Technical Guide

NETCONF and RESTCONF

Feature Overview and Configuration Guide

Introduction

This guide describes the network management protocols NETCONF and RESTCONF and how to configure them on AlliedWare Plus™ devices.

Both of these protocols are built upon a network management framework outlined in RFC 6244. This framework employs YANG data models (defined in RFC 6020 and RFC 7950) to represent configuration and device states. Additionally, it uses the NETCONF protocol (defined in RFC 6241) to enable client software to interact with and modify the data models.

RFC 8040 describes RESTCONF, another protocol that can be used to manipulate data models expressed in the same way as for NETCONF. While these protocols are different in many ways, there are also many similarities between them, not least of which is the commonality of YANG-based data models.

These protocols are used to modify device configurations by altering data represented by YANG models. They are also employed to retrieve configurations and read device status by accessing data defined in YANG models. These operations are carried out through a limited set of RPC (Remote Procedure Call) calls, which can deal with extensive data, depending on the device's supported data models.

Products and software version that apply to this guide

This guide applies to AlliedWare Plus™ products that support NETCONF and RESTCONF, running version **5.5.3-2.1** or later. To see whether your product supports these protocols, see the following documents:

- The product's Datasheet
- The product's Command Reference

These documents are available from the above links on our website at alliedtelesis.com

Licensing

All products that run AlliedWarePlus software have NETCONF and RESTCONF included in their builds. Use of NETCONF and RESTCONF is not subject to a feature license.

Contents

Feature overview

NETCONF and RESTCONF offer standardized methods for managing network devices. These protocols themselves are standardized, and to some degree, the data models they work with are standardized as well.

Data models have three primary sources:

1. The IETF (Internet Engineering Task Force) provides standard data models.
2. The OpenConfig Consortium has created numerous models, often covering similar areas as the IETF.
3. Vendors commonly develop specific models tailored to their devices.

Some of the benefits of using NETCONF and RESTCONF include:

- They allow users to employ standards based tools to manage networks, and use of standard models allows consistent management of devices across the network, even when those devices are produced by different vendors.
- They offer a programmable management interface. While it's possible to achieve this with CLI-based tools, using a standard protocol and well-defined YANG data models simplifies the process of developing management applications and makes them more widely applicable without requiring extensive modifications.
- They operate in a client-server model, with the network device being managed acting as the server, and the management applications acting as the clients.
- The protocols are secure, using SSH (NETCONF) and HTTPS (RESTCONF) as their transport layers, and user authentication can be carried out with a variety of methods.
- Because they use standard protocols, there are many tools available for helping create client applications, including development libraries and full blown applications.
- They can be tied into users' provisioning systems or management applications. The protocols are well suited to the task of gathering network telemetry, being programmable and carrying data in the standard formats of XML (NETCONF) and Json (RESTCONF).

How the NETCONF and RESTCONF features work

NETCONF

The NETCONF server runs passively on the network device, waiting for sessions to be initiated by clients.

A typical session will run as follows:

- The client (for example software running on a PC) starts an SSH connection to the server (the network device being managed).
- On successful establishment of the SSH connection, the client and server exchange NETCONF Hello messages.
- The client makes RPC calls to retrieve data or edit the device's configuration.
- The client makes a <close-session> RPC call to terminate the NETCONF session.

The exchange of Hello messages is an important step in the session. The client and server exchange capabilities and negotiate the protocol version to be used. In particular, the server (network device being managed) lists the data models it supports, including release information, so the client knows exactly how the device is to be managed. A list of capabilities is provided, and since many of these capabilities are optional, this is essential information for the client to manage the device.

After a successful Hello exchange, the client can send RPC calls to the server to perform various actions. The NETCONF server responds with either an OK message, an error message, or data from the server's modelled data.

The RPC calls (i.e. operations) available to the Allied Telesis implementation of NETCONF include:

OPERATION	DESCRIPTION
<get>	Retrieve running configuration and device state information.
<get-config>	Retrieve all or part of a specified configuration data store.
<edit-config>	Edit a configuration data-store by creating, deleting, merging, or replacing content.
<lock>	Lock an entire configuration data-store of a device.
<unlock>	Release a configuration data-store lock previously obtained with the <lock> operation.
<close-session>	Request graceful termination of a NETCONF session.
<kill-session>	Force the termination of a NETCONF session.

