Notes from Monday (10/12/2015 – 10:00-11:00 am ET)

Attendees:

- Kent Watsen
- Ignas Bagdonas
- Anu Nair
- Eric Voit
- Andy Bierman
- Susan Hares

Purpose of the Meeting: Review the revision of the I2R protocol strawman

Definition of the ephemeral requirements:

1. Yang language extension to configuration

[Sue]: Text suggests configuration information is defined by "config=true" and "ephemeral=true" which is meant to apply to the node level statement within the ephemeral data store. Should this have wider scope or less scope? Examples are:

i. I2RS protocol independent data modules: RIB, Topology, FB-RIB
ii. I2RS ephemeral additions to protocol features:

[Sue]: One example of I2RS ephemeral changes to protocol BGP feature – such as cost community addition to existing route or a BGP Peer. One can think of the cost-community as a blue marker added to a BGP route.

[Kent]: This change could require a change to the yang language. Actually, this could be provided by the yang extension, but I get ahead of myself.

[Eric]: The I2RS pub/sub work is identifying a target data store for the subscription and the target node within the subscription. Can this be applied to the I2RS ephemeral datastore? Should we allow the ephemeral comments (tag of "config=true and ephemeral=true") in all leaves of the datastore? Should we limit where these are applicable?

[Sue]: Can you explain this further?

[Eric]: The ephemeral data store is where we plan to have the config=false statements. I'm concerned about statements which are applicable only to the ephemeral data store. It would be useful to make the syntax applicable other places.

[Eric]: Kent – I was hearing from you that these statements would only be available to the ephemeral database? Is this where you going with these statements?

[Kent]: Not that these would be just applicable to the ephemeral data store, but that these commands would indicate which nodes are available within the ephemeral data store. One idea that we have consider for some time is the "leaf" level statement of ephemeral=true. Susan is raising an issue for an existing BGP peer to be in the config=true in the normal configuration, and then additions to this structure being in the ephemeral database. How do you do this? We need a keyword to define specific objects as being ephemeral in the configuration.

[Kent]: Not that we need to do something different in the language for the "config=true" datastore for NETCONF, but we are marking the these nodes as "ephemeral" so these nodes are detected.

- [Kent]: We should have a leaf level statement (for the BGP use case) and a modules statement (for I2RS modules). We should have a conformance type statement in the yang library or conformance statements where NETCONF/RESTCONF could do something dynamically to register this information. In these cases, the conformance statement would make this a function of the netconf/restconf server (aka I2RS Agent) rather than the module definitions to make this information available ephemerally.

- [Eric]: This makes sense to me. This is why I was poking at the subscription part. I want to know how much of these ephemeral node requirements are specific to a I2RS client's specific permissions (aka role-based permissions) specific on the I2RS client-I2RS agent exchange. Doing the permissions for ephemeral at the leaf level would be absolute but you would have different access role-based modules reducing this access per I2RS client attaching to the I2RS agent (netconf server).

- [Sue]: This role-based access has been in the I2RS architecture since the beginning so I am interested in this as well. Kent where do I look up the conformance statements that you are referring to in the netconf/restconf document.

- [Kent]: Historically in the NETCONF message the Hello contained the list of capabilities advertising what the netconf server (I2RS agent) supports. Introduced with RESTCONF (because RESTCONF is session less) is a yang library. The yang library has its own draft (draft-ietf-netconf-yang-library-01).

- [Sue]: Therefore, HELLO capabilities are for NETCONF and the netconf yang library statement is for RESTCONF.

- [Kent]: The netconf library statement is for RESTCONF, but the thinking is that NETCONF could use the yang library statement because it is more powerful than th Hello information.

- [Andy]: The netconf yang library is used in netconf 1.1. The Hello message is different for yang 1.1.

- [Andy]: The higher level issue here is if a netconf-based I2RS client sees a config=true, then the netconf-based/I2RS client is going to try NETCONF on this connection. At the NETMOD interim in New York City, we discussed this for a while. It came out of the New York City interim, that it is up to the operator to decide if something is persistent or ephemeral or both. It was not up to the data-model designer.

