruleNumber" type="xs:unsignedInt"/>
<xs:element name="access" type="NcxAccessType" default="none"/>
<xs:element name="groupList" type="NcxAccessNameList"/>
<xs:element name="dataTarget" type="NcxAccessNodeList"/>
<xs:complexType>
<xs:choice>
<xs:element name="nodeNameList" type="NcxAccessNodeList"/>
<xs:element name="xpathExpr" type="xs:string"/>
<xs:element name="all"/>
</xs:choice>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
<!-- dataRules Type -->
<xs:complexType name="NcxAccessDataTable">
  <xs:annotation>
    <xs:documentation>
      NCX Access Control Rules for Configuration Data
    </xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:element name="dataRule" type="NcxAccessDataRule" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>

<!-- notificationRule Type -->
<xs:complexType name="NcxAccessNotificationRule">
  <xs:annotation>
    <xs:documentation>
      One NCX Notification Access Control Rule
    </xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:element name="ruleNumber" type="xs:unsignedInt"/>
    <xs:element name="ruleType" type="NcxAccessRuleType"/>
    <xs:element name="groupList" type="NcxAccessNameList"/>
    <xs:element name="eventTarget">
      <xs:complexType>
        <xs:choice>
          <xs:element name="eventNodeList" type="NcxAccessNodeList"/>
          <xs:element name="xpathExpr" type="xs:string"/>
          <xs:element name="all"/>
        </xs:choice>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:complexType>

<!-- notificationRules Type -->
<xs:complexType name="NcxAccessNotificationTable">
  <xs:annotation>
    <xs:documentation>
      NCX Access Control Rules for Notifications.
    </xs:documentation>
  </xs:annotation>
</xs:complexType>
<xs:complexType name="securityPSType"/>

<!-- blank type template for a security object -->
<xs:complexType name="securityPS" type="securityPSType" abstract="true"/>

<!-- abstract template for the security application contents -->
<xs:complexType name="securityAppType">
  <xs:sequence>
    <xs:element ref="securityPS" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>

<!-- accessControl Type -->
<xs:complexType name="accessControlPSType">
  <xs:annotation>
    <xs:documentation Parameters for NETCONF Access Control. </xs:documentation>
  </xs:annotation>
  <xs:complexContent>
    <xs:extension base="securityPSType">
      <xs:sequence>
        <xs:element name="configCapabilities" type="NcxAccessConfigCapabilities">
          <xs:annotation>
            <xs:documentation>NCX ACM configuration supported by the agent</xs:documentation>
            <xs:appinfo>
              <ncx:max-access>read-only</ncx:max-access>
            </xs:appinfo>
          </xs:annotation>
        </xs:element>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<xs:element name="profile" type="NcxAccessConfigCapabilities">
  <xs:annotation>
    <xs:documentation>
      NCX Global Access Control Profile Parameters
    </xs:documentation>
    <xs:appinfo>
      <ncx:max-access>read-only</ncx:max-access>
    </xs:appinfo>
  </xs:annotation>
</xs:element>

<xs:element name="groups" type="NcxAccessGroupTable">
  <xs:annotation>
    <xs:documentation>
      NCX Group Membership Table
    </xs:documentation>
    <xs:appinfo>
      <ncx:max-access>read-create</ncx:max-access>
    </xs:appinfo>
  </xs:annotation>
  <xs:key name="groupsKey">
    <xs:selector xpath="group"/>
    <xs:field xpath="group/name"/>
  </xs:key>
</xs:element>

<xs:element name="rpcRules" type="NcxAccessRpcTable">
  <xs:annotation>
    <xs:documentation>
      NCX RPC Execution Access Control Table
    </xs:documentation>
    <xs:appinfo>
      <ncx:max-access>read-create</ncx:max-access>
    </xs:appinfo>
  </xs:annotation>
  <xs:key name="rpcRulesKey">
    <xs:selector xpath="rpcRule"/>
    <xs:field xpath="rpcRule/ruleNumber"/>
  </xs:key>
</xs:element>

<xs:element name="dataRules" type="NcxAccessDataTable">
  <xs:annotation>
    <xs:documentation>
      NCX Data Model Access Control Table
    </xs:documentation>
    <xs:appinfo>
      <ncx:max-access>read-create</ncx:max-access>
    </xs:appinfo>
  </xs:annotation>
</xs:element>
<xs:element name="accessControl" type="accessControlPSType" substitutionGroup="securityPS"/>

<!-- element definition for the accessDeniedEvent notification not really accessible via the data model!!! Just contained in the ‘notification’ element!!! -->
<xs:element name="accessDeniedEvent" type="NcxAccessDeniedEvent"/>

</xs:schema>
7. IANA Considerations

There are two actions that are requested of IANA:

1. register data model schema namespace URI

2. register data model schema URL and store associated XSD

[namespace URI TBD]

[namespace URL TBD]
8. Security Considerations

This entire document discusses access control requirements and mechanisms for restricting NETCONF protocol behavior within a given session.

Configuration of the access control system is highly sensitive to system security. An agent may choose not to allow any user configuration to some portions of it, such as the global security level, or the groups which allowed access to system resources.

If the agent chooses to allow user configuration of the access control system, then only sessions using the ‘root’ administrative role should be allowed to have write access to the data model.

If the agent chooses to allow user retrieval of the access control system configuration, then only sessions using the ‘root’ administrative role should be allowed to have read access to the data model.

Xpath expressions can be used for configuration of most access control rules. Care must be taken by the administrator to understand the ramifications of all terms within the Xpath expression, and make sure that it represents the intended evaluation criteria.

There is a risk that invocation of non-standard RPC methods will have undocumented side effects. An administrator should construct access control rules such that the configuration database is protected from such side effects. Also, such RPC methods should never be invoked by a session using the ‘root’ administrative role.

There is a risk that attackers may exploit bugs or side effects related to Xpath expression evaluation, which open security holes (i.e., grant access to groups that are not intended to have access) on a periodic or predictable basis, allowing an attacker more access than intended for short periods of time.

There is a risk that non-standard RPC methods, or even the standard <get> operation, may return data which ‘aliases’ or ‘copies’ sensitive data from a different data object. In this case, the namespace and/or the element name will not match the values for the sensitive data, which is then fully or partially copied into a different namespace and/or element. An administrator should avoid using data models which use this practice.

An administrator should restrict write access to all configurable objects within this data model. It is suggested that only sessions using the ‘root’ administrative role be permitted to configure the
data model defined in this document.

If write access is allowed for configuration of access control rules, then care must be taken not to disrupt the access control security, such as deleting an access control rule which denies access to ‘all’. It is suggested that the entire set of rules (e.g., `<dataRules>` object, be replaced in one operation. For example, the `<dataRules>` object could be replaced by using the `<edit-config>` operation with a ‘replace’ operation attribute in the `<dataRules>` element.

An administrator should restrict read access to the following objects within this data model, which reveal access control configuration: which could be considered sensitive.

- groups
- rpcRules
- dataRules
- notificationRules
9. References

9.1. Normative References


9.2. Informative References

Author’s Address

Andy Bierman
netconfcentral.org
Simi Valley, CA
USA

Email: ietf@andybierman.com