Note that a number of standard RPC calls are not supported in the Allied Telesis implementation:

OPERATION	DESCRIPTION
<copy-config>	Copy an entire configuration data-store to another configuration data-store.
<delete-config>	Delete a configuration data-store.

These are not supported due to the fact that the Allied Telesis implementation only supports the <running> data-store, so the concepts of copy-config and delete-config are not relevant.

Note the difference between <close-session> and <kill-session> RPC calls. The <close-session> is sent by a client to close the session on which the request is made. In contrast, the <kill-session> RPC call is made to terminate a different session.

RESTCONF

The RESTCONF server runs passively on the network device, waiting for sessions to be initiated by clients.

A typical session will run as follows:

- The client (for example software running on a PC) starts an HTTPS connection to the server (the network device being managed).
- On successful establishment of the HTTPS connection, the client can request the root of the RESTCONF API by getting the URI "/.well-known/host-meta".
- The client can then request a list of YANG models supported by the server by accessing the data stored in the ietf-yang-library.
- The client makes calls to retrieve data or edit the device's configuration using a simple set of methods such as GET, DELETE, PATCH, OPTIONS, HEAD, POST and PUT.
- The client terminates the RESTCONF session by closing the HTTPS session.

The RESTCONF standard allows implementations to specify where the RESTCONF API is located. As the URI for RESTCONF is defined by the implementation, a syntax “{+restconf}” is often used to refer to the root of the RESTCONF data.

OPERATION	DESCRIPTION
GET	Retrieve running configuration and device state information.
DELETE	Delete a target resource specified by the supplied path.
HEAD	Retrieve just the header fields (which contain metadata for a resource).
OPTIONS	Retrieve the list of supported methods on the server.
PATCH	Merge the contents of the message-body with the target resource.
POST	Create a data resource.
PUT	Create or replace a target data resource.

Limitations

NETCONF has a lot of moving parts, and has been extended since its original definition. A number of the features and capabilities of NETCONF are optional and in a few cases, Allied Telesis has also deviated slightly from the standard. This section documents the most important areas in which the Allied Telesis implementation of NETCONF may differ from other implementations.

There are a number of reasons why Allied Telesis may have chosen to deviate from full compliance in certain areas, including, but not limited to:

- **Performance** - we have decided not to implement some NETCONF features on certain Allied Telesis devices with relatively low-powered CPUs to enhance their performance, as these features demand significant processing power.
- **Utility** - some of the NETCONF features are more obscure, and Allied Telesis feels that not all of these are all that useful to users.
- **Configuration architecture** - Allied Telesis devices have a configuration model that supports only a running configuration. In contrast, NETCONF supports startup and candidate configuration as well. Until such time as Allied Telesis changes its architecture to support candidate and startup configuration, there does not seem to be much point in supporting it in the protocol. Note that these are optional features, and that at least two other management protocols, RESTCONF and gNMI, support only a running configuration.

NETCONF

Supported

As reported in the Hello message, the following capabilities are supported in Allied Telesis's current implementation of NETCONF:

- v1.1 of NETCONF, but not v1.0
  `urn:ietf:params:netconf:base:1.1`
- writing of configuration to the running data-store.
  `urn:ietf:params:netconf:capability:writable-running:1.0`
- the with-defaults capability as defined in RFC 6243.
  `urn:ietf:params:netconf:capability:with-defaults:1.0`

Not supported

The following capabilities are not supported in Allied Telesis's current implementation of NETCONF:

- a candidate data-store is not consistent with the configuration architecture of AlliedWare Plus.
  `urn:ietf:params:netconf:capability:candidate:1.0 - .`
- confirmed-commit is likely to impact the performance of some Allied Telesis devices. It is also considered to not provide as much utility to users as more basic capabilities.
  `urn:ietf:params:netconf:capability:confirmed-commit:1.1`
- rollback-on-error is not consistent with the configuration architecture of AlliedWare Plus.
  `urn:ietf:params:netconf:capability:rollback-on-error:1.0`
- confirmed-commit is likely to impact the performance of some Allied Telesis devices.
  `urn:ietf:params:netconf:capability:validate:1.1`
- startup is not consistent with the configuration architecture of AlliedWare Plus.
  `urn:ietf:params:netconf:capability:startup:1.0`
- startup is not consistent with the configuration architecture of AlliedWare Plus.
  `urn:ietf:params:netconf:capability:url:1.0`
- xpath is likely to impact the performance of some Allied Telesis devices and is not supported in software version 5.5.3-2, but will be supported from version 5.5.4-0 onward.
  `urn:ietf:params:netconf:capability:xpath:1.0`

