Restconf Adapter
nb-bierman <-> RFC 8040

Summary

	Adaptation Requirement	Remarks
Common	1. Mapping of root resource from “/restconf” to “/rests” 2. Mapping of /<list-name>/<key> to /<list-name>=<key> in URL Same requirement for leaf-list **Note: Would need to obtain YANG Schema from odl-yang-tools and do parsing ourselves or look for some existing utilities. Also Parsing would need to consider applicable YANG constructs like Augmentation** 3. Retrieval of YANG schema: a. Adapt changes to URL b. Adapt change to header “accept” c. Adapt changes to response body 4. **Possible Blocker:** Response error message translation.	1. To explore further: For Response error message blocker, needs to cross check ODL implementation 2. Open Point: For content-type difference captured by Martin. Needs to cross verify with him.
Notification	1. Available Streams Discovery: Mapping of API and response body. 2. Event Notification Subscription **Note**: RFC 8040 has introduced concept of “access” in discovered stream and during subscription, client must subscribe using location information from “access”. As there could be multiple access, adaptation of this is NOT POSSIBLE and could be a **Possible Blocker.** 3. Get event notifications: Mapping of response body (only namespace)	1. Open Point**:** Needs to check ODL implementation to understand which RFC it follows for Notification. RFC 8650 is another RFC which defines dynamic subscriptions over RESTCONF [RFC8040]
RPC	1. Only root resource adaptation is needed.	1. Open Point: For namespace difference note captured by Martin, needs to cross verify with him.
Data: GET	1. URL adaptation/mapping 2. Query parameter mapping: “select mapped to fields” & “format mapped to “Accept” header”	1. Based on current understanding of RFC/Draft, response data of GET shouldn’t differ for 8040 & Bierman. But it would be good to cross-verify this in ODL/IETF YANG WG to ensure we are not leaving any border scenario

Summary

	Adaptation Requirement	Remarks
Data: Post/Put/Delete	1. URL adaptation would be needed.	1. Open Point: For namespace difference note captured by Martin in yang augment scenarios needs to be discussed.
Data: Plain Patch	1. URL adaptation. 2. Request body adaptation would be needed if ODL supports older version of bierman prior to bierman-04	1. Open Point: Bierman version support in ODL.
Data: YANG patch	1. API adaptation 2. Request Body also needs to be adapted for following a. Namespace b. Target if its a list **Note: This would need YANG schema based parsing to figure out target type in request body.**	

Notes:

1. RFC 8040 supports YANG version 1.1, but nb-bierman is only for YANG version 1.0. As adapter job is to support YANG applications using bierman, thereby additional features of RFC 8040 is not required considered for this analysis
2. Many parts of analysis is based on IETF documents (RFC 8040 & draft-bierman), but actual implementation of ODL could differ from the standard.
3. There is a possibility that some remote scenario would have been missed, would need to confirm few points with IETF YANG WG or with ODL community.

Suggestion: Implementing such adapter based on current analysis doesn't looks like a trivial job as would need YANG schema based adaptation/translation of URI and request body (for YANG Patch). Also solution for blockers needs to be found. Hence if we considering deprecating bierman after few releases, than leveraging ODL code would need much lesser efforts.

Common: API path

	RFC 8040	Bierman	Adapter Task
Root Resource Discovery	Top level resource is “/rests”	Top level resource is “/restconf”	Mapping of root resource from “/restconf” to “/rests”
List in resource identifiers (URL)	api-path = root *("/" (api-identifier / list-instance)) root = string ;; replacement string for {+restconf} api-identifier = [module-name ":"] identifier module-name = identifier list-instance = api-identifier "=" key-value *("," key-value) key-value = string ;; constrained chars are percent-encoded string = <an unquoted string> identifier = (ALPHA / "_") *(ALPHA / DIGIT / "_" / "-" / ".")	api-path = "/" | ("/" api-identifier 0*("/" (api-identifier | key-value))) api-identifier = [module-name ":"] identifier module-name = identifier key-value = string ;; An identifier MUST NOT start with ;; (('X'|'x') ('M'|'m') ('L'|'l')) identifier = (ALPHA / "_") *(ALPHA / DIGIT / "_" / "-" / ".") string = <an unquoted string>	Mapping of /<list-name>/<key> to /<list-name>=<key> 1. Would need to obtain YANG Schema from odl-yang-tools and do parsing ourselves or look for some existing utilities. Parsing would need to consider things like Augmentation 2. Do the URL conversion **Open Point:** Functionality with Leaf-list

