NETCONF Data Modeling with YANG

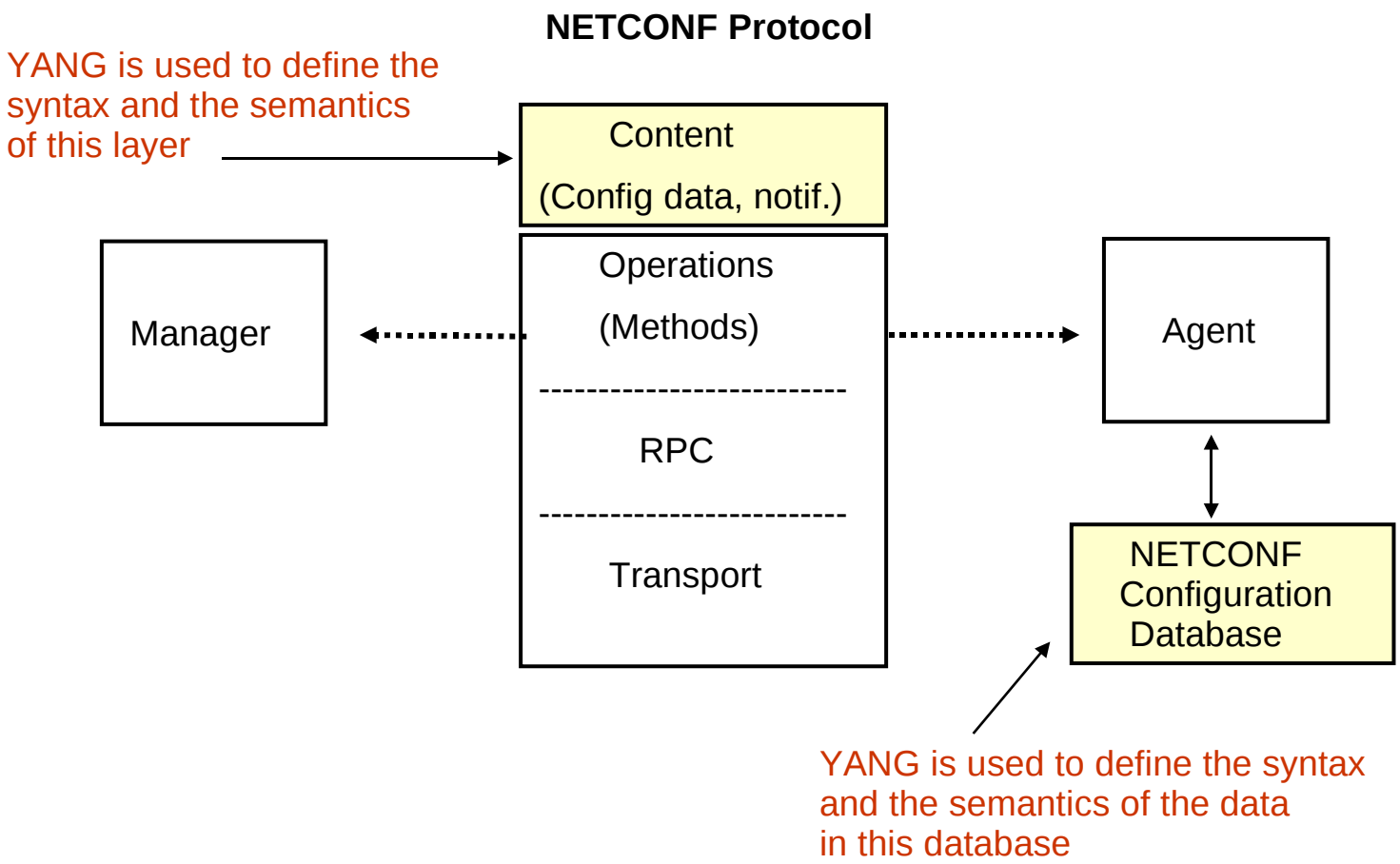
Part 1: Getting Started

Andy Bierman
draft: March 19, 2009

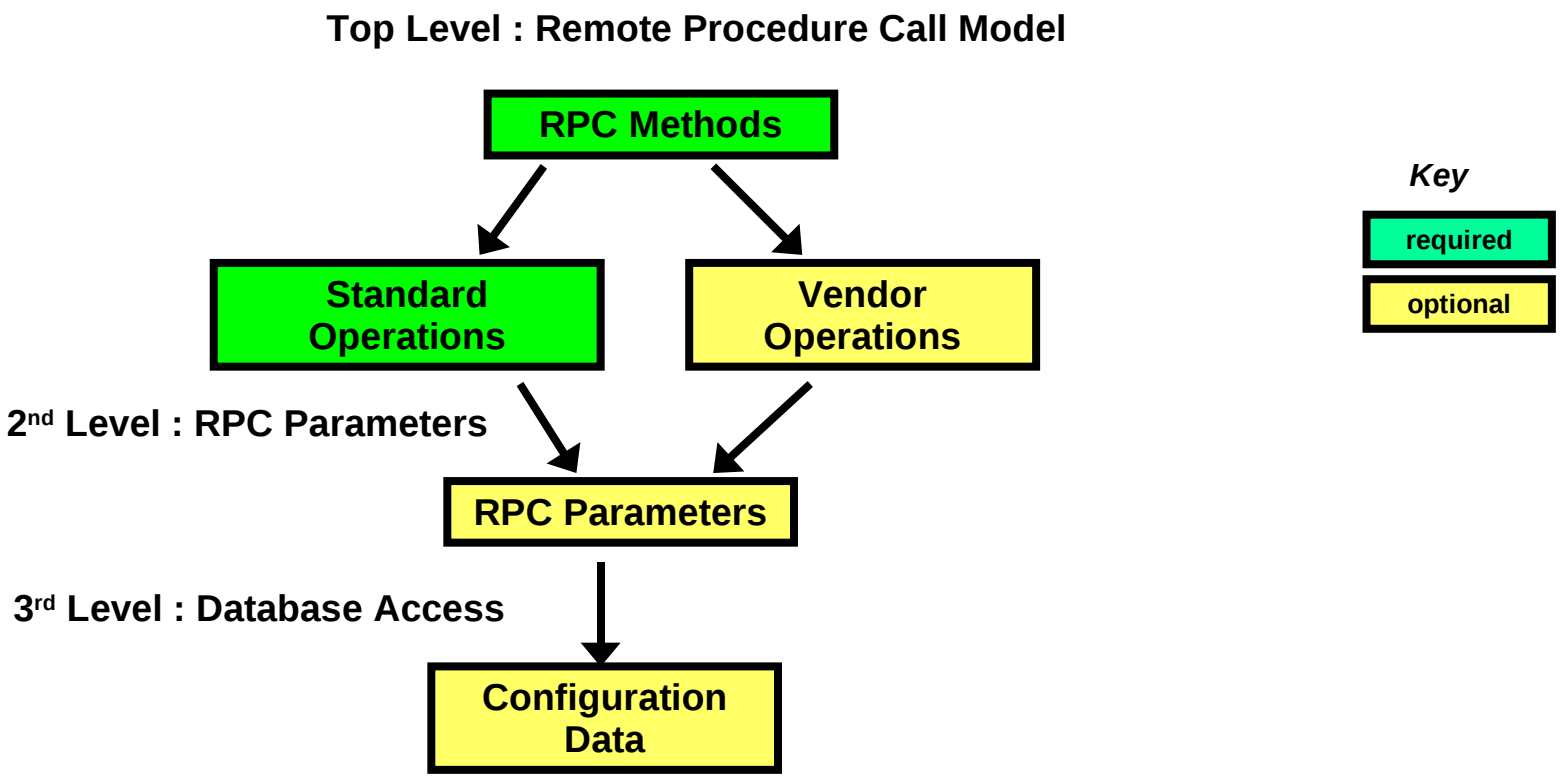
Contents

- NETCONF Content Summary
- Data Module Organization
- Module Structure
- Built-in Types
- Typedefs
- Leafs
- Leaf-lists
- Containers
- Lists
- Choices

