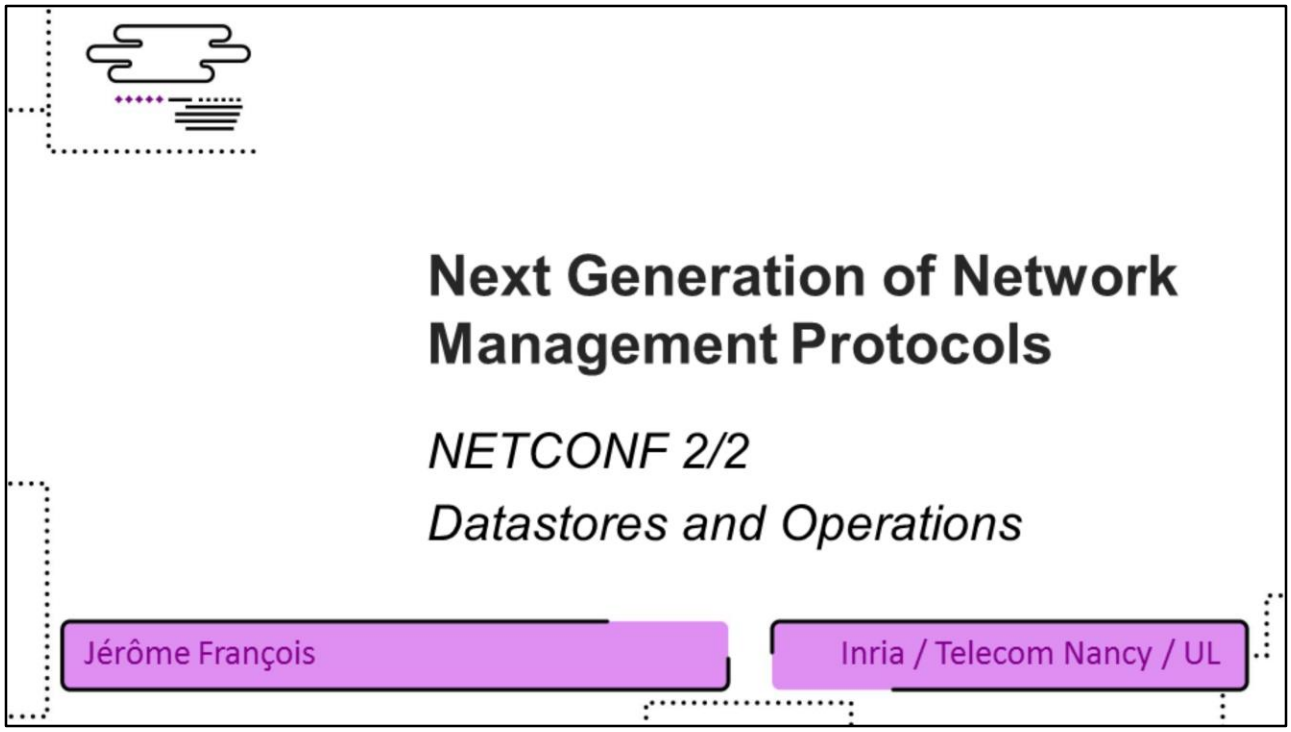
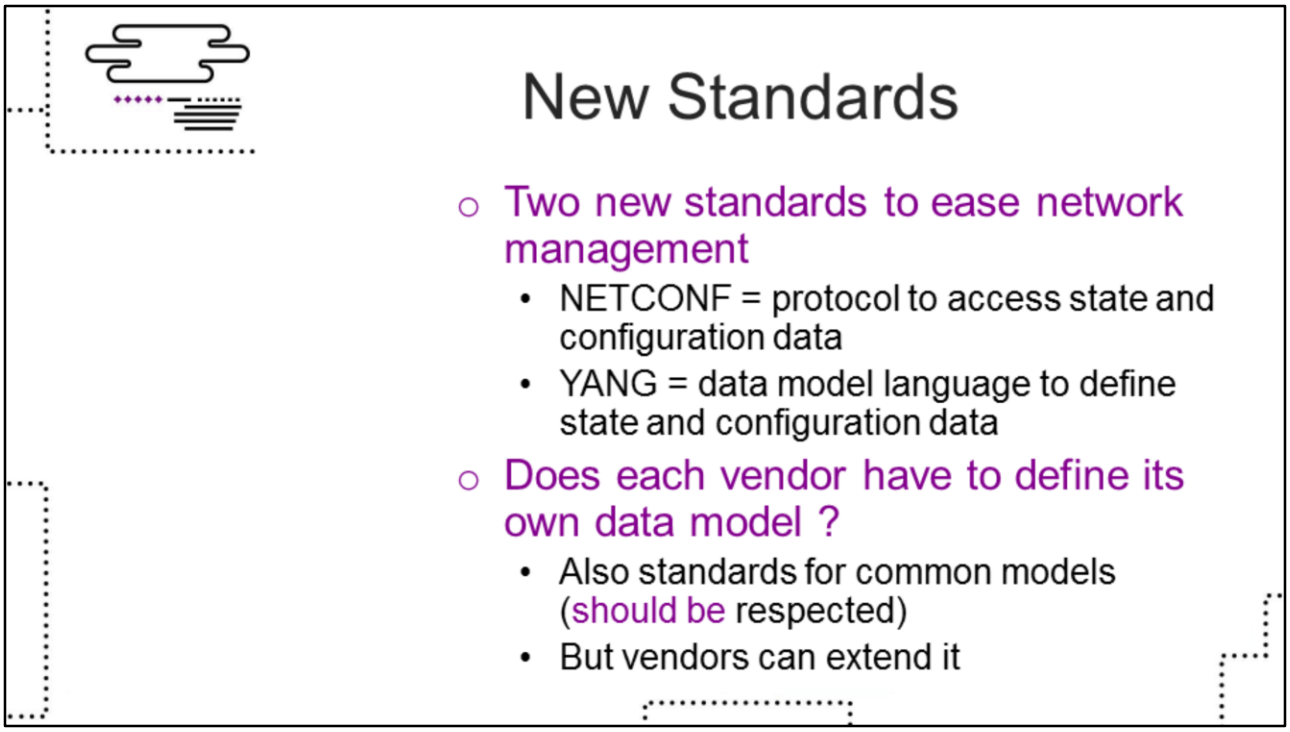
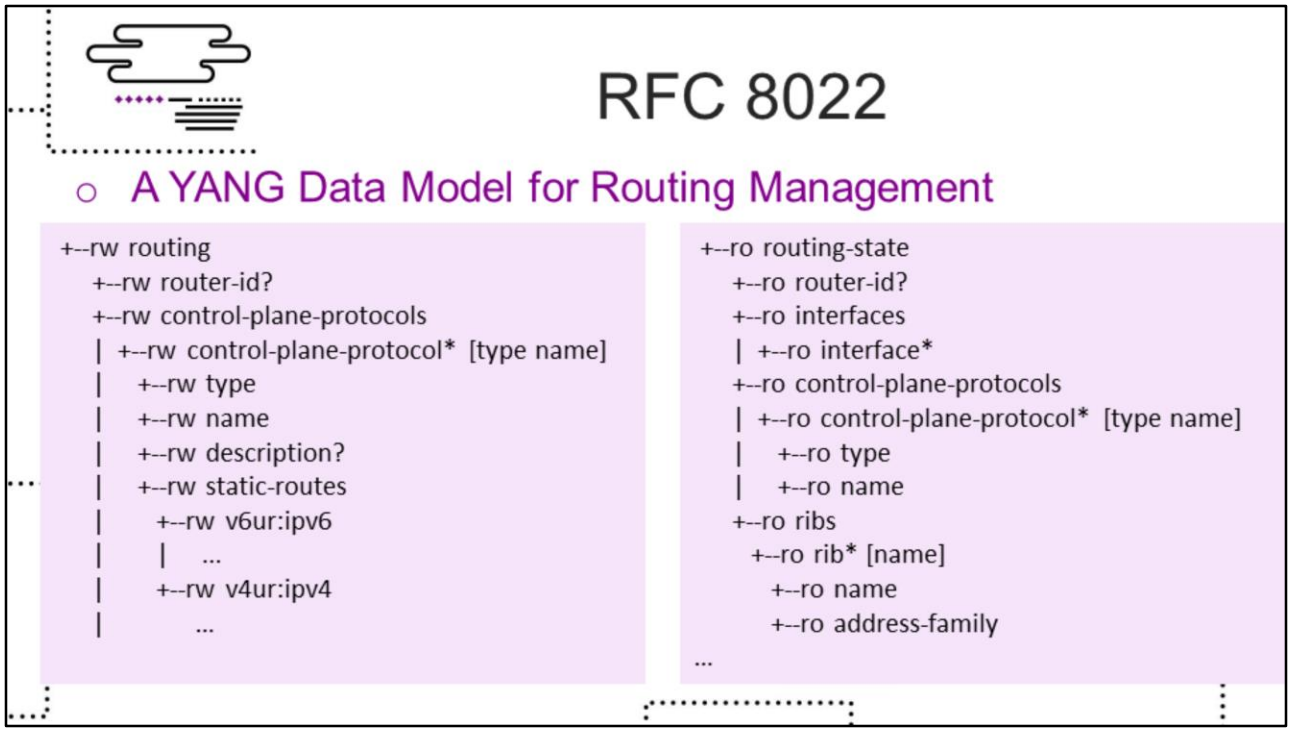


