Inside RESTCONF:

To Capitalize on the New Network Programmability Standard, Vendors Should Know How RESTCONF Works, Where it Can Be Used, and Where it Shouldn't

tail-f

Table of Contents

Introduction

Organizations in all industries want to lower operating costs and decrease time-to-value of new services—and they're looking to network programmability to do it. By automating the configuration and management of network elements (both physical and virtual), organizations can eliminate huge amounts of manual effort, and create and provision services much more quickly. To accomplish this, however, southbound network elements must be addressable by northbound systems such as orchestration platforms and software defined networking (SDN) controllers. As a result, network equipment vendors are looking for the best approach to enable their network elements to communicate with these systems.

In modern networks, programmability is typically based on data models defined in YANG [RFC 7950]. For several years, service providers have used NETCONF [RFC6241] as the protocol of choice to access and manipulate YANG data for the management and configuration of network elements in large-scale service provider environments. NETCONF remains a powerful, flexible protocol to accomplish this. As more enterprises look to create programmable data center environments, however, there has been growing interest in finding alternate approaches based on REST APIs. This market demand makes sense: many traditional enterprise IT programmers have little background in NETCONF and incorrectly envision a steep learning curve to use NETCONF libraries and tools effectively. They are, however, intimately familiar with using REST APIs to programmatically access remote Web services, and they appreciate the ability to use a protocol with which they are already comfortable.

Today, programmers can use REST calls and libraries to address southbound network elements in the same way they use REST to call other types of resources. However, there is no single, standard implementation. Different vendors implement their own proprietary RESTful APIs—much like the way that different vendors can all claim they use CLI, but implement it in unique ways for their products. For these reasons, RESTCONF [RFC8040] was born.

RESTCONF is a recently standardized REST-like protocol over HTTP (or HTTPS) for accessing data defined in YANG, using the datastores defined in NETCONF. RESTCONF is not meant to replace NETCONF but to provide a simplified interface that follows REST principles and is compatible with resource-oriented network element abstractions. It is envisioned for use cases like SDN controller integration, application automation, and operations support system/business support system (OSS/BSS) integration.

RESTCONF standardizes the use of REST techniques to manipulate the data described in YANG data models (the same data used by NETCONF to configure network elements). Unlike NETCONF, however, RESTCONF runs over HTTP Web protocol and uses the familiar verbs of HTTP—"push," "put," "get," "patch," and others—to make changes to network elements. Most importantly to programmers familiar with REST, it allows them to begin using YANG data models to automate their environments, while using the REST-based tools and knowledge they have today.

For network equipment vendors, it's important to understand how RESTCONF works, how it compares to NETCONF, and the tradeoffs in using one versus the other. This paper provides an overview of RESTCONF, including features and methods defined in the new standard. It also discusses some of the limitations of RESTCONF in contrast to NETCONF and the scenarios where RESTCONF can provide an effective solution.

If network element vendors want their products to be applicable to a broad range of customers and use cases, they should make sure that they fully understand what RESTCONF can and cannot do. In most cases, they will likely conclude that, while RESTCONF support can provide a valuable complementary option for integrating their elements into automated environments, it should not be viewed as a NETCONF replacement.

Note: To get the most out of this white paper you should be familiar with NETCONF and YANG.

For more details on NETCONF, visit: https://tools.ietf.org/html/rfc6241

To learn more about YANG, visit: https://tools.ietf.org/html/7950

You can also view NETCONF and YANG tutorials as part of the free ConfD Training Videos by visiting: http://www.tail-f.com/confd-training-videos/

RESTCONF Overview

RESTCONF emerged in response to the broad popularity of REST in enterprise networking and data center environments. RESTful APIs are popular for a number of reasons: They are scalable, perform relatively well, provide a simple and uniform interface, and they are easy to port to different platforms. A huge number of RESTful APIs are now in use, and while different vendors implement their own proprietary versions of REST, all share some common properties. Every RESTful API:

- Is based on a client-server model
- Is stateless (The server doesn't maintain any state.)
- Uses textual representation (typically XML or JSON)
- Uses resources specified in uniform resource identifiers (URIs)
- Uses the pre-defined verbs in the HTTP protocol (such as GET, POST, PUT, PATCH, and others)

However, while all RESTful APIs share some commonalities, it's important to note that RESTful does not mean "standard APIs." Different RESTful APIs may use different HTTP options and may handle HTTP verbs, methods of authentication, encoding, and message serialization in entirely different ways.

