TTC技術レポート
Technical Report

TR-1083

NETCONF に関する技術報告書

Technical Report on Network Configuration Protocol (NETCONF)

第 1 版
2020 年 3 月 5 日制定

一般社団法人
情報通信技術委員会

THE TELECOMMUNICATION TECHNOLOGY COMMITTEE

TTC Telecommunication Technology Committee

目次

＜参考＞

１．国際勧告等の関連

本技術レポートは、RFC6241 を調査したものである。

２．上記国際勧告等に対する追加項目等

なし

３．改版の履歴

版数	制定日	改版内容
第 1 版	2020 年 3 月 5 日	制定

４．参考文献

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/info/rfc6241>.

５．工業所有権

本標準に関わる「工業所有権等の実施の権利に係る確認書」の提出状況は、TTC ホームページでご覧になれます。

６．技術レポート作成部門

第 1 版 ： 企業ネットワーク専門委員会

I 本技術レポートの概要

これまで、ネットワークの状態可視化には、広く SNMP (Simple Network Management Protocol) が使用されてきた。しかし、SNMP の最新バージョンである SNMPv3 は 1999 年に登場したレガシーなプロトコルであり、様々な課題を持っている。そのため、IETF では 2001 年頃から継続して SNMP に代わるネットワーク管理プロトコルの議論が行われている。

2001 年当時から言われている SNMP の課題のひとつは、SNMP は装置の状態取得に使われることが多く、装置の設定に関してはサポートされている機能が少ないという点である。例えば、SNMP はトランザクションの仕組みを持っておらず、設定の妥当性の確認やロールバックといったことができない。また、バイナリベースのプロトコルであり、プログラマがあまり親しみのない SNMP 独自のプロトコルを理解しなければならないため、独自の MIB に対して操作を行うことへの敷居がやや高いという課題もあった。

これらの課題を解決するために、IETF では 2002 年頃から SNMP に代わるネットワーク管理プロトコルとして、NETCONF/YANG の標準化が進められてきた。

NETCONF/YANG は、近年の大規模データセンターの普及でネットワークの仮想化が進んだことにより、より柔軟にネットワークを運用管理が可能な技術として、SDN (Software Defined Networking) や NFV (Network Function Virtualization) で利用されるようになった。国内では一部の機器が NETCONF/YANG をサポートしているのみであるが、海外では Cisco や Juniper Networks といった大手ベンダを始めとして、既に多くのルータベンダが NETCONF/YANG をサポートしている。

また、最近の議論では管理対象デバイスから管理装置に YANG データをプッシュする、YANG push の議論も進んでいる。2020 年 2 月現在、YANG push の RFC は一部を除きまだドラフトの状態であるが、改版は進んでおり標準化も近いと予測される。

本報告書では、NETCONF の最新仕様である RFC6241「Network Configuration Protocol (NETCONF)」のセクション 1 からセクション 8 について、日本語に翻訳している。翻訳文中に出てくる付録については、本報告書の翻訳対象外のため、RFC6241 の原文を参照されたい。

II RFC6241 の和訳

概要

この文書で定義されるネットワークコンフィグレーションプロトコル (NETCONF) は、ネットワークデバイスのコンフィグレーションをインストール、操作、および削除するためのメカニズムを提供する。これは、コンフィグレーションデータならびにプロトコル・メッセージを表すために、拡張マークアップ言語 (XML) ベースのデータ・エンコーディングを使用する。NETCONF プロトコルの操作は、リモートプロシージャコール (RPC) として実現される。この文書は RFC 4741 を廃止する。

1. はじめに

NETCONF プロトコルは、ネットワークデバイスの管理やコンフィグレーションデータの検索、新しいコンフィグレーションデータのアップロード、操作をすることができるシンプルなメカニズムを定義する。このプロトコルは、デバイスが完全で正式なアプリケーションプログラミングインタフェース (API) を公開することを可能にする。アプリケーションは、このわかりやすい API を使用して、完全なコンフィグレーションデータセットおよび部分的なコンフィグレーションデータセットを送受信することができる。

NETCONF プロトコルは、リモートプロシージャコール (RPC) の仕組みを使用する。クライアントは、RPC を XML [W3C.REC-xml-20001006] でエンコードし、セキュアなコネクション型セッションを使用してサーバに送信する。サーバは、XML でエンコードされたレスポンスを送信する。リクエストおよびレスポンスの両方の内容は、XML DTD または XML スキーマ、または両方で完全に記述され、両方の当事者が交換に用いられた構文制約を認識することを可能にする。

NETCONF の重要な側面は、管理プロトコルの機能性がデバイスのネイティブな機能を厳密に反映することを可能にすることである。これにより、実装コストが削減され、新しいケイパビリティへのタイムリーなアクセスが可能になる。さらに、アプリケーションは、デバイスのネイティブユーザインタフェースの構文コンテンツと意味コンテンツの両方にアクセスすることができる。

NETCONF は、クライアントが、サーバによってサポートされるプロトコル拡張のセットを発見することを可能にする。これらの「ケイパビリティ」は、クライアントが、デバイスによって公開された機能を利用するために、その挙動を調整することを可能にする。ケイパビリティ定義は、非集中方式で容易に拡張することができる。標準ケイパビリティおよび非標準ケイパビリティは、意味論的および構文論的に厳密に定義することができる。ケイパビリティについては、セクション 8 で説明する。

NETCONF プロトコルは、システムにおける自動コンフィグレーションの構成要素である。XML は共通語を交換するための仕組みであり、階層型コンテンツを提供する柔軟で完全に規定されたエンコードメカニズムを提供する。NETCONF は、XSLT [W3C.REC-xslt-19991116] などの XML ベースの変換技術と協力して使用され、完全なコンフィグレーションおよび部分的なコンフィグレーションの自動生成のためのシステムを提供することができる。システムは、ネットワーキング・トポロジ、リンク、ポリシー、顧客、およびサービスに関するデータを取得するために 1 つまたは複数のデータベースを参照することができる。これらのデータは、1 つまたは複数の XSLT スクリプトを使用して、タスク指向のベンダ独立データスキーマから、ベンダ、製品、オペレーティングシステム、およびソフトウェアリリースなどの固有の形式に変換することができる。結果として生じるデータは、NETCONF プロトコルを使用してデバイスに渡すことができる。

本文書におけるキーワード「MUST」、「MUST NOT」、「REQUIRED」、「SHALL」、「SHALL NOT」、「SHOULD」、「SHOULD NOT」、「RECOMMENDED」、「MAY」、「OPTIONAL」は、[RFC2119] の記述どおりに解釈される。

1.1. 用語

- 候補コンフィグレーションデータストア：デバイスの現在のコンフィグレーションに影響を与えることなく操作することができ、ランニングコンフィグレーションデータストアにコミットすることができるコンフィグレーションデータストア。すべてのデバイスが候補コンフィグレーションデータストアをサポートしているわけではない。
- ケイパビリティ：基本的な NETCONF の仕様を補う機能。
- クライアント：サーバ上のプロトコル操作を呼び出す。さらに、クライアントは、サーバから通知を受信するためにサブスクライブすることができる。
- コンフィグレーションデータ：システムをデフォルト状態から現在の状態に変換するために必要な書き込み可能なデータセット。
- データストア：情報を格納し、アクセスするための概念的な場所。データストアは、例えば、ファイル、データベース、フラッシュメモリロケーション、またはそれらの組合せを使用して実装され得る。
- コンフィグレーションデータストア：デバイスを初期デフォルト状態から特定の動作状態にするために必要なコンフィグレーションデータの完全なセットを保持するデータストア。
- メッセージ：セッションを介して送信されるプロトコル要素。メッセージは、整形式の XML 文書である。
- 通知：サーバによって特定のイベントが認識されたことを示す、サーバによって開始されるメッセージ。
- プロトコル操作：NETCONF プロトコル内で使用される特定のリモートプロシージャコール。
- リモートプロシージャコール (RPC)：<rpc> および <rpc-reply> メッセージを交換することによって実現される。
- ランニングコンフィグレーションデータストア：デバイス上で現在アクティブな完全なコンフィグレーションを保持するコンフィグレーションデータストア。ランニングコンフィグレーションデータストアは常に存在する。

- サーバ：クライアントによって呼び出されたプロトコル操作を実行する。さらに、サーバは、クライアントに通知を送信することができる。
- セッション：クライアントとサーバは、セキュアなコネクション型セッションを使用してメッセージを交換する。
- スタートアップコンフィグレーションデータストア：ブート時にデバイスによってロードされたコンフィグレーションを保持するコンフィグレーションデータストア。スタートアップコンフィグレーションデータストアとランニングコンフィグレーションデータストアを分離しているデバイスにのみ存在する。
- 状態データ：読み取り専用のステータス情報や収集された統計など、コンフィグレーションデータではないシステム上の追加データ。
- ユーザ：クライアントの認証 ID。クライアントの認証 ID は、一般に、NETCONF ユーザ名と呼ばれる。

1.2. プロトコル概観

NETCONF は、クライアントとサーバとの間の通信を容易にするために、シンプルな RPC ベースのメカニズムを使用する。クライアントは、通常、ネットワークマネージャの一部として実行されているスクリプトやアプリケーションである。サーバは、通常、ネットワークデバイスである。「デバイス」および「サーバ」という用語は、「クライアント」および「アプリケーション」と同様に、本文書では交換可能に使用される。

NETCONF セッションは、ネットワーク管理者やネットワークコンフィグレーションアプリケーションとネットワークデバイスとの間の論理接続である。デバイスは少なくとも 1 つの NETCONF セッションをサポートしなければならない (MUST) 。また、複数のセッションをサポートすべきである (SHOULD) 。グローバルコンフィグレーション属性は、許可されたセッション中に変更することができ、すべてのセッションに影響が表示される。セッション固有属性は、変更を行ったセッションにのみ影響する。

NETCONF は、図 1 に示すように、概念的に 4 つのレイヤに分割することができる。

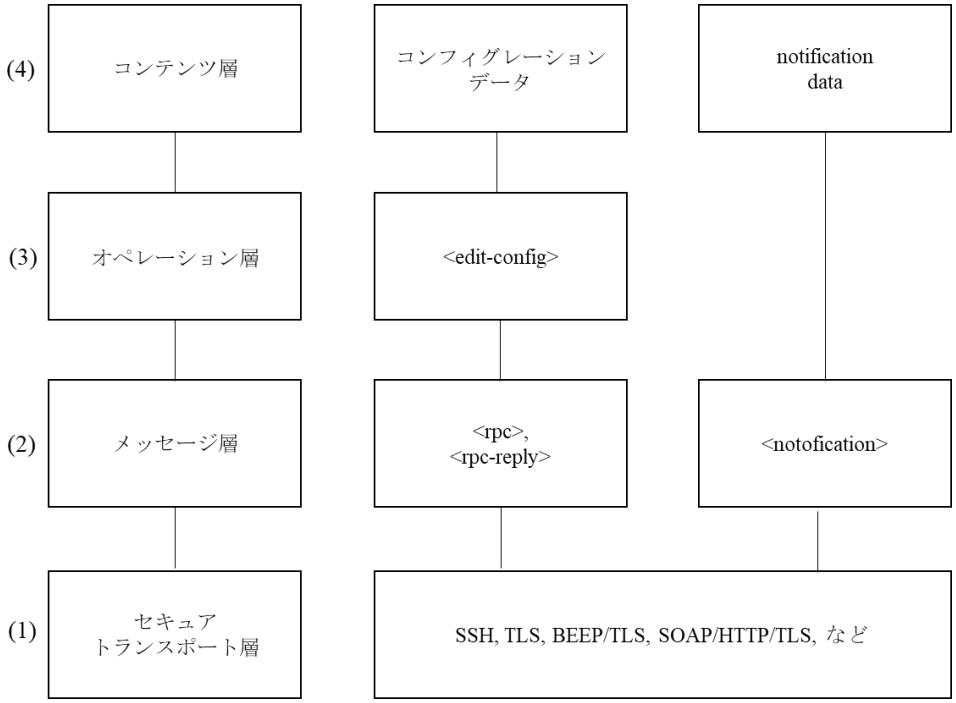

図 1: NETCONF プロトコルレイヤ

(1) セキュア・トランスポート層は、クライアントとサーバとの間の通信経路を提供する。NETCONF は、基本要件のセットを提供する任意のトランスポートプロトコル上に階層化することができる。セクション 2 では、これらの要件について説明する。

(2) メッセージ層は、RPC および通知をエンコードするための単純なトランスポート独立フレーム機構を提供する。セクション 4 は RPC メッセージを文書化し、[RFC5717] は通知を文書化する。

(3) オペレーション層は、XML 符号化パラメータを有する RPC メソッドとして呼び出される基本プロトコルオペレーションのセットを定義する。セクション 7 は、基本プロトコル操作のリストを詳述する。

(4) コンテンツ層は、この文書の範囲外である。NETCONF データモデルの標準化には別途取り組むことが予想される。

YANG データモデリング言語[RFC6020]は、NETCONF データモデルおよびプロトコル操作を指定するために開発され、図 1 のオペレーション層およびコンテンツ層をカバーしている。

1.3. ケイパビリティ

NETCONF ケイパビリティは、基本的な NETCONF 仕様を補う機能のセットである。ケイパビリティは、URI (uniform resource identifier) [RFC3986] によって識別される。

ケイパビリティは、デバイスの基本的な動作を増強し、追加の動作と内部動作で許可されるコンテンツの両方を記述する。クライアントは、サーバのケイパビリティを発見し、それらのケイパビリティによって定義された任意の追加動作、パラメータ、およびコンテンツを使用することができる。