Dans ce nouveau cours, nous allons donc voir comment utiliser NETCONF.



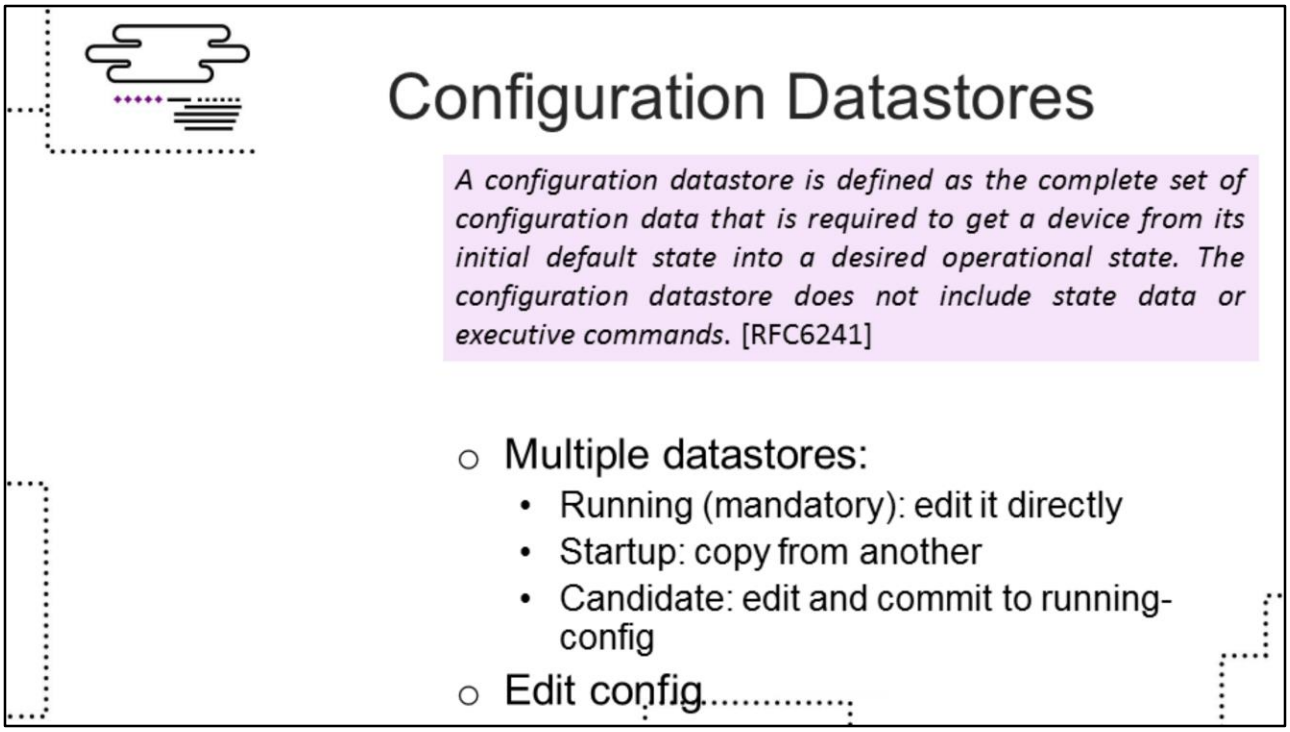
Pour rappel, NETCONF est un protocole de gestion ou supervision réseau qui permet de manipuler les données de configuration et d'état, ces dernières étant définies dans un modèle de données grâce à YANG. Vous vous demandez surement si chaque équipementier doit définir son propre modèle. Si c'était le cas, alors un des objectifs de NETCONF aurait été manqué puisqu'il est né du constat de la multiplication des interfaces propriétaires. En fait, l'IETF définit aussi des modèles de données standards, pour couvrir les besoins d'équipements classiques sans que cela n'empêche chaque équipementier d'étendre les modèles pour couvrir des fonctionnalités spécifiques.



Ici, vous avez un exemple. Le RFC 8022 définit le modèle de données pour les routeurs, ou pour être plus précis tout équipement qui peut intégrerait des capacités de routage.
Vous trouverez souvent ce type de représentation qui est bien plus condensée et permet de mettre facilement en évidence la hiérarchie des éléments.