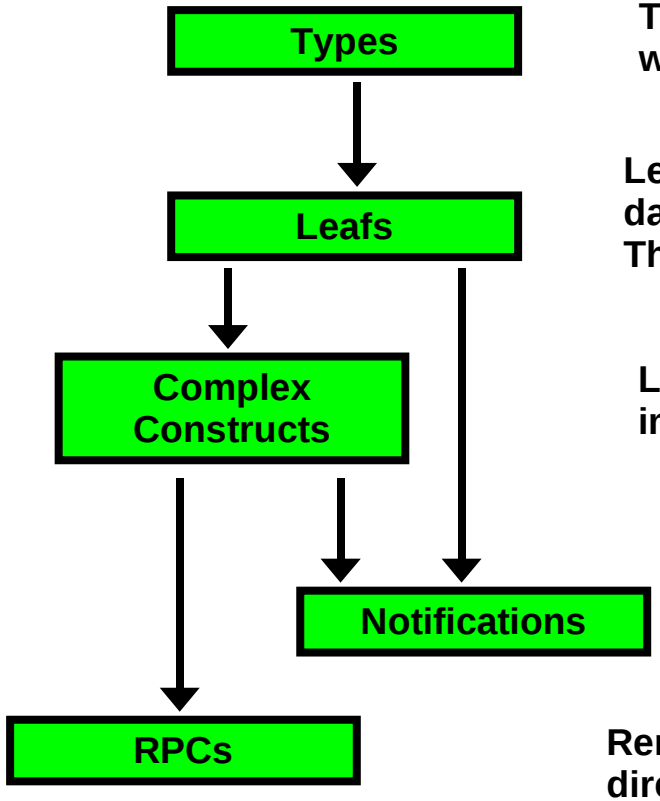
NETCONF Functional Layers



NETCONF Concepts (Top-down)



NETCONF Concepts (Bottom-up)



Types are abstractions (like TCs in SMI), which represent templates for leafs.

Leafs represent the smallest unit of conceptual data that the NETCONF protocol can manipulate. This corresponds to an XML simple type.

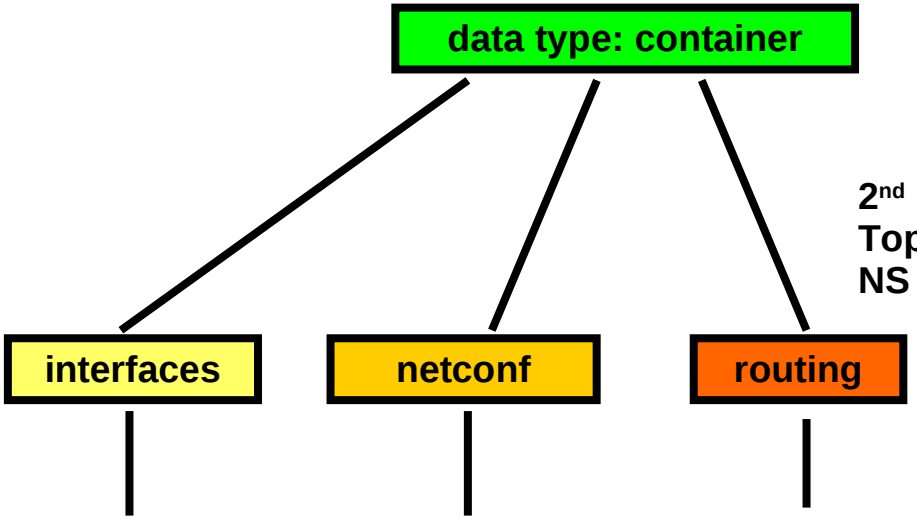
Leaf-lists, containers, lists, and choices provide important data manipulation capabilities

Notifications can contain data of any type

Remote procedure calls are used for all direct manager access to NETCONF databases

NETCONF XML Content Model

Top Level : conceptual <config> element, NETCONF namespace



2nd Level :
Top level objects, each in its own NS

3rd Level : Child nodes (if any)

- **Data models will be developed and maintained in modules**
- **Each module defines its own namespace for all typedefs, data objects, RPCs, and notifications**