ケイパビリティ定義は、1 つまたは複数の従属ケイパビリティに名前を付けることができる。ケイパビリティをサポートするために、サーバは、それが依存する任意のケイパビリティをサポートしなければならない (MUST) 。

セクション 8 では、クライアントがサーバのケイパビリティを発見することを可能にするケイパビリティ交換を定義する。また、セクション 8 では、この文書で定義されたケイパビリティのセットの一覧を掲載する。

追加のケイパビリティは、外部ドキュメントにおいていつでも定義することができ、ケイパビリティのセットを時間と共に拡張することができる。標準化団体は、標準化されたケイパビリティを定義することができ、実装では、独自のケイパビリティを定義することができる。ケイパビリティ URI は、命名衝突を避けるために、命名権限を十分に区別しなければならない (MUST) 。

1.4. コンフィグレーションデータと状態データの分離

実行中のシステムから取り出すことができる情報は、コンフィグレーションデータと状態データの 2 つのクラスに分けられる。コンフィグレーションデータは、システムをその初期デフォルト状態からその現在の状態に変換するのに必要な書き込み可能なデータセットである。状態データは、読み取り専用ステータス情報および収集された統計などの、コンフィグレーションデータではないシステム上の追加データである。デバイスがコンフィグレーション操作を実行しているとき、状態データが含まれていれば、いくつかの問題が発生してしまう。

- コンフィグレーションデータセットの比較は、異なる統計のような無関係なエントリによって支配される。
- 受信データは、読み取り専用データを書き込もうとする試みなど、無意味なリクエストを含むことができる。
- データセットが大きくなる。
- アーカイブされたデータには、読み取り専用データの値が含まれる可能性があり、アーカイブされたデータを復元するために必要な処理が複雑になる。

これらの問題に対処するために、NETCONF プロトコルは、コンフィグレーションデータと状態データの差を認識し、それぞれの動作を提供する。<get-config> 操作はコンフィグレーションデータのみを取得するが、<get> 操作はコンフィグレーションデータと状態データを取得する。

NETCONF プロトコルは、デバイスを所望の実行状態にするために必要な情報に焦点を合わせていることに留意されたい。他の重要な永続的データを含めることは、実装固有である。例えば、ユーザファイルおよびデータベースは、NETCONF プロトコルによってコンフィグレーションデータとして扱われない。

例えば、ユーザ認証データのローカルデータベースがデバイスに記憶される場合には、それがコンフィグレーションデータに含まれるかどうかは、処理系に依存する。

2. トランスポートプロトコルの要件

NETCONF は、RPC ベースの通信の仕組みを使用する。クライアントは、一連の 1 つ以上の RPC リクエストメッセージを送信し、これにより、サーバは、対応する一連の RPC レスポンスメッセージでレスポンスする。

NETCONF プロトコルは、必要な機能セットを提供する任意のトランスポートプロトコル上に階層化することができる。これは、特定のトランスポートプロトコルに拘束されるものではなく、マッピングが、特定のプロトコルを介してそれをどのように実施することができるかを定義することを可能にする。

トランスポートプロトコルは、セッションタイプ (クライアントまたはサーバ) を NETCONF プロトコル層に示すメカニズムを提供しなければならない (MUST) 。

このセクションでは、NETCONF のベースとなるトランスポートプロトコルに要求される特性を詳述する。

2.1. コネクション指向の操作

NETCONF は、コネクション指向であり、ピア間の永続的なコネクションを必要とする。このコネクションは、信頼できる、順序付けられたデータ配信を提供しなければならない。NETCONF 接続は、プロトコル操作間で長期にわたり持続する。

さらに、特定のコネクションのためにサーバからリクエストされたリソースは、コネクションが閉じたときに自動的に解放されなければならず、障害回復をより単純にし、より強固にしなければならない。例えば、ロックがクライアントによって獲得されると、そのロックは、それが明示的に解放されるか、またはコネクションが終了したとサーバが判断するまで持続する。クライアントがロックを保持している間に接続が終了した場合、サーバは任意で適切な回復を実行することができる。<lock> 操作については、セクション 7.5 で詳しく説明する。

2.2. 認証、インテグリティ、および機密性

NETCONF 接続は、認証、データ完全性、機密性、およびリプレイ保護を提供しなければならない (MUST) 。NETCONF は、このケイパビリティのために、トランスポートプロトコルに依存する。NETCONF ピアは、適切なレベルのセキュリティと機密性が、この文書とは独立して提供されると仮定する。例えば、基礎となるプロトコルに応じて、Transport Layer Security (TLS) [RFC5246] または Secure Shell (SSH) [RFC4251] を使用して接続を暗号化することができる。

NETCONF 接続は認証されなければならない (MUST) 。トランスポートプロトコルは、クライアントに対するサーバの認証、およびサーバに対するクライアントの認証を担当する。NETCONF ピアは、基礎となるトランスポートプロトコルが持つ十分に信頼できるメカニズムによって接続の認証情報が検証され、ピアの ID が十分に証明されていると仮定する。

NETCONF の 1 つの目標は、デバイスのネイティブ・インターフェースの機能に密接に従う、デバイスへのプログラマチック・インターフェースを提供することである。したがって、基礎となるプロトコルは、デバイス上で利用可能な既存の認証メカニズムを使用することが期待される。例えば、RADIUS [RFC2865] をサポートするデバイス上の NETCONF サーバは、NETCONF セッションを認証するために RADIUS の使用を許可することができる。

認証プロセスは、認証されたクライアントの ID をサーバに提供しなければならない (MUST) 。クライアントの認証 ID は、一般に、NETCONF ユーザ名と呼ばれる。ユーザ名は、[W3C.REC-xml-20001006] のセクション 2.2 からの「Char」プロダクションに一致する文字列である。ユーザ名を導出するために使用されるアルゴリズムは、トランスポートプロトコルに固有であり、さらに、トランスポートプロトコルによって使用される認証機構に固有である。トランスポートプロトコルは、他の NETCONF 層によって使用されるユーザ名を提供しなければならない (MUST) 。

付与されたクライアントのアクセス許可は、その NETCONF ユーザ名によって識別され、NETCONF サーバのコンフィグレーションの一部である。これらの許可は、NETCONF セッションの残りの間に実施されなければならない (MUST) 。アクセス制御がどのように構成されるかの詳細は、本文書の有効範囲外である。

2.3. 必須トランスポートプロトコル

NETCONF の実装は、SSH トランスポートプロトコルマッピング [RFC6242] をサポートしなければならない (MUST) 。

3. XML に関する考慮事項

XML は、NETCONF のエンコードフォーマットとして使用され、複雑な階層データを、従来のテキストツールと XML に固有のツールの両方を用いて読み取り、保存し、操作することができるテキストフォーマットで表現することを可能にする。

すべての NETCONF メッセージは、UTF-8 [RFC3629] でエンコードされた、整形式の XML でなければならない (MUST) 。ピアが <rpc>メッセージを受信し、それが整形式の XML ではないか、UTF-8 でエンコードされていない場合、"malformed-message" エラーでレスポンスすべきである (SHOULD) 。何らかの理由でレスポンスを送信できない場合、サーバはセッションを終了しなければならない (MUST) 。

NETCONF メッセージは、XML 宣言で始まってもよい (MAY) ([W3C.REC-xml-20001006]のセクション 2.8 を参照) 。

このセクションでは、NETCONF における XML 関連の少数の考慮事項について説明する。

3.1. ネームスペース

すべての NETCONF プロトコル要素は、以下の名前空間で定義される。

urn:ietf:params:xml:ns:netconf:base:1.0

NETCONF ケイパビリティ名は、URI [RFC3986] でなければならない (MUST) 。NETCONF ケイパビリティについては、セクション 8 で説明する。

3.2. Document Type Declarations (DTD)

DTD ([W3C.REC-xml-20001006] のセクション 2.8 を参照) は、NETCONF コンテンツに現れてはならない (MUST NOT) 。

4. RPC モデル

NETCONF プロトコルは、RPC ベースの通信モデルを使用する。NETCONF ピアは、<rpc> 要 素と <rpc-reply> 要素を使用して、トランスポートプロトコルに依存しない NETCONF リクエストとレスポンスのフレーム機構を提供する。

メッセージ層の RPC および XML エンコードについては、正式には、付録 B の XML スキーマで定義される。

4.1. <rpc> 要素

<rpc> 要素は、クライアントからサーバに送信される NETCONF リクエストを表すために使用される。

<rpc> 要素は必須属性 "message-id" を持ち、これは RPC の送信者が選んだ文字列で、単調増加する整数でエンコードする。RPC の受信側は、この文字列をデコードまたは解釈せず、単に、結果として生じる <rpc-reply> メッセージ内の "message-id" 属性として使用されるようにそれを保存する。送信者が文字列を変更せずに返したい場合、送信者は、[W3C.REC-xml-20001006] で定義されている XML

属性値正規化規則に従って、"message-id" 値が正規化されていることを確認しなければならない (MUST) 。例:

```
<rpc message-id="101"
       xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <some-method>
    <!-- method parameters here... -->
  </some-method>
</rpc>
```

<rpc> 要素に追加の属性が存在する場合、NETCONF ピアは <rpc-reply> 要素でそれらを変更せずに返さなければならない (MUST) 。これは、任意の「xmlns」属性を含む。

RPC の名前とパラメータは、<rpc> 要素の内容としてエンコードされる。RPC の名前は <rpc> 要素の内部の要素であり、任意のパラメータはこの要素の内部でエンコードされる。

次の例では、<my-own-method> と呼ばれるメソッドを呼び出す。このメソッドは、”14” という値を持つ <my-first-parameter> と ”fred” という値を持つ <another-parameter> という 2 つのパラメータからなる。

```
<rpc message-id="101"
       xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <my-own-method xmlns="http://example.net/me/my-own/1.0">
    <my-first-parameter>14</my-first-parameter>
    <another-parameter>fred</another-parameter>
  </my-own-method>
</rpc>
```

次の例では、<zip-code> パラメータが "27606-0100" の <rock-the-house> メソッドを呼び出す。

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <rock-the-house xmlns="http://example.net/rock/1.0">
    <zip-code>27606-0100</zip-code>
  </rock-the-house>
</rpc>
```

次の例では、パラメータを指定せずに NETCONF の <get> メソッドを呼び出す。

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

4.2. <rpc-reply> 要素

<rpc-reply> メッセージは<rpc>メッセージのレスポンスとして送信される。

<rpc-reply> 要素は必須属性 "message-id" を持ち、このレスポンスのもとになる <rpc> の "message-id" 属性の値と等しい。

また、NETCONF サーバは、<rpc-reply> 要素の <rpc> 要素に含まれる、変更されていない追加属性を返さなければならない。

レスポンスデータは、<rpc-reply> 要素に対する 1 つ以上の子要素としてエンコードされる。

例:

以下の<rpc>要素は、NETCONF の <get> メソッドを呼び出し、"user-id" と呼ばれる追加属性を含む。"user-id" 属性が NETCONF 名前空間にないことに注意する。返された <rpc-reply> 要素は、"user-id" 属性とリクエストされたコンテンツを返す。

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
     xmlns:ex="http://example.net/content/1.0"
     ex:user-id="fred">
  <get/>
```

```
</rpc>

<rpc-reply message-id="101"
      xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
      xmlns:ex="http://example.net/content/1.0"
      ex:user-id="fred">
  <data>
    <!-- contents here... -->
  </data>
</rpc-reply>
```

4.3. <rpc-error> 要素

<rpc-error> 要素は、<rpc> リクエストの処理中にエラーが発生した場合に、<rpc-reply> メッセージで送信される。

サーバが <rpc> リクエストの処理中に複数のエラーに遭遇した場合、<rpc-reply> は複数の <rpc-error> 要素を含んでもよい (MAY) 。ただし、リクエストに複数のエラーが含まれている場合、サーバは複数の <rpc-error> 要素を検出または報告する必要はない。サーバは、特定のシーケンスにおける特定のエラー状態をチェックする必要はない。処理中にエラー状態が発生した場合、サーバは <rpc-error> 要素を返さなければならない (MUST) 。

サーバは、<rpc-error> 要素内で、クライアントが十分なアクセス権を持っていないアプリケーションレベルまたはデータモデル固有のエラー情報を返してはならない (MUST NOT) 。

<rpc-error> 要素には、以下の情報が含まれる 。

error-type : エラーが発生した概念層を定義する。下記からひとつ。

- transport (セキュア・トランスポート層)
- rpc (メッセージ層)
- protocol (オペレーション層)
- application (コンテンツ層)

error-tag : エラー状態を識別する文字列を含む。許容される値については、付録 A を参照されたい。

error-severity : デバイスによって決定された、エラーの重大度を識別する文字列が格納される。下記からひとつ。

- error
- warning

「warning」を利用する <error-tag> の値は、この文書で定義されていないことに注意すること。このフィールドは将来の使用のために予約されている。

error-app-tag : データモデル固有または実装固有のエラー条件が存在する場合、それを識別する文字列が格納される。この要素は、特定のエラー状態に適切なアプリケーションエラータグを関連付けることができない場合には存在しない。データモデル固有と実装固有の両方の error-app-tag が存在する場合、サーバによってデータモデル固有の値が、使用されなければならない (MUST) 。

error-path : 絶対表記の XPath 式 [W3C.REC-xpath-19991116] で、特定の <rpc-error> 要素で報告されているエラーに関連付けられているノードへの要素パスを指定する。この要素は、適切なペイロード要素またはデータストアノードが特定のエラー状態に関連付けられない場合には、存在しない。

