Technical Guide

AlliedWare Plus™ RESTful API

Developer Guide

Introduction

The RESTful API provides a convenient and reliable communication interface to Allied Telesis products. Developers of device and network management applications can use this interface to, among other things:

- Configure AlliedWare Plus devices
- View their configurations
- Query the devices' environmental state
- View information the devices have learned from the network
- Transfer files to/from the devices
- Upgrade devices' firmware
- Reboot devices
- Retrieve audit information from devices

This guide describes how to access the API, an overview of the API's syntax and a few practical programming examples. It is written for developers and assumes an understanding of RESTful APIs and the tools used to access them. It does not go in to detail about the features exposed by the API. For an understanding of these features see the relevant Feature Overview and Configuration Guides available in the Allied Telesis document library, www.alliedtelesis.com/library .

Contents

Products and software version that apply to this guide

This guide applies to all AlliedWare Plus™ products, running version **5.4.6** or later.

However, features available to the API varies between products and versions. For details, see the following documents:

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

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

Getting Started

Configuring the device for the API

The RESTful API requires that the HTTP service is enabled on the device and the device is accessible via the switch ports and/or out-of-band management port.

To enable the HTTP service:

Step 1: Enter Privileged Exec mode.

```
awplus> enable
```

Step 2: Enter Global Configuration mode.

```
awplus# configure terminal
```

Step 3: Enable the HTTP service.

```
awplus(config)# service http
```

If your device is not already configured with an IP address then configure one on the default VLAN (vlan1).

Step 1: Enter Privileged Exec mode.

```
awplus> enable
```

Step 2: Enter Global Configuration mode.

```
awplus# configure terminal
```

Step 3: Enter Interface Configuration mode for the vlan1 interface.

```
awplus(config)# interface vlan1
```

Step 4: Enter the IP address and mask.

```
awplus(config)# ip address 192.168.0.1/24
```

For more information about configuring your device see the Getting Started with AlliedWare Plus Feature Overview and Configuration Guide.

Accessing the API

The API provides an XML definition of all the endpoints that it exposes, this is both device and version specific. This definition can be retrieved by pointing your browser to:

`https://<ip address>/api.xml`

You will be prompted for a username and password. The username and password you provide in response to this prompt must be those of a user account that the target device can authenticate, either by its own internal user database, or (if configured) via a RADIUS or TACACS+ server. The default user account on Allied Telesis devices is manager:friend.

Output 1: Sample API definition (collapsed)

```
- <MODULE xmlns="https://github.com/alliedtelesis/apteryx xmlns:xs...>
  + <NODE name="aaa">...</NODE>
  + <NODE name="dns">...</NODE>
  + <NODE name="domain-lists" help="Collection of domain-name lists...
  + <NODE name="gui">...</NODE>
  + <NODE name="ntp">...</NODE>
  + <NODE name="ovs">...</NODE>
  + <NODE name="radius">...</NODE>
  + <NODE name="applications" help="List of applications">...</NODE>
  + <NODE name="atmf">...</NODE>
  + <NODE name="entities">...</NODE>
  + <NODE name="fiber-monitor">...</NODE>
  + <NODE name="ip">...</NODE>
  + <NODE name="interface">...</NODE>
  + <NODE name="routing">...</NODE>
</MODULE>
```

Note: As the definition is product and version specific, check your own device to see which features are exposed by the API.

The XML description is only available at the root of the API. Browsing the nodes of the API will return a JSON string listing nodes and their child nodes.

A URL with a slash at the end returns just the direct children of a node:

`https://<ip address>/api/<node name>/`

whereas a URL without a slash returns all of a node's children.

`https://<ip address>/api/<node name>`

Compare the output from the following:

`https://<ip address>/api/dns/`

```
{"dns": ["relay-settings"]}
```

`https://<ip address>/api/dns`

```
{"dns": {
    "relay-settings": {
        "debug": "0",
        "dead-time": "3600",
        "cache-size": "0",
        "relay": "0",
        "cache-timeout": "1800",
        "max-retry": "2",
        "timeout": "3"
    }
}}
```

API definition syntax

The following example of a device feature, DNS relay, illustrates the API definition syntax.

Enabling DNS relay on your device provides the capability for it to act as a local virtual DNS server. It can then service DNS lookup requests sent to it from local hosts. Acting as a DNS Relay, the switch will usually relay the requests to an external, or upstream, DNS server. By default, DNS Relay is disabled.

