Update from the NETMOD Datastore Design Team

draft-nmdsdt-netmod-revised-datastores-00

IETF 97

Agenda

- Recap of datastore design team discussions
- Protocol implications (seeking WG input)
  - NETCONF
  - RESTCONF

Design Team Solution

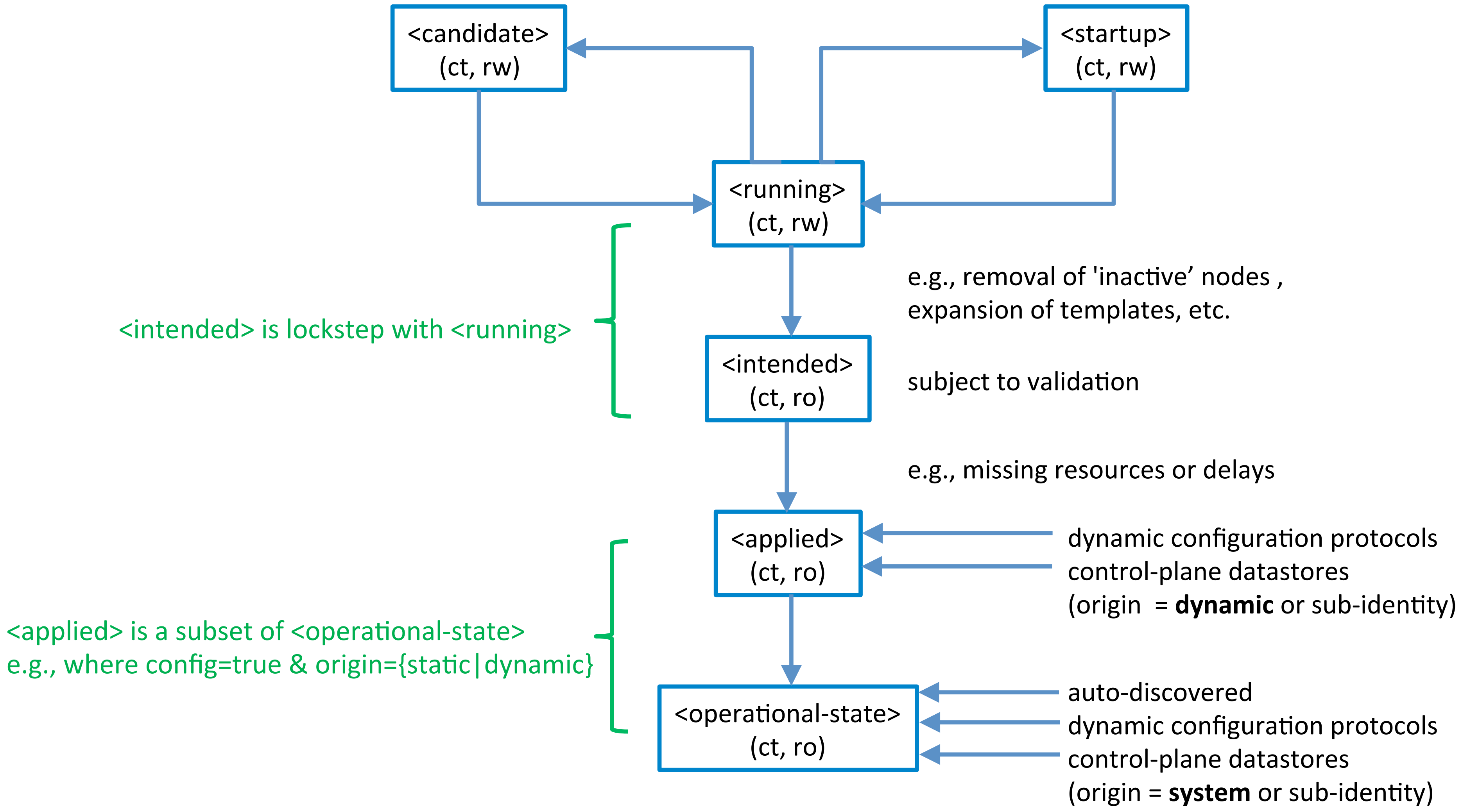
- Use three new, well-defined datastores:

  <intended>

  <applied>

  <operational-state>

- All three are read-only


<candidate>
(ct, rw)
<startup>
(ct, rw)
<running>
(ct, rw)
e.g., removal of 'inactive' nodes , expansion of templates, etc.
<intended> is lockstep with <running>
<intended>
(ct, ro)
subject to validation
e.g., missing resources or delays
<applied>
(ct, ro)
dynamic configuration protocols
control-plane datastores
(origin = **dynamic** or sub-identity)
<applied> is a subset of <operational-state>
e.g., where config=true & origin={static|dynamic}
<operational-state>
(ct, ro)
auto-discovered
dynamic configuration protocols
control-plane datastores
(origin = **system** or sub-identity)