Common: Schema resource

	RFC 8040	Bierman	Adapter Task

Retrieve Schema Resource (YANG modules supported by Server)	GET /restconf/data/ietf-yang-library:modules-state HTTP/1.1
	Host: example.com
	Accept: application/yang-data+json
	
	{
	"ietf-yang-library:modules-state": {
	"module-set-id": "5479120c17a619545ea6aff7aa19838b036ebbd7",
	"module": [
	{
	"name": "foo",
	"revision": "2012-01-02",
	"schema": "https://example.com/modules/foo/2012-01-02",
	"namespace": "http://example.com/ns/foo",
	"feature": [
	"feature1",
	"feature2"
	],
	"deviation": [
	{
	"name": "foo-dev",
	"revision": "2012-02-16"
	}
	],
	"conformance-type": "implement"
	}
	]
	}
	}
		GET /restconf/modules
		HTTP/1.1
		Host: example.com
		Accept: application/yang.api+json
		
		{
		"ietf-restconf:modules": {
		"module": [
		{
		"name": "foo",
		"revision": "2012-01-02",
		"namespace": "http://example.com/ns/foo",
		"feature": [
		"feature1",
		"feature2"
		]
		}
		]
		}
		}
			1. Adapt changes to URL
			2. Adapt change to header “accept”
			3. Adapt changes to response body
			
			Open Point: Whether ODL support this API

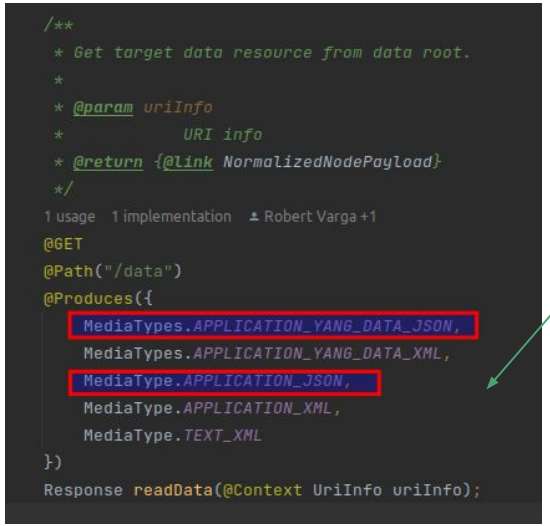
Common: Media type - Open Point

https://wiki.onap.org/display/DW/Switch+from+Biermann-RestConf+to+RFC8040+interface

Get NetConf Topology from **configuration** datastore

```
GET {{baseUrl}}/restconf/config/network-topology:network-topology/topology-netconf
Authorization: Basic {{user}} {{password}}
Accept: application/json
```

```
GET {{baseUrl}}/rests/data/network-topology:network-topology/topology=topology-netconf?content=config
Authorization: Basic {{user}} {{password}}
Accept: application/yang-data+json
```



```
/**
 * Get target data resource from data root.
 *
 * @param uriInfo
 *            URI info
 * @return {@link NormalizedNodePayload}
 */
1 usage  1 implementation  Robert Varga +1
@GET
@Path("/data")
@Produces({
    MediaTypes.APPLICATION_YANG_DATA_JSON,
    MediaTypes.APPLICATION_YANG_DATA_XML,
    MediaType.APPLICATION_JSON,
    MediaType.APPLICATION_XML,
    MediaType.TEXT_XML
})
Response readData(@Context UriInfo uriInfo);
```

In ODL implementation of 8040, API supports both application/json and application/yang-data+json media types. So Modifying the media types in adapter is not required.

Need to confirm with Martin Skorupski to see if we are missing something.

Common: Error Response message

	RFC 8040	Bierman	Adapter Task

Error Response
(Application for all requests)
	+--ro errors
	+--ro error*
	+--ro error-type enumeration
	+--ro error-tag string
	+--ro error-app-tag? string
	+--ro error-path? instance-identifier
	+--ro error-message? string
	+--ro error-info
	
	Example:
	{
	"ietf-restconf:errors": {
	"error": [
	{
	"error-type": "protocol",
	"error-tag": "invalid-value",
	"error-path": "/example-ops:input/delay",
	"error-message": "Invalid input parameter"
	}
	]
	}
	}
		+--ro errors
		+--ro error
		+--ro error-type enumeration
		+--ro error-tag string
		+--ro error-app-tag? string
		+--ro error-path? data-resource-identifier
		+--ro error-message? string
		+--ro error-info
		