RPCs not supported

The following RPCs are not consistent with the configuration architecture of AlliedWare Plus:

- <copy-config>, <delete-config> - (RFC 6241)
- <discard-changes>, <validate>, <commit>, <cancel-commit> - (RFC 6241)
- <get-data>, <edit-data> - (RFC 8526)

Likely to affect performance on some Allied Telesis devices:

- <create-subscription>, <notification> - (RFC 5277).
- <get-schema> - (RFC 6022)

The NETCONF server can support multiple simultaneous sessions, but there is a limit to their number. The default limit is 4 simultaneous sessions, with the number being settable to any number in the range of 1 to 10. Creating a NETCONF session that exceeds the active sessions limit will result in termination with an error.

RESTCONF

There are no known limitations to the RESTCONF implementation.

How to use NETCONF

To use NETCONF, all that is required is to set up the SSH server aspects of NETCONF and to enable NETCONF, assuming other aspects of configuration such as IP addresses and routing are configured correctly. Since NETCONF uses SSH as a transport layer, the commands required to set up NETCONF mirror those used to setup the SSH server.

The general steps required are as follows:

- Set up basic IP connectivity
- Add users and user authentication options
- Add allowed and denied users for NETCONF
- Configure logging and maximum sessions options for NETCONF
- Enable the NETCONF server

At this point the device will be waiting for clients to start NETCONF sessions by creating SSH sessions to the relevant TCP port (which for NETCONF is port 830). Once enabled, the operation of NETCONF can be monitored by inspecting the system log (which will show messages for whichever RPC calls have been configured to be logged) or running the commands to show NETCONF sessions or counters.

NETCONF is available on all AlliedWare Plus devices, and does not require a license on any device. Netconf operates the same way on all AlliedWare Plus devices, but different devices will show different information in their Hello packets, based on the data models supported and in some cases on which features in the data model are supported.

How to use RESTCONF

To use NETCONF, all that is required is to enable HTTP and RESTCONF services and setup any PKI certificates required for authentication.

The general steps required are as follows:

- Set up basic IP connectivity
- Enable the HTTP service
- Add users and user authentication options
- Set the HTTP secure port if this is not going to be set to 443
- Configure any required HTTP trustpoints
- Enable the RESTCONF server

At this point the device will be waiting for clients to start RESTCONF sessions. Once enabled, the operation of RESTCONF can be monitored by inspecting the system log (which will show messages for whichever RPC calls have been configured to be logged).

RESTCONF is available on all AlliedWare Plus devices, and does not require a license on any device. RESTCONF operates the same way on all AlliedWare Plus devices, but different devices can show different information, based on the data models supported and in some cases on which features in the data model are supported.

Configuration examples

NETCONF

The following commands configure a device from a blank state. This program initiates four sessions and performs various RPCs.

The network device communicates with the network via its eth0 interface, address 198.51.100.6. The server is at address 203.0.113.33. The router device used to provide network access is at 198.51.100.1, with subnet mask of 255.255.255.0 (/24).

```
awplus>ena
awplus#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
awplus(config)#interface eth2
awplus(config-if)#ip address 198.51.100.6/24
awplus(config-if)#exit
awplus(config)#ip route 203.0.113.0/24 198.51.100.1
awplus(config)#user netconf priv 15 password netconf_test
awplus(config)#netconf allow-users netconf 203.0.113.33
awplus(config)#netconf log get
awplus(config)#netconf log lock
awplus(config)#service netconf
awplus(config)#exit
awplus#copy run netconf_test.cfg
Copying...
Successful operation
awplus#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
awplus(config)#boot config-file netconf_test.cfg
awplus(config)#exit
```

RESTCONF

The following commands configure a device from a blank state. The network device communicates with the network via its eth0 interface, address 198.51.100.6. The server is at address 203.0.113.33. The router device used to provide network access is at 198.51.100.1, with subnet mask of 255.255.255.0 (/24).