- [Sue]: I'm not sure that "leaving it up to the operator" works. We've defined protocol independent data models which are purely ephemeral. The operator can choose to use these data models or not use these data models. In BGP, we have defined additional functions in BGP which must have a leaf for the "cost community" addition that is ephemeral. The operator can choose to use this feature in the data model or not to use this feature in the data model. However, I have not seen anything not coming from an I2RS data model or a config data model with ephemeral notation. Has anyone seen a model which is not data model dependent?

- Eric/Kent: No.

- [Sue]: (note added later): As I consider I2RS interim's comment, it makes good sense for a third class of functions in BGP: Adding peers or routes ephemerally. So we have three cases:

    - Protocol independent I2RS modules (defined as whole modules) – which the operator cannot make "non-ephemeral";
    - Protocol functions (for example BGP) which are only ephemeral functions and not configuration functions (which the operator cannot make ephemeral"), and
    - Protocol functions which can be ephemeral or non-ephemeral.

- [Sue]: (a note added later): These concepts clear up a lot of confusion for the protocol modules I wrote. I think it helps the I2RS Topology-Teas Topology modules.

- Andy: OK. This is something that Dan Bogdanovic mentioned.

- Sue: I will check with Dan Bogdanovic to find out what he was concerned about.

- Andy: On implementation point of view, it is true. If an implementation can handle configuration, then the only difference on handling ephemeral data is not persisted over a reboot. It simply leaves a step out of the process. From this process viewpoint, operators can specify what is ephemeral and what is not. However, from a conformance point of view a totally role-based feature does not make sense. You could have every single config-true leaf also having an ephemeral statement. It is a certain amount of work to describe this performance. Juergen is talking about a completely new method that does not exist.

- [Andy]: If you have an ephemeral keyword that changes the way conformance works, then I'm concerned about having a NETCONF tool that will treat the ephemeral configuration as part of the configuration data store. Before, we were going to use config=false to separate the I2RS ephemeral data store from the configuration. Now, we are using just the "ephemeral statement" with "config=True". How does NETCONF know not to touch the ephemeral portions of the datastore.

- [Sue]: Any thoughts about this work?

- [Kent]: Something statement in the conformance statement in the yang library will solve this issue. You mentioned that the operators will have the decision to determine what can be access ephemerally or note. This leads away from using leaf-level or module level ephemeral statements to doing this at the conformance level.

- [Kent] Somehow in my mind, I have peace with this idea conformance level even though I have not completely thought it out. I have always concerned the leaf level ephemeral statement to be quirky. I do not have a better idea, and I was continuing with the leaf level ideal until I could do this better. In my mind, anything that the I2RS client can access in the server/I2RS agent can be ephemeral. However, this means that by default everything is ephemeral.

- [Kent]: Do other people like this idea or not? It would make it easy to determine whether config=true space is available ephemeral. It would always be available ephemerally. We would have to address the extra things which are only available ephemerally. In this situation, the conformance level statement could help.

- [Andy]: We've brought up the issue before, but the NETCONF did not want to be left out. A yang module must be understood by all protocols that understand yang. Therefore, there should be nothing that should be put in a yang module which indicates this is an I2RS model. We are questioning this wisdom.

- [Kent]: It goes to the yang Meta-model that I sent. If the notion of an ephemeral data store exists orthogonal to any protocol, then it is not protocol specific. If it just so happens, that only protocol we are discussion is the I2RS protocol – then this is just an instance of the global model. Other protocols could access this as well.

- [Kent]: For example, if we have a module that is only I2RS that module would be tagged accessible in the ephemeral data store. This is not the same as saying this is only accessible to the I2RS protocol, but at this point it time it is equivalent statement.

- [Andy]: This is a good point. For some data models, the “config=true” version which is overwritten by an ephemeral value. We simply have to classify the data clearly so it indicates which nodes can be overwritten by an ephemeral model.

- [Kent]: Yes. We should add more examples on the conformance level statements to show how contents of the modules are available in the ephemeral data store.

- [Sue]: I can add text to the document based on the draft-ietf-netconf-yang-library-01.pdf modules we have

```
+--ro modules
      +-- ro module *[name revision]
         +--ro name  yang: yang-identifier
         +--ro revision  union;
         +--ro schema?   inet:uri
         +--ro namespace   inet:uri
         +--ro feature*    yang:yang-identifier
         +--ro deviation [name revision]
         |  +-- ro name   yang:yang-identifier
         |  +-- ro revision union
         +--ro conformance enumeration
         +--ro submodule [name revision]
              +--ro name yang:yang-identifier
              +--ro revision  union
              +--ro schema?  inet:uri
```

Conformance is an enum that indicates the module implements a set of modules.