XPath 式は、以下のコンテキストで解釈される。

- 名前空間宣言の集合は、<rpc-error> 要素の有効範囲内のものである。
- 変数バインディングのセットは空である。
- 関数ライブラリは、コア関数ライブラリである。

コンテキストノードは、報告されるエラーに関連付けられたノードに依存する。

- ペイロード要素をエラーに関連付けることができる場合、コンテキストノードは、rpc リクエストの文書ノード (すなわち <rpc> 要素) である。
- そうでない場合、コンテキストノードは、すべてのデータモデルのルート、すなわち、すべてのデータモデルからの最上位ノードを子として有するノードである。

error-message : エラー状態を説明するための、人間が理解するために適した文字列を含む。この要素は、特定のエラー状態に対して適切なメッセージが提供されない場合には存在しない。この要素は、[W3C.REC-xml-20001006] で定義され、[RFC3470] で議論されている "xml:lang" 属性を含むべきである。

error-info : プロトコルまたはデータモデル固有のエラー内容が含まれる。この要素は、そのようなエラー内容が特定のエラー状態に対して提供されない場合には、存在しない。付録 A のリストは、各エラーのためのいくつかの必須のエラー情報内容を定義する。プロトコルで規定された内容の後、データモデル定義は、特定のアプリケーション層エラー情報が error-info コンテナに含まれることを規定してもよい (MAY) 。実装は、拡張された、および/または実装固有のデバッグ情報を提供するために、追加の要素を含んでもよい (MAY) 。

付録 A は、標準的な NETCONF エラーを列挙する。

例：<rpc> 要素が "message-id" 属性なしで受信された場合、エラーが返される。この場合にのみ、NETCONF ピアが <rpc-reply> 要素の "message-id" 属性を省略してもよいことに注意する。

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get-config>
    <source>
      <running/>
    </source>
  </get-config>
</rpc>

<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <rpc-error>
    <error-type>rpc</error-type>
    <error-tag>missing-attribute</error-tag>
    <error-severity>error</error-severity>
    <error-info>
      <bad-attribute>message-id</bad-attribute>
      <bad-element>rpc</bad-element>
    </error-info>
  </rpc-error>
</rpc-reply>
```

以下の <rpc-reply> は、複数の <rpc-error> 要素を返す場合を示している。

このセクションの例で使用されるデータモデルは、<name> 要素を使用して、<interface> 要素の複数のインスタンスを区別することに注意してください。

```
<rpc-reply message-id="101"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
  xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
  <rpc-error>
    <error-type>application</error-type>
    <error-tag>invalid-value</error-tag>
    <error-severity>error</error-severity>
    <error-path xmlns:t="http://example.com/schema/1.2/config">
      /t:top/t:interface[t:name="Ethernet0/0"]/t:mtu
    </error-path>
    <error-message xml:lang="en">
      MTU value 25000 is not within range 256..9192
    </error-message>
  </rpc-error>
  <rpc-error>
    <error-type>application</error-type>
    <error-tag>invalid-value</error-tag>
    <error-severity>error</error-severity>
    <error-path xmlns:t="http://example.com/schema/1.2/config">
      /t:top/t:interface[t:name="Ethernet1/0"]/t:address/t:name
    </error-path>
    <error-message xml:lang="en">
      Invalid IP address for interface Ethernet1/0
    </error-message>
  </rpc-error>
</rpc-reply>
```

4.4. <ok>要素

<rpc> リクエストの処理中にエラーまたは警告が発生せず、操作からデータが返されなかった場合、<ok> 要素は <rpc-reply> メッセージで送信される。例:

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

4.5. パイプライン

NETCONF の <rpc> リクエストは、管理対象デバイスによって逐次処理されなければならない (MUST) 。追加の <rpc> リクエストは、前のリクエストが完了する前に送信されてもよい (MAY) 。管理対象デバイスは、リクエストが受信された順序でのみレスポンスを送信しなければならない (MUST) 。

5. コンフィグレーションモデル

NETCONF は、オペレーションの初期セットと、ベースを拡張するのに使用できる多数のケイパビリティを提供する。NETCONF ピアは、セクション 8.1 で説明されるようにセッションが開始されると、デバイスケイパビリティを交換する。

5.1. コンフィグレーションデータストア

NETCONF は、1 つまたは複数のコンフィグレーションデータストアの存在を定義し、それらに対するコンフィグレーション操作を可能にする。コンフィグレーションデータストアは、デバイスをその初期デフォルト状態から所望の動作状態にするために必要とされるコンフィグレーションデータの完全なセットとして定義される。コンフィグレーションデータストアは、状態データまたは実行コマンドを含まない。

ランニングコンフィグレーションデータストアは、ネットワークデバイス上で現在アクティブな完全なコンフィグレーションを保持する。このタイプのコンフィグレーションデータストアはデバイス上にひとつだけ存在し、また、常に存在する。NETCONF プロトコル操作は、<running> 要素を使用してこのデータストアを参照する。

基本モデルには <running> コンフィグレーションデータストアのみが存在する。追加のコンフィグレーションデータストアは、ケイパビリティによって定義されてもよい (MAY) 。このようなコンフィグレーションデータストアは、ケイパビリティをアドバタイズするデバイスでのみ使用可能である。

セクション 8.3 および 8.7 のケイパビリティは、それぞれ <candidate> および <startup> コンフィグレーションデータストアを定義する。

5.2. データモデリング

データモデリングおよびコンテンツの問題は、NETCONF プロトコルの範囲外である。デバイスのデータモデルはアプリケーションに周知であり、双方の当事者は、データのレイアウト、包含、キーイング、ルックアップ、置換、および管理などの問題、ならびにデータモデルによって課される任意の他の制約を認識していると仮定する。

NETCONF は、デバイスのデータモデルに固有の <config> 要素内の設定データを搬送する。プロトコルは、その要素の内容を不透明なデータとして扱う。デバイスは、ケイパビリティを使用して、デバイスが実装するデータモデルのセットをアナウンスする。ケイパビリティ定義は、データモデルによって課される動作および制約を詳述する。

デバイスおよびマネージャは、標準データモデルと独自データモデルの両方を含む複数のデータモデルをサポートすることができる。

6. サブツリーフィルタリング

6.1. 概要

XML サブツリーフィルタリングは、アプリケーションが <get> または <get-config> 操作のために <rpc-reply> に含める特定の XML サブツリーを選択できるようにするメカニズムである。包含、単純なコンテンツの完全一致、および選択のためのフィルタの小さなセットが提供され、これは、いくつかの有用であるが、非常に限定された選択メカニズムも可能にする。サーバは、処理中にデータモデル固有の方式 (semantics) を利用する必要がなく、単純で集中化された実装戦略を可能にする。

概念的には、サブツリーフィルタは、フィルタ選択基準 (criteria) を表すゼロ以上の要素サブツリーから構成される。サブツリーおよびルートへの要素のパス (path) がフィルタ出力に含まれるかどうかが判定するために、サブツリー内の各包含レベルで、兄弟 (sibling) ノードのセットがサーバによって論理的に処理さる。

サブツリーフィルタで指定された各ノードは、包含フィルタを表す。サーバ上の指定されたデータストア内の基礎となるデータモデル内の関連ノードのみが、フィルタによって選択される。フィルタ絶対パス名が <filter> より下のレイヤから始まるように調整されることを除いて、フィルタデータに与えられた要素の選択基準 (criteria) および階層に一致する場合、ノードが選択される。

応答メッセージは、フィルタによって選択されたサブツリーのみを含む。特定の選択されたサブツリー内の要求に存在した任意の選択基準も、応答に含まれる。フィルタにおいてリーフノードとして表現されるいくつかの要素は、フィルタ出力において拡張される(すなわち、サブツリーが含まれる)ことに留意されたい。要求に同じデータを選択する複数のフィルタサブツリー式が含まれている場合、応答で特定のデータインスタンスが重複することはない。

6.2. サブツリー・フィルター・コンポーネント

サブツリーフィルタは、XML 要素とその XML 属性から構成される。サブツリーフィルタには、以下の 5 つのタイプのコンポーネントが存在する。

- 名前空間の選択
- 属性一致式
- 包含ノード
- 選択ノード
- コンテンツ一致ノード

6.2.1. 名前空間の選択

<filter> 要素内の特定のノードに関連付けられた XML 名前空間が、基礎となるデータモデルと同じである場合、名前空間は (フィルタの目的で) 一致するとみなされる。名前空間の選択を単独で使用することはできないことに注意してください。フィルタ出力に含まれる要素があれば、少なくとも 1 つの要素をフィルタに指定しなければならない (MUST) 。

XML 名前空間ワイルドカードメカニズムは、サブツリーフィルタリングのために定義される。<filter> 要素内の要素が名前空間によって修飾されていない (例えば、xmlns="") 場合、サーバは、そのサブツリーフィルタノードを処理するときに、それがサポートするすべての XML 名前空間を評価しなければならない (MUST) 。このワイルドカードメカニズムは、XML 属性には適用できない。

フィルタ要素を基礎となるデータモデル内の要素と比較する場合、修飾された名前空間の接頭部値は関連しないことに注意してください。

例:

```
<filter type="subtree">
  <top xmlns="http://example.com/schema/1.2/config"/>
</filter>
```

この例では、<top> 要素は選択ノードであり、"http://example.com/schema/1.2/config" 名前空間内のこのノードと (基礎となるデータモデルからの) 任意の子ノードのみがフィルタ出力に含まれる。

6.2.2. 属性一致式

サブツリーフィルタに現れる属性は、「属性一致式」の一部である。任意の数の (非修飾または修飾の) XML 属性が、任意のタイプのフィルタノードに存在してもよい (MAY) 。そのノードに通常適用可能な選択基準に加えて、選択されたデータは、そのノードで指定されたすべての属性について、一致する値を持たなければならない (MUST) 。要素が指定された属性を含むように定義されていない場合、フィルタ出力では選択されない。

例:

```
<filter type="subtree">
  <t:top xmlns:t="http://example.com/schema/1.2/config">
    <t:interfaces>
      <t:interface t:ifName="eth0"/>
    </t:interfaces>
  </t:top>
</filter>
```

この例では、<top> および <interfaces> 要素は包含ノードであり、<interface> 要素は選択ノードであり、「ifName」は属性一致式である。「http://example.com/schema/1.2/config」名前空間内の「interface」ノードのみが、「eth0」を持つ「ifName」属性を持ち、「top」ノード内の「interface」ノード内に存在する。

6.2.3. 包含ノード

サブツリーフィルタ内に子要素を含むノードは、「包含ノード」と呼ばれる。各子要素は、別の包含ノードを含む、任意のタイプのノ

ードとすることができる。サブツリーフィルタで指定された各包含ノードについて、指定された名前空間、要素階層、および任意の属性一致式に正確に一致するすべてのデータ・モデル・インスタンスが、フィルタ出力に組み込まれる。

例:

```
<filter type="subtree">
  <top xmlns="http://example.com/schema/1.2/config">
    <users/>
  </top>
</filter>
```

この例では、<top> 要素は包含ノードである。

6.2.4. 選択ノード

フィルタ内の空のリーフノードは、「選択ノード」と呼ばれ、基礎となるデータモデル上の「明示的選択」フィルタを表す。兄弟ノードのセット内に任意の選択ノードが存在すると、フィルタは、指定されたサブツリーを選択し、基礎となるデータモデル内の兄弟ノードのセット全体の自動選択を抑制する。フィルタリングのために、空のリーフノードは、空のタグ (例えば、<foo/>) または明示的な開始タグおよび終了タグ (例えば、<foo> </foo>) のいずれかで宣言することができる。空白文字はこの形式では無視される。

例:

```
<filter type="subtree">
  <top xmlns="http://example.com/schema/1.2/config">
    <users/>
  </top>
</filter>
```

この例では、<top> 要素は包含ノードであり、<users> 要素は選択ノードである。設定データストアのルートである <top> 要素内に存在する "http://example.com/schema/1.2/config" 名前空間内の "users" ノードのみがフィルタ出力に含まれる。

6.2.5. コンテンツ一致ノード

単純なコンテンツを含むリーフノードは、「コンテンツ一致ノード」と呼ばれる。これは、フィルタ出力のために、その兄弟ノードのいくつかまたはすべてを選択するために使用され、リーフノード要素コンテンツ上の完全一致フィルタを表す。コンテンツ一致ノードには、以下の制約が適用される。

- コンテンツ一致ノードは、ネストされた要素を含んではならない (MUST NOT) 。
- 複数のコンテンツ一致ノード (すなわち、兄弟ノード) は、「AND」式で論理的に結合される。
- 混合コンテンツのフィルタリングはサポートされていない。
- リストコンテンツのフィルタリングはサポートされていない。
- 空白のみのコンテンツのフィルタリングはサポートされていない。
- コンテンツ一致ノードは、空白以外の文字を含まなければならない (MUST) 。空の要素 (例えば、<foo></foo>) は、選択ノード (例えば、<foo/>) として解釈される。
- 先頭および末尾の空白文字は無視されますが、テキスト文字ブロック内の空白文字は無視または変更されない。

サブツリーフィルタ式内のすべての指定された兄弟コンテンツ一致ノードが「真」である場合、フィルタ出力ノードは以下のように選択される。

- 兄弟セット内の各コンテンツ一致ノードは、フィルタ出力に含まれる。
- 任意の包含ノードが兄弟セットに存在する場合、それらはさらに処理され、ネストされたフィルタ基準も満たされる場合に含まれる。
- 任意の選択ノードが兄弟セットに存在する場合、それらのすべてがフィルタ出力に含まれる。
- 選択ノードの任意の兄弟ノードが、概念データ構造(例えば、リストキーリーフ)のインスタンス識別子コンポーネントである場合、それらもフィルタ出力に含めることができる (MAY) 。
- そうでない場合(すなわち、フィルタ兄弟セット内に選択ノードまたは包含ノードがない場合)、基礎となるデータモデル内のこのレベルで定義されたすべてのノード (および、もしあれば、それらのサブツリー) がフィルタ出力内で返される。

兄弟コンテンツ一致ノード・テストのいずれかが「偽」である場合、その兄弟セットに対してそれ以上のフィルタ処理は実行されず、コンテンツ一致ノードを含む兄弟サブツリーのいずれもフィルタによって選択されない。

例:

```
<filter type="subtree">
  <top xmlns="http://example.com/schema/1.2/config">
    <users>
      <user>
        <name>fred</name>
      </user>
    </users>
  </top>
</filter>
```

この例では、<users> ノードと <user> ノードは両方とも包含ノードであり、<name> はコンテンツ一致ノードである。<name> の兄弟ノードが指定されていない (したがって包含ノードや選択ノードが指定されていない) ため、<name> の兄弟ノードはすべてフィルタ出力に返される。要素の段階的組織構造に一致し、<name> 要素が "fred" に等しい "http://example.com/schema/1.2/config" 名前空間内の "ユーザ" ノードのみがフィルタ出力に含まれる。

6.3. サブツリーフィルタ処理

フィルタ出力 (選択されたノードのセット) は、最初は空である。

各サブツリーフィルタは、フィルタ出力において (すべての子ノードと共に) 選択されるデータモデルの部分を表す、1 つまたは複数のデータモデルフラグメントを含むことができる。

各サブツリー・データ・フラグメントは、サーバによって、サーバによってサポートされる内部データモデルと比較される。サブツリー・データ・フラグメント・フィルタ全体 (フィルタで指定されたルートから最も内側の要素まで) が、サポートされているデータモデルの対応する部分と正確に一致する場合、そのノードとそのすべての子ノードが結果データに含まれる。

サーバは、ルートから始まりリーフノードまで、同じ親ノード (兄弟セット) を有するすべてのノードを一緒に処理する。フィルタ内のルート要素は、共通の親を持たない場合であっても、同じ兄弟セット内で考慮される (同じ名前空間内にあると仮定する) 。

各兄弟セットについて、サーバは、どのノードがフィルタ出力に含まれる (または潜在的に含まれる) か、およびどの兄弟サブツリーがフィルタ出力から除外される (枝刈りされる) かを決定する。サーバは、まず、兄弟セット内にどのタイプのノードが存在するかを判定し、そのタイプに関する規則に従ってノードを処理する。兄弟セット内のいずれかのノードが選択された場合、プロセスは、各選択されたノードの兄弟セットに再帰的に適用される。このアルゴリズムは、フィルタ内で指定されたすべてのサブツリー内のすべての兄弟セットが処理されるまで続く。

6.4. サブツリーのフィルタリング例

6.4.1. フィルタなし

<get> 操作でフィルタを除外すると、データモデル全体が返される。

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

<rpc-reply message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <data>
    <!-- ... entire set of data returned ... -->
  </data>
</rpc-reply>
```

6.4.2. 空フィルタ

空のフィルタは、コンテンツ一致または選択ノードが存在しないので、何も選択しない。これはエラーではない。これらの例で使用される <filter> 要素の「type」属性については、7.1 セクションでさらに説明する。

```
<rpc message-id="101"
       xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get>
    <filter type="subtree">
    </filter>
  </get>
</rpc>
<rpc-reply message-id="101"
       xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <data>
  </data>
</rpc-reply>
```

6.4.3. 完全な <users>サブツリーを選択する。

この例のフィルタは、1 つの選択ノード (<users>) を含むので、そのサブツリーだけがフィルタによって選択される。この例は、以下のほとんどのフィルタ例で、完全に入力された <users> データモデルを表す。実際のデータモデルでは、<company-info> は、特定のホストまたはネットワークのユーザのリストとともに返される可能性は低い。

メモ: このドキュメントで使用されているフィルタおよび設定の例は、名前空間「http://example.com/schema/1.2/config」に表示される。この名前空間のルート要素は <top> である。<top> 要素とその子孫は、コンフィグレーションデータモデルの例のみを表す。

```
<rpc message-id="101"
       xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get-config>
    <source>
      <running/>
    </source>
    <filter type="subtree">
      <top xmlns="http://example.com/schema/1.2/config">
        <users/>
      </top>
    </filter>
  </get-config>
</rpc>

<rpc-reply message-id="101"
       xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <data>
    <top xmlns="http://example.com/schema/1.2/config">
      <users>
        <user>
          <name>root</name>
          <type>superuser</type>
          <full-name>Charlie Root</full-name>
          <company-info>
            <dept>1</dept>
            <id>1</id>
          </company-info>
        </user>
        <user>
          <name>fred</name>
          <type>admin</type>
          <full-name>Fred Flintstone</full-name>
          <company-info>
            <dept>2</dept>
```

```
            <id>2</id>
          </company-info>
        </user>
        <user>
          <name>barney</name>
          <type>admin</type>
          <full-name>Barney Rubble</full-name>
          <company-info>
            <dept>2</dept>
            <id>3</id>
          </company-info>
        </user>
      </users>
    </top>
  </data>
</rpc-reply>
```

以下のフィルタ要求は、コンテナ <users> が 1 つの子要素 (<user>) を定義しているためにのみ、同じ結果を生成する。

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get-config>
    <source>
      <running/>
    </source>
    <filter type="subtree">
      <top xmlns="http://example.com/schema/1.2/config">
        <users>
          <user/>
        </users>
      </top>
    </filter>
  </get-config>
</rpc>
```

6.4.4. <users>サブツリー内のすべての<name>要素を選択する。(選択ノードの例)

このフィルタには、2 つの包含ノード (<users>、<user>) と 1 つの選択ノード (<name>) が含まれている。同じ兄弟セット内の <name> 要素のすべてのインスタンスがフィルタ出力で選択される。クライアントは、<name> がこの特定のデータ構造においてインスタンス識別子として使用されることを知る必要があるかもしれないが、サーバは、要求を処理するためにそのメタデータを知る必要はない。

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get-config>
    <source>
      <running/>
    </source>
    <filter type="subtree">
      <top xmlns="http://example.com/schema/1.2/config">
        <users>
          <user>
            <name/>
          </user>
        </users>
      </top>
```

```
      </filter>
    </get-config>
  </rpc>

  <rpc-reply message-id="101"
       xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <data>
      <top xmlns="http://example.com/schema/1.2/config">
        <users>
          <user>
            <name>root</name>
          </user>
          <user>
            <name>fred</name>
          </user>
          <user>
            <name>barney</name>
          </user>
        </users>
      </top>
    </data>
  </rpc-reply>
```

6.4.5. １つの特定の <user> エントリ（コンテンツ一致ノードの例）

このフィルタには、2 つの包含ノード (<users>、<user>) と 1 つのコンテンツ一致ノード (<name>) が含まれている。<name> の値が「fred」に等しい <name> を含む兄弟セットのすべてのインスタンスが、フィルタ出力で選択される。

```
  <rpc message-id="101"
       xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <get-config>
      <source>
        <running/>
      </source>
      <filter type="subtree">
        <top xmlns="http://example.com/schema/1.2/config">
          <users>
            <user>
              <name>fred</name>
            </user>
          </users>
        </top>
      </filter>
    </get-config>
  </rpc>

  <rpc-reply message-id="101"
       xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <data>
      <top xmlns="http://example.com/schema/1.2/config">
        <users>
          <user>
            <name>fred</name>
            <type>admin</type>
            <full-name>Fred Flintstone</full-name>
```

```
            <company-info>
              <dept>2</dept>
              <id>2</id>
            </company-info>
          </user>
        </users>
      </top>
    </data>
  </rpc-reply>
```

6.4.6. 特定の<user>エントリからの特定の要素（コンテンツ一致ノードと選択ノードの複合例）

このフィルタには、2 つの包含ノード (<users>、<user>) 、1 つのコンテンツ一致ノード (<name>) 、および 2 つの選択ノード (<type>、<full-name>) が含まれてい。<name> の値が「fred」に等しい <name> を含む同じ兄弟セット内の <type> 要素および <full-name> 要素のすべてのインスタンスがフィルタ出力で選択される。兄弟セットに選択ノードが含まれているため、<company-info> 要素は含まれない。

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get-config>
    <source>
      <running/>
    </source>
    <filter type="subtree">
      <top xmlns="http://example.com/schema/1.2/config">
        <users>
          <user>
            <name>fred</name>
            <type/>
            <full-name/>
          </user>
        </users>
      </top>
    </filter>
  </get-config>
</rpc>

<rpc-reply message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <data>
    <top xmlns="http://example.com/schema/1.2/config">
      <users>
        <user>
          <name>fred</name>
          <type>admin</type>
          <full-name>Fred Flintstone</full-name>
        </user>
      </users>
    </top>
  </data>
</rpc-reply>
```

6.4.7. 複数のサブツリー

このフィルタは、3 つのサブツリー (name=root、fred、barney) を含む。

「root」サブツリーフィルタは、2 つの包含ノード (<users>、<user>) 、1 つのコンテンツマッチノード (<name>) 、および 1 つの選

択ノード (<company-info>) を含む。サブツリー選択基準が満たされ、「ルート」の company-info サブツリーのみがフィルタ出力で選択される。

「fred」サブツリーフィルタは、3 つの包含ノード (<users>、<user>、<company-info>) 、1 つのコンテンツ一致ノード (<name>) 、および 1 つの選択ノード (<id>) を含む。サブツリー選択基準が満たされ、「fred」の company-info サブツリー内の <id> 要素だけがフィルタ出力で選択される。

「barney」サブツリーフィルタは、3 つの包含ノード (<users>、<user>、<company-info>) 、2 つのコンテンツ一致ノード (<name>、<type>) 、および 1 つの選択ノード (<dept>) を含む。ユーザ「barney」が「スーパーユーザ」ではないため、サブツリー選択基準は満たされず、「barney」のサブツリー全体 (その親 <user> エントリを含む) はフィルタ出力から除外される。

```
<rpc message-id="101"
       xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get-config>
    <source>
      <running/>
    </source>
    <filter type="subtree">
      <top xmlns="http://example.com/schema/1.2/config">
        <users>
          <user>
            <name>root</name>
            <company-info/>
          </user>
          <user>
            <name>fred</name>
            <company-info>
              <id/>
            </company-info>
          </user>
          <user>
            <name>barney</name>
            <type>superuser</type>
            <company-info>
              <dept/>
            </company-info>
          </user>
        </users>
      </top>
    </filter>
  </get-config>
</rpc>

<rpc-reply message-id="101"
           xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <data>
    <top xmlns="http://example.com/schema/1.2/config">
      <users>
        <user>
          <name>root</name>
          <company-info>
            <dept>1</dept>
            <id>1</id>
          </company-info>
        </user>
        <user>
```

```
            <name>fred</name>
            <company-info>
              <id>2</id>
            </company-info>
          </user>
        </users>
      </top>
    </data>
  </rpc-reply>
```

6.4.8. 属性名を持つ要素（属性一致式の例）

この例では、フィルタは、1つの包含ノード (<interfaces>) 、1つの属性一致式 (「ifName」) 、および1つの選択ノード (<interface>) を含む。「eth0」に等しい「ifName」属性を有する <interface> サブツリーのすべてのインスタンスが、フィルタ出力において選択される。フィルタデータ要素および属性は、「ifName」属性が「schema/1.2」名前空間の一部と見なされないため、修飾される。

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get>
    <filter type="subtree">
      <t:top xmlns:t="http://example.com/schema/1.2/stats">
        <t:interfaces>
          <t:interface t:ifName="eth0"/>
        </t:interfaces>
      </t:top>
    </filter>
  </get>
</rpc>

<rpc-reply message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <data>
    <t:top xmlns:t="http://example.com/schema/1.2/stats">
      <t:interfaces>
        <t:interface t:ifName="eth0">
          <t:ifInOctets>45621</t:ifInOctets>
          <t:ifOutOctets>774344</t:ifOutOctets>
        </t:interface>
      </t:interfaces>
    </t:top>
  </data>
</rpc-reply>
```

「ifName」が属性ではなく子ノードである場合、以下の要求は同様の結果を生成する。

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get>
    <filter type="subtree">
      <top xmlns="http://example.com/schema/1.2/stats">
        <interfaces>
          <interface>
            <ifName>eth0</ifName>
          </interface>
        </interfaces>
```

```
        </top>
      </filter>
    </get>
  </rpc>
```

7. プロトコル動作

NETCONF プロトコルは、デバイスコンフィグレーションを管理し、デバイス状態情報を取り出すための低レベル動作の小さなセットを提供する。基本プロトコルは、コンフィグレーションデータストアの取得、定義、コピー、削除の動作を提供する。デバイスによって通知された機能に基づいて、追加の動作が提供される。

基本プロトコルは、以下のプロトコル動作を含む。

- get
- get-config
- edit-config
- copy-config
- delete-config
- lock
- unlock
- close-session
- kill-session

プロトコル動作は、「動作がサポートされていない」を含む様々な理由で失敗する可能性がある。イニシエータは、操作が常に成功することを想定すべきではない (SHOULD NOT) 。任意の RPC 応答の戻り値は、エラー応答についてチェックされるべきである (SHOULD) 。

プロトコル操作の構文および XML エンコーディングは、付録 C の YANG モジュールで正式に定義されている。以下のセクションでは、各プロトコル操作のセマンティクスについて説明する。

7.1. <get-config>

説明：

指定されたコンフィグレーションデータストアのすべてまたは一部を取得する。

パラメータ:

source : <running/> など、照会されるコンフィグレーションデータストアの名前。

filter : このパラメータは、取得するデバイスコンフィグレーションデータストアの部分を識別する。このパラメータが存在しない場合、コンフィグレーション全体が戻される。<filter> は、オプションで "type" 属性を含んでもよい。この属性は、<filter> 内で使用されるフィルタリング構文のタイプを示す。NETCONF におけるデフォルトフィルタリングメカニズムは、サブツリーフィルタリングと呼ばれ、セクション 6 で説明される。「サブツリー」の値は、このタイプのフィルタリングを明示的に識別する。NETCONF ピアが:xpath 機能 (セクション 8.9) をサポートしている場合、<filter> の "select" 属性に XPath 式が含まれていることを示すために、"xpath" の値を使用してもよい。

肯定応答：デバイスが要求を満たすことができる場合、サーバは、クエリの結果とともに <data> を含む <rpc-reply> を送信する。
否定応答： <rpc-error> は、何らかの理由で要求を完了できない場合に <rpc-reply> に含まれる。

例： <users> サブツリー全体を取得するには：