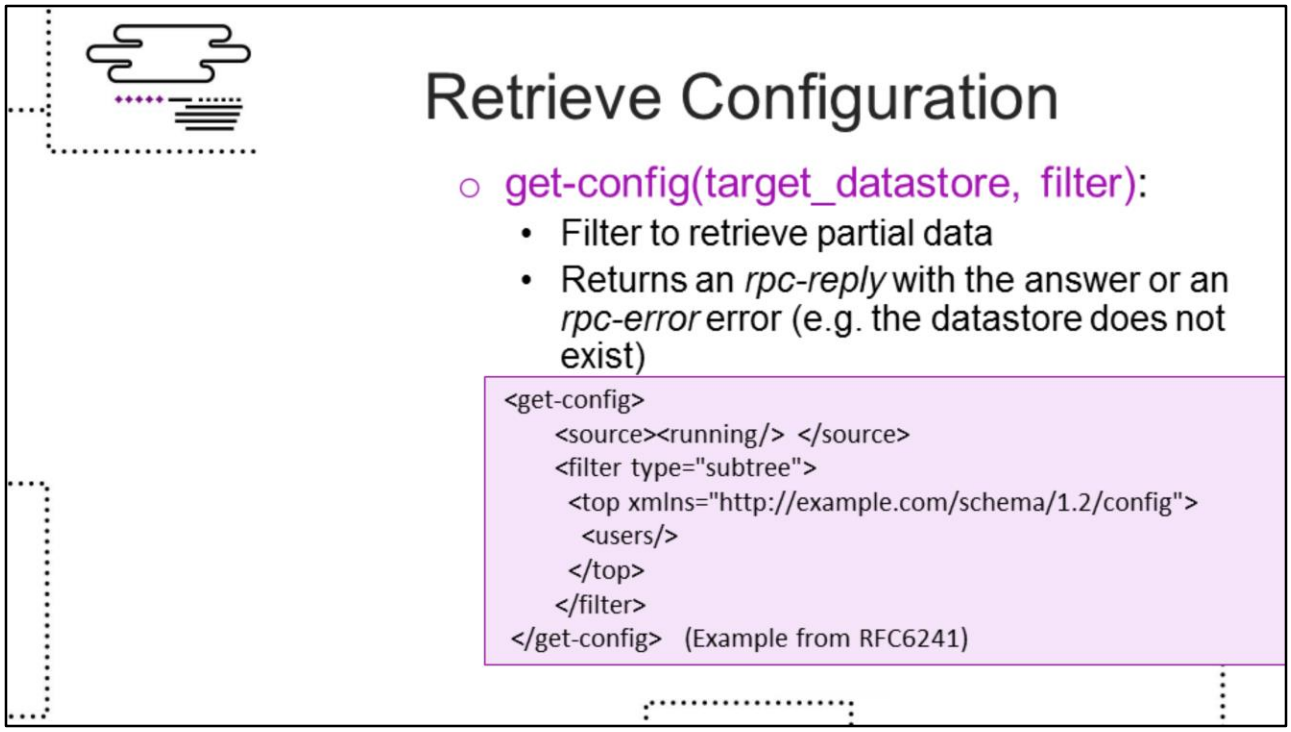
On voit donc ici les données de configurations préfixée rw pour read-writabe. La syntaxe est assez intuitive et courante, le point d'interrogation signifie que l'id du routeur est optionnel. L'étoile sur un nœud interne dénote une liste et on trouve entre parenthèse la clé utilisée, c'est ce qu'on voit ici pour le protocole de plan de contrôle dans lequel on peut définir des routes statiques.
On voit d'ailleurs que, pour ces routes statiques, on utilise soit des adresses de type IPv4 ou IPv6

Maintenant les variables d'état en lecture seule ou ro pour read only. Vous remarquerez sans doute qu'on retrouve certains éléments. Attention, ce n'est pas la même chose, à gauche, les éléments vont vous permettre de configurer un protocole de routage, à droite de récupérer son état et donc on retrouve effectivement des éléments commun comme le nom du protocole. Par contre, dans les états on retrouve les routes qui ont été inférées par le protocole de routage et inscrites dans les tables de routage ou Routing Information Base, RIB, en anglais



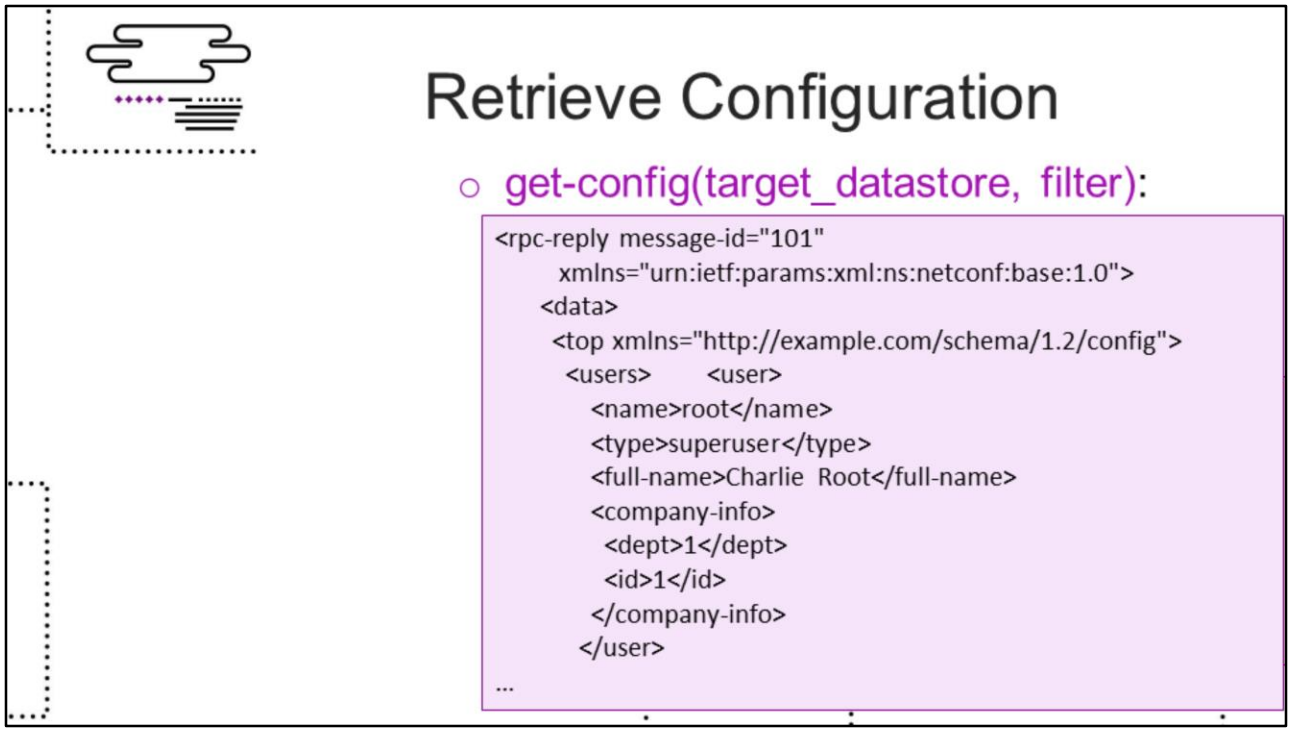
Une autre notion essentielle de NETCONF est le configuration datastore, que l'on pourrait traduire littéralement par dépôt de données, de configuration. Pour une définition très schématique et même caricaturale, on pourrait dire qu'un datatstore de configuration est un fichier de configuration, comme vous avez déjà pu en manipuler pour configurer des logiciels ou services sur vos machines.

La définition officielle dit simplement qu'un datastore est un ensemble de données de configuration qui permettent de passer un équipement, d'un état initial à un état opérationnel.
Un équipement compatible NETCONF a en principe plusieurs datastores et au moins le running, c'est-à-dire la configuration courante. Vous pouvez également avoir une configuration de démarrage, souvent copiée à partir d'une autre configuration, voire de la configuration courante, une fois que l'on s'est assuré que tout va bien. La configuration candidate, comme son nom l'indique, représente une configuration que l'on prépare, avec des opérations d'édition, en vue ensuite de la mettre en production ou plutôt en exécution grâce à commit. A l'inverse, les modifications apportées à la configuration running sont appliquées immédiatement.Pour connaître quelles sont les datastores disponibles, l'équipement ou le service configuré va l'indiquer lors de l'échange des capacités ou capabilities au début la session. Changer ou éditer une configuration s'apparentent donc à modifier le dépôt de données de configuration



