NETCONF Server and RESTCONF Server Configuration Models

draft-ietf-netconf-server-model-09

NETCONF Virtual Interim

May 18, 2016

Open Issues

1. How to split this draft into several drafts?
2. How complete do the SSH/TLS models need to be?
3. How to address the semi-configurable aspects of the keychain model?

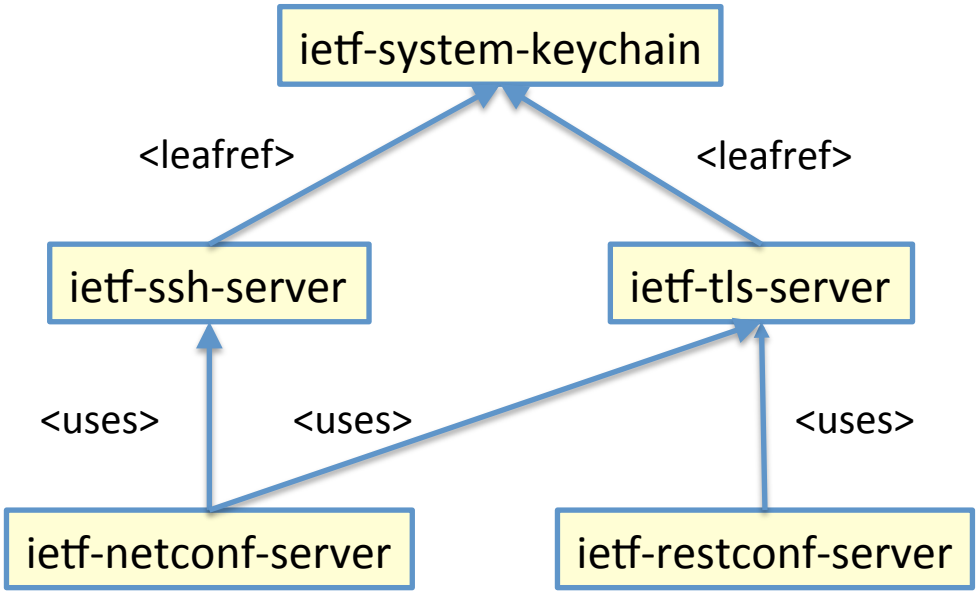
Let's discuss…

Issue #1

How to split this draft into several drafts?