RESTCONF is an attempt to address these issues. It provides a uniform, standardized way for Web applications to access the configuration data, state data, data-model-specific Remote Procedure Call (RPC) operations, and event notifications within a network element.

The RESTCONF protocol operates on the configuration datastores defined in NETCONF. It defines a set of Create, Read, Update, Delete (CRUD) operations that can be used to access these datastores. The YANG language defines the syntax and semantics of datastore content, operational data, protocol operations, and REST operations that are used to access the hierarchical data within a datastore. In NETCONF, YANG data nodes are identified with XPath expressions, starting from the document root to the target resource. RESTCONF uses URI-encoded path expressions to identify the YANG data nodes.

RESTCONF Features

The RESTCONF protocol operates on a hierarchy of resources, each of which can be thought of as a collection of data and a set of allowed methods operating on that data. Resources are accessed via a set of URIs using syntax specified in RFC 8040. The set of YANG modules supported by the server determine the RPC operations, top-level data nodes, and event notification messages supported by the server.

The RESTCONF protocol does not include a data resource discovery mechanism. Instead, the definitions within the YANG modules advertised by the server are used to construct an RPC operation and data resource identifiers.

Resources

RESTCONF resources include:

- **Root Resource Discovery:** RESTCONF supports root resource discovery, allowing implementations to specify where the RESTCONF API resource is located. When first connected, clients retrieve the "/.well-known/host-meta" and use the link contained in the resource in subsequent RESTCONF requests.
- **RESTCONF Media Types:** The RESTCONF protocol defines two application-specific media types, yang-data+xml and yang-data+json, for encoding of the YANG data.
- **API Resource:** The RESTCONF API resource contains the root resource for the RESTCONF datastore and operation resources. It is the top-level resource located at "/restconf". The API resource has three child resources, as shown in Table 1.

Table 1. RESTCONF API Resources

Resource	Description
data	The data resource contains all data resources specified by the YANG-models supported by the RESTCONF server.
operation	The operation resource provides access to the data-model-specific RPC operations supported by the server.
yang-library-version	The yang-library-version resource identifies the revision date of the "ietf-yang-library" YANG module that is implemented by the server.

- **Datastore Resource:** The datastore resource represents the combined configuration and operational state data resources that can be accessed by a RESTCONF client. The datastore resource is handled by the RESTCONF server and cannot be created or deleted by clients.

- **Data Resource:** A data resource represents a YANG data node that is a descendant node of a datastore resource. Each YANG-defined data node can be uniquely targeted by the request-line of an HTTP method. Containers, leafs, leaf-list entries, and list entries are all data resources. Data nodes are identified using absolute XPath-expressions starting from the document root to the targets resource. List entries are identified by the name of the list followed by "=" and the value of the key(s).

- **Operation Resource:** An operation resource represents an RPC operation defined with the YANG "rpc" statement or a data-model-specific action defined with a YANG 1.1 "action" statement. An operation is invoked using a POST method on the resource. All operation resources representing RPC operations supported by the server are found in the "/restconf/operations" subtree, while operation resources representing YANG actions are identified in the "/restconf/data" subtree matching their location in the YANG-model.

- **Schema Resource:** Clients can retrieve YANG modules from the server. In order to retrieve a YANG module, a client first retrieves the URL for the relevant schema which is stored in the "schema" leaf in the module entry in the yang-library.