Example 0: Module is all config, no ephemeral
        Conformance = existing conformance

Example 1: Entire protocol Independent module is ephemeral
        Conformance 1 = ephemeral on every node,

Leaf-list [Yang: yang-identifier]

Example 2: ephemeral state is allowed for BGP peers
Conformance = 2 ephemeral state on submodule
Leaf-list [yang: yang-
Submodule [bgp-peers 1]

Example 3: ephemeral modules is loaded upon operator request
List deviation [name revision] d

[Sue]: Welcome to Jeff and Jie. Let me catch you up with the work in the ephemeral data store issues:

- Ephemeral state is not unique to I2RS work,
- Ephemeral data store is not intended to survive a reboot. Config information is denoted by config=true. Ephemeral nodes will be denoted by an ephemeral statement at the module level, and at the module.

[Jeff]: Did you mean at the leaf and module level only? Or did you mean block scope? Container may

[Sue]: do you think the container level as well?

[Kent]: Node-level statement is more accurate. It would apply to containers, lists, leaf-lists, and groupings.

[Jeff]: Node level is a lot clearer.

[Jeff]: Did the text that indicated if you tag something above as ephemeral, nothing below it is non-ephemeral get added to the text.

[Sue]: No, it did not. I will add it.

[Andy]: It is the subtree, so when you are tagging the node you are tagging the entire subtree as being ephemeral.

[Kent]: Just like with configuration false. If you tag a node as configuration false, the entire subtree is configuration false.

[Jeff]: If you have a reference out of the base Yang specification on how that gets applied it would save Sue having to make up some text.

[Sue]: I'd love a pointer so I can put this in.

2) Error statements

[Sue]: Should groupings be tagged in syntax? Let me discuss this after I have discussed the three error conditions: "stop-on-error", "continue-on-error", "all-or-nothing". In our discussions at Joel considered that the "stop-on-error" and "continue-on-error" were only valid for data which was "non-grouped" writes of data. Do we want to go for all-or-nothing only? Eric are you using all-or-nothing?

[Eric]: No I'm not.

[Jeff]: For Juniper implementation, all-or-nothing is easy because we are applying it to a candidate. If you an implementation that does not have the candidate approach, all-or-nothing maybe trickier because it means you have to keep the diff of what you are applying. You would have to back it out if something failed. For Andy and Eric, what does that imply for you guys.

[Andy]: The question I raised are clients taking advantage of this. We're not really seeing this point. The client would have to clever coding to recover from a partial write. The client will have to figure what was left out and how much that mattered. I have not found this level of sophistication in the document. Also, there is a question if the errors during the partial process are recoverable or not. In any decent implementation, there is a question – "should I try this again" or are we done? This type of question in the client has to get rippled through the processing. For resources or low memory, it is random. You cannot write client code that will pre-plan for spurious errors caused by low memory. The partial tends not to get used. The "all-or-nothing" is what is reliable.

Kent: The stop-on-error is only useful in the cases where the data send in the message was disjoint already. A partial application makes sense, but I2RS is intended to be a light-weight protocol which makes changes easy. If the I2RS protocol is a light weight protocol, then the difference between sending 1 message and N-messages should be small.

Eric: I totally agree with this point. A lot of what we have been doing is a lot of small read changes and small writes. We expect that the system will over-time will converge. We have not been building applications which use "continue-on-error" models for us.

Kent: Right, the point of this conversation is that we could remove complexity if we did not support "continue-on-error" models. I think we can remove this because the use for continue on error that Joel Halpern mention is not a use case we need to support. Instead clients can send multiple messages.

Jeff: I think the message is more fundamental. I agree from the client viewpoint the "all-or-nothing" is the simpliest message because it either works or it does not work. It looks very API like and that is a happy thing for us. Does the back-out of the all or nothing in the client complicate the client or cause speed reductions in the I2RS agent side? Do we care about the issues with backing out.

Andy: The validation certainly impacts speed.