		Example:
		{
		"ietf-restconf:errors": {
		"error": {
		"error-type": "protocol",
		"error-tag": "lock-denied",
		"error-message": "Lock failed, lock already held",
		"error-path": "/example-jukebox:jukebox/library
		/artist/Foo%20Fighters/album/Wasting%20Light
		/song/Burning%20Light"
		}
		}
		}
			This could be **BLOCKER** as error response for RFC 8040 its a list but in bierman it's a container (single instance).
			Not sure why RFC 8040 has changed it to list and its required in which all scenarios.
			
			Note: Would need to check ODL implementation to see in which scenario a multi instance or error is being sent as an output.

Notification

	RFC 8040	Bierman	Adapter Task
Retrieve Server supported event Stream	GET /restconf/data/ietf-restconf-monitoring:restconf-state/\ streams HTTP/1.1 Host: example.com Accept: application/yang-data+xml The server might send the following response: HTTP/1.1 200 OK Content-Type: application/yang-data+xml <?xml version="1.0" encoding="UTF-8"?> <streams xmlns="urn:ietf:params:xml:ns:yang:ietf-restconf-monitoring"> <stream> <name>NETCONF</name> <description>default NETCONF event stream</description> <replay-support>true</replay-support> <replay-log-creation-time>\ 2007-07-08T00:00:00Z\</replay-log-creation-time> <access> <encoding>xml</encoding> <location>https://example.com/streams/NETCONF\</location> </access> <access> <encoding>json</encoding> <location>https://example.com/streams/NETCONF-JSON\</location> </access> </stream> ... </streams>	GET /restconf/streams HTTP/1.1 Host: example.com Accept: application/yang.api+xml The server might send the following response: HTTP/1.1 200 OK Content-Type: application/yang-data+xml <?xml version="1.0" encoding="UTF-8"?> <streams xmlns="urn:ietf:params:xml:ns:yang:ietf-restconf"> <stream> <name>NETCONF</name> <description>default NETCONF event stream</description> <replay-support>true</replay-support> <replay-log-creation-time>2007-07-08T00:00:00Z</replay-log- creation-time> <events /> </stream> ... </streams>	• Both API & Response body needs adaptation RFC 8040 has introduced concept of “access” and during subscription client must subscribe using location. As there could be multiple access, adaptation of this is **NOT POSSIBLE and could be a BLOCKER.**

Notification

	RFC 8040	Bierman	Adapter Task
Subscribing to receive notifications	The RESTCONF client can then use this URL value to start monitoring the event stream: GET /streams/NETCONF HTTP/1.1 Host: example.com Accept: text/event-stream Cache-Control: no-cache Connection: keep-alive	GET /restconf/streams/stream/NETCONF/events HTTP/1.1 Host: example.com Accept: text/event-stream Cache-Control: no-cache Connection: keep-alive	RFC 8040 has introduced concept of “access” and during subscription client must subscribe using location. As there could be multiple access, adaptation of this is **NOT POSSIBLE and could be a BLOCKER.** **Open Point:** Needs to check ODL implementation to understand which RFC it follows for Notification. RFC 8650 is another RFC which defines dynamic subscriptions over RESTCONF [RFC8040]
Receiving event notifications	<?xml version="1.0" encoding="UTF-8"?> <notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0"> <eventTime>2013-12-21T00:01:00Z</eventTime> <event xmlns="http://example.com/event/1.0"> <event-class>fault</event-class> <reporting-entity> <card>Ethernet0</card> </reporting-entity> <severity>major</severity> </event> </notification>	<?xml version="1.0" encoding="UTF-8"?> <notification xmlns="urn:ietf:params:xml:ns:yang:ietf-restconf"> <event-time>2013-09-30T00:01:00Z</event-time> <event xmlns="http://example.com/event/1.0"> <eventClass>fault</eventClass> <reportingEntity> <card>Ethernet0</card> </reportingEntity> <severity>major</severity> </event> </notification>	1. Namespace in response body needs adaptation.

RPC operation

RFC 8040	Bierman	Adapter Task

POST {+restconf}/operations/<operation>
The <operation> field identifies the module name and rpc identifier string for the desired operation.
 	