- **Event stream (notification) resource:** An event stream resource represents a source for system-generated event notifications. Each stream is created and modified by the server only. A client can retrieve a stream resource or initiate a long-poll server-sent event stream.

RESTCONF Methods

The RESTCONF protocol uses HTTP methods to identify the CRUD operations requested for a particular resource. Access control mechanisms are used to limit which RESTCONF CRUD operations can be used. In particular, RESTCONF is compatible with the NETCONF Access Control Model (NACM) [RFC6536]. The RESTCONF server converts the resource path to the corresponding YANG instance identifier and then applies the NACM access control rules to RESTCONF messages using this information.

Table 2 summarizes the various RESTCONF operations and how they map, when applicable, to their corresponding NETCONF operations.

Table 2. RESTCONF and NETCONF Operations

RESTCONF Operation	Description	Corresponding NETCONF Operation
HEAD	Get without a body	<none>
OPTION	Discover which operations are supported by a data resource	<none>
GET	Retrieve data and meta data	<get>, <get-config>
POST	Create a data resource	<edit-config> (nc:operation="create")
POST	Invoke an RPC operation	Call RPC directly
PUT	Create or replace a data resource	<edit-config> (nc:operation="create/replace"), <copy-config> (PUT on datastore)
PATCH	Create or update but not delete a data resource	<edit-config> (nc:operation depends on patch content)
DELETE	Delete a data resource	<edit-config> (nc:operation="delete")

RESTCONF methods include:

- **HEAD:** The HEAD method is used by the client to retrieve just the header fields (which contain the metadata for a resource) that would be returned for the comparable GET method, without the response message-body.
- **OPTION:** The OPTIONS method is used by the client to discover which methods are supported by the server for a specific resource.

- **GET:** The GET method is used by the client to retrieve data and metadata for a resource. It is supported for all resource types, except operation resources. The RESTCONF server only returns data the client is allowed to read.

- **POST:** The POST method is used by the client to create a data resource or invoke an operation resource. The server uses the target resource type to determine how to process the request. If the target resource is a datastore, a top-level configuration resource is created. If the target resource is a data resource, a child resource is created. And, if the target resource is an operation, the RPC is invoked. Insert and point query parameters give clients control over placement of new list and leaf-list elements when the lists are declared as "ordered-by user" in the YANG model. A POST request will fail if the target resource already exists.

- **PUT:** The PUT method is used by the client to create or replace the target data resource. Both the POST and PUT methods can be used to create data resources. The differences are that for POST, the client provides the resource identifier for the resource that will be created rather than its parent, and that PUT can be used when the resource already exists.

- **PATCH:** RESTCONF uses the HTTP PATCH method defined in [RFC7589] to provide an extensible framework for resource patching mechanisms. The PATCH method is used to modify existing resources. If the target resource doesn't exist, the PATCH request will fail. Plain PATCH can be used to create or update, but not delete, a child resource within the target resource. (Note: a more powerful patch mechanism, YANG-Patch, is currently being developed: [draft-ietf-netconf-yang-patch-14])

- **DELETE:** The DELETE method is used to delete a data resource. The data resource must exist. RESTCONF does not provide an operation corresponding to NETCONF's <edit-config> (nc:operation="remove"), which silently ignores attempts to delete non-existing resources.

Queries

RESTCONF allows clients to supply zero or more "query parameters" in the request URI. The query parameters can be used to tweak the meaning of the request in different ways, for example whether to retrieve only configuration data, only operational state data, or both with a GET request.

Each RESTCONF operation allows zero or more query parameters to be present in the request URI. Which specific parameters are allowed will depend on the resource type, and sometimes the specific target resource used in the request. The query parameters supported by RESTCONF are shown in Table 3.