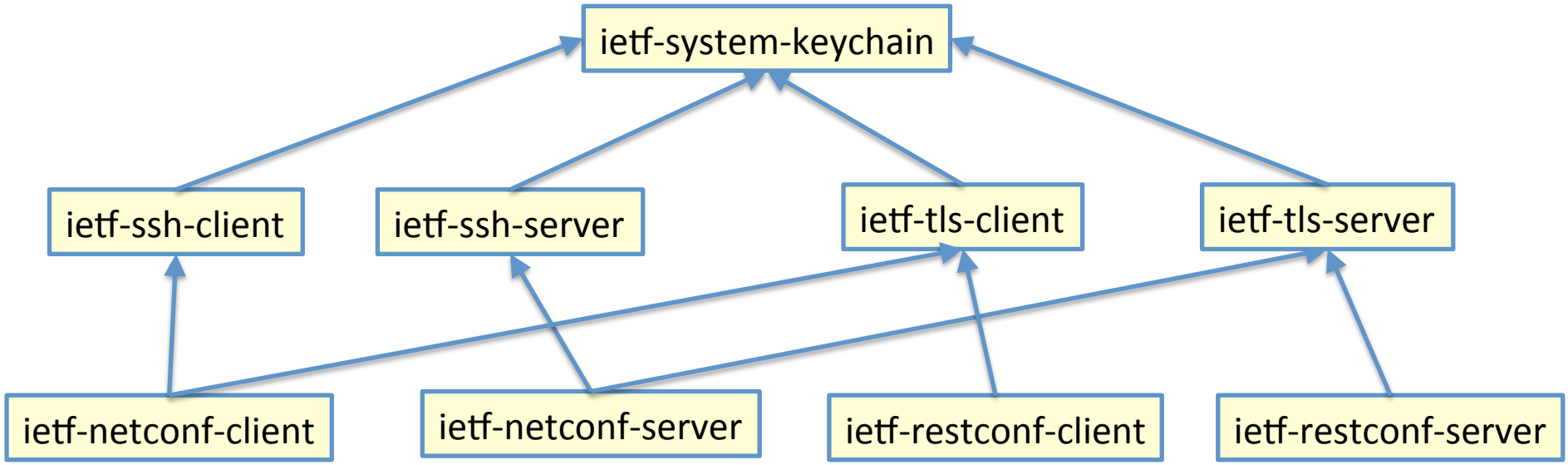
Current

This is the diagram that is in Section 3 in the draft...with s/augment/uses/g' fixed.



From IETF 95 Meeting

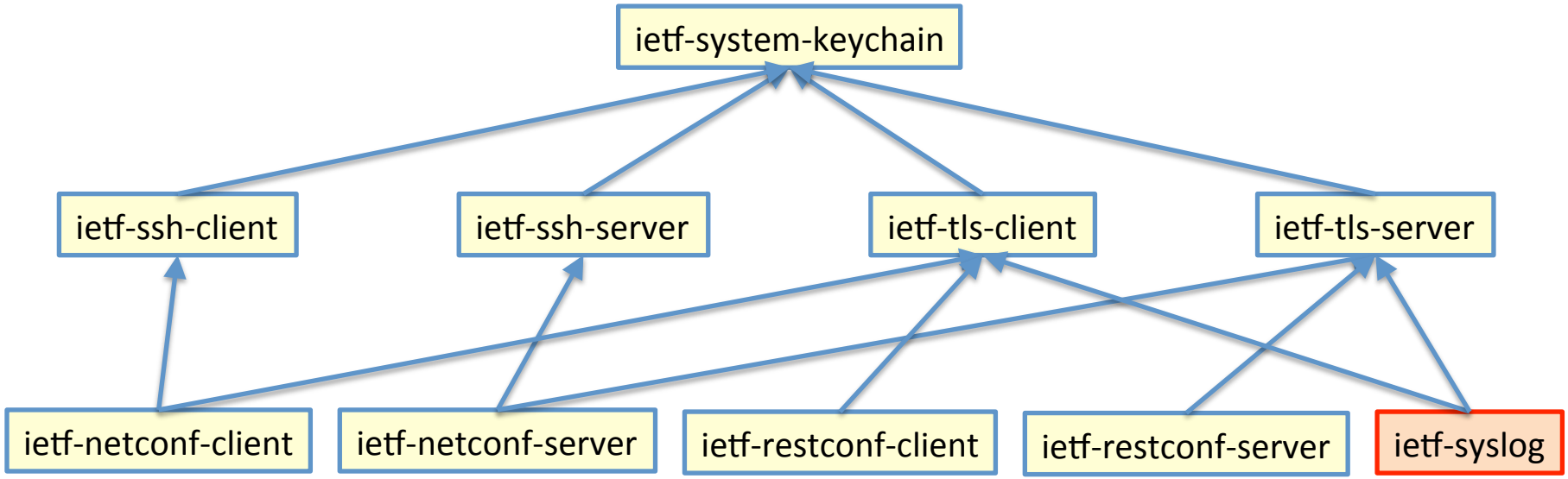
Discussion was to also define the “ietf-*-client” modules as well...