Jeff: From a validation do you mean does this patch apply or something deeper than this level?

Kent: Our favorite example is an ephemeral edit that has a reference to LSP ID. If the LSP-ID has a config-false value. Let's assume the LSP-ID does not exist. At the time you sent the original message the LSP-ID existed, but during the time the message was sent from the I2RS client to the I2RS Agent the LSP-ID disappeared. Therefore when the I2RS agent on the remote system receives the message we would expect the edit to fail. This is a referential edit constraint.

- What level of validation are we talking about? We all agree on syntactical validation. Perhaps we also agree to validate that the write-query conforms to the data model. This is a level above these validations – it is does the I2RS write have referential validity checks that ensures the LSP-ID exists.
- Is this level of validity checks required?

Andy: Joel indicated that the I2RS agent should send notifications when the referential validity failed. (Or maybe it was Alia). The LSP-ID existed and now it goes away. The data is not being applied because the LSP-ID is not there. It is like an ACL with dangling references. Do you send a notification at this point.

Kent: Surely, if the LSP-ID when the write occurs everything is good. If the LSP-ID disappears, it should be that the I2RS agent sends a message to the I2RS client to clean up its state. If the LSP-ID disappears before the write comes in, are you viewing this case the same as the case where the write occurred earlier and then the LSP-ID disappears? We should then send notifications on the failed request.

Sue: My understanding is the notification in the first case is for a referenced LSP-ID going away. In the second case, it is a notification that the referential write is failing to be active due to the

Jeff: What do we care about for validation of context? There is a MUST. If we do validation at a level of validation constraints, then it will slow down the responses in the system (as Andy pointed out). The secondary thought is if you do care about the speed impact of the "MUST" you do not put the "MUST" in the yang syntax, but instead count on the method of the "commit" is not valid if the constraint is not satisfied:

- Any example of this methodology is the static route which is committed with a nexthop referencing an interface. If the interface does not exist then the commit will still go through, but the static route does not become active.
- For validation purposes, we apply the exact same semantics.

- In summary, if the I2RS work we are trying to do is geared for speed, do not put and constraints in it. We do no forbid people from doing constraints if it makes sense.

Sue: I can write-up something on that if it is what people think.

- Constraints can slow you do a large transaction or you are worried about timeliness.
- You can remove all checks, but this leads to the problem of mystery state if the invalidity causes you to fail.

Andy: Jeff is talking about leafref validation.

- In Yang 1.1, we added instance-required=false which allows leafref validation not to be done,
- Just to make sure everyone understands how leafref works, if I say my leaf points over there to a specific name – then my value has to be one of the values actually used. Leafref validation means checking that the value in this variable is actually being used in the thing you are pointing at.
- The leafref validation can be expensive to process. Maybe people have code which is more clever than our code, but this is the pain points for my company's product. When people have lots and lots of leafrefs – then all these leafrefs must be validated during writes.
- I agree with Jeff that you should not put in constraints that really do not matter. You do not want these constraints in the description cause rather than the machine readable work. This was the whole point of the Yang moving this work out of the description clauses to the machine readable causes.

Jeff: The example I gave about the static routes is a generally applicable thing. Is there any discussion inside of netmod about showing the dependencies? In this case there is a dependency between static route and the interface so the route can be active. This type of relationship is not present in yang at the moment.

Andy: Not really, there is no concern about implementation performance. This type of work has not been a concern of netmod/netconf up to this time.

Jeff: If it is not a concern of netmod at this date, then I2RS does not have to solve it at this point. We simply indicate if something (like the interface) is not satisfied, then things do not work.

Andy: ODL operates in this fashion. ODL trusts the data because it has been pre-validate. What really matters is what the operator expectations for the resulting write. The cases are :

- Variable is in effective if get a successful write, or
- Variable could be in effective if get a successful write if every constraint.

Andy: What I'm hearing is that I2RS is like a sharp knife which those who use it should understand its strengths and dangers. There is a possibility that things like leafref will not be validated. This is a lot of hand-waving to throw at the yang doctors.

Jeff: It sounds like this hand waving is expected by the yang doctors.

Kent: We need to take this hand-waving and convert it to normative text.

Jeff: If it is a general problem, should I2RS do it.

Sue: If we do not create normative text, then we run into the problem of getting other people to do it. If we start out with normative text, then the yang doctors refine the text.