```
<rpc message-id="101"
       xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get-config>
    <source>
      <running/>
    </source>
    <filter type="subtree">
      <top xmlns="http://example.com/schema/1.2/config">
        <users/>
      </top>
```

```
      </filter>
    </get-config>
  </rpc>

  <rpc-reply message-id="101"
       xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <data>
      <top xmlns="http://example.com/schema/1.2/config">
        <users>
          <user>
            <name>root</name>
            <type>superuser</type>
            <full-name>Charlie Root</full-name>
            <company-info>
              <dept>1</dept>
              <id>1</id>
            </company-info>
          </user>
          <!-- additional <user> elements appear here... -->
        </users>
      </top>
    </data>
  </rpc-reply>
```

セクション 6 は、サブツリーフィルタリングの追加の例を含む。

7.2. <edit-config>

説明：

<edit-config> は、指定されたコンフィグレーションのすべてまたは一部を、指定されたターゲットコンフィグレーションデータストアにロードする。この操作により、新しいコンフィグレーションを、ローカル・ファイル、リモート・ファイル、またはインラインを使用するなど、いくつかの方法で表すことができる。ターゲットコンフィグレーションデータストアが存在しない場合、ターゲットコンフィグレーションデータストアが作成される。

NETCONF ピアが:url 機能 (セクション 8.8) をサポートしている場合、<config> パラメータの代わりに <url> を表示できる。デバイスは、ソースコンフィグレーションおよびターゲットコンフィグレーションを分析し、要求された変更を実行する。ターゲット設定は、<copy-config> メッセージのように、必ずしも置き換える必要はない。その代わりに、ターゲットコンフィグレーションは、ソースのデータおよび要求された動作に従って変更される。

<edit-config> が、基礎となるデータモデル内の同じ概念ノードに適用される複数のサブオペレーションを含む場合、オペレーションの結果は未定義である (すなわち、NETCONF プロトコルの範囲外である) 。

属性：

operation : <config> サブツリーは、セクション 3.1 で定義された NETCONF 名前空間に属する "operation" 属性を含んでもよい。この属性は、操作を実行するためのコンフィグレーション内のポイントを識別し、<config> サブツリー全体の複数の要素に現れる可能性がある。「operation」属性が指定されていない場合、コンフィグレーションはコンフィグレーションデータストアにマージされる。「operation」属性は、以下の値のうちの 1 つを有する。

merge : この属性を含む要素によって識別されるコンフィグレーションデータは、<target> パラメータによって識別されるコンフィグレーションデータストア内の対応するレベルのコンフィグレーションとマージされる。これはデフォルトの動作である。

replace : この属性を含む要素によって識別されるコンフィグレーションデータは、<target> パラメータによって識別されるコンフィグレーションデータストア内の関連するコンフィグレーションを置き換える。そのようなコンフィグレーションデータがコンフィグレーションデータストアに存在しない場合、そのコンフィグレーションデータが作成される。ターゲット設定全体を置き換える <copy-config> とは異なり、<config> パラメータに実際に存在する設定のみが影響を受ける。

create : この属性を含む要素によって識別されるコンフィグレーションデータは、コンフィグレーションデータがコンフィグレーションデータストア内にまだ存在しない場合に限り、コンフィグレーションに追加される。コンフィグレーションデータが存在する場合、<rpc-error> は、<error-tag> の値が "data-exists" で返される。

delete : この属性を含む要素によって識別されるコンフィグレーションデータは、コンフィグレーションデータが現在コンフィグ

レーションデータストア内に存在する場合に限り、コンフィグレーションから削除される。コンフィグレーションデータが存在しない場合、<rpc-error> は、<error-tag> の値が "data-missing" で返される。

remove : コンフィグレーションデータがコンフィグレーションデータストアに存在する場合、この属性を含む要素によって識別されるコンフィグレーションデータはコンフィグレーションから削除される。コンフィグレーションデータが存在しない場合、「削除」動作はサーバに無視される。

パラメータ：

target: <running/> や <candidate/> など、編集するコンフィグレーションデータストアの名前。

default-operation : この <edit-config> リクエストのデフォルトの操作 (「operation」属性で説明されている) を選択する。<default-operation> パラメータのデフォルト値は"merge"である。<default-operation> パラメータはオプションですが、指定されている場合は、次のいずれかの値になる。

merge : <config> パラメータ内の設定データが、ターゲットデータストア内の対応するレベルの設定とマージされる。これはデフォルトの動作である。

replace : <config> パラメータ内の設定データは、ターゲットデータストア内の設定を完全に置き換える。これは、以前に保存した設定データをロードする場合に便利である。

none : 着信設定データが「operation」属性を使用して異なる操作を要求しない、および使用するまで、ターゲットデータストアは <config> パラメータの設定の影響を受けない。<config> パラメータの設定に、ターゲットデータストアに対応するレベルがないデータが含まれている場合、<rpc-error> に data-missing の <error-tag> の値が返される。「none」を使用することにより、「delete」のような操作で、削除する要素の親階層が意図せずに作成されることが回避できる。

test-option : <test-option> は、デバイスが:validate:1.1 機能 (セクション 8.6) を通知する場合にのみ指定されてもよい (MAY) 。<test-option> には、次のいずれかの値がある。test-then-set: 設定を試みる前に検証テストを実行する。検証エラーが発生した場合は、<edit-config> 操作を実行してはいけない。これはデフォルトの test-option である。

test-then-set : 検証テストを行わずに最初にセットを実行する。

test-only : 設定を試行せずに、検証テストのみを実行する。

error-option : <error-option> には、次のいずれかの値がある。

stop-on-error : 最初のエラー時に <edit-config> を中止する。これはデフォルトの error-option である。

continue-on-error : エラー時に設定データの処理を続行する。エラーが記録され、エラーが発生した場合は否定応答が生成される。

rollback-on-error : エラー重大度 <rpc-error> が生成されるようなエラー条件が発生した場合、サーバは <edit-config> の処理を停止し、指定されたコンフィグレーションをこの <edit-config> の開始時に完全な状態に復元する。このオプションでは、セクション 8.5 で説明する:rollback-on-error 機能をサーバがサポートする必要がある。

config : デバイスのデータモデルの 1 つによって定義されるコンフィグレーションデータの階層。コンテンツは、デバイスが適切なデータモデルを検出することを可能にするために、適切な名前空間に配置されなければならず (MUST) 、コンテンツは、その能力定義によって定義されるように、そのデータモデルの制約に従わなければならない (MUST) 。機能については、セクション 8 で説明する。

肯定応答：デバイスが要求を満たすことができた場合、<ok> を含む <rpc-reply> が送信される。

否定応答：何らかの理由で要求を完了できない場合、<rpc-error> 応答が送信される。

例：このセクションの <edit-config> の例では、<interface> の複数のインスタンスが存在し、各 <interface> 内の <name> によってインスタンスが区別される単純なデータモデルを使用する。

ランニングコンフィグレーションで「Ethernet0/0」という名前のインタフェースで MTU を 1500 に設定する。

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config>
    <target>
      <running/>
    </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>
```

実行中の設定に「Ethernet0/0」という名前のインタフェースを追加し、以前のインタフェースをその名前で置き換える。

```
  <rpc message-id="101"
       xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <edit-config>
      <target>
        <running/>
      </target>
      <config xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
        <top xmlns="http://example.com/schema/1.2/config">
          <interface xc:operation="replace">
            <name>Ethernet0/0</name>
            <mtu>1500</mtu>
            <address>
              <name>192.0.2.4</name>
              <prefix-length>24</prefix-length>
            </address>
          </interface>
        </top>
      </config>
    </edit-config>
  </rpc>

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

ランニングコンフィグレーションから「Ethernet0/0」という名前のインタフェースのコンフィグレーションを削除する。

```
  <rpc message-id="101"
       xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <edit-config>
      <target>
        <running/>
      </target>
      <default-operation>none</default-operation>
      <config xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
        <top xmlns="http://example.com/schema/1.2/config">
          <interface xc:operation="delete">
            <name>Ethernet0/0</name>

          </interface>
        </top>
      </config>
    </edit-config>
```

```
</rpc>

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

OSPF エリアからインタフェース 192.0.2.4O を削除する (同じエリアに設定されている他のインタフェースは影響を受けない) 。:

```
<rpc message-id="101"
      xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config>
    <target>
      <running/>
    </target>
    <default-operation>none</default-operation>
    <config xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
      <top xmlns="http://example.com/schema/1.2/config">
        <protocols>
          <ospf>
            <area>
              <name>0.0.0.0</name>
              <interfaces>
                <interface xc:operation="delete">
                  <name>192.0.2.4</name>
                </interface>
              </interfaces>
            </area>
          </ospf>
        </protocols>
      </top>
    </config>
  </edit-config>
</rpc>

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

7.3. <copy-config>

説明：コンフィグレーションデータストア全体を作成するか、または別の完全なコンフィグレーションデータストアの内容で置き換える。ターゲットデータストアが存在する場合、上書きされる。または、許可されれば、新しいものが作成される。

NETCONF ピアが：url 機能 (セクション 8.8) をサポートしている場合、<url> は <source> または <target> パラメータとして表示できる。

デバイスが：writable-running capability を通知しても、<copy-config> の <target> パラメータとして <running/> コンフィグレーションデータストアをサポートしないことを選択してもよい。デバイスは、<source> と <target> の両方のパラメータが <url> を使用する場合、リモートからリモートへのコピー操作をサポートしないことを選択してもよい。<source> パラメータと <target> パラメータが同じ URL またはコンフィグレーションデータストアを識別する場合、"invalid-value" を含むエラータグでエラーを返さなければならない (MUST) 。

パラメータ：

target： <copy-config> の宛先として使用するコンフィグレーションデータストアの名前。

source： <copy-config> のソースとして使用するコンフィグレーションデータストアの名前、またはコピーする完全なコンフィグレーションを含む <config> 。

肯定応答：デバイスが要求を満たすことができた場合、<ok> を含む <rpc-reply> が送信される。
否定応答： <rpc-error> は、要求が何らかの理由で完了できない場合、<rpc-reply> 内に含まれる。
例:

```
<rpc message-id="101"
      xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <copy-config>
    <target>
      <running/>
    </target>
    <source>
      <url>https://user:password@example.com/cfg/new.txt</url>
    </source>
  </copy-config>
</rpc>

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

7.4. <delete-config>

説明：コンフィグレーションデータストアを削除する。<running> コンフィグレーションデータストアを削除できない。
NETCONF ピアが:url 機能 (セクション 8.8) をサポートしている場合、<url> は <target> パラメータとして表示できる。
パラメータ：
target: 削除するコンフィグレーションデータストアの名前。
肯定応答：デバイスが要求を満たすことができた場合、<ok> を含む <rpc-reply> が送信される。
否定応答： <rpc-error> は、要求が何らかの理由で完了できない場合、<rpc-reply> 内に含まれる。
例:

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

7.5. <lock>

説明：<lock>は、クライアントがデバイスのコンフィグレーションデータストアシステム全体をロックすることを可能にする。そのようなロックは、短い期間であることが意図され、クライアントが、他の NETCONF クライアント、非 NETCONF クライアント (例えば、SNMP およびコマンドラインインターフェース (CLI) スクリプト) 、およびユーザとの対話を心配することなく、変更できるようにすることを目的とする。

既存のセッションまたは他のエンティティがロックターゲットのいずれかの部分でロックを保持している場合、コンフィグレーションデータストアをロックしようとする試みは失敗しなければならない (MUST) 。

ロックが獲得されると、サーバは、このセッションによって要求されたもの以外のロックされたリソースへの変更を防止しなければならない (MUST) 。リソースを変更するための SNMP および CLI 要求は、適切なエラーで失敗しなければならない。

ロックの持続時間は、ロックが獲得されたときに始まり、ロックが解放されるか、または NETCONF セッションが閉じるまで続くものとして定義される。セッション終了は、クライアントによって明示的に実行されるか、または基礎となるトランスポートの

失敗、単純な非アクティビティタイムアウト、またはクライアント側での不正行為の検出などの基準に基づいてサーバによって暗示的に実行される。これらの基準は、実装および基礎となるトランスポートに依存する。

<lock> は必須パラメータ <target> をとる。<target> パラメータは、ロックされるコンフィグレーションデータストアの名前を指定する。ロックが有効な場合、ロックされた設定データストアで <edit-config> を使用し、<copy-config> のターゲットとしてロックされた設定を使用することは、他の NETCONF セッションでは許可されない。さらに、システムは、これらのロックされたコンフィグレーションリソースが、SNMP および CLI などの他の非 NETCONF 管理動作によって変更されないことを保証する。<kill-session> を使用して、別の NETCONF セッションが所有するロックを強制的に解除することができる。他のエンティティによって保持されているロックを解除する方法を定義することは、この文書の範囲外である。