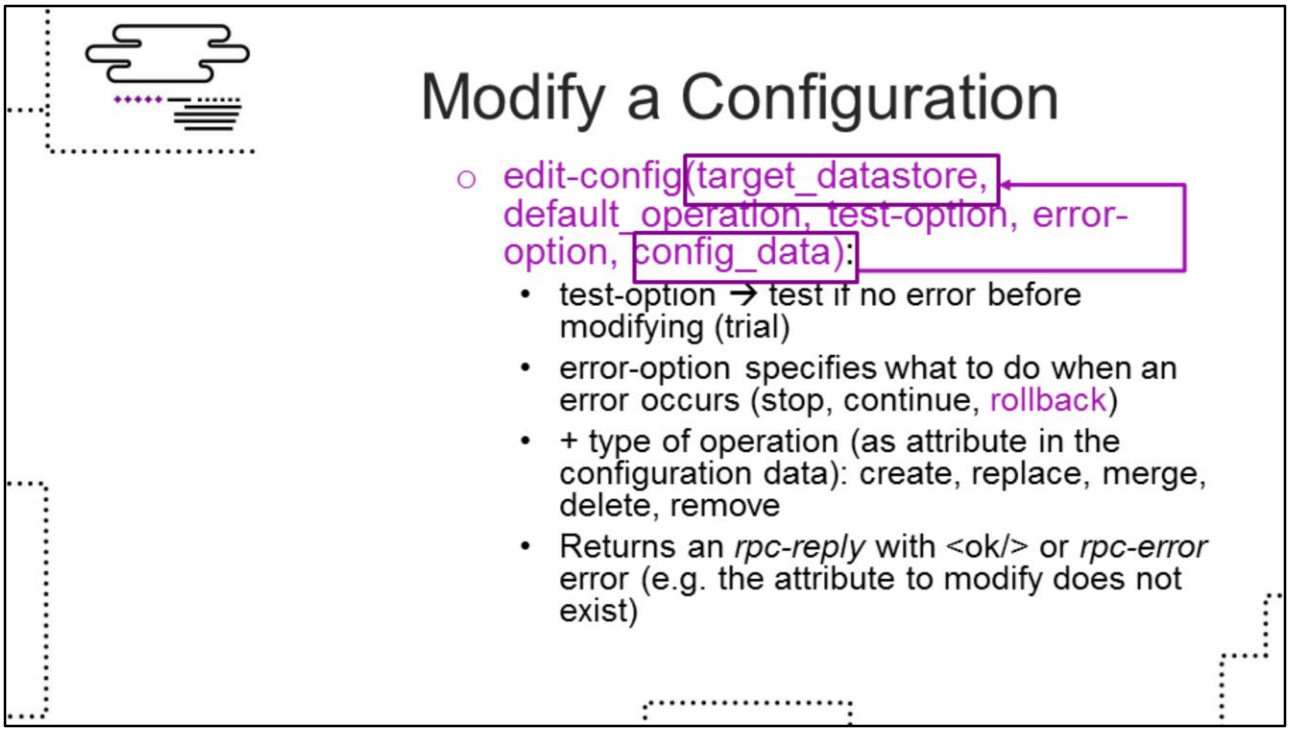
Vous vous souvenez sûrement du cours précèdent qui a rapidement introduit le protocole NETCONF et notamment l'utilisation d'appels RPCs pour exécuter des procédures distantes, dans notre cas à nous des opérations de configuration toutes encodées sous un format XML. Regardons ici le premier cas, il s'agit de récupérer des données grâce à l'opération get-config.

Dans cet exemple on voit bien les deux paramètres à utiliser. Premièrement, on précise le datastore, ici running, puis un filtre pour sélectionner les données qui nous intéresse, ici tout le sous-arbre enracinée dans la balise top avec les utilisateurs. Ici on va donc récupérer uniquement le nœud users qui a été défini dans l'espace de nommage indiqué, ainsi que ses enfants.



En cas de succès, on récupère le sous-arbre qui correspond à users et qui décrit donc les utilisateurs, comme par exemple Charlie Root. En cas d'erreur on aurait reçu une erreur avec rpc-error

Pour clarifier, il y a à la fois l'espace de nommage des commandes de NETCONF, le premier xmlns et puis l'espace de nommage des données de configuration, le deuxième xmlns dans cet exemple qui fait référence à un modèle de données YANG.



Il est possible de manipuler en écriture des configurations grâce à l'operation edit-config. Cette opération a plusieurs paramètres, d'une part la configuration à modifier, c'est à dire le target-datastore, et les opérations à appliquer, c'est à dire les config data, celles-ci vont donc s'appliquer sur la configuration mentionnée. On a également deux autres paramètres importants afin de faciliter la supervision de réseaux pour éviter de tout gérer à la main, notamment les erreurs. On peut tout d'abord demander à ce que la configuration soit testée avant d'être appliquée. Il ne s'agit pas ici de faire un test fonctionnel bien entendu mais de vérifier que ce que vous souhaitez modifier ou accéder existe bien. L'autre option error-option est plus flexible parce qu'elle définit le comportement à adopter lorsqu'une erreur est rencontrée: à savoir soit on s'arête, on continue ou on l'revient en arrière sur les opérations qui avaient déjà été appliquées.

Le type d'opération à appliquer est un attribut spécial que l'on va retrouver dans les données de configurations c'est-à-dire les config-data. En effet, config-data va décrire toutes les données de configuration à modifier, mais pour chacune d'entre elle, on peut spécifier un type d'opération à appliquer distinct. Comme vous pouvez le voir, on peut appliquer une opération de création de remplacement ou de fusion, merge. Notez bien deux opérations distinctes pour supprimer un élément. Delete a la particularité de retourner une erreur si l'élément n'existe pas contrairement à remove qui est silencieux.

Pour revenir un peu plus haut, on note aussi la présence d'un paramètre pour définir l'opération par défaut quand aucune opération n'est précisée pour les données de configuration. Et dans le cas où cela n'est pas spécifier, l'opération par défaut est la fusion. Sinon vous pouvez utilier replace pour remplacer une données ou none. None a pour objectif de n'appliquer aucune modification effective au datastore. Cela sert soit à des fins de tests, par exemple pour vérifier l'existence d'un élément (une réponse d'erreur, rpc-error, nous indiquerait alors le contraire) ou sinon pour obliger à préciser dans la partie config-data la ou les opérations à appliquer.