While RESTCONF uses HTTP as its communication protocol, it is strongly recommended that the secure version of the protocol is used (HTTPS). The instructions here deliberatly turn off the non-secure version of HTTP:

```
awplus>ena
awplus#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
awplus(config)#interface eth2
awplus(config-if)#ip address 198.51.100.6/24
awplus(config-if)#exit
awplus(config)#ip route 203.0.113.0/24 198.51.100.1
awplus(config)#user rest priv 15 password restconf_test
awplus(config)#http port none
awplus(config)#service http
awplus(config)#service restconf
awplus(config)#exit
awplus#copy run restconf_test.cfg
Copying...
Successful operation
awplus#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
awplus(config)#boot config-file restconf_test.cfg
awplus(config)#exit
```

Monitoring

NETCONF

This is an example output of the show commands before sessions start:

```
awplus#show netconf session
No Netconf sessions found
Maximum number of simultaneous sessions: 4

awplus#show netconf counters
Netconf counters
Global counters
-----------------------------------------------
 Netconf start time: 2023-10-03T04:04:15Z+00:00
 In bad Hellos:                0
 Sessions started:             0
 Dropped sessions:             0
 In RPCs:                      0
 In bad RPCs:                  0
 Out RPC errors:               0
 Out Notifications:            0
-----------------------------------------------
No Netconf sessions found
```

Once sessions start running, we see the following output:

```
show netconf session
Netconf sessions
ID        Username     Lock  Login time (UTC)            Remote address
--------------------------------------------------------------------------------
24734     netconf      R     2023-10-03T04:05:00Z+00:00  203.0.113.33:55930
24735     netconf      -     2023-10-03T04:05:02Z+00:00  203.0.113.33:55946
24736     netconf      -     2023-10-03T04:05:04Z+00:00  203.0.113.33:40712
24737     netconf      -     2023-10-03T04:05:06Z+00:00  203.0.113.33:40718
--------------------------------------------------------------------------------
Maximum number of simultaneous sessions: 4

awplus#show netconf counters
Netconf counters
Global counters
-----------------------------------------------
 Netconf start time: 2023-10-03T04:04:15Z+00:00
 In bad Hellos:                0
 Sessions started:             5
 Dropped sessions:             1
 In RPCs:                      1
 In bad RPCs:                  0
 Out RPC errors:               4
 Out Notifications:            0
-----------------------------------------------
Session counters
ID        Username     Login time (UTC)            Remote address
--------------------------------------------------------------------------
24734     netconf      2023-10-03T04:05:00Z+00:00  203.0.113.33:55930
 In RPCs:                      1
 In bad RPCs:                  0
 Out RPC errors:               1
 Out Notifications:            0
24735     netconf      2023-10-03T04:05:02Z+00:00  203.0.113.33:55946
 In RPCs:                      0
 In bad RPCs:                  0
 Out RPC errors:               1
 Out Notifications:            0
24736     netconf      2023-10-03T04:05:04Z+00:00  203.0.113.33:40712
 In RPCs:                      0
 In bad RPCs:                  0
 Out RPC errors:               1
 Out Notifications:            0
24737     netconf      2023-10-03T04:05:06Z+00:00  203.0.113.33:40718
 In RPCs:                      0
 In bad RPCs:                  0
 Out RPC errors:               1
 Out Notifications:            0
--------------------------------------------------------------------------
```

Clearing a NETCONF session

To clear a session, use the command **clear netconf session**.

For example, to clear session ID 24741:

```
awplus#show netconf session
Netconf sessions
ID         Username     Lock  Login time (UTC)            Remote address
--------------------------------------------------------------------------------
24739      netconf      -     2023-10-03T04:09:01Z+00:00  203.0.113.33:59910
24740      netconf      -     2023-10-03T04:09:03Z+00:00  203.0.113.33:59922
24741      netconf      R     2023-10-03T04:09:05Z+00:00  203.0.113.33:44344
24742      netconf      -     2023-10-03T04:09:07Z+00:00  203.0.113.33:44348
--------------------------------------------------------------------------------
Maximum number of simultaneous sessions: 4

awplus#clear netconf session 24741

awplus#show netconf session
Netconf sessions
ID         Username     Lock  Login time (UTC)            Remote address
--------------------------------------------------------------------------------
24739      netconf      -     2023-10-03T04:09:01Z+00:00  203.0.113.33:59910
24740      netconf      -     2023-10-03T04:09:03Z+00:00  203.0.113.33:59922
24742      netconf      -     2023-10-03T04:09:07Z+00:00  203.0.113.33:44348
--------------------------------------------------------------------------------
Maximum number of simultaneous sessions: 4
```

You can see that session 24741 (which had a lock on the <running> data-store) has been cleared.

RESTCONF

As RESTCONF is layered over the HTTP service, there are no direct commands to display 'restconf.' Access is logged through the standard logging for HTTP requests.