POST /restconf/operations/example-ops:reboot HTTP/1.1
Host: example.com
Content-Type: application/yang-data+json
{
"example-ops:input" : {
"delay" : 600,
"message" : "Going down for system maintenance",
"language" : "en-US"
}
}
	POST /restconf/operations/example-ops:reboot HTTP/1.1
	Host: example.com
	Content-Type: application/yang.data+json
	{
	"example-ops:input" : {
	"delay" : 600,
	"message" : "Going down for system maintenance",
	"language" : "en-US"
	}
	}
		1. Only root resource adaptation is needed.

https://wiki.onap.org/display/DW/Switch+from+Biermann-RestConf+to+RFC8040+interface

Create new NetworkElement connection with SDN-R **data-provider**.

```
POST {{baseUrl}}/restconf/operations/data-provider:create-network-element-connection
Authorization: Basic {{user}} {{password}}
Accept: application/json
Content-Type: application/json

{
  "data-provider:input": {
    "id": "new-mountpoint-name",
    "node-id": "new-mountpoint-name",
    "host": "10.10.10.10",
    "port": "830",
    "username": "netconf",
    "password": "netconf",
    "is-required": "true"
  }
}

# Note: the http body is the same, but pay attention to the namespace of the input and output object.
```

```
POST {{baseUrl}}/rests/operations/data-provider:create-network-element-connection
Authorization: Basic {{user}} {{password}}
Accept: application/yang-data+json
Content-Type: application/yang-data+json

{
  "data-provider:input": {
    "id": "new-mountpoint-name",
    "node-id": "new-mountpoint-name",
    "host": "10.10.10.10",
    "port": "830",
    "username": "netconf",
    "password": "netconf",
    "is-required": "true"
  }
}

# Note: the http body is the same! Just the URL is different.
```

Open Point

Needs discussion with Martin Skorupski

Needs to understand the Note provided with this example

Data: GET

Note: Based on current understanding of RFC/Draft, response data of GET shouldn't differ for 8040 & Bierman. But it would be good to cross-verify this in ODL/IETF YANG WG to ensure we are not leaving any border scenario.

	RFC 8040	Bierman	Adapter Task

To retrieve configuration data	GET
	/restconf/data/example-events:events?content=config
	HTTP/1.1
	Host: example.com
	Accept: application/yang-data+json
	
	The server might respond as follows:
	{
	"example-events:events": {
	"event": [
	{
	"name": "interface-up",
	"description": "Interface up notification count"
	}
	]
	}
	}
		GET /restconf/config/example-jukebox:jukebox/
		library/artist/Foo%20Fighters/album?format=json HTTP/1.1
		Host: example.com
		Accept: application/yang.data+json
		
		The server might respond:
		{
		"album": {
		"name": "Wasting Light",
		"genre": "example-jukebox:alternative",
		"year": 2011
		}
		}
			URL adaptation (translation)

To retrieve only non configuration data	GET
	/restconf/data/example-events:events?content=nonconfig
	Host: example.com
	Accept: application/yang-data+json
	The server might respond as follows:
	{
	"example-events:events": {
	"event": {
	"name": "interface-up",
	"event-count": 42
	}
	}
	}
		GET /restconf/operational/example-jukebox:jukebox/library
		Host: example.com
		Accept: application/yang.data+json
		The server might respond:
		{
		"example-jukebox:library": {
		"artist-count": 42,
		"album-count": 59,
		"song-count": 374
		}
		}
			URL adaptation (translation)

Data: Get: Query parameters

	RFC 8040	Bierman	Adapter Task
Select (beieram) Fields (RFC8040)	The "fields" query parameter is used to optionally identify data nodes within the target resource to be retrieved in a GET method. For example, assume that the target resource is the "album" list. retrieve only the "label" and "catalogue-number" of the "admin" container within an album, use "fields=admin(label;catalogue-number)". "/" is used in a path to retrieve a child node of a node. example, to retrieve only the "label" of an album, use "fields=admin/label"	The "select" query parameter is used to specify an expression which can represent a subset of all data nodes within the target resource. GET /restconf?select=version&format=json HTTP/1.1 Host: example.com Accept: application/yang.api+json The server might respond as follows. { "ietf-restconf:version": "1.0" }	• Map “select” to “fields”
format	Not supported	• The "format" parameter is used to specify the format of any content returned in the response. • This parameter MAY be used instead of the "Accept" header to identify the format desired in the response. • The "format" parameter is only supported for the GET and HEAD methods. • If the "format" parameter is present, then it overrides the Accept header, if present. Example GET /restconf/config/example-routing:routing?format=json Host: example.com	• Map “format” to “Accept” header

