YANG Patch and RESTCONF

tail-f a Cisco company

Table of Contents

1 **Introduction** 3

2 **REST** 3

3 **RESTCONF** 4

4 **YANG Patch** 5

5 **Conclusion** 7

6 **For More Information** 7

YANG Patch and RESTCONF

1. Introduction

REST APIs are a popular and ubiquitous style of interface implemented using standard HTTP methods, such as GET, POST, PUT and DELETE. These methods can be used to change and query resources represented by URIs (Uniform Resource Identifiers).

Given the popularity of REST APIs and the fact that many engineers are comfortable working with REST APIs, the IETF NETCONF Working Group developed RESTCONF (RFC 8040) as an alternative to NETCONF. RESTCONF follows RESTful principles and provides a standard REST interface that clients can use towards YANG data models, rather than non-standardized, proprietary REST interfaces that vary from device to device.

RESTCONF, as defined in RFC 8040, does have limitations when compared to NETCONF. One such limitation not being able to combine different CRUD (Create, Read, Update, Delete) operations against YANG instance data within a single operation. Using RESTCONF as defined in RFC 8040, this would require several different calls, when the underlying platform may want to see the changes as a single collection rather than a discrete set of changes.

In order to enable this combining of various operations into a single HTTP request, the IETF NETCONF Working Group created a new media type, the YANG Patch media type, which is used with the HTTP PATCH method. This application note will introduce and describe this new media type.

2. REST

REST has become a common and popular style for defining client-server interactions using the standard HTTP requests GET, PUT, POST, PATCH and DELETE. The various requests all take an identifier to a resource using a URI. Other than some general principles on the semantics of the HTTP methods, and some core principles, REST is an architectural style, and not a fixed protocol, so there is no official standard for a RESTful API.

This had led RESTful APIs to take many different forms at the discretion of the system designers. This is true also for device vendors, including physical and virtual devices, who may have designed REST APIs for their products. These REST APIs may have family resemblances, but the differences require developers creating clients which work with these REST APIs to be aware of the differences and have to write different code for different devices even in a single-vendor network.

3. RESTCONF

NETCONF is a powerful and flexible protocol for providing programmable device interfaces, but as device vendors sought to extend programmable interfaces into the enterprise and data center markets, they recognized the need to provide a standards-based protocol based on those market's experience with REST APIs. The IETF NETCONF Working Group created RESTCONF, RFC 8040, a REST API for manipulating YANG data models that could provide many but not all of the capabilities of NETCONF while staying true to RESTful API design principles.

In a previously published whitepaper, "Inside RESTCONF", we have provided detailed information about RESTCONF, including a comparison to NETCONF. Therefore in this application note, we will only provide a summary focusing on the HTTP operations that can modify the configuration and showing the corresponding NETCONF RPC for these operations.

The following table is taken from the table “CRUD Methods in RESTCONF” in section 4 “RESTCONF Methods” of RFC 8040.

RESTCONF	**NETCONF**
POST	<edit-config> (nc:operation=”create”)
PUT	<edit-config> (nc:operation=”create/replace”)
PATCH	<edit-config> (nc:operation depends on PATCH content)
DELETE	<edit-config> (nc:operation=”delete”)

While POST can be used to invoke an operation, here we will only focus on its use for creating a new data resource. With POST, if the data that we are attempting to create already exists, then the POST will fail.

PUT can also be used to create a new data resource. However, compared to POST, if the resource already exists, then PUT will replace it instead of reporting an error.

PATCH can be used to overwrite data with more precision than PUT or POST, but it cannot delete data.

DELETE will delete the specified target resource and will return an error if the resource does not exist, i.e. it will not silently ignore an attempt to delete non-existing resources.

4. YANG Patch

We have seen that each operation in NETCONF has a corresponding RESTCONF operation. However, one of the powers of NETCONF is that a single <edit-config> RPC can contain multiple different edit operations against any YANG instance data. With RESTCONF, this may require multiple operations and, due to the nature of RESTCONF, this will not be seen by the underlying platform as single transaction. This may require careful sequencing of operations, having to rollback previously successful operations if there is a later failure, and other issues that could be avoided if the operations could be combined in a single request.

It would be useful to be able to combine the capabilities of the different RESTCONF methods into a single operation, so that RESTCONF can handle an arbitrary set of updates.

In order to support this, the IETF NETCONF Working Group created the YANG Patch Media Type (RFC 8072) to be used with the PATCH HTTP method, that gives you the ability to specify an arbitrary set of operations. The media types defined in RFC 8072 are `application/yang-patch+xml` and `application/yang-patch+json`. You can choose to use either one depending on whether you want JSON or XML encodings of the YANG data.

