YANG-API Implementation Observations

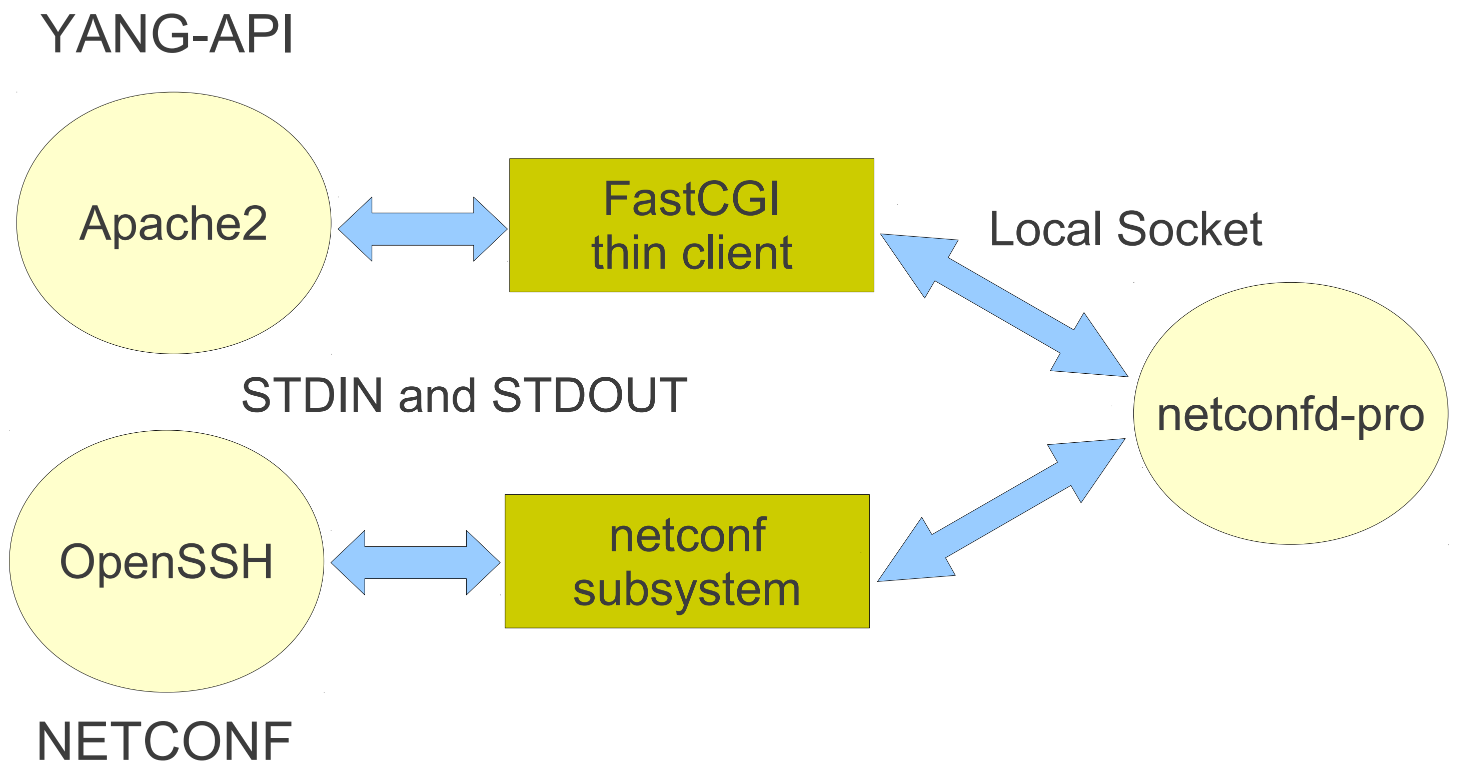
draft-bierman-netconf-yang-api-01
IETF 86, March 2013

Andy Bierman
March 12, 2013

Agenda

- Implementation Summary
- Implementation and Compatibility Issues

Implementation Summary



draft-bierman-netconf-yang-api-01.txt
draft-lhotka-netmod-yang-json-00.txt

Features Implemented

- All methods implemented (OPTIONS, HEAD, GET, POST, PUT, PATCH, DELETE)
- All query parameters implemented (config, depth, format, insert, point, select)
- Most HTTP headers implemented
- All error handling implemented (<errors> and HTTP Status)
- All **/yang-api** fields (modules, datastore, operations, version)
- All server NETCONF operations are available in YANG-API
- draft-lhotka-netmod-yang-json-00 for JSON encoding

Features Not Yet Implemented

- JSON support
  - JSON message body input
  - JSON output of POST (<get> or <get-config>)
  - depth parameter support
- Accept header
  - strong media typing not used yet
- Range, Content-Range headers
- optional-key YANG extension

Variations

- Resource definition changed so every node is a considered a resource
- Depth parameter changed to default=2 and applies to all child nodes the same
- Select parameter implemented as XPath expression, using the target resource node(s) as the document root
- Operations on a leaf-list do not require a message body since the value is in the resource URI
- Added a <data> wrapper to prevent invalid XML from being returned for GET operations on data resources

New Parameter 'test'