Data: POST, Delete & PUT

	RFC 8040	Bierman	Adapter Task

Create a new data resource	POST /restconf/data/example-jukebox:jukebox/library HTTP/1.1
	Host: example.com
	Content-Type: application/yang-data+json
	
	{
	"example-jukebox:artist": {
	"name": "Foo Fighters"
	}
	}
		POST /restconf/config/example-jukebox:jukebox/library HTTP/1.1
		Host: example.com
		Content-Type: application/yang.data+json
		
		{
		"example-jukebox:artist": {
		"name": "Foo Fighters"
		}
		}
			URL adaptation (translation)

Delete an existing resource	DELETE /restconf/data/example-jukebox:jukebox/\ library/artist=Foo%20Fighters/album=Wasting%20Light HTTP/1.1 Host: example.com	DELETE /restconf/config/example-jukebox:jukebox/ library/artist/Foo%20Fighters/album/Wasting%20Light HTTP/1.1 Host: example.com	URL adaptation (translation)

Replace an existing resource	PUT /restconf/data/example-jukebox:jukebox/\
	library/artist=Foo%20Fighters/album=Wasting%20Light
	Host: example.com
	Content-Type: application/yang-data+json
	{
	"example-jukebox:album":
	{
	"name": "Wasting Light",
	"genre": "example-jukebox:alternative",
	"year": 2011
	}
	}
		PUT /restconf/config/example-jukebox:jukebox/
		library/artist/Foo%20Fighters/album/Wasting%20Light
		Host: example.com
		If-Match: b3830f23a4c
		Content-Type: application/yang.data+json
		{
		"example-jukebox:album": {
		"name": "Wasting Light",
		"genre": "example-jukebox:alternative",
		"year": 2011
		}
		}
			URL adaptation (translation)

Data: POST, Delete & PUT - Open Point

https://wiki.onap.org/display/DW/Switch+from+Biermann-RestConf+to+RFC8040+interface

Edit Config operation:merge

```
@node=Core14-ONF-NTS-Manager

PUT {{baseUrl}}/restconf/config/network-topology:network-topology/topology/topology-netconf/node/{{node}}/yang-ext:mount/network-topology-simulator:simulator-config/notification-config
Authorization: Basic {{user}} {{password}}
Accept: application/json
Content-Type: application/json

{
  "network-topology-simulator:notification-config": {
    "network-topology-simulator:is-netconf-available": true,
    "network-topology-simulator:ves-heartbeat-period": 0,
    "network-topology-simulator:is-ves-available": false,
    "network-topology-simulator:fault-notification-delay-period":
[
      60, 50, 40, 10
    ]
  }
}

# Pay attention to the namespaces!!!
```

```
@node=Core14-ONF-NTS-Manager

PUT {{baseUrl}}/rests/data/network-topology:network-topology/topology=topology-netconf/node={{node}}/yang-ext:mount/network-topology-simulator:simulator-config/notification-config
Authorization: Basic {{user}} {{password}}
Accept: application/yang-data+json
Content-Type: application/yang-data+json

{
  "network-topology-simulator:notification-config": {
    "is-netconf-available": true,
    "ves-heartbeat-period": 0,
    "is-ves-available": false,
    "fault-notification-delay-period": [
      60, 50, 40, 10
    ]
  }
}

# Pay attention to the namespaces!!!
```

This is an YANG Augmentation scenario. In which module name for every leaf looks unnecessary.

Need to check with Martin on why module-name (namespace) needs to be specified for every YANG leaf, because it is already specified for top level container “notification-config” for nb-bierman.

Ideally XML/JSON serialization/deserialization of data shouldn’t change for bierman and RFC 8040, can also cross check with ODL community.

Data: POST, Delete & PUT - Open Point

https://wiki.onap.org/display/DW/Switch+from+Biermann-RestConf+to+RFC8040+interface

Create MountPoint

This is again YANG Augmentation scenario, In which module name for every leaf looks necessary.

Need to check with Martin on why module-name (namespace) NEED NOT to be specified for every Augmented YANG leaf for RFC 8040.