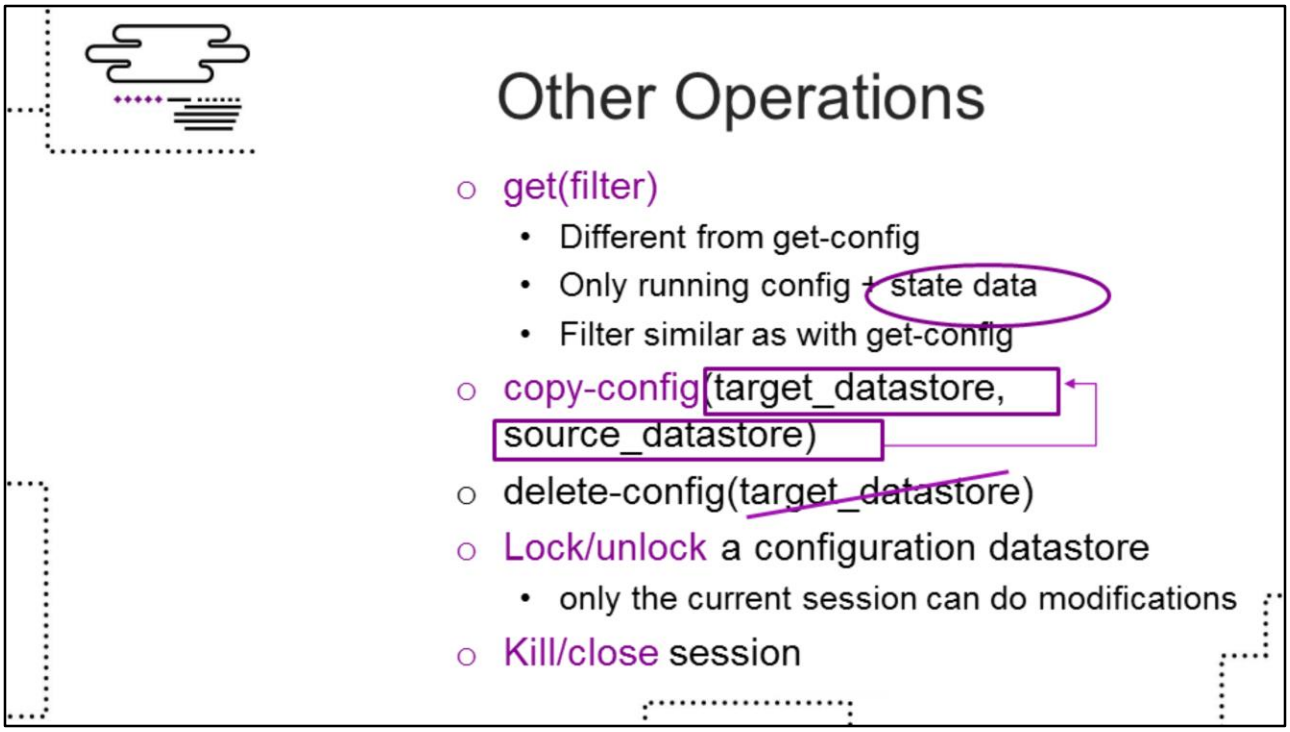
Quelques précisions, sur l'opération de fusion. Celle-ci a pour comme son nom l'indique de fusionner les informations des deux configurations, et donc d'ajouter les éléments manquants dans la configuration visée mais aussi de remplacer ceux existant quand de nouvelles valeurs sont fournies.

Example [RFC6241]

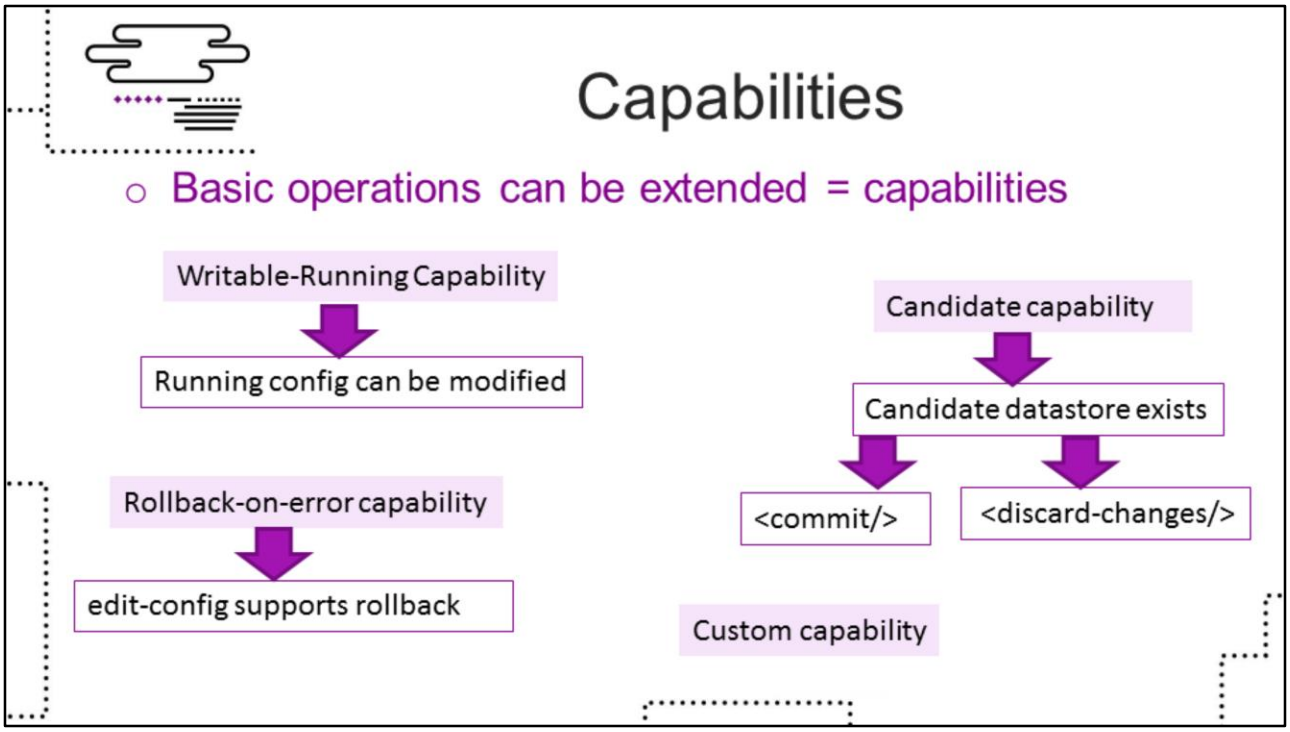
```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <edit-config>
    <target>
     <candidate/>
    </target>
    <config>
     <top xmlns="http://example.com/schema/1.2/config">
      <interface>
       <name>Ethernet0/0</name>
       <mtu>1500</mtu>
      </interface>
     </top>
    </config>
   </edit-config>
  </rpc>
```

```
<rpc-reply message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <ok/>
  </rpc-reply>
```

Après un long discours, un petit exemple.
Ici on cherche donc à modifier la configuration candidate et plus spécifiquement configurer une interface avec le nom Ethernet0/0 avec une MTU (Maximum Transmission Unit) de 1500 octets. Aucune opération spécifique n'est précisée et c'est donc l'opération de fusion par défaut qui sera appliquée.
Le résultat quand tout se déroule bien , une réponse rpc-reply avec simplement ok.



Il y a également d'autres opérations et, pour une description complète avec en particulier les différents paramètres et leurs effets, la lecture du RFC 6241 est recommandée.
La commande get est similaire à get-config et s'appuie donc sur le paramètre filter pour sélectionner des données à récupérer. Cependant cette commande est reservée pour accéder à la configuration actuelle ainsi qu'aux données d'état.
Copy-config comme son nom l'indique, permet de copier tout un datasatore dans un autre de la même façon que delete-config permet tout simplement de supprimer un datastore
Bien évidemment, cette dernière opération ne peut pas s'appliquer sur la configuration courante en cours d'exécution.
Enfin il est également possible de se réserver l'accès exclusif en écriture sur un datastore. L'utilité d'une telle commande est évidemment d'éviter tout problème de conflit en cas de modifications parallèles. Enfin pour terminer la session, deux commandes sont utilisables: kill et close. kill est un peu plus brutal puisque dans ce cas-ci toutes les opérations en cours sont interrompues sans en attendre leur terminaison.



Nous avons donc vu les opérations minimales que doit implémenter un serveur NETCONF. Comme déjà mentionné, les fonctionnalités peuvent être étendues par le biais de la déclaration de nouvelle capacités ou capabilities en anglais. Trois exemples sur cette diapositive.