以下の条件のいずれかが真である場合、ロックは許可されてはならない。

* ロックは、任意の NETCONF セッションまたは別のエンティティによって既に保持されている。
* ターゲット設定は <candidate> で、既に変更されており、これらの変更はコミットまたはロールバックされていない。
* ターゲットコンフィグレーションは <running> であり、別の NETCONF セッションが進行中の確認コミットを有する (セクション 8.4) 。

サーバは、<ok> または <rpc-error> のいずれかで応答しなければならない (MUST) 。
ロックを保持しているセッションが何らかの理由で終了した場合、ロックはシステムによって解放される。

パラメータ：
target : ロックするコンフィグレーションデータストアの名前。
肯定応答: デバイスが要求を満たすことができた場合、<ok> を含む <rpc-reply> が送信される。
否定応答： <rpc-error> は、何らかの理由で要求を完了できない場合に <rpc-reply> に含まれる。
ロックが既に保持されている場合、<error-tag> は "lock-denied" となり、<error-info> はロック所有者の <session-id> を含む。ロックが非 NETCONF エンティティによって保持される場合、0 (ゼロ) の <session-id> が含まれる。ターゲットの一部であってもロックを実行する他のエンティティは、(グローバルである) NETCONF ロックがそのターゲット上で取得されるのを防止することに留意されたい。

例：以下の例は、ロックの取得の成功を示している。

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

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

例：以下の例は、ロックがすでに使用されているときに、ロックを獲得しようとして失敗したことを示している。

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

<rpc-reply message-id="101"
```

```
        xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
      <rpc-error> <!– lock failed -->
        <error-type>protocol</error-type>
        <error-tag>lock-denied</error-tag>
        <error-severity>error</error-severity>
        <error-message>
          Lock failed, lock is already held
        </error-message>
        <error-info>
          <session-id>454</session-id>
          <!-- lock is held by NETCONF session 454 -->
        </error-info>
      </rpc-error>
    </rpc-reply>
```

7.6. <unlock>

説明： <unlock> は、<lock> で以前に取得した設定ロックを解除するために使用される。

<unlock> 操作は、以下のいずれかの条件が満たされた場合には成功しない。

* 指定されたロックは、現在アクティブではない。
* <unlock> 操作を発行するセッションは、ロックを取得したセッションと同じではない。

サーバは、<ok> または <rpc-error> のいずれかで応答しなければならない (MUST) 。

パラメータ:

target : ロック解除するコンフィグレーションデータストアの名前。

NETCONF クライアントは、ロックしていないコンフィグレーションデータストアのロックを解除することはできない。

肯定応答： <ok> を含む <rpc-reply> が送信される。

否定応答： <rpc-error> は、何らかの理由で要求を完了できない場合に <rpc-reply> に含まれる。

例:

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

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

7.7. <get>

説明：ランニングコンフィグレーションおよびデバイス状態情報を取得する。

パラメータ：

filter：このパラメータは、取得するシステムコンフィグレーションおよび状態データの部分を指定する。このパラメータが存在しない場合、すべてのデバイスコンフィグレーションおよび状態情報が戻される。

<filter> は、オプションで "type" 属性を含んでもよい。この属性は、<filter> 内で使用されるフィルタリング構文のタイプを示す。

NETCONF におけるデフォルトフィルタリングメカニズムは、サブツリーフィルタリングと呼ばれ、セクション 6 で説明される。「サブツリー」の値は、このタイプのフィルタリングを明示的に識別する。

NETCONF ピアが：xpath 機能 (セクション 8.9) をサポートしている場合、<filter> の "select" 属性が XPath 式を含むことを示すために、"xpath" の値を使用してもよい。

肯定応答： <rpc-reply> が送信される。<data> セクションには、適切なサブセットが含まれている。

否定応答： <rpc-error> は、何らかの理由で要求を完了できない場合に <rpc-reply> に含まれる。

例 ：

```
<rpc message-id="101"
      xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get>
    <filter type="subtree">
      <top xmlns="http://example.com/schema/1.2/stats">
        <interfaces>
          <interface>
            <ifName>eth0</ifName>
          </interface>
        </interfaces>
      </top>
    </filter>
  </get>
</rpc>

<rpc-reply message-id="101"
      xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <data>
    <top xmlns="http://example.com/schema/1.2/stats">
      <interfaces>
        <interface>
          <ifName>eth0</ifName>
          <ifInOctets>45621</ifInOctets>
          <ifOutOctets>774344</ifOutOctets>
        </interface>
      </interfaces>
    </top>
  </data>
</rpc-reply>
```

7.8. <close-session>

説明：NETCONF セッションの正常な終了を要求する。

NETCONF サーバが <close-session> リクエストを受信すると、セッションを正常に終了する。サーバは、セッションに関連するあらゆるロックおよびリソースを解放し、関連する接続を適切に閉じる。<close-session> リクエストの後に受信された NETCONF リクエストは無視される。

肯定応答： <ok>を含む <rpc-reply> が送信される。

否定応答： <rpc-error> は、何らかの理由で要求を完了できない場合に <rpc-reply> に含まれる。

例：

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

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

7.9. <kill-session>

説明：NETCONF セッションの強制終了。

NETCONF エンティティは、オープンセッションの <kill-session> 要求を受信すると、現在処理中の操作を中止し、セッションに関連するロックとリソースを解放し、関連する接続を閉じる。確認コミット (セクション 8.4) の処理中に NETCONF サーバが

<kill-session> 要求を受信した場合、確認コミットが発行される前の状態にコンフィグレーションを復元しなければならない。そうでない場合、<kill-session> 操作は、ロックを保持しているエンティティによって行われたコンフィグレーションまたは他のデバイス状態変更をロールバックしない。

パラメータ:

session-id : 終了する NETCONF セッションのセッション識別子。この値が現在のセッション ID と等しい場合、「無効値」エラーが返される。

肯定応答：デバイスが要求を満たすことができた場合、<ok> を含む <rpc-reply> が送信される。

否定応答： <rpc-error>は、何らかの理由で要求を完了できない場合に <rpc-reply> に含まれる。

例：

```
<rpc message-id="101"
      xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <kill-session>
    <session-id>4</session-id>
  </kill-session>
</rpc>

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

8. ケイパビリティ

このセクションでは、クライアントまたはサーバが実装してもよいケイパビリティのセットを定義する (MAY) 。各ピアは、初期ケイパビリティ交換中にそれらを送信することによって、そのケイパビリティをアドバタイズする。各ピアは、使用する可能性のあるケイパビリティのみを理解する必要があり、他のピアから受信した必要としない、または理解できない任意のケイパビリティを無視しなければならない (MUST) 。

追加のケイパビリティは、付録 D のテンプレートを使用して定義することができます。将来のケイパビリティ定義は、標準化団体によって標準として公開することも、独自の拡張として公開することもできる。

NETCONF ケイパビリティは、URI で識別される。基本ケイパビリティは、RFC 3553 [RFC 3553] に記載されている方法に従って URN を使用して定義される。このドキュメントで定義されているケイパビリティは、次の形式を持つ。

urn:ietf:params:netconf:capability:{name}:1.x

ここで、{name} はケイパビリティの名前である。ケイパビリティに複数のバージョンに存在する場合、ケイパビリティは、略して :{name}、または :{name}:{version} を使用して、議論および電子メールで参照されることが多い。例えば、foo ケイパビリティは、形式名 "urn:ietf:params:netconf:capability:foo:1.0" を有し、":foo " と呼ばれる。省略形は、プロトコル内で使用してはならない (MUST NOT) 。

8.1. ケイパビリティ交換

ケイパビリティは、セッション確立中に各ピアによって送信されるメッセージ内でアドバタイズされる。NETCONF セッションがオープンされるとき、各ピア (クライアントとサーバの両方) は、そのピアのケイパビリティのリストを含む <hello> 要素を送信しなければならない (MUST) 。各ピアは、少なくとも基本 NETCONF ケイパビリティ "urn:ietf:params:netconf:base:1.1" を送信しなければならない (MUST) 。ピアは、複数のプロトコルバージョンをサポートしていることを示すために、以前の NETCONF バージョンのケイパビリティを含めてもよい (MAY) 。

両方の NETCONF ピアは、他のピアが共通のプロトコルバージョンをアドバタイズしたことを検証しなければならない (MUST) 。プロトコルバージョンケイパビリティ URI を比較する場合、URI 文字列の末尾にパラメータが符号化されている場合には、基本部分のみが使用される。共通のプロトコルバージョンケイパビリティが見つからない場合、NETCONF ピアはセッションを継続してはならない（MUST NOT)。共通する複数のプロトコルバージョン URI が存在する場合、最も高い番号の (最新の) プロトコルバージョンを両方のピアで使用しなければならない (MUST) 。

<hello> 要素を送信するサーバは、この NETCONF セッションのセッション ID を含む <session-id> 要素を含まなければならない (MUST) 。<hello> 要素を送信するクライアントは、<session-id> 要素を含めてはならない (MUST NOT) 。

<session-id> 要素を持つ <hello> メッセージを受信したサーバは、NETCONF セッションを終了しなければならない (MUST) 。同様に、サーバの <hello> メッセージで <session-id> 要素を受信しないクライアントは、(最初に <close-session> を送信せずに) NETCONF セッションを終了しなければならない (MUST) 。

以下の例では、サーバは、基本 NETCONF ケイパビリティ、基本 NETCONF ドキュメントで定義されている 1 つの NETCONF ケイ

パビリティ、および1つの実装固有ケイパビリティを通知する。

```
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <capabilities>
    <capability>
      urn:ietf:params:netconf:base:1.1
    </capability>
    <capability>
      urn:ietf:params:netconf:capability:startup:1.0
    </capability>
    <capability>
      http://example.net/router/2.3/myfeature
    </capability>
  </capabilities>
  <session-id>4</session-id>
</hello>
```

各ピアは、接続がオープンになるとすぐに <hello> 要素を同時に送信する。ピアは、自身のセットを送信する前に、相手側からケイパビリティセットを受信するのを待ってはならない (MUST NOT) 。

8.2. Writable-Running (書き込み可能な実行) ケイパビリティ

8.2.1. 説明

: writable-running ケイパビリティは、デバイスが <running> コンフィグレーションデータストアへの直接書き込みをサポートしていることを示す。つまり、<edit-config> および <copy-config> 操作がサポートされ、<running> コンフィグレーションがターゲットになる。

8.2.2. 依存性

なし。

8.2.3. ケイパビリティ識別子

: writable-running ケイパビリティは、以下のケイパビリティ文字列で識別される。

urn:ietf:params:netconf:capability:writable-running:1.0

8.2.4. 新規操作

なし。

8.2.5. 既存操作の変更

8.2.5.1. <edit-config>

: writable-running ケイパビリティは、<edit-config> 操作を変更して、<running> 要素を <target> として受け入れる。

8.2.5.2. <copy-config>

: writable-running ケイパビリティは、<copy-config> 操作を変更して、<running> 要素を <target> として受け入れる。

8.3. Candidate Configuration (候補コンフィグレーション) ケイパビリティ

8.3.1. 説明

: candidate ケイパビリティは、デバイスが、デバイスの現在のコンフィグレーションに影響を与えることなく操作することができるコンフィグレーションデータを保持するために使用される候補コンフィグレーションデータストアをサポートすることを示す。候補コンフィグレーションは、コンフィグレーションデータを作成し、操作するための作業場所として働く完全なコンフィグレーションデータセットである。所望のコンフィグレーションデータを構築するために、このデータに追加、削除、および変更を行うことができる。<commit> 操作は、デバイスのランニングコンフィグレーションに候補コンフィグレーションの値を設定する。<commit> 操作は、いつでも実行されてもよい (MAY) 。

<commit> 操作は、候補コンフィグレーションの現在の内容をランニングコンフィグレーションに効果的に設定する。これは、単純なコピーとしてモデル化することができるが、いくつかの理由で、別個の操作として行われる。高レベルの概念をファーストクラスのオペレーションとして維持する際に、開発者は、クライアントが何を要求しているか、およびサーバが何を実行しなければならないか

の両方をより明確に見ることができる。これは、意図をより明白に保ち、特別な場合をより複雑にせず、操作間の相互作用をより簡単にする。例えば、:confirmed-commit:1.1 ケイパビリティ (セクション 8.4) は、"copy confirmed" 操作として意味をなさない。

候補コンフィグレーションは、複数のセッション間で共有することができる。クライアントが、候補コンフィグレーションが共有されていないという特定の情報を持たない限り、他のセッションが同時に候補コンフィグレーションを修正できると仮定しなければならない (MUST) 。したがって、クライアントは、候補コンフィグレーションを修正する前に、その候補コンフィグレーションをロックすることが賢明である。

クライアントは、<discard-changes> 操作を実行することによって、候補コンフィグレーションに対するコミットされていない変更を破棄することができる。この操作は、候補コンフィグレーションの内容をランニングコンフィグレーションの内容に戻す。

8.3.2. 依存性

なし。

8.3.3. ケイパビリティ識別子

:candidate ケイパビリティは、以下のケイパビリティ文字列によって識別される。

urn:ietf:params:netconf:capability:candidate:1.0

8.3.4. 新規操作

8.3.4.1. <commit>

説明：

候補コンフィグレーションのコンテンツが完了すると、コンフィグレーションデータをコミットすることができ、データセットをデバイスの残りの部分に発行し、新しいコンフィグレーションで記述された挙動に適合するようにデバイスに要求する。