Table 3. RESTCONF Query Parameters

Query Parameter	Methods	Description
content	GET, HEAD	select config and/or non-config resources
depth	GET, HEAD	request limited subtree depth in the reply content
fields	GET, HEAD	request a subset of the target resource content
filter	GET, HEAD	boolean notification filter for event stream resources
insert	PUT, POST	insertion mode for “ordered-by user” data resources
point	PUT, POST	insertion point for “ordered-by user” data resources
start-time	GET, HEAD	replay buffer start-time for event stream resources
stop-time	GET, HEAD	replay buffer stop-time for event stream resources
with-defaults	GET, HEAD	control the retrieval of default values

Messages

The RESTCONF protocol uses HTTP messages, and a single HTTP message corresponds to a single protocol method (Table 4). In general, messages can perform a single task on a single resource, such as retrieving a resource or editing a resource.

Table 4. RESTCONF HTTP Messages

RESTCONF URI Structure			
method	entry	resource	query
<OP>	</restconf>	<path>	<query>
mandatory	mandatory	mandatory	mandatory

RESTCONF messages are encoded in HTTP according to [RFC7230]. RESTCONF message content is encoded in either JSON or XML format and is sent in the HTTP message-body. The request input content encoding format is identified with the "Content-Type" header field. Clients may supply the "Accept" header field to indicate to the server which format it prefers for the reply.

Caching

Since datastore contents can change at unpredictable times, RESTCONF responses are usually not cached. Instead, clients rely on "Etag" and/or "Last-Modified" fields returned by the server in the HTTP-header to determine if it has the most resent version of a resource.

Notifications

The RESTCONF protocol supports YANG-defined event notifications similar to NETCONF notifications. A RESTCONF server that supports notifications provides a stream resource for each available notification stream. A RESTCONF client can retrieve the list of supported event streams from a RESTCONF server using the GET method on the "stream" list and use the gathered information to subscribe to notifications. Clients subscribe to RESTCONF events by sending an HTTP GET request for the URL representing the event with the "Accept" type "text/event-stream". The server will treat the connection as an event stream, using the Server-Sent Events [W3C.REC-eventsource-20150203] transport strategy.

Query parameters can be used to filter the event stream or control replay similar to NETCONF event notifications.

RESTCONF Limitations

While RESTCONF can be used to address southbound network elements using familiar RESTful operations, it does have limitations. For example, there are inconsistencies between JSON and XML when retrieving multiple instances from a list, which is possible with JSON but not possible with XML.

Additionally, the ordering of name/value pairs in a JSON object is undefined, which makes it more difficult to compare to configurations with simple text-based tools like grep. Thus, to compare configurations in JSON format, a tool that understands JSON syntax is needed.

RESTCONF Interactions with NETCONF

When a server supports both NETCONF and RESTCONF, the protocols can interact in relation to edit operations. It is, for example, possible that locks are in use on a RESTCONF server, even though RESTCONF doesn't provide any operations that can be used to manipulate these locks. In this case, the RESTCONF operations will not be granted write access to data resources within a datastore.

Another example is interactions between RESTCONF and NETCONF in how the configuration data stores are used. If the network element supports :candidate, all edits to configuration nodes in "/restconf/data" are performed in the candidate configuration datastore, and, per the RFC, the candidate is automatically committed to running immediately after each successful edit. Any edits from other sources that are in the candidate datastore will also be committed. Furthermore, if a confirmed commit procedure is in progress by a NETCONF client, then a RESTCONF commit will act as the confirming commit.

Can RESTCONF Replace NETCONF?

RESTCONF can provide a simple, effective mechanism to enable programmability in certain use cases—especially when used to allow northbound systems to communicate with SDN controllers. When used to directly address southbound network elements, however, it offers just a subset of NETCONF functionality. The concept of a "transaction" in RESTCONF, for example, is much narrower than in NETCONF. This, along with RESTCONF's lack of a mechanism to validate configuration changes, makes it far more limited than NETCONF for automating many aspects of complex, real-world environments.

For example, NETCONF features for datastores, locking, transactions (both local and network-wide), "remove" edit operation, and explicit validation are all missing in RESTCONF. The lack of transaction support, in particular, can have significant implications.