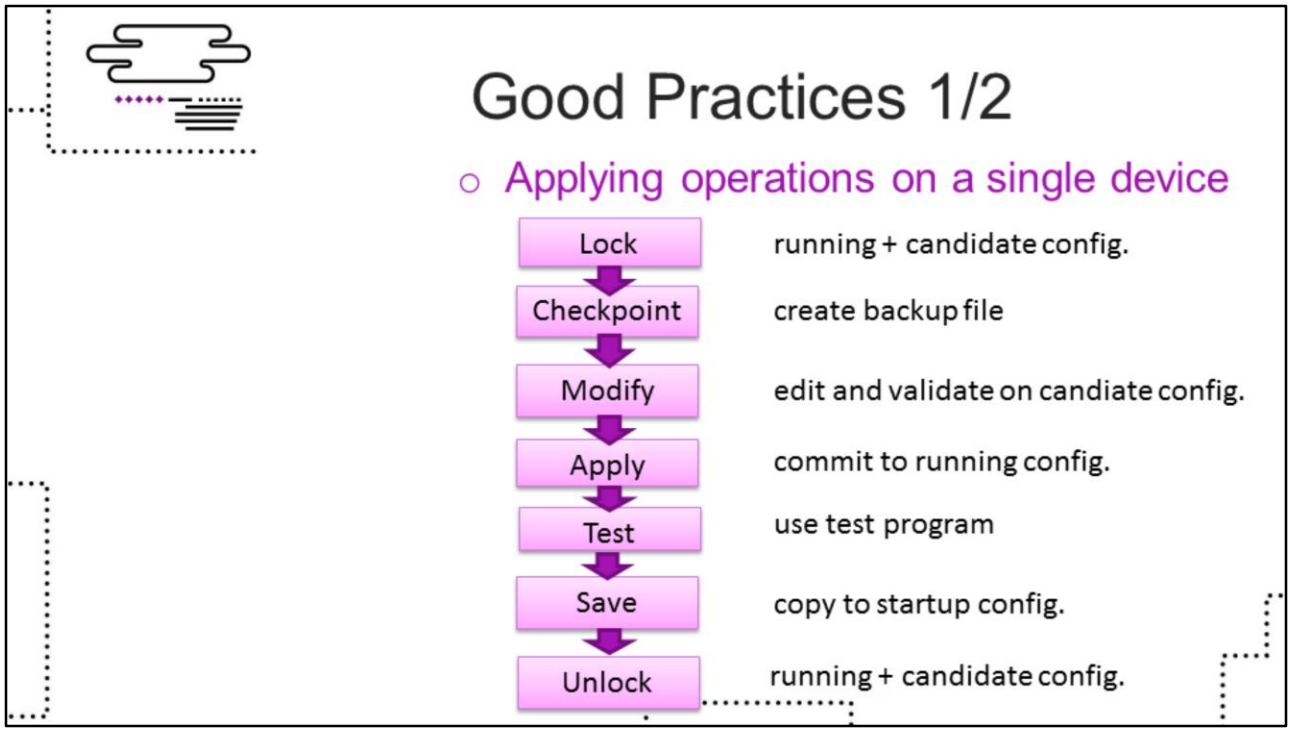
Writable-running capability signifie tout simpelement qu'il est possible de modifier la configuration courante à chaud

Rollback-on-error capability est une des capacités les plus intéressantes. Si, lors de l'application d'une nouvelle configuration avec edit-config, une erreur se produit alors on peut revenir en arrière et donc annuler toutes les modifications qui ont déjà été effectuées pour revenir dans l'état initial. C'est bien la même fonctionnalité que je vous ai décrite lors de la description de la commmande get-config où il était possible de spécifier ce comportement en cas d'erreur en paramètre. Pour utiliser un tel paramètre, il faut que donc le serveur ait annoncé cette capacité auparavant.

Candidate capability induit d'avoir un datastore candidate pour préparer une configuration sans toucher à la configuration en cours d'utilisation.
Deux opérations principales sont associées. Commit permet d'appliquer la nouvelle configuration. En cas de succès, elle remplace donc la configuration en cours d'exécution et, en cas d'erreur, la configuration en cours n'est pas modifiée.
Imaginons maintenant que vous n'êtes pas satisfait de la nouvelle configuration

candidate et que vous voulez revenir à zero, discard-changes permet de revenir à une configuration candidate identique à la running config.

Rien n'empêche bien entendu de développer de nouvelles fonctionnalités. Idéalement, l'objectif est de les proposer ensuite pour la standardisation afin d'éviter la mutliplication de capacités similaires définies indépendamment les unes des autres.

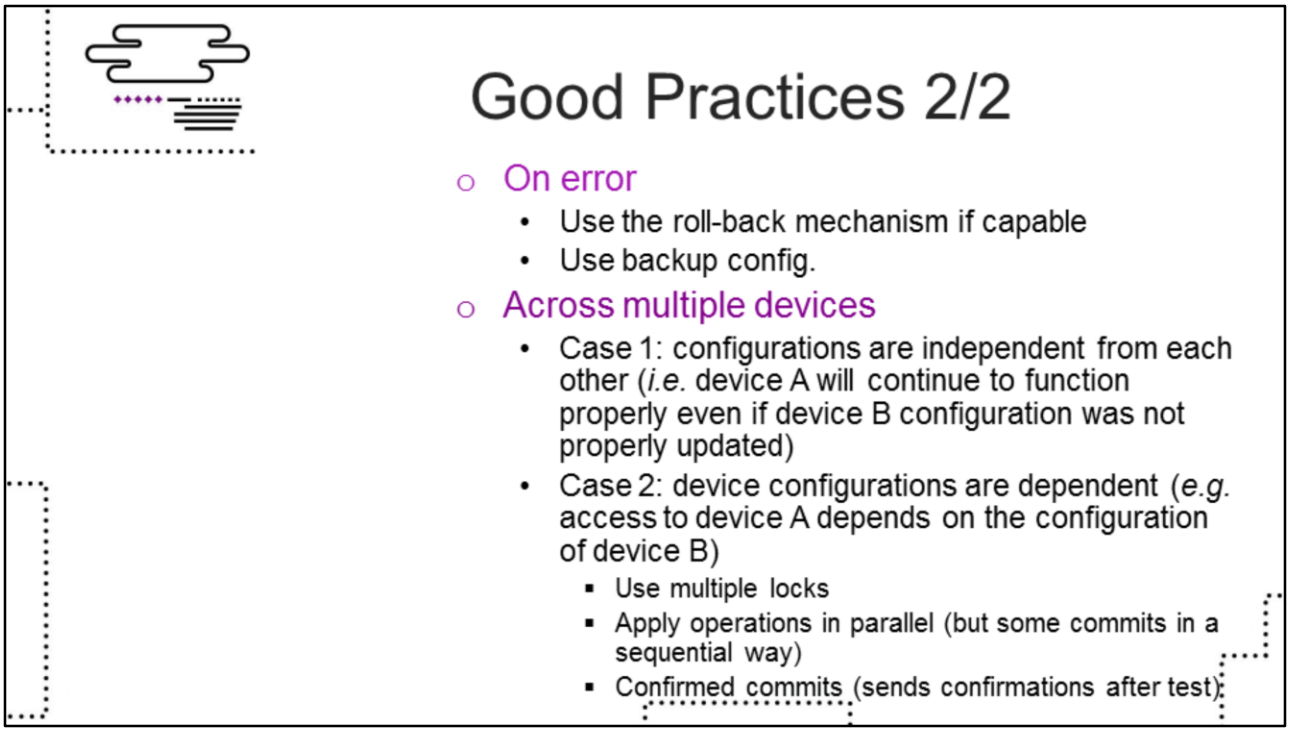


Je vais maintenant résumer ce qu'il est conseillé de faire lorsque vous souhaitez configurer un équipement.
Tout d'abord, on verrouille les modifications des données de configuration. Attention, assurez vous bien ensuite que vous êtes capable de réaliser les étapes suivantes en un temps court. Vous devez déjà avoir réfléchi et préparé vos opérations de modifications car le verrou est exclusif et donc bloquant.