Basic YANG Module Structure

bold = mandatory
plain = optional
black = keywords
blue = your text

```
module interfaces {
    yang-version 1;
    namespace "some-unique-URI";
    prefix if;

    import foo-module { prefix foo; }
    include bar-submodule;

    organization "your organization/company name here";
    contact "your name and email address here";
    description   "Module summary here";
    reference  "Module references here";

    revision 2008-03-05 { description "initial version."; }

    <data model definitions here, in any order>
}
```

YANG Definitions

The basic building blocks for YANG data models

Definition	Description
typedef	Refined type definition: Parent can be built-in or another refined type.
grouping	Reusable set of objects, paired with ‘uses’
object	Protocol accessible data: container, leaf, leaf-list, list, choice, etc.
rpc	Remote Procedure Call Method definition
notification	NETCONF Notification content definition

YANG Numeric Data Types

Numeric content for leaf and leaf-list objects

YANG Type	bits	sign	XSD Type	SMI Type
int8	8	Y	byte	Integer32
uint8	8	N	unsignedByte	Unsigned32
int16	16	Y	short	Integer32
uint16	16	N	unsignedShort	Unsigned32
int32	32	Y	int	Integer32
uint32	32	N	unsignedInt	Unsigned32
int64	64	Y	long	N/A
uint64	64	N	unsignedLong	Counter64
float32	32	Y	float	N/A
float64	64	Y	double	N/A

YANG String Types

String content for leaf and leaf-list objects

YANG Type	XSD Type	SMI Type
string	string	OCTET STRING
binary	base64	OCTET STRING
enumeration	string	INTEGER
bits	list	BITS
instance-identifier	string	OCTET STRING

YANG Special Types

XML content for leaf and leaf-list objects

YANG Type	XSD Type	SMI Type
empty	no type	N/A
boolean	boolean	Integer32
leafref	string	foreign leaf
union	union	N/A
identityref	string	IDENTITY

YANG String Encoding

- Strings can be specified in several ways:
  - » **Double Quoted String**: whitespace allowed, but it is adjusted (escape character sequences replaces, whitespace trimmed)
  - » **Single Quoted String**: whitespace allowed, and is preserved
    (this form is safest for pattern strings)
  - » **Unquoted String**: no whitespace allowed
    (Language tokens like ‘{‘ and ‘}’ are not allowed.)
- Strings can be specified in fragments if desired:
  - » “foobarbaz” is the same as “foo” + “bar” + “baz”
  - » Mixing forms is allowed: “foo” + bar + ‘baz’
- For comparison purposes, the quotes are ignored
  - » “foo” is equivalent to foo or ‘foo’
  - » “foo bar “ is not the same as ‘foo bar ‘ (extra space)

Identifiers in YANG Modules

Built-in definitions

type int32;

Built-in keywords and type names are not imported.
No prefix is allowed when they are used.

From the current module

type fooType; OR type foo:fooType

Local definitions such as type names and groupings are not imported.
The current module prefix is allowed, but not required.

From a different submodule

include footypes;
type fooType; OR type foo:fooType

An include statement must be present.
The current module prefix is allowed, but not required.

From a different module

import bartypes { prefix bar; }
type bar:barType;

An import statement must be present, with a unique prefix.
The declared module prefix must be present when identifiers are used.

Interface Table Example

```
<rpc message-id="101">
  <edit-config>
    <target><running/></target>
    <default-operation>none</default-operation>
     <config>
       <interfaces>
          <interface>
             <name>eth0</name>
             <mtu operation="replace">1500</mtu>
          </interface>
       </interfaces>
    </config>
  </edit-config>
</rpc>
```

rpc method: edit-config

input parameters from edit-config rpc

```
container interfaces {
  list interface {
     key  name;
     leaf name { type string; }
     leaf mtu { type uint32; }
  }
}
```

Set the MTU for Interface 'eth0'

YANG Types vs. SMI TCs

YANG Typedef

- Just data abstractions
- Can be derived from other types
- Simple types can be restricted
- Can be imported and shared without regard for namespace
- Default value can be defined
  - » Override any previous default
- Units clause can be defined
- YANG extensions could be used to declare a display-hint

