YANG-API Protocol

draft-bierman-netconf-yang-api-00
IETF 84, July 2012

Andy Bierman
Martin Bjorklund
July 25, 2012

Agenda

- What is YANG-API?
- Defining content with YANG
- NETCONF integration
- Separation of meta-data and data
- Scalable transaction model

What is YANG-API

- Programmable interface for WEB application developers
  - uses REST operations over HTTP on same conceptual content as the NETCONF server
  - uses JSON* or XML encoding of message body content (*draft-lhotka-yang-json-01.txt)
  - uses same YANG modules for YANG-API content as NETCONF
  - REST transaction model scales from simple/direct to complex/multiple candidates

Simple Example

This example operation would retrieve 2 levels of configuration datanodes that exist within the top-level "jukebox" resource.

```
GET /yang-api/datastore/jukebox?depth=2 HTTP/1.1
Host: example.com
Accept: application/vnd.yang.data+json
```

The server might respond:

```
HTTP/1.1 200 OK
Date: Mon, 23 Apr 2012 17:11:30 GMT
Server: example-server
Cache-Control: no-cache
Pragma: no-cache
Content-Type: application/vnd.yang.data+json

{ "jukebox" :
   { "library" : { "artist" : { "index" : 1, "name" : "Foo Fighters" } },
     "player" : { "gap" : 0.5 }  }  }
```

Simple Resource Model

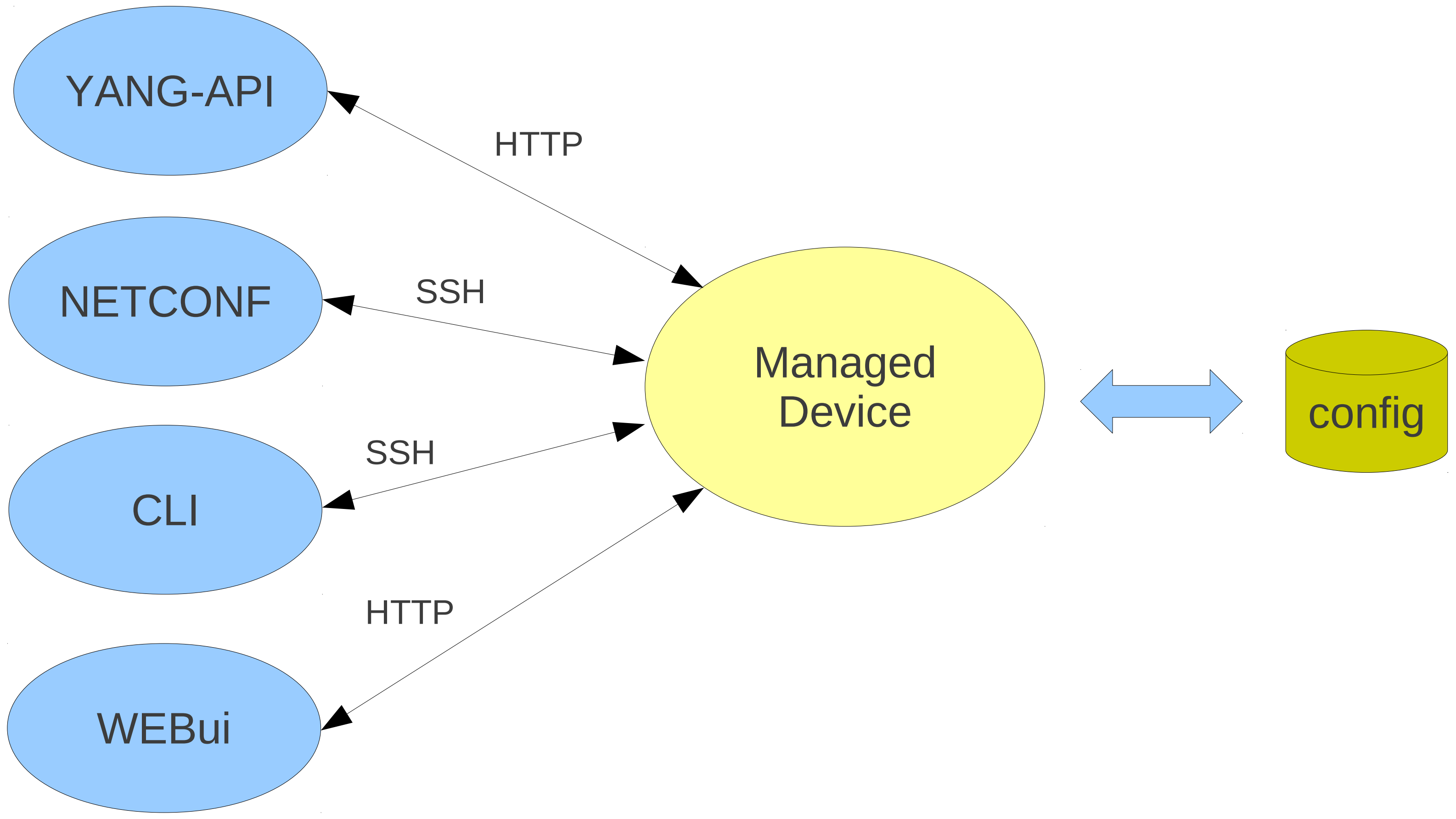
- HTTP/REST operations are applied to resources
  - 1 operation on 1 resource at a time
  - URL identifies operation, target resource, and query parameters for the operation
  - Extensible message headers provide meta-data and can refine the operations
  - Message body contains new data or operation input parameters
  - Reply uses HTTP error codes and includes <rpc-error> in the message body for errors

Defining Content with YANG

- YANG data nodes are mapped to REST resources
- YANG rpc statements are mapped to HTTP POST operations
- YANG notification statements are not supported (no notification mechanisms defined in YANG-API)

NETCONF Integration

YANG-based Framework



Separation of Meta-data

- HTTP headers are used to modify operations
  - IfMatch and ETag used together to confirm shared resource has not been modified unexpectedly
  - Range used to return subset of all list entries
  - If-Modified-Since is used to optimize polling retrievals
  - Other headers used, all according to HTTP protocol requirements

Scalable Transaction Model

- :writable-running → direct edit of /yang-api/datastore contents
- :candidate → multiple edits of shared candidate through the /yang-api/transaction contents, followed by commit operation
- multiple candidates → multiple edits of private candidate through the /yang-api/transaction contents, followed by update, then commit operations

Backup Slides

REST vs. NETCONF Operations

HTTP	NETCONF
OPTIONS	none
HEAD	none
GET	get, get-config
POST	edit-config (operation=”create”)
POST	custom RPC operation
PUT	edit-config (operation=”replace”)
PATCH	edit-config (operation=”merge”)
DELETE	edit-config (operation=”delete”)

Resource URI Map

```
/yang-api
    /capabilities
        /edit-model
        /persist-model
        /transaction-model
   /datastore
        /<top-level-data-nodes> (config=true or false)
  /modules
       /module  (leaf-list of URI string)
  /operations
       /lock-datastore
       /save-datastore
       /unlock-datastore
       /<YANG rpc-stmt operations>
  /transaction/<transaction-id>
      /commit
      /datastore
      /<top-level-data-nodes> (config=true)
      /discard-changes
      /exclusive-mode
      /update
      /validate
 /version
```

Optional Keys

- HTTP returns the ID in a Location header when a resource is created, so not all key leafs need to be provided by the client.
- api:optional-key YANG extension lets YANG module authors select key leafs for which the server can select an appropriate value if that key is missing from the message body in the POST operation.