Ensuite, on crée une sauvegarde de la configuration actuelle qui pourra alors être réinstallée en cas de problème
A partir de cet instant on peut donc commencer les modifications, idéalement en passant par la configuration candidate si cette fonctionnalité est proposée.
Notez aussi une étape de validation, qui est une fois de plus une capacité optionnelle, et qui permet de vérifier bonne définition vos modifications avant de les appliquer.
Vient ensuite l'étape de test. Ici la nouvelle configuration s'exécute, et a donc normalement été validée mais cela ne garantit en rien que le résultat obtenu est bien ce que vous souhaitiez. Imaginons que vous vouliez ajouter des routes par défaut pour certains préfixes et que vous êtes trompé dans la définition de ces dernier,s sans vous en rendre compte au moment de faire edit-config, c'est ici que vous devez vous en apercevoir grâce à des tests fonctionnels, par exemple avec traceroute pour vérifier vos routes statiques

Une fois ces tests réalisés et concluants, la configuration peut être considérée comme

et être copiée dans la configuration de démarrage de façon à être celle utilisée pour tout prochain redémarrage.
On finit alors par lever le verrou exclusif. Tout n’est pas toujours possible et ce modèle doit donc être adapté aux besoins.



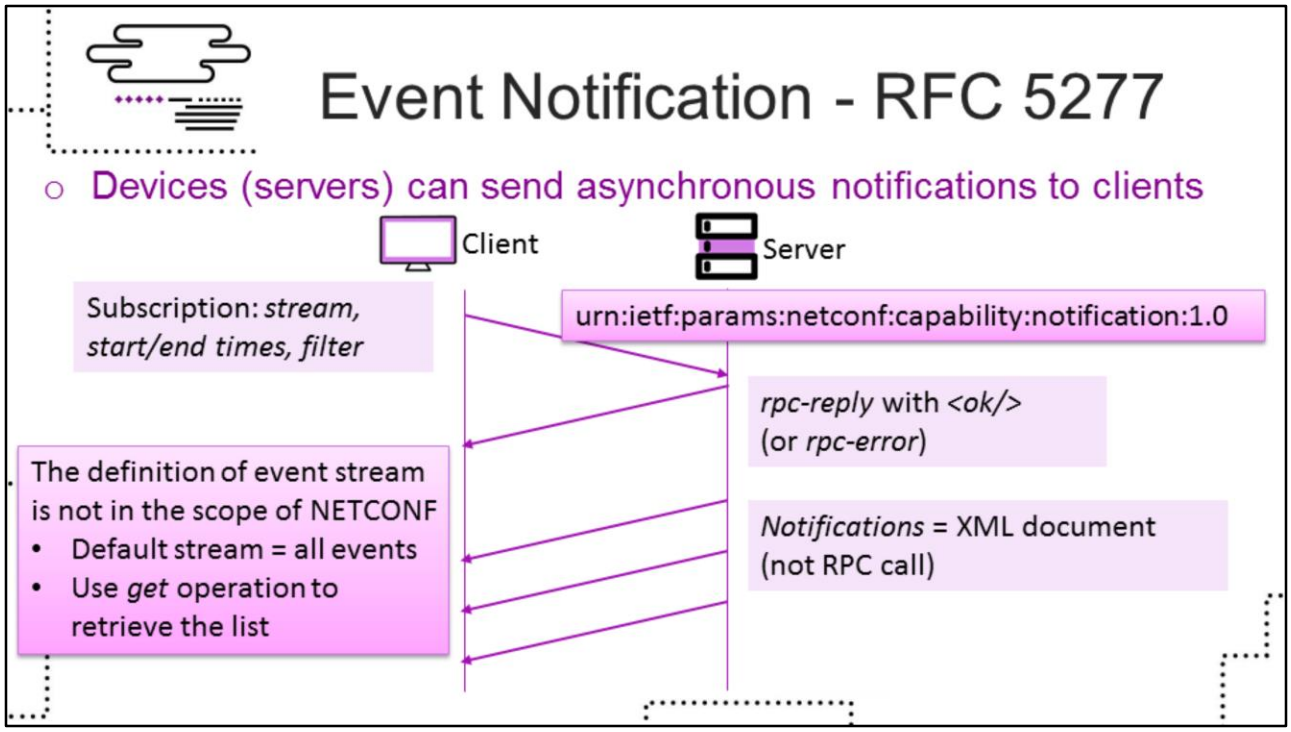
A cela s'ajoute l'utilisation du mécanisme de rollback, si ce dernier est disponible, et l'utilisation de configurations sauvegardées en cas de problèmes qui apparaîtraient. On pourrait imaginer, dans le cas précèdent, que les test fonctionnels n'ont pas une couverture assez grande et qu'une erreur n'est visible que bien plus tard. Dans ce cas, on peut revenir à une configuration sauvegardée en attendant de corriger les erreurs de la nouvelle configuration.

Dans la pratique, il est souvent nécessaire de modifier la configuration à plusieurs endroits en même temps. Dans le cas simple où les configurations de chaque équipement sont indépendantes, on peut appliquer la méthodologie précédente de manière individuelle et donc en parallèle si l'on souhaite le faire rapidement.

Dans le second cas souligné ici, c'est un peu plus compliqué puisqu'il existe des dépendances entre les équipements. Dans ce cas, il est recommandé de verrouiller en parallèle ces différents équipements, de faire les opérations et les appliquer parfois dans un ordre bien précis. Par exemple, une première modification pourrait malencontreusement couper la connectivité entre votre client NETCONF et les équipements suivants à configurer. Cela peut être le cas si vous changez la configuration d'un firewall qui va bloquer vos connexions suivantes. Petit rappel: chaque équipement à superviser héberge un serveur NETCONF.

Enfin, il existe d'autres mécanismes, d'autres capacités, pour faciliter des retours

arrières. Par exemple, confirmed-commit impose de confirmer un commit dans un délai limité. Une fois de plus, on peut imaginer le cas où une modification entraîne une perte de connectivité de l'équipement ce qui le rend complètement inaccessible pour le configurer. Il reviendra alors de lui-même à son état initial car il ne recevra pas de confirmation du commit.



Jusque là, nous avons vu comment il était possible pour un client de se connecter sur un serveur, au niveau de l'équipement à superviser, afin configurer ce dernier et de récupérer des données d'état.

Ici est donc exposé le cas inverse où un équipement remonte des informations vers le client. Cela doit cependant se faire au sein d'une session NETCONF dont l'ouverture reste à la charge du client. En effet, le serveur ne va pas de lui même envoyer toutes les notifications possibles au client.

C'est donc au client dans un premier temps de souscrire au service de notifications. Il peut notamment spécifier quels flux d'évènements l'intéresse si l'équipement en propose plusieurs. On peut également limiter les évènements remontés à ceux qui remplissent certaines conditions grâce au paramètre filter. Cela permet dans le cas du monitoring d'avoir une observation avec un grain plus ou moins large et de customiser les informations par rapport à ses besoins. Assez classiquement avec ce type de mécanisme on peut restreindre l'intervalle de temps pour recevoir les notifications.