Ideally XML/JSON serialization/deserialization of data shouldn't change for bierman and RFC 8040, can also cross check with ODL community.

```
@node=my-new-mount-point-2
PUT {{baseUrl}}/restconf/config/network-topology:network-topology/topology/topology-netconf/node/{{node}}
Authorization: Basic {{user}} {{password}}
Accept: application/json
Content-Type: application/json

{
  "network-topology:node": [
    {
      "node-id": "{{node}}",
      "netconf-node-topology:host": "127.0.0.1",
      "netconf-node-topology:port": 830,
      "netconf-node-topology:password": "netconf",
      "netconf-node-topology:username": "netconf",
      "netconf-node-topology:sleep-factor": 1.5,
      "netconf-node-topology:tcp-only": false,
      "netconf-node-topology:reconnect-on-changed-schema": true,
      "netconf-node-topology:default-request-timeout-millis": 60000,
      "netconf-node-topology:connection-timeout-millis": 20000,
      "netconf-node-topology:max-connection-attempts": 100,
      "netconf-node-topology:between-attempts-timeout-millis": 2000,
      "netconf-node-topology:keepalive-delay": 120,
      "netconf-node-topology:concurrent-rpc-limit": 0,
      "netconf-node-topology:actor-response-wait-time": 5
    }
  ]
}
```

```
@node=my-new-mount-point-2
PUT {{baseUrl}}/rests/data/network-topology:network-topology/topology=topology-netconf/node={{node}}
Authorization: Basic {{user}} {{password}}
Accept: application/yang-data+json
Content-Type: application/yang-data+json

{
  "network-topology:node": [
    {
      "node-id": "{{node}}",
      "host": "127.0.0.1",
      "port": 830,
      "password": "netconf",
      "username": "netconf",
      "sleep-factor": 1.5,
      "tcp-only": false,
      "reconnect-on-changed-schema": true,
      "default-request-timeout-millis": 60000,
      "connection-timeout-millis": 20000,
      "max-connection-attempts": 100,
      "between-attempts-timeout-millis": 2000,
      "keepalive-delay": 120,
      "concurrent-rpc-limit": 0,
      "actor-response-wait-time": 5
    }
  ]
}
```

Data: Plain Patch

	RFC 8040	Bierman - 02	Adapter Task
Plain Patch an existing resource(merge)	To replace just the "year" field in the "album" resource (instead of replacing the entire resource with the PUT method), the client might send a plain patch as follows: PATCH /restconf/data/example-jukebox:jukebox/\ library/artist=Foo%20Fighters/**album**=Wasting%20Light <**album** xmlns="http://example.com/ns/example-jukebox"> <year>2011</year> </**album**>	PATCH /restconf/config/example-jukebox:jukebox/ library/artist/Foo%20Fighters/**album**/Wasting%20Light HTTP/1.1 { "example-jukebox:year" : 2011 } **Note: In Bierman02 body is like shown above. In bierman04 body is same with RFC8040.**	Needs to check implementation in ODL for Patch follows Bierman-02 or Bierman-04 Based on that what to adapt can be figured out.

Data: YANG Patch

	RFC 8040	Bierman	Adapter Task

YANG Patch	PATCH /restconf/data/example-jukebox:jukebox/\
	
	library/artist=Foo%20Fighters/album=Wasting%20Light
	\
	HTTP/1.1
	Host: example.com
	Accept: application/yang-data+json
	Content-Type: application/yang-patch+json
	
	{
	"ietf-yang-patch:yang-patch": {
	"patch-id": "add-songs-patch-2",
	"edit": [
	{
	"edit-id": "edit1",
	"operation": "create",
	"target": "/song=Rope",
	"value": {
	"song": [
	{
	"name": "Rope",
	"location": "/media/rope.mp3",
	"format": "MP3",
	"length": 259
	}
	]
	}
	}
	]
	}
	}
		PATCH
		/restconf/config/example-jukebox:jukebox/
		library/artist/Foo%20Fighters/album/Wasting%2
		0Light
		HTTP/1.1
		Host: example.com
		Accept: application/yang.patch-status+json
		Content-Type: application/yang.patch+json
		
		{
		"ietf-restconf:yang-patch": {
		"patch-id": "add-songs-patch-2",
		"edit": [
		{
		"edit-id": 1,
		"operation": "create",
		"target": "/song",
		"value": {
		"song": {
		"name": "Rope",
		"location": "/media/rope.mp3",
		"format": "MP3",
		"length": 259
		}
		}
		}
		]
		}
		}
			1. API adaptation
			2. Request Body also needs to be adapted for following
			a. Namespace
			b. Target if its a list
			Note: This would need YANG schema based parsing to figure out target type.

Thanks