Since IETF 95 Meeting

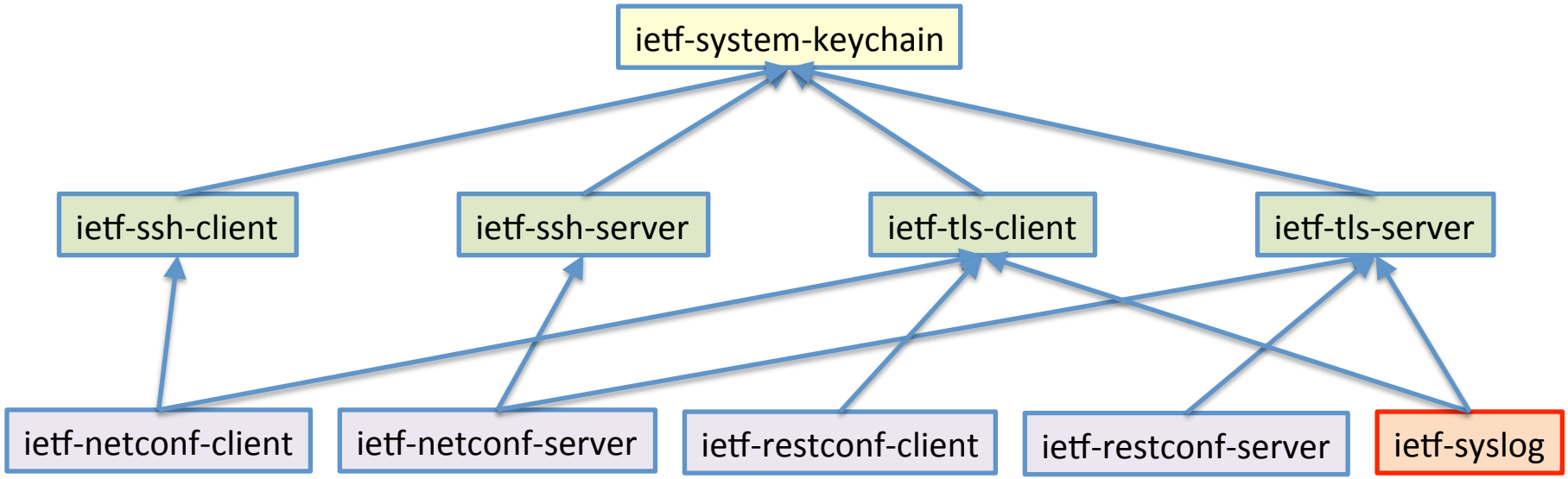
Added ietf-syslog, from draft-ietf-netmod-syslog

- note: syslog model is both a client and a server



Proposal #1

This is minimum viable solution:

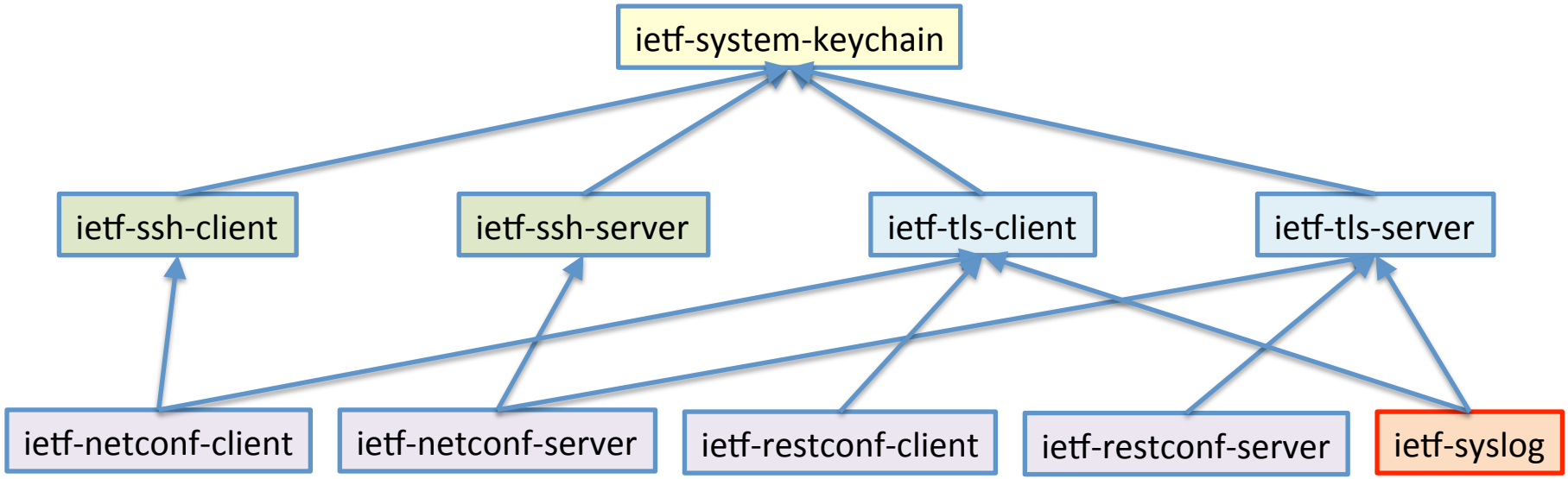


3 drafts (not including the 'syslog' draft):

- draft-ietf-netconf-system-keychain
- draft-ietf-netconf-ssh-tls-client-server
- draft-ietf-netconf-netconf-restconf-client-server

Proposal #2

This allows future servers (e.g., ietf-syslog) to reference the SSH and/or TLS modules drafts as needed:

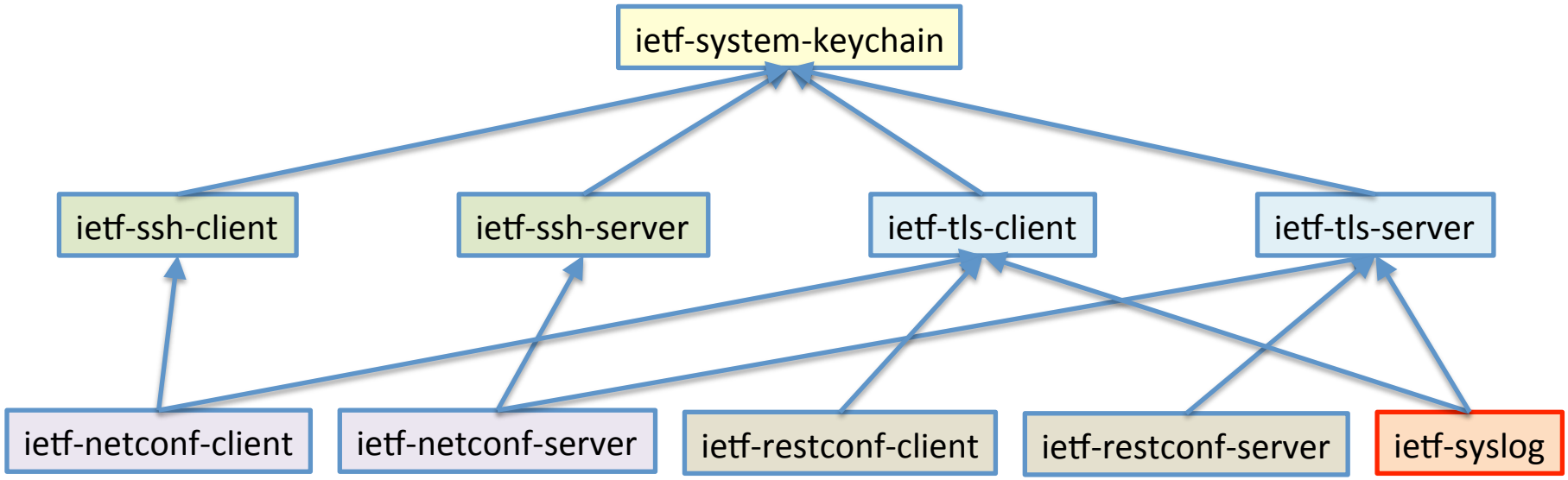


4 drafts (not including the 'syslog' draft):

- draft-ietf-netconf-system-keychain
- draft-ietf-netconf-ssh-client-server
- draft-ietf-netconf-tls-client-server
- draft-ietf-netconf-netconf-restconf-client-server

Proposal #3

This is a pretty good partitioning, with everything on well-defined boundaries:

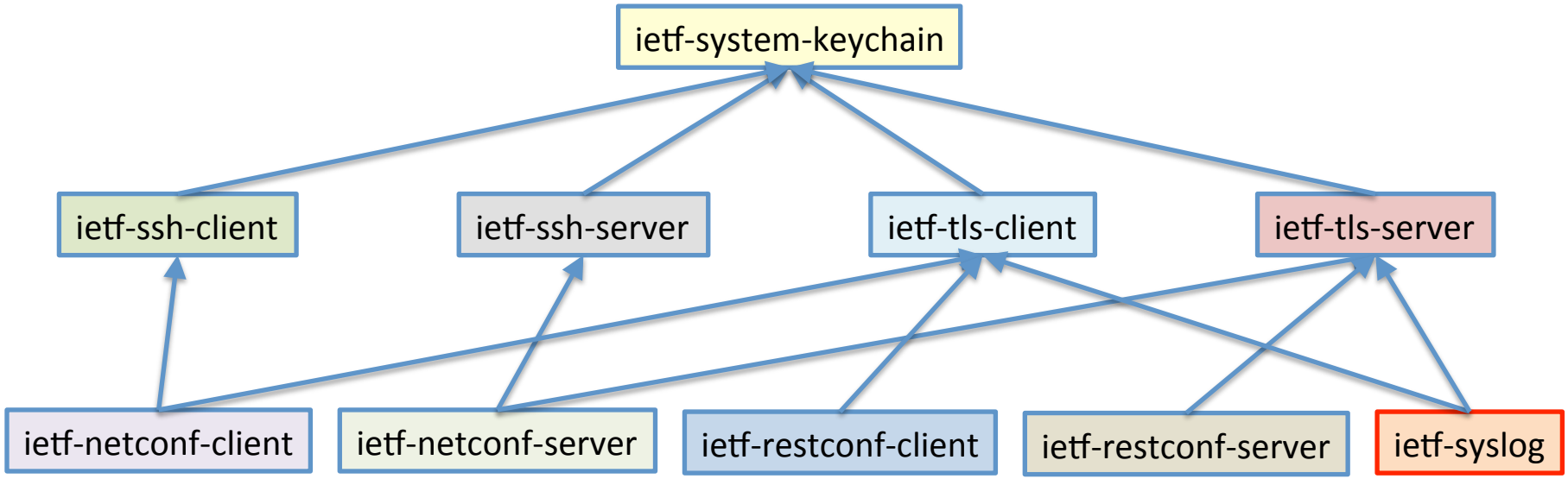


5 drafts (not including the 'syslog' draft):

- draft-ietf-netconf-system-keychain
- draft-ietf-netconf-ssh-client-server
- draft-ietf-netconf-tls-client-server
- draft-ietf-netconf-netconf-client-server
- draft-ietf-netconf-restconf-client-server

Proposal #4

Okay, this is going too far:

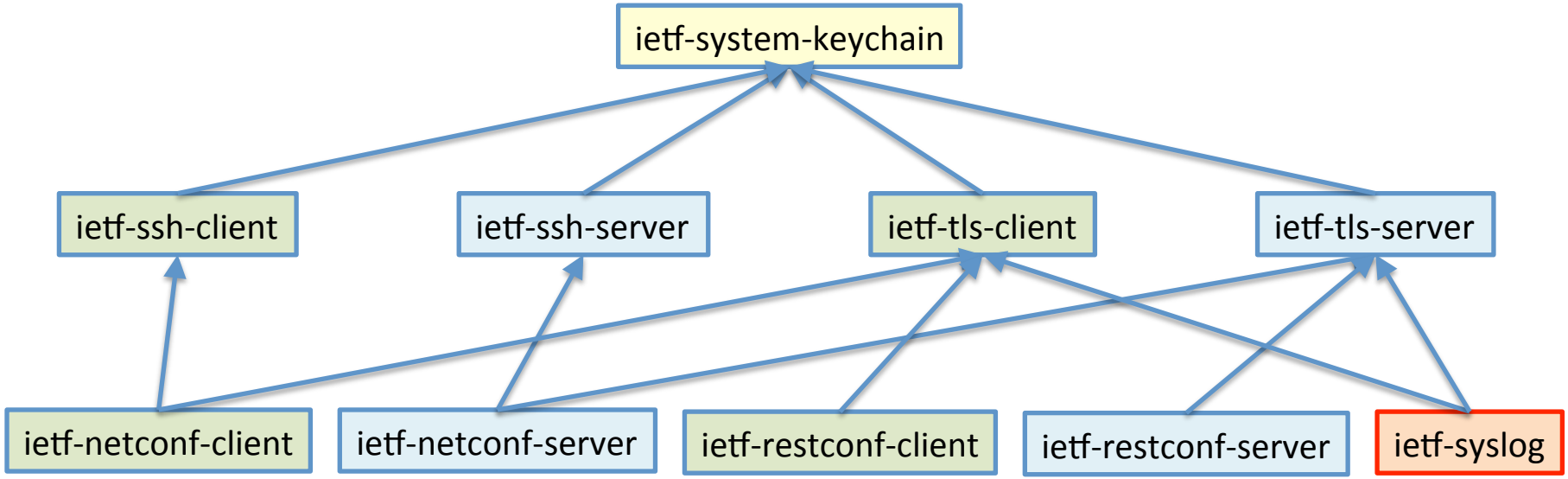


9 drafts (not including the 'syslog' draft):

- draft-ietf-netconf-system-keychain
- draft-ietf-netconf-ssh-client
- draft-ietf-netconf-ssh-server
- draft-ietf-netconf-tls-client
- draft-ietf-netconf-tls-server
- draft-ietf-netconf-netconf-client
- draft-ietf-netconf-netconf-server
- draft-ietf-netconf-restconf-client
- draft-ietf-netconf-restconf-server

Proposal #5

This doesn't align very well with the layering inherent in the modules:



3 drafts (not including the 'syslog' draft):

- draft-ietf-netconf-system-keychain
- draft-ietf-netconf-client-models
- draft-ietf-netconf-server-models

Issue #2

How complete do the SSH and TLS models need to be?

- The current draft defines a minimum subset of SSH/TLS server config
  - It does not have config knobs provided by various SSH/TLS server implementations
  - But being just groupings, they're designed to be mixed into actual server models
    - For instance, an OpenSSH server model might use/extend the ietf-ssh-server
- This issue seems similar to a module that needs to supported many vendors
  - Do we use LCD and expect augmentations to fill in missing parts when needed?
  - Or make an effort to fill in more and use feature statements to enable unsupported parts to be left out?
- Thoughts?

Issue #3

How to address the semi-configurable aspects of the keychain model?

- The current draft defines action statements such as ‘generate-private-key’ and ‘load-private-key’
- Private keys are currently unavailable in the model, but they could be added and protected by nacm:default-deny-all
  - but not all keys! (see next comment)
- That said, some private keys are never available (e.g., stored by a TPM). So for these systems, backup/restore (RMA) is impossible.

```
module: ietf-system-keychain
   +--rw keychain
      +--rw private-keys
      |  +--rw private-key* [name]
      |  |  +--rw name                                  string
      |  |  +--ro algorithm?                            kc:algorithms
      |  |  +--ro key-length?                           uint32
      |  |  +--ro public-key                            binary
      |  |  +--rw certificate-chains
      |  |  |  +--rw certificate-chain* [name]
      |  |  |     +--rw name                            string
      |  |  |     +--rw certificate*                    binary
      |  |  +---x generate-certificate-signing-request
      |  |     +---w input
      |  |     |  +---w subject                          binary
      |  |     |  +---w attributes?                      binary
      |  |     +--ro output
      |  |        +--ro certificate-signing-request     binary
      |  +---x generate-private-key
      |  |  +---w input
      |  |     +---w name                               string
      |  |     +---w key-usage?                         enumeration
      |  |     +---w algorithm                          kc:algorithms
      |  |     +---w key-length?                        uint32
      |  +---x load-private-key
      |     +---w input
      |        +---w name           string
      |        +---w private-key    binary
      +--...
```

That’s all folks!