Une fois de plus la souscription passe par un appel RPC et le serveur devrait donc répondre avec une réponse ok quand il n'y a pas d'erreur. Sinon une erreur est retournée, par exemple dans le cas où la fin spécifiée de la souscription est antérieure au début ou lorsque seule une fin est indiquée mais non une date de

début. Autre exemple, lorsque la souscription est relative à un flux, stream en anglais, inexistant.

Le serveur enverra ensuite les évènements correspondant aux flux d'évènements auquel a souscrit le client. Il faut cependant noter qu'il n'est pas ici question d'appels RPC mais simplement d'envoyer un document XML. En effet, il n'est pas question ici de déclencher automatiquement des procédures au niveau du client. Le mécanisme de notifications n'est pas une fonctionnalité obligatoire de dans la norme. Les équipements qui le propose doivent donc spécifiquement l'annoncer au moment de l'établissement de la session NETCONF. Il n'y a pas non plus de mécanisme de type keep-alive au sein de la sesssion NETCONF, car, je vous le rappelle, celle-ci s'exécute au dessus d'un autre protocole de session qui en a en fait la charge.

Notez que la définition des flux, des streams, n'est pas normée. Chaque équipement ou chaque fabricant a donc la liberté des les définir et notamment de spécifier quelles informations ils contiennent. Pour en connaître la liste, le client peut alors utiliser l'opération get. Il existe cependant un flux par défaut contenant tous les évènements.

Example [RFC5277]

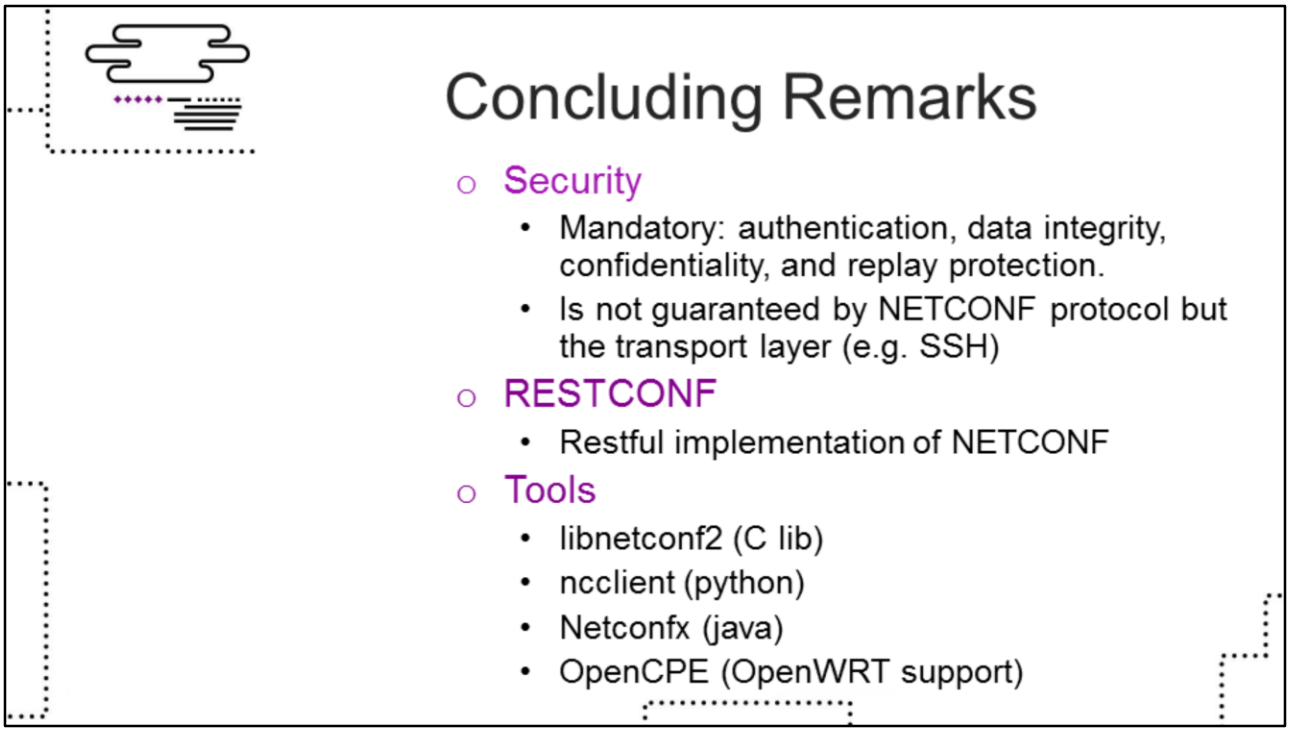
```
<netconf:rpc netconf:message-id="101"
xmlns:netconf="urn:ietf:params:xml:ns:netconf:base:1.0">
      <create-subscription
xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
       <filter netconf:type="subtree">
        <event xmlns="http://example.com/event/1.0">
         <eventClass>fault</eventClass>
         <severity>critical</severity>
        </event>
        <event xmlns="http://example.com/event/1.0">
         <eventClass>fault</eventClass>
         <severity>major</severity>
        </event></filter>
      </create-subscription>
     </netconf:rpc>
```

```
<notification
xmlns="urn:ietf:params:xml:ns:netconf:no
tification:1.0">
    <eventTime>2007-07-
08T00:02:00Z</eventTime>
    <event
xmlns="http://example.com/event /1.0">
      <eventClass>fault</eventClass>
      <reportingEntity>
        <card>Ethernet2</card>
      </reportingEntity>
      <severity>critical</severity>
    </event>
  </notification>
```

Voici un exemple. A gauche, le client souscrit aux notification sans précisé le stream. Il cherche donc à recevoir l'ensemble des notifications mais filtrées selon certaines conditions. Comme vous pouvez le constater ici, on veut remonter les différentes fautes qui peuvent avoir lieu mais seulement lorsque que leur sévérité est critique ou majeure. Il s'agit donc d'être informer des plus gros problèmes et éviter d'avoir trop de bruits en recevant des notifications pour les fautes mineures.

A droite, une notification envoyée par le serveur. On retrouve un document XML où l'on obsèrve bien un évènement de la classe fault avec une séverité égale à critical. Dans le cas illustré, le problème vient de l'interface Ethernet2. Comme expliqué précédemment, ce que contient exactement le message (sa syntaxe et sa sémantique) est dépendant de l'équipment. La seule contrainte est l'horodatage qui est défini hors des balises event.

Encore une dernière remarque sur les notifications, il est possible de recevoir des notifications passées grâce à la fonctionnalité de rejeu. En effet, c'est un intérêt majeur de spécifier une date de début, lors de la souscription, antérieure à la date courante. Bien entendu, les capacités d'un équipement peuvent limiter le nombre d'évènements mis en cache mais cela est spécifique à l'implémentation et donc hors du champs de la norme de NETCONF en elle-même.



Quelques petite remarques pour conclure. La sécurité est aussi importante et fait partie intégrante des besoins exprimés pour la gestion de réseau par les opérateurs. NETCONF s'appuie sur des standards existants comme SSH. Récemment, une spécification NETCONF basée sur REST a été proposée sous le nom de RESTCONF, celle-ci devant simplifier l'utilisation de NETCONF. Notez toutefois qu'elle est plus limitée comme l'absence des verrous. Je vous conseille donc de jeter un œil à la norme décrite dans le RFC 8040. Vous avez également ici quelques implémentations existantes qui sont listées.