Kent: The hand-waving I thought we were talking about was the amount of hand-waving that took place regarding the amount of leafref validation that takes place.

- Let's start out with an explicit indication that no "leafref" validation would be performed.
- The MUST statement could be disabled as well.
- Instance identifier – the check to see if what you are pointing to actually exists could be disabled.

Jeff: By removing these checks you indicate that the safeties are not present.

Kent: We have a new datastore in the ephemeral datastore. The intent is for it to go fast, so I can assume that these complicated validations that make it go slow. This is the point that I raised privately to Jeff Haas last week. However, I question if the speed of the validations is the longest pole in the tent of "slowness".

- We can make the validation of the configuration super-fast, but the distributing of the data out to the dataplane can be very slow.
- We could be optimizing the I2RS client-I2RS agent which could be only 10% of the time to get information to/from dataplane, and the I2RS agent-dataplane the 90% of the time.

Sue: You are right, the distribution from the I2RS agent to the I2RS data-plane is outside the I2RS scope.

Andy: The answer may be that new systems will be designed to handle these I2RS agent to I2RS dataplane work. When I stated that current systems do not have these features, stated that new new systems would be built to quicken the speed of the transfer in order to be viable I2RS systems. The older demons were written to handle 1 thing at a time because human speed of CLI typing with a single line only focused on 1 line at a time. The notion of bulk needs to be added to these older demons/devices in order to get faster performance. This is the same way that the notification systems will need to be beefed up in order to support this work.

Jeff: totally agreed.

Andy: I guess I was wrong telling Alia – just don't do that. It is true in today's system and not true in the next generation of systems.

Sue: This is a good point. I think we've gone to the notification piece. Do we continue to support all three error handlings ("all-or-nothing", "continue-on-error", "stop-on-error"), or do we just support "all-or-nothing". Is this a reasonable thing to take to the

Jeff: All-or-nothing is a wonderful idea.

Kent: me to!

Sue: Does anyone will consider this a problem?

[silence]

Sue: I will send to the i2rs-proto-dt list this comment, and ask people to object by 8pm. If I do not hear objections, I will start a mail thread on the I2RS list.

Kent: I think that "all-or-nothing" is a low bar. We are saying that validation for leafref, MUST, and instance identifier is not there. Under how many cases will the write actually fail? Not many … The instance would have to be pretty bad to fail this case.

Sue: You would have to be pretty bad.

Kent: It would be really bad so that the syntax does not hold up. We are saying that normally all would be checked.

Sue: Other than you hit on a priority problem.

Kent: What about Juergen?

Sue: Juergen asked why not RESTCONF instead of NETCONF/RESTCONF. Do I float that by the list? We've talk that the features that we plan to use will be in RESTCONF. NETCONF only supports "all-or-nothing". Jeff and I in our private discussion have always used RESTCONF as an example for all the features. We have always use RESTCONF as a way.

Kent: I'm not concerned about supporting NETCONF and RESTCONF. The statement suggests there are things that cannot be done with NETCONF. This is just how you defined the ephemeral datastore. We can make similar statements until NETCONF becomes compliant. I do have a personal feeling. It is more for the I2RS.

Andy: If we pick NETCONF, we have things we have to specify.

- For example, copy config. The question is can I copy the ephemeral datastore off to a URL. It is useful, but is it really needed. Can I copy the ephemeral datastore to the candidate or vice-versa? What happens if all of these datastore
- There

Kent: We go with our starting statement of “locking” and copy is not supported. Just disappear support for anything that causes problems with the ephemeral datastore. If what remains is so simple, that NETCONF can support it then we have a workable solution.

Jeff: To work back from what you are saying, is that we indicate our functional requirements. If someone objects, then suggest they provide text to describe the interaction.

Kent: Exactly.

Jeff: We do not get the NETCONF text perfect, but allow people to work with the text.

Andy: The anti-RESTCONF argument is that people are already using NETCONF. We want just 1 protocol, and we want config=True and ephemeral both with the same protocol. RESTCONF is new

Sue: This leads us back to continuing with RESTCONF/NETCONF and “all-or-nothing”.

Kent: I should jump into meeting.

Sue: I’ve got enough to create another draft. I think we can stop here.