SMIv2 TEXTUAL-CONVENTION

- Just data abstractions
- Cannot be derived from other TCs
- Simple types can be restricted
- Can be imported and shared
- Default value cannot be defined
  - » DEFVAL must be in the OBJECT-TYPE macro instead
- UNITS can be defined
- DISPLAY-HINT can be defined

YANG Instance Identifiers

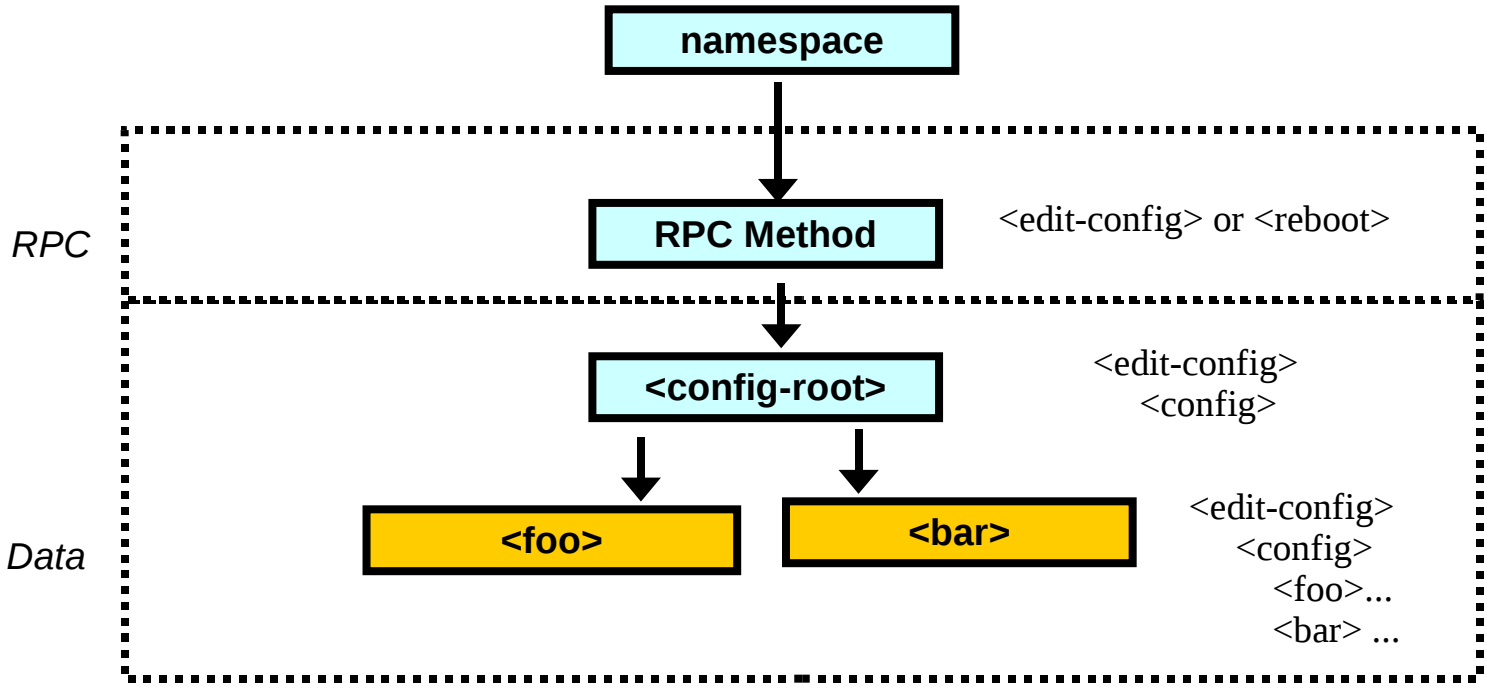
- All instance IDs can be specified with Xpath:
  - » Xpath absolute path expression
- The path from the conceptual <config> root to the object instance is the identifier value
  - » a canonical format conversion algorithm is needed
  - » a canonical instance order algorithm is needed
- Example:
  - » # ID for address (2nd in unnamed list)
    /interfaces/interface[name='eth0']/addresses/address[2]
  - » # ID for phone
    /foo/bar[x=3][y=4]/baz/acme-stuff[name='fred']/phone