NETCONF's support for transactions means that whenever an operator makes a configuration change, the entire change set either succeeds or fails. This transaction support means that controllers and orchestrators don't have to worry about the order that changes are applied to network elements, vastly simplifying network element management. For these reasons, transaction support is in huge demand by network operators and service providers—who may hesitate to adopt network elements that don't provide it. RESTCONF has a much more limited notion of a transaction than NETCONF, because it doesn't allow a transaction to span multiple RESTCONF operations. Instead, every edit operation is its own transaction.

RESTCONF and NETCONF also differ in the ability to explicitly validate configuration changes. NETCONF supports validation of configuration changes such as types, allowed value ranges, and string formats, as well as semantic aspects, such as relationships between elements in the data model. RESTCONF only support validation as part of commits. This lack of explicit validation poses significant challenges in more complex environments, where correct and consistent network element configuration data is truly critical

for the correct operations of the element. Misconfiguring a network element may lead to a situation where the element is no longer connected to the network.

NETCONF also supports network wide transactions. That is, transactions that span multiple network elements, all changes are either successfully applied to all network elements, or no changes are applied to any element. In NETCONF, transactions spanning multiple network elements also use the same semantics as local transactions. Here again, RESTCONF is limited. With no support for the explicit use of the candidate datastore and no support for validation without committing, it can't support network-wide transactions.

When Does RESTCONF Make Sense?

There are certainly situations where RESTCONF is useful, even to directly address southbound network elements. In scenarios where northbound systems are only reading operation state data and statistics from the element—and not using it to change configurations, where transaction support becomes essential—RESTCONF is a simple, effective solution. Similarly, if RESTCONF is used exclusively for simple configuration changes—where it is acceptable to address elements one at a time, in separate individual RESTCONF operations—it can also be effective. For scenarios like these, when programmers are already well versed in the HTTP/REST model of interacting with servers, are fluent in JavaScript, XML, JSON, and other Web-technologies, and only need the simplified transactional model, then RESTCONF is a good choice.

In many, if not most other scenarios, full programmability will require features not supported by RESTCONF, such as support for locks, direct manipulation of datastores, and explicit validation. All of these cases can benefit from a solution with full transaction support, including network-wide transactions, so it is better to use NETCONF.

Conclusion

Market demand for programmability and automation will only grow in the coming years. Indeed, for network element vendors that don't want to be relegated to niche use cases, support for southbound configuration and management protocols will become a core customer requirement. While RESTCONF provides just a subset of NETCONF functionality, the emergence of the new standard should be viewed as an overall positive. It gives customers more options, and it demonstrates that network programmability has applicability well beyond large-scale service provider environments.

To capitalize on this market trend, however, network element vendors need to fully understand how RESTCONF can be used most effectively by their customers, as well as those scenarios where it's likely to fall short. To provide the most choice and flexibility, most vendors would be well served by enabling their elements for both RESTCONF and NETCONF. If choosing just one protocol to support however, vendors should choose NETCONF. Relying solely on RESTCONF—and giving customers an element cannot support full transactions, configuration validation, and other NETCONF features—could significantly limit the market for that product.

Get Started

The ConfD Management Agent from Tail-f makes it easy to enable both NETCONF and RESTCONF support, and deliver network elements that are suitable for a broad range of programmable network environments and use cases, from the simplest to the most advanced.

To learn more, visit http://www.tail-f.com/management-agent/.

Resources

RFC 8040 - RESTCONF
RFC 6241 - NETCONF
RFC 7950 – YANG

More Tail-f white papers and resources:

- NETCONF: The Programmable Interface for SDN and NFV
- ConfD from Tail-f: A Vital Piece of the NFV/SDN Puzzle
- Try ConfD for Free: ConfD Basic
- Watch ConfD Training Videos

tail-f

www.tail-f.com
info@tail-f.com

Corporate Headquarters

Sveavagen 25
111 34 Stockholm
Sweden
+46 8 21 37 40