- Contains an XPath expression treated as a boolean
  - document root = running config root
  - context node = target resource node(s)
- Used with or without the ”select” parameter for ”needle-in-a-haystack” filtering (works like XSLT)
- GET /yang-api/datastore/interfaces/interface?
  test=type='fast-ethernet'&select=name|counters&
  config=false
  - get the name and counters for all fast-ethernet interfaces

Issues

- Resource vs. <config> subtree operations
- JSON vs. XML encoding issues
- Simplified transaction model
- Entity tags and last modified timestamps
- Pre-condition Issues

Resource Based Operations

- Pros:
  - Server can set config=false and with-defaults=report-all from the resource URI
  - Message body often not required, simplifying API usage
  - More efficient encoding than XML sub-tree
- Cons:
  - Can only access one resource subtree at a time

Comparing Message Sizes

- JSON vs. XML
  - Comparing just message body chunked encoding length; no indentation or newlines

Data Resource	XML	JSON	% diff
/netconf-state	14658	9089	-38%
/interfaces?config=false	1129	600	-47%
/?config=false (root)	24338	15807	-35%

JSON vs. XML Issues

- Pros:
  - Smaller message encoding size
- Cons:
  - Streaming output implementation of JSON can be complicated because of context-specific encodings (array, object, comma, null)
- Compatibility Issues:
  - Attributes cannot be encoded
  - <data> container may be needed in XML but not JSON; added to JSON anyway

Simplified Editing Model

Target=candidate: 2 - 9 NETCONF vs 1 YANG-API requests

lock R	lock C	lock S	edit-config	commit	copy-config R to S	unlock S	unlock C	unlock R

Target=running: 1 - 6 NETCONF vs 1 YANG-API requests

lock R	lock S	edit-config	copy-config R to S	unlock S	unlock R

- Same transaction model for all servers
- Simple editing requires only 1 request
- Implicit locking allows any locking implementation

Entity Tags

- ETag header:
  - Supported for every <running> config data node
  - Implementation up to the server (opaque string)
  - Returned for the config=true target resource in non-error responses
  - If-Match and If-None-Match unmet preconditions will cause an <error> with ”412 Precondition Failed” status for edit operations
  - If-Match and If-None-Match unmet preconditions will cause a ”304 Not Modified” status for retrieval operations

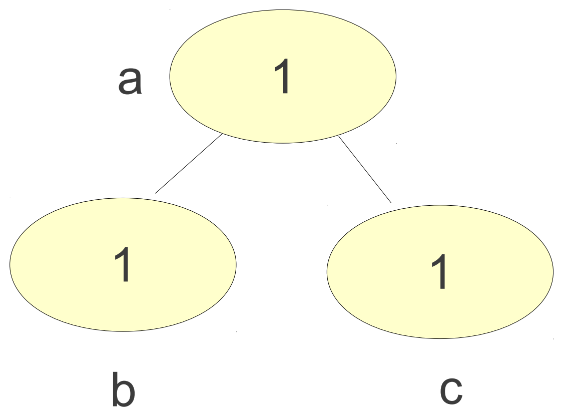
Timestamps

- Last-Modified header:
  - Supported for every <running> config data node
  - Returned for the config=true target resource in non-error responses
  - If-Modified-Since and If-Unmodified-Since unmet preconditions will cause an <error> with ”412 Precondition Failed” status for edit operations
  - If-Modified-Since and If-Unmodified-Since unmet preconditions will cause a ”304 Not Modified” status for retrieval operations

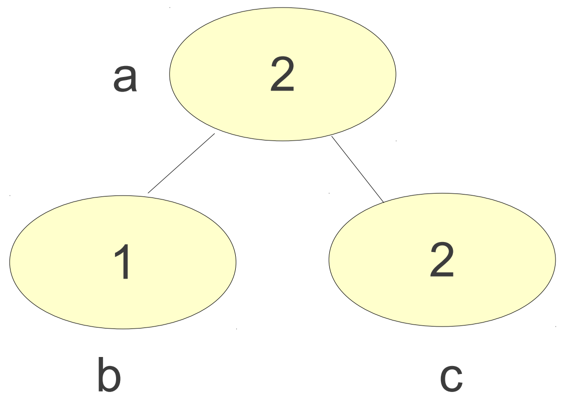
Pre-Condition Issues

- Whole Resource vs. Filtered Resource
  - HTTP procedures say to return the entire requested resource if pre-conditions pass (and 304 Not Modified not returned)
  - Is it more efficient for data resources to return only the descendant data resources instead of all of them?
  - E.g: Return node /a/b for If-Modified-Since time 2 on /a

Target Resource: .a Edit 1



Target Resource: /a/c Edit 2



Pre-condition Failed Error

- 412 Precondition Failed:
  - Supposed to only return 412 if the there would not be any errors returned
  - Used in editing with If-Match, If-None-Match, If-Modified-Since and If-Unmodified-Since headers
  - Cannot really implement this requirement because the commit can fail and if done for real then undone the device and the network could be adversely affected
  - Implement all paramater checking, then a full <validate> and return 412 if no errors so far

Compatibility Issues

- PATCH operation
  - not sure all tools support it (e.g., Poster does not but Postman does)
- Browsers cache replies using LastModified and ETag
  - Will send If-UnmodifiedSince and If-None-Match because bot LastModified and ETag previously sent
  - Causes server to persist both attributes for each node if stability across reboots is desired