Extra Slides

- Work-in-progress

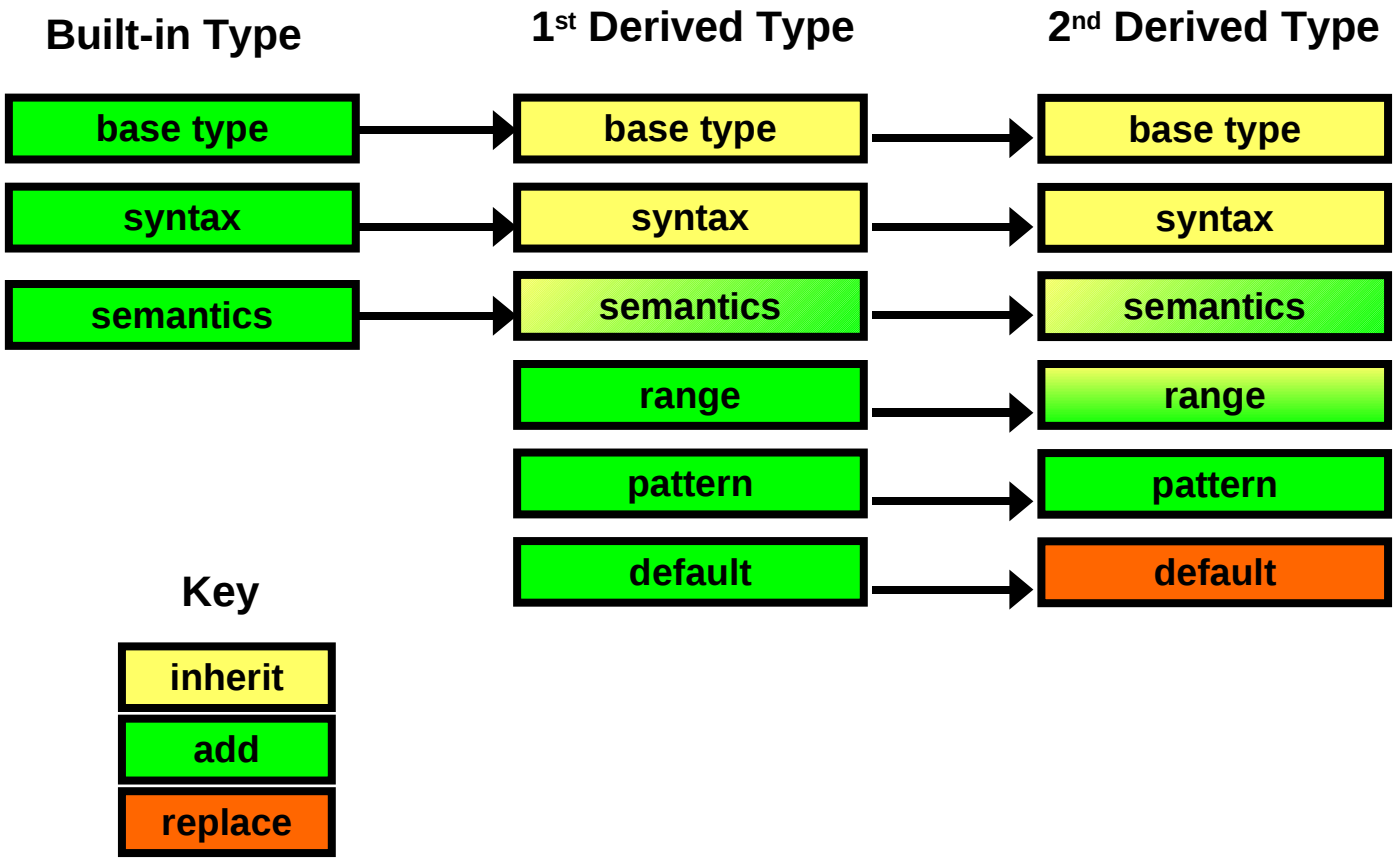
NETCONF Database Access Control Model

2 Step Access Model: First RPC, then Data

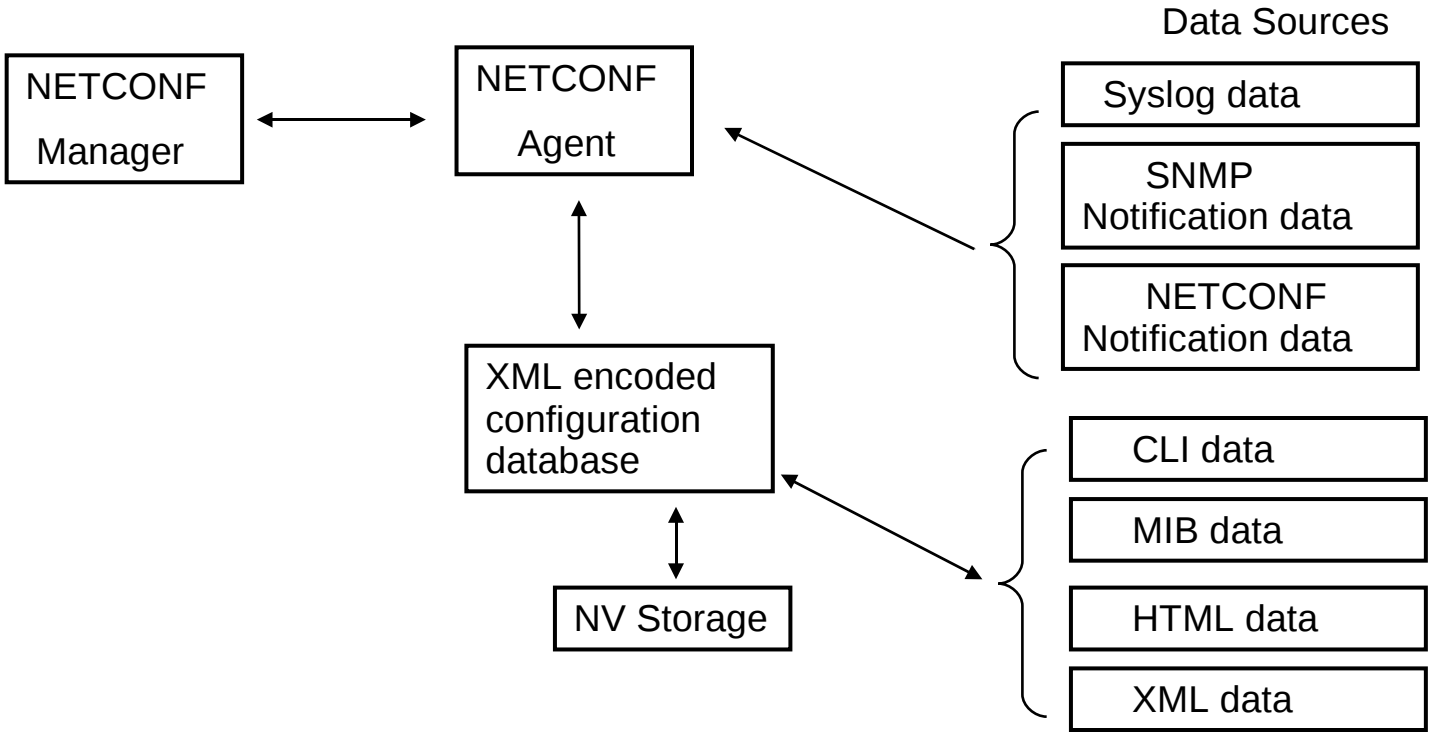


YANG does not address the NETCONF access control model at this time

YANG Derived Types



NETCONF as a Convergence Point?



- The manager uses one protocol, and only or two NETCONF sessions for all NM activity
- The agent packages or fully converts various data sources to NETCONF/XML format
- The configuration database is a conceptual hierarchical tree encoded in XML
- The data sources provide the raw data formats, which continue to be accessible