Below is the YANG tree diagram for the `yang-patch` container as specified in the RFC. A “?” indicates an optional node and a "*" indicates a list with the list key in square brackets.

```
+---- yang-patch
      +---- patch-id string
      +---- comment? string
      +---- edit* [edit-id]
         +---- edit-id          string
         +---- operation        enumeration
         +---- target      target-resource-offset
         +---- point?      target-resource-offset
         +---- where?      enumeration
         +---- value?
```

In order to use YANG Patch, the RESTCONF client needs to supply a unique patch-id, which can be any string. This patch-id is returned in the response for correlation purposes. This is followed by an optional comment. The URI of the PATCH operation can specify the datastore resource itself or it can be a data resource within the datastore resource, so that each edit operation applies sub-resources within that data resource.

The edit operations themselves form an ordered collection of edits. Each individual edit operation has a unique edit-id for one of following edit operations taken from section 2.5 of RFC 8072:

Operation	Description
create	Create a new data resource, or return an error if it already exists
delete	Delete a data resource, or returnan error if it does not exist
insert	Insert a new user-ordered data resource
merge	Merge the value with the target, creating it if it does not exist
move	Reorder the target resource
replace	Replace the data resource with the value
remove	Delete a data if it exists, but no error if it does not exist

The `target` identifies the resource that this edit is to be applied to. If the URI specified in PATCH is a datastore resource, then `target` is treated as an absolute path expression. Otherwise, the `target` is treated as a relative path expression to the resource specified in the URI.

For insert or move operations, `point` and where are used to control the insertion or move point and where the operation is performed.

For those operations where data is to be supplied, `value` will contain the data.

The set of operations specified by YANG Patch will either all succeed or not be applied at all by the RESTCONF server. i.e. The contents of the PATCH will be treated as one transaction.

The RESTCONF server will return YANG Patch Status using a special YANG Patch Status media type. Below is the YANG tree diagram for the `yang-patch-status` container.

```
+---- yang-patch-status
     +---- patch-id      string
     +---- (global-status)?
     |  +--:(global-errors}
     |  |  +---- errors
     |  |      +---- error*
     |  |         +---- error-type           enumeration
     |  |         +---- error-tag            string
     |  |         +---- error-app-tag?       string
     |  |         +---- error-path?          instance-identifier
     |  |         +---- error-message?       string
     |  |         +---- error-info?
     |  +--:(ok)
     |     +---- ok?              empty
     +---- edit-status
        +---- edit* [edit-id]
            +---- edit-id      string
            +---- (edit=-status-choice)?
               +--:(ok)
```

```
|  +---- ok?          empty
+--:(errors)
   +---- errors
      +---- error*
         +---- error-type          enumeration
         +---- error-tag           string
         +---- error-app-tag?      string
         +---- error-path?         instance-identifier
         +---- error-message?      string
         +---- error-info?
```

The `patch-id` is the value supplied in the corresponding request and can be used to correlate the response with the request.

For a successful set of operations, the RESTCONF server can return a `global-status` of `ok`.

If there is a failure, then the RESTCONF server can return error information globally in `global-status` and for specific edit operations in `edit-status`.

There are several examples in Appendix A of RFC 8072, which you are encouraged to look at to see YANG Patch in action. There is also a ConfD example using YANG Patch in `$CONFD_DIR/examples.confd/restconf/yang-patch`. Please consult the README file in that directory for instructions on how to run the example.

5. Conclusion

RESTCONF provides a standard RESTful API that vendors can implement to make it easy for customers to work with a uniform REST interface across all devices rather than a diverse set of vendor-defined proprietary REST interfaces. One weakness of RESTCONF, as defined in RFC 8040, when compared with NETCONF, is in its inability to combine multiple different edit operations into a single operation and therefore a single transaction.

To remedy this, the IETF NETCONF Working Group created the YANG Patch Media Type in RFC 8072. Support for YANG Patch is not mandatory for a RESTCONF server, but for those RESTCONF servers that do implement it, YANG Patch provides a powerful technique for a client to provide updates to a RESTCONF servers. A client could even use YANG Patch to make all changes to a RESTCONF server and never have to resort to any of the other HTTP operations.

6. For More Information

For more information about ConfD, visit https://www.tail-f.com.

RFC 8040 "RESTCONF Protocol": https://datatracker.ietf.org/doc/html/rfc8040

RFC 8072 "YANG Patch Media Type": https://datatracker.ietf.org/doc/html/rfc8072

tail-f a Cisco company

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

Corporate Headquarters

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