候補コンフィグレーションをデバイスの新しい現在のコンフィグレーションとしてコミットするには、<commit> 操作を使用する。

<commit> 操作は、候補コンフィグレーションに含まれるコンフィグレーションデータを実施するようにデバイスに命令する。デバイスが候補コンフィグレーションデータストア内のすべての変更をコミットできない場合、ランニングコンフィグレーションは変更されないままでなければならない (MUST) 。デバイスがコミットに成功した場合、ランニングコンフィグレーションは、候補コンフィグレーションの内容で更新されなければならない (MUST) 。

ランニングコンフィグレーションまたは候補コンフィグレーションが現在別のセッションによってロックされている場合、<commit> 操作は、<error-tag> 値が "in-use" で失敗しなければならない (MUST) 。

システムに :candidate ケイパビリティがない場合、<commit> 操作は使用できない。

肯定応答：

デバイスが要求を満たすことができた場合、<ok>要素を含む<rpc-reply>が送信される。

否定応答:

要求が何らかの理由で完了できない場合、<rpc-error>要素が<rpc-reply>に含まれる。

例:

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

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

8.3.4.2. <discard-changes>

クライアントが、候補コンフィグレーションをコミットしないと決定した場合、<discard-changes> 操作を使用して、候補コンフィグレーションを現在ランニングコンフィグレーションに戻すことができる。

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

この操作は、ランニングコンフィグレーションの内容で候補コンフィグレーションをリセットすることによって、コミットされてい

ない変更を破棄する。

8.3.5. 既存操作の変更

8.3.5.1. <get-config>、<edit-config>、<copy-config>、<validate>

候補コンフィグレーションは、<get-config>、<edit-config>、<copy-config>、または <validate> 操作のソースまたはターゲットとして、<source> または <target> パラメータを使用できる。<candidate> 要素は、コンフィグレーションの候補を示すために使用される。

```
<rpc message-id="101"
       xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get-config>
    <source>
      <candidate/>
    </source>
  </get-config>
</rpc>
```

8.3.5.2. <lock>と<unlock>

<candidate> 要素を <target> パラメータとして使用する <lock> 操作を使用して、候補コンフィグレーションをロックすることができる。

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

同様に、<candidate> 要素を <target> パラメータとして使用して、候補コンフィグレーションのロックが解除される。

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

クライアントが、候補コンフィグレーションに対する未解決の変更で障害を起こすと、回復が困難になる可能性がある。容易な回復ができるようにするために、<unlock> 操作で明示的に行っても、セッション障害から暗黙的に行っても、ロックが解除されたときに未解決の変更が破棄される。

8.4. Confirmed Commit (確認中コミット) ケイパビリティ

8.4.1. 説明

:confirmed-commit:1.1 ケイパビリティは、サーバが <cancel-commit> 操作と、<commit> 操作の <confirm-timeout>、<persist>、および <persist-id> パラメータをサポートすることを示す。<commit> 操作の詳細については、セクション 8.3 を参照してください。

コミットの確定がタイムアウト時間内 (デフォルトでは 600 秒=10 分) に発行されない場合、確認中 <commit> 操作は元に戻されなければならない (MUST) 。コミットの確定は、<confirmed>パラメータなしの <commit> 操作である。タイムアウト時間は、<confirm-timeout> パラメータで調整できる。タイマが満了する前にフォローアップ確認中 <commit> 操作が発行されると、タイマは新しい値 (デフォルトでは 600 秒) にリセットされる。コミットの確定とフォローアップ確認中 <commit> 操作の両方で、コンフィグレーションに追加の変更を導入してもよい (MAY) 。

確認中コミット操作で <persist> 要素が指定されない場合、任意のフォローアップコミットとコミットの確定は、確認中コミットを

発行したのと同じセッションで発行されなければならない (MUST) 。もし <persist> 要素が確認中 <commit> 操作で与えられるなら、フォローアップコミットとコミットの確定はどのセッションでも与えられ、<persist> 要素の与えられた値に等しい値を持つ <persist-id> 要素を含まなければならない (MUST) 。

サーバが：startup ケイパビリティ も通知する場合、スタートアップ の変更を保存するには、ランニングからスタートアップへの <copy-config> も必要である。

確認タイムアウトが満了する前に、確認中コミットを発行したセッションが何らかの理由で終了した場合、確認中コミットが <persist> 要素も含まない限り、サーバは、確認中コミットが発行される前の状態にコンフィグレーションを復元しなければならない (MUST) 。

確認タイムアウトが満了する前に何らかの理由でデバイスがリブートした場合、サーバは、確認中コミットが発行される前の状態にコンフィグレーションを復元しなければならない (MUST) 。

コミットの確定が発行されない場合、デバイスは、そのコンフィグレーションを、確認中コミットの発行前の状態に戻す。確認中コミットをキャンセルし、確認タイムアウトが満了するのを待たずに変更を元に戻すには、クライアントは <cancel-commit> 操作を使用して、確認中コミットが発行される前の状態にコンフィグレーションを復元できる。

共有コンフィグレーションの場合、この機能を使用すると、コンフィグレーションロック機能を使用しない限り (つまり、<edit-config> 操作が開始される前にロックが取得されない限り) 、他のコンフィグレーション変更 (たとえば、他の NETCONF セッションを介した) が不注意に変更または削除される可能性がある。したがって、共有コンフィグレーションデータストアでこの機能を使用するためには、コンフィグレーションロックも使用すべきであることが強く推奨される (SHOULD) 。

このケイパビリティのバージョン 1.0 は [RFC4741] で定義されている。本書では、バージョン 1.1 が定義されており、新しい操作 <cancel-commit> と、2 つの新しいオプションパラメータ <persist> と <persist-id> を追加することにより、バージョン 1.0 を拡張する。古いクライアントとの下位互換性のために、この仕様に準拠するサーバは、バージョン 1.1 に加えてバージョン 1.0 を通知してもよい (MAY) 。

8.4.2. 依存性

:confirmed-commit:1.1 ケイパビリティは、:candidate ケイパビリティもサポートされている場合にのみ関連する。

8.4.3. ケイパビリティ識別子

:confirmed-commit:1.1 ケイパビリティは、以下のケイパビリティ文字列によって識別される。

urn:ietf:params:netconf:capability:confirmed-commit:1.1

8.4.4. 新規操作

8.4.4.1. <cancel-commit>

説明:

進行中の確認中コミットをキャンセルする。<persist-id> パラメータが与えられない場合、<cancel-commit> 操作は、確認中コミットを発行したのと同じセッション上で発行されなければならない (MUST) 。

パラメータ：

persist-id :

永続的な確認中コミットをキャンセルする。この値は、<persist> パラメータで <commit> 操作に与えられた値と等しくなければならない (MUST) 。値が一致しない場合、操作は "invalid-value" エラーで失敗する。

肯定応答：

デバイスが要求を満たすことができた場合、<ok> 要素を含む <rpc-reply> が送信される。

否定応答:

要求が何らかの理由で完了できない場合、<rpc-error> 要素が <rpc-reply> に含まれる。

例：

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

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

```
</rpc-reply>

<rpc message-id="102"
        xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <cancel-commit/>
</rpc>

<rpc-reply message-id="102"
        xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <ok/>
</rpc-reply>
```

8.4.5. 既存操作の変更

8.4.5.1. <commit>

: confirmed-commit:1.1 ケイパビリティを使用すると、<commit> 操作に 4 つの追加パラメータを使用することができる。

パラメータ：

confirmed :

確認中 <commit> 操作を実行する。

confirm-timeout :

確認中コミットのタイムアウト時間 (秒単位) 。指定されていない場合、確認タイムアウトはデフォルトで 600 秒になる。

persist :

確認中コミットをセッション終了後に残し、進行中の確認中コミットにトークンを設定する。

persist-id :

以前の <commit> 操作のトークンを使用して、任意のセッションからのフォローアップ確認中コミットまたはコミットの確定を発行するために使用する。

例：

```
<rpc message-id="101"
        xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <commit>
    <confirmed/>
    <confirm-timeout>120</confirm-timeout>
  </commit>
</rpc>

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

例：

```
<!-- start a persistent confirmed-commit -->
<rpc message-id="101"
        xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <commit>
    <confirmed/>
    <persist>IQ,d4668</persist>
  </commit>
</rpc>

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

```
</rpc-reply>

<!-- confirm the persistent confirmed-commit,
        possibly from another session -->
<rpc message-id="102"
        xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <commit>
    <persist-id>IQ,d4668</persist-id>
  </commit>
</rpc>

<rpc-reply message-id="102"
        xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <ok/>
</rpc-reply>
```

8.5. Rollback-on-Error (エラー時のロールバック) ケイパビリティ

8.5.1. 説明

このケイパビリティは、サーバが <error-option> パラメータの <edit-config> 操作への "rollback-on-error" 値をサポートすることを示す。

共有コンフィグレーションの場合、このケイパビリティを使用すると、コンフィグレーションロック機能を使用しない限り (つまり、<edit-config> 操作が開始される前にロックが取得されない限り) 、他のコンフィグレーション変更 (たとえば、他の NETCONF セッションを介した) が不注意に変更または削除される可能性がある。したがって、共有コンフィグレーションデータストアでこのケイパビリティを使用するには、コンフィグレーションロックも使用することを強くお勧めする。

8.5.2. 依存性

なし。

8.5.3. ケイパビリティ識別子

:rollback-on-error ケイパビリティは、以下のケイパビリティ文字列によって識別されます。

urn:ietf:params:netconf:capability:rollback-on-error:1.0

8.5.4. 新規操作

なし。

8.5.5. 既存操作の変更

8.5.5.1. <edit-config>

rollback-on-error ケイパビリティにより、<edit-config> 操作の <error-option> パラメータに "rollback-on-error " 値を指定できる。

```
<rpc message-id="101"
        xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config>
    <target>
      <running/>
    </target>
    <error-option>rollback-on-error</error-option>
    <config>
      <top xmlns="http://example.com/schema/1.2/config">
        <interface>
          <name>Ethernet0/0</name>
          <mtu>100000</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>
```

8.6. Validate (検証) ケイパビリティ

8.6.1. 説明

検証は、コンフィグレーションをデバイスに適用する前に、構文エラーおよびセマンティックエラーについて完全なコンフィグレーションをチェックすることからなる。

このケイパビリティがアドバタイズされると、デバイスは <validate> プロトコル操作をサポートし、少なくとも構文エラーをチェックする。さらに、このケイパビリティは、<edit-config> 操作の <test-option> パラメータをサポートし、提供されている場合は、少なくとも構文エラーをチェックする。

このケイパビリティのバージョン 1.0 は [RFC4741] で定義されている。本書では、バージョン 1.1 が定義されており、<edit-config> 操作の <test-option> パラメータに新しい値 "test-only" を追加することによって、バージョン 1.0 を拡張する。古いクライアントとの下位互換性のために、この仕様に準拠するサーバは、バージョン 1.1 に加えてバージョン 1.0 を通知してもよい (MAY) 。

8.6.2. 依存性

なし。

8.6.3. ケイパビリティ識別子

:validate:1.1 ケイパビリティは、以下のケイパビリティ文字列によって識別される。

urn:ietf:params:netconf:capability:validate:1.1

8.6.4. 新規操作

8.6.4.1. <validate>

説明：

このプロトコル操作は、指定されたコンフィグレーションの内容を検証する。

パラメータ：

source :

検証するコンフィグレーションデータストアの名前 (<candidate> または検証する完全なコンフィグレーションを含む <config> 要素など) 。

肯定応答:

デバイスが要求を満たすことができた場合、<ok> 要素を含む <rpc-reply> が送信される。

否定応答:

要求が何らかの理由で完了できない場合、<rpc-error> 要素が <rpc-reply> に含まれる。

<validate> 操作は、構文エラー、パラメータの欠落、未定義のコンフィグレーションデータへの参照、または基礎となるデータモデルによって確立されたルールの他の違反など、いくつかの理由で失敗する可能性がある。

例：

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

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

</rpc-reply>

8.6.5. 既存操作の変更

8.6.5.1. <edit-config>

:validate:1.1 ケイパビリティは、<test-option> パラメータを受け入れるように <edit-config> 操作を変更する。

8.7. Distinct Startup (明確なスタートアップ) ケイパビリティ

8.7.1. 説明

デバイスは、別個のランニングコンフィグレーションデータストアおよびスタートアップコンフィグレーションデータストアをサポートする。スタートアップコンフィグレーションは、起動時にデバイスによってロードされる。ランニングコンフィグレーションに影響する操作は、スタートアップコンフィグレーションへ自動的にコピーされない。<running> から <startup> への明示的な <copy-config> 操作を使用して、スタートアップコンフィグレーションを現在のランニングコンフィグレーションの内容に更新する。NETCONF プロトコル操作は、<startup> 要素を使用したスタートアップデータストアを参照する。

8.7.2. 依存性

なし。

8.7.3. ケイパビリティ識別子

:startup ケイパビリティは、以下のケイパビリティ文字列によって識別される。

urn:ietf:params:netconf:capability:startup:1.0

8.7.4. 新規操作

なし。

8.7.5. 既存操作の変更

8.7.5.1. 一般

:startup ケイパビリティは、<startup/> コンフィグレーションデータストアをいくつかの NETCONF 操作の引数に追加する。サーバは、以下の追加値をサポートしなければならない (MUST) 。

演算	パラメータ	注記
<get-config>	<source>	
<copy-config>	<source> <target>	
<lock>	<target>	
<unlock>	<target>	
<validate>	<source>	validate:1.1 が アドバタイズされる場合
<delete-config>	<target>	デバイスを工場 デフォルトに リセットします