Origin Attribute

- "origin" attribute describes the source of data
  - Appears on each node
  - Value defined as a YANG identity:

    static – data comes from <intended>

    dynamic – data from dynamic datastore

    data-model – value comes from data model

    system – system-controlled data
  - Use of identities allows for some extensibility
    - dhcp based on dynamic, etc

Datastore draft status

- NETMOD had a poll for consensus for adoption:
  - Had solid support which will be confirmed on ML
  - Would like to know if NETCONF WG also supports?

- Any questions, comments, concerns?

Agenda

- Recap of datastore design team discussions

- Protocol implications (seeking WG input)
  - General considerations
  - NETCONF
  - RESTCONF

General Considerations

- The design team has focussed on getting agreement on the data stores
- Not much time spent discussing protocol impact
- Would like to start the discussion now

Following slides are to start that conversation

NETCONF implications

- Mechanism required to advertise support for 3 new datastores:
  - <intended>, <applied>, <operational-state>
  - All are optional to implement
  - But solution only really makes sense if at least <operational-state> is implemented
- <get-config> can be used for <intended> & <applied>
  - And <operational-state> too?

<get-data> and origin meta-data

- <get-data> is a new operation to return the contents of any datastore
  - For config datastores, is equivalent to <get-config> (but not including origin meta-data)

- How to return origin meta-data?
  - Probably an optional parameter to <get-data> operation (default: not included)?
  - And <get-config>? But wouldn't apply to all config datastores (e.g. <running>, <candidate>, etc)

<get>

- Can <get> be deprecated?
  - Obsoleted by <operational-state>
  - Doesn't make much sense once <operational-state> is defined (or even now ☺)
  - Is handling <get> as a <get-data> request on <operational-state> a valid pragmatic deprecation approach?
  - Or do servers that support <operational-state> still need to support <get> (i.e. <running> + all config false)?

What about other NC operations?

- We don’t think that any need to be supported since these new datastores are all read only.
- But what about <validate/>?

RESTCONF implications

Bringing up Phil’s comment:

- Disagreement of goals of RESTCONF
  - Does it need all NETCONF capabilities?
  - Or is it the “Easy” button?

- Is this the right time to discuss this?

{+restconf}/data resource

- Represents combined config and state data
- Equivalent to bundling <running> together with <operational-state>
- Much like <get> operation of NETCONF

- Should this design be deprecated?

Some options to select datastore

1. Use a query parameter to choose the datastore
2. Or let the datastore be explicit in the path:

   e.g. {+restconf}/ds/running/…

   {+restconf}/ds/operational-state/…

   etc

2+. Could also define {+restconf}/data as:

- POST/PUT/… implicitly updates running/candidate
- GET/etc implicitly reads from operational-state

Fetching origin meta-data

- Should the origin meta-data be made available through RESTCONF at all?
- If so, presumably need an extra query parameter to choose whether to return it (default no)?

Agenda

- Recap of datastore design team discussions
- Protocol implications (seeking WG input)
  - General considerations
  - NETCONF
  - RESTCONF

Any last questions?

BACKUP

<intended>

- Content driven from <running>
  - templates/scripts/etc expanded, if supported
  - inactive nodes are removed, if supported
  - May be identical (if system doesn’t support above)
- Must be valid configuration
- Feeds into <applied>
  - E.g. can be thought of as “pre-applied”
- [Only "config true" nodes]

<applied> (#5.2)

- Currently active in-use configuration data
- Complete view of "config true" nodes
  - Where origin is static or dynamic (no defaults)
- Data may be removed:
  - Missing resources (aka ephemeral interfaces)
- Data may be added:
  - Non-"traditional" configuration sources:
    - DHCP, Dynamic Datastores, 802.1x, etc

<operational-state> (#5.3)

- "The whole enchilada"
  - All nodes, "config true" and "config false"
- Currently active in-use values
- "config true" nodes are marked with the origin attribute
- Constraints from data models do not apply
- <applied> is subset of <operational-state>
  - Where @origin is "static" or "dynamic"

Implications (#6)

- Define new DSs
- Device advertising support for DSs
  - NETCONF: capability exchange
  - RESTCONF: ??
- <get/> is deprecated

  (And there was much rejoicing!)

  Also {+restconf}/data

  Needs parameter for <operational-state>
- Clarification
  - YANG constraints apply to <intended>