スタートアップコンフィグレーションを保存するには、<copy-config> 操作を使用して、<running> コンフィグレーションデータストアを <startup> コンフィグレーションデータストアにコピーする。

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <copy-config>
    <target>
      <startup/>
    </target>
```

```
      <source>
        <running/>
      </source>
    </copy-config>
  </rpc>
```

8.8. URL ケイパビリティ

8.8.1. 説明

NETCONF ピアは、<source> および <target> パラメータ内の <url> 要素を受け入れることができる。このケイパビリティは、サポートされる URL 方式を示す URL 引数によってさらに識別される。

8.8.2. 依存性

なし。

8.8.3. ケイパビリティ識別子

:url ケイパビリティは、以下のケイパビリティ文字列によって識別される。

urn:ietf:params:netconf:capability:url:1.0?scheme={name,...}

:url ケイパビリティ URI は、NETCONF ピアがどのスキームをサポートしているかを示すスキーム名のカンマで区切られたリストが割り当てられた"scheme"引数を含まなければならない (MUST) 。例:

urn:ietf:params:netconf:capability:url:1.0?scheme=http,ftp,file

8.8.4. 新規操作

なし。

8.8.5. 既存操作の変更

8.8.5.1. <edit-config>

:url ケイパビリティは、<edit-config> 操作を変更して、<config> パラメータの代替として<url> 要素を受け入れる。

URL が参照するファイルには、変更するコンフィグレーションデータ階層が含まれており、"urn:ietf:params:xml:ns:netconf:base:1.0" 名前空間の要素 <config> の下に XML でエンコードされる。

8.8.5.2. <copy-config>

:url ケイパビリティは、<copy-config> 操作を変更して、<url> 要素を <source> および <target> パラメータの値として受け入れる。

url が参照するファイルには、"urn:ietf:params:xml:ns:netconf:base:1.0" 名前空間の要素 <config> の下に XML でエンコードされた完全なデータストアが含まれる。

8.8.5.3. <delete-config>

:url ケイパビリティは、<delete-config> 操作を変更して、<url> 要素を <target> パラメータの値として受け入れる。

8.8.5.4. <validate>

:url ケイパビリティは、<validate> 操作を変更して、<source> パラメータの値として <url> 要素を受け入れる。

8.9. XPath ケイパビリティ

8.9.1. 説明

XPath ケイパビリティは、NETCONF ピアが <filter> 要素で XPath 式の使用をサポートしていることを示す。XPath は、[W3C.REC-xpath-19991116] に記載されている。

XPath 式で使用されるデータモデルは、XPath 1.0 [W3C.REC-xpath-19991116] で使用されるものと同じであり、XSLT 1.0 ([W3C.REC-xslt-19991116] 、セクション 3.1) によって使用されるものと同じルートノード子の拡張を持つ。具体的には、ルートノードは、任意の数の要素ノードをその子として持っていてもよいことを意味する (MAY) 。

XPath 式は、以下のコンテキストで評価される。

- 名前空間宣言のセットは、<filter> 要素のスコープ内のものである。
- 変数バインディングのセットは、データモデルによって定義される。このような変数バインディングが定義されていない場合、セットは空である。
- 関数ライブラリは、コア関数ライブラリにデータモデルによって定義される任意の関数を加えたものである。

o　コンテキストノードはルートノードである。

XPath 式はノード・セットを返さなければならない (MUST) 。ノード・セットを返さない場合、操作は "invalid-value" エラーで失敗する。

応答メッセージは、フィルタ式によって選択されたサブツリーを含む。このような各サブツリーについて、データモデルルートノードからサブツリーまでのパスが応答メッセージに含まれる。サブツリーを一意に識別するために必要な任意の要素または属性も応答メッセージに含まれる。特定のデータインスタンスは、応答において複製されない。

8.9.2.　依存性

なし。

8.9.3.　ケイパビリティ識別子

:xpath ケイパビリティは、以下のケイパビリティ文字列によって識別される。

urn:ietf:params:netconf:capability:xpath:1.0

8.9.4.　新規操作

なし。

8.9.5.　既存操作の変更

8.9.5.1.　<get-config> と <get>

:xpath ケイパビリティは、<get> および <get-config> 操作を変更して、<filter> 要素の "type " 属性の値 "xpath " を受け入れる。"type" 属性が "xpath" に設定されるとき、"select" 属性は <filter> 要素上に存在しなければならない (MUST) 。"select" 属性は、XPath 式として扱われ、返されたデータをフィルタリングするために使用される。この場合、<filter> 要素自体は空でなければならない (MUST) 。

select 式の XPath 結果は、ノード・セットでなければならない (MUST) 。ノード・セット内の各ノードは、基礎となるデータモデル内のノードに対応しなければならない (MUST) 。各ノードを適切に識別するために、以下の符号化規則が定義される。

o　結果ノードのすべての祖先ノードは、最初に符号化されなければならない (MUST) 。そのため、応答で返される <data> 要素は、基礎となるデータモデルに従って、完全に指定されたサブツリーのみを含む。
o　結果ノードの任意の兄弟ノードまたは祖先ノードが、概念データ構造内の特定のインスタンスを識別するために必要とされる場合、これらのノードもまた、応答の中で符号化されなければならない (MUST) 。

例：

```
<rpc message-id="101"
       xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get-config>
    <source>
      <running/>
    </source>
    <!-- get the user named fred -->
    <filter xmlns:t="http://example.com/schema/1.2/config"
            type="xpath"
            select="/t:top/t:users/t:user[t:name='fred']"/>
  </get-config>
</rpc>

<rpc-reply message-id="101"
           xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <data>
    <top xmlns="http://example.com/schema/1.2/config">
      <users>
        <user>
          <name>fred</name>
          <company-info>
```

```
            <id>2</id>
          </company-info>
        </user>
      </users>
    </top>
  </data>
</rpc-reply>
```

9. セキュリティ考察

このセクションでは、基本NETCONFメッセージ層とNETCONFプロトコルの基本操作に関するセキュリティの考慮事項を提供する。NETCONF トランスポートのセキュリティ考慮事項は、トランスポートドキュメントで提供され、NETCONF によって操作されるコンテンツのセキュリティ考慮事項は、データモデルを定義するドキュメントで見つけることができる。

この文書は、そのようなスキームがメタデータモデルまたはデータモデルに結び付けられる可能性が高いため、許可スキームを指定しない。実装者は、NETCONF で包括的な許可スキームを提供すべきである (SHOULD) 。

NETCONF サーバを介した個々のユーザの許可は、他のインタフェースに 1 対 1 にマッピングしてもしなくてもよい。第 1 に、データモデルは互換性がないかもしれない。第 2 に、セキュア・トランスポート層で利用可能なメカニズム (例えば、SSH、ブロック・エクステンシブル・エクスチェンジ・プロトコル (BEEP) など) に基づいて許可することが望ましい場合がある。

さらに、コンフィグレーションに対する操作は、それらの操作が操作されているファイルまたはオブジェクトに対するグローバル・ロックによっても保護されない場合、意図しない結果をもたらす可能性がある。例えば、ランニングコンフィグレーションがロックされていない場合、部分的に完全なアクセスリストが、未知の候補コンフィグレーションから候補コンフィグレーションのロックの所有者にコミットされる可能性があり、セキュアでないデバイスまたはアクセス不可能なデバイスのいずれかにつながる。

コンフィグレーション情報は、その性質上、非常に敏感である。平文で完全性チェックなしに送信することにより、デバイスは、古典的な盗聴及び誤ったデータ注入攻撃に対して開放される。コンフィグレーション情報は、多くの場合、パスワード、ユーザ名、サービス記述、およびトポロジ情報を含み、これらはすべて機密である。このため、このプロトコルは、他の管理インタフェースで経験すると予想されるあらゆる攻撃方法に十分な注意を払って慎重に実装される必要がある (SHOULD) 。

したがって、プロトコルは、機密性と認証の両方のオプションを最小限にサポートしなければならない (MUST) 。基礎となるプロトコル (SSH、BEEP など) は、必要に応じて、機密性と認証の両方を提供することが予想される。さらに、NETCONF セッションの各終端の ID が、任意の所与の要求に対する許可を決定するために、他方で利用可能であることが予想される。また、承認の目的のために利用可能にされる、トランスポートおよび暗号化方法などの追加情報を容易に想像することができる。NETCONF 自体は、再認証する手段を提供せず、認証をはるかに少なくする。このようなアクションはすべて、下位レイヤで行われる。

異なる環境は、認証の前後で異なる権利を可能にすることができる。したがって、許可モデルは、この文書では指定されない。操作が適切に許可されていない場合、単純な「アクセス拒否」で十分である。認証情報は、コンフィグレーション情報の形態で交換することができ、これは、接続のセキュリティを保証するためのより多くの理由であることに留意されたい。

これまで述べてきたように、いくつかのオペレーションは、本質的に他の操作よりも明らかにより敏感であることを認識することが重要である。たとえば、スタートアップコンフィグレーションまたはランニングコンフィグレーションに対する <copy-config> は、明らかに通常のプロビジョニング操作ではありませんが、<edit-config> は通常のプロビジョニング操作ではない。そのようなグローバル操作は、個人が実行する権限を持たない情報の変更を許可しないものとしなければならない (MUST) 。例えば、ユーザ A がインタフェースで IP アドレスを設定することを許可されていないが、ユーザ B が <candidate> コンフィグレーションのインタフェースで IP アドレスを設定した場合、ユーザ A は <candidate> コンフィグレーションをコミットすることを許可されてはならない (MUST NOT) 。

同様に、誰かが「特定の場所で URL ケイパビリティを介してコンフィグレーションを書く」と言うだけで、これは、要素が適切な許可なしにそれを行うことを意味しない。

<lock> 操作は、NETCONF が、管理者の少なくともいくらかの信頼を有するシステムによる使用を意図されていることを示す。この文書に明記されているように、トップ階級の人がアクセスできないコンフィグレーションの部分をロックすることが可能である。結局、コンフィグレーション全体がロックされる。この問題を緩和するために、2 つのアプローチがある。違反しているセッションのセッション識別子を知っている場合、NETCONF 内から別の NETCONF セッションをプログラムで強制終了することが可能である。ロックを解除する他の可能な方法は、デバイスのネイティブユーザインタフェース内に機能を提供することである。これらの 2 つのメカニズムは、認証、許可、およびアカウンティング (AAA) サーバから違反ユーザを除去することによって改善することができる競合状態に悩まされている。しかしながら、そのような解決策は、SSH 公開/秘密鍵ペアが使用されるような、すべての展開シナリオにおいて有用ではない。

10. IANA Considerations

[RFC6241] の原文参照のこと。

11. Contributors

[RFC6241] の原文参照のこと。

12. Acknowledgements

[RFC6241] の原文参照のこと。

13. 参考文献

13.1. 参考文献 (Normative)

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3553] Mealling, M., Masinter, L., Hardie, T., and G. Klyne, "An IETF URN Sub-namespace for Registered Protocol Parameters", BCP 73, RFC 3553, June 2003.

[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO10646", STD 63, RFC 3629, November 2003.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, January 2004.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.

[RFC5717] Lengyel, B. and M. Bjorklund, "Partial Lock Remote Procedure Call (RPC) for NETCONF", RFC 5717, December 2009.

[RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.

[RFC6021] Schoenwaelder, J., "Common YANG Data Types", RFC 6021, October 2010.

[RFC6242] Wasserman, M., "Using the NETCONF Configuration Protocol over Secure Shell (SSH)", RFC 6242, June 2011.

[W3C.REC-xml-20001006] Sperberg-McQueen, C., Bray, T., Paoli, J., and E. Maler, "Extensible Markup Language (XML) 1.0 (Second Edition)", World Wide Web Consortium REC-xml-20001006, October 2000, <http://www.w3.org/TR/2000/REC-xml-20001006>.

[W3C.REC-xpath-19991116] DeRose, S. and J. Clark, "XML Path Language (XPath) Version 1.0", World Wide Web Consortium Recommendation REC-xpath-19991116, November 1999, <http://www.w3.org/TR/1999/REC-xpath-19991116>.

13.2. 参考文献 (Informative)

[RFC2865] Rigney, C., Willens, S., Rubens, A., and W. Simpson, "Remote Authentication Dial In User Service (RADIUS)", RFC 2865, June 2000.

[RFC3470] Hollenbeck, S., Rose, M., and L. Masinter, "Guidelines for the Use of Extensible Markup Language (XML) within IETF Protocols", BCP 70, RFC 3470, January 2003.

[RFC4251] Ylonen, T. and C. Lonvick, "The Secure Shell (SSH) Protocol Architecture", RFC 4251, January 2006.

[RFC4741] Enns, R., "NETCONF Configuration Protocol", RFC 4741, December 2006.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

[W3C.REC-xslt-19991116] Clark, J., "XSL Transformations (XSLT) Version 1.0", World Wide Web Consortium Recommendation REC-xslt-19991116, November 1999, <http://www.w3.org/TR/1999/REC-xslt-19991116>.

付録 A. NETCONF Error List

[RFC6241] の原文参照のこと。

付録 B. XML Schema for NETCONF Messages Layer

[RFC6241] の原文参照のこと。

付録 C. YANG Module for NETCONF Protocol Operations

[RFC6241] の原文参照のこと。

付録 D. Capability Template

[RFC6241] の原文参照のこと。

付録 E. Configuring Multiple Devices with NETCONF

[RFC6241] の原文参照のこと。

付録 F. Changes from RFC 4741

[RFC6241] の原文参照のこと。