Output 2: DNS node definition

```
- <NODE name="dns">
  - <NODE name="relay-settings" help="Settings for DNS relay">
    - <NODE name="relay" mode="rw" default="0" help="DNS relay status"
       pattern="^(0|1)$">
        <VALUE name="disabled" value="0" help="DNS relay is disabled"/>
        <VALUE name="enabled" value="1" help="DNS relay is enabled"/>
      </NODE>
    - <NODE name="debug" mode="rw" default="0" help="DNS relay debug"
       pattern="^(0|1)$">
        <VALUE name="disable" value="0" help="Disable DNS relay
         debug"/>
        <VALUE name="enable" value="1" help="Enable DNS relay debug"/>
      </NODE>
    - <NODE name="max-retry" mode="rw" default="2" help="Maximum number
       of retries when sending to a DNS server"
       pattern="^(0|([1-9][0-9]?|100))$"/>
    - <NODE name="timeout" mode="rw" default="3" help="The time to wait
       for a response from a DNS server" pattern="^(0|([1-9][0-9]{0,2}
       |[12][0-9]{3}|3[0-5][0-9]{2}|3600))$"/>
    - <NODE name="dead-time" mode="rw" default="3600" help="An
       unresponsive server will be ignored for dead-time (in seconds)
       before any more requests are forwarded to it" pattern="^([6-9]
       [0-9]|[1-9][0-9]{2,3}|[1-3][0-9]{4}|4[0-2][0-9]{3}|43[01]
       [0-9]{2}|43200)$"/>
    - <NODE name="cache-size" mode="rw" default="0" help="The number of
       successful DNS lookups that will be cached"
       pattern="^(0|([1-9][0-9]{0,2}|1000))$"/>
    - <NODE name="cache-timeout" mode="rw" default="1800" help=
       "Individual entries in the cache will be timed out after
       cache-timeout" pattern="^([6-9][0-9]|[1-9][0-9]{2}|[12][0-9]
       {3}|3[0-5][0-9]{2}|3600)$"/>
    - <NODE name="source-interface-name" mode="rw" help="If defined,
       forwarded DNS requests will use a source IP address obtained from
       the defined interface"/>
   </NODE>
</NODE>
```

Table 1-1: API definition, field name descriptions

FIELD NAME	DESCRIPTION
NODE	Nodes can have child nodes or values. Nodes named "*" are parent nodes to lists. See **List syntax** for additional information.

Table 1-1: API definition, field name descriptions

FIELD NAME	DESCRIPTION

VALUE	Value nodes give a name and description to a value. For example a node that takes either 0 or 1 may have these values descried as:
	`<VALUE name="disabled" value="0" help="Feature is disabled"/>`
	`<VALUE name="enabled" value="1" help="Feature is enabled"/>`

name	The name of the node or value.
default	Default value for the feature
mode	■ **r - read** The value of the node may only be read through the API. ■ **w - write** The value of the node may be changed through the API.
help	This field describes what a node is used for and how to use it. See the relevant AlliedWare Plus Feature Overview and Configuration Guide for a fuller explanation of the feature.
pattern	The pattern field is a regular expression that matches all possible values that may appear in the node. The regex syntax used is POSIX Extended Regular Expression. Some notable aspects: ■ Special characters have their special meaning when unescaped. ■ The anchor characters (^ and $) only match the start and end of the string, not line breaks. ■ "." and "[^...]" **do** match new line characters. ■ "\d" is NOT supported. The regex must include the "start of string" character (^) and "end of string" character ($) to ensure the whole value matches the expression.

List syntax

Nodes with `name="*"` are parents to lists. You can access a list item by using the name of the child node. For example, consider the interface definition snippet below:

```
- <NODE name="interface">
  + <NODE name="if-alias" help="Interface index to name">...</NODE>
  - <NODE name="interfaces" help="Interface List">
    - <NODE name="*" help="Interface Name">
      - <NODE name="name" mode="r" help="Interface name"/>
      - <NODE name="if-index" mode="r" help="Interface Index"/>
...
```

To see a list of all the interfaces on the device browse:

`https://<ip address>/api/interface/interfaces/`

```
{"interfaces": ["port1.0.2","port1.0.1","vlan1",...]}
```

To see a single interface append the name of one of the interfaces to the URL used above:

`https://<ip address>/api/interface/interfaces/vlan1`

```
{"vlan1": {
    "name": "vlan1",
    "status": {
        "mtu": "1500",
        "oper-status": "6",
...
```

Sometimes a list item's name is more complex than just the child node's "name" field. Where this is the case the format of the name is specified in the node's help field. An example of this is the "neighbours" list; the item name is a combination of the neighbor's IP and MAC addresses.

```
<NODE name="neighbours" help="IPv6 neighbors">
- <NODE name="*" help="name-pattern: ip_mac">
  - <NODE name="ip" mode="r" help="Neighbor's IPv6 Address"/>
  - <NODE name="phys-address" mode="r" help="Neighbor's physical
     address (MAC, IPv4 or IPv6)"/>
...
```

This will produce an item name similar in form to the one below:

`https://<ip address>/api/interface/interfaces/vlan1/status/ipv4/neighbours/`

```
{"neighbours": ["10.0.0.5_00:00:5e:00:53:f2"]}
```

Examples of Accessing the API

Configure DNS relay using cURL

cURL is a command line utility for getting or posting files to a URL. For more information, and to download the utility, see https://curl.haxx.se/.

To retrieve the API definition using cURL:

`curl -u manager:friend -k https://<ip address>/api.xml`

GET:
Get the DNS Relays settings (note the ending slash to only return direct children):

`curl -u manager:friend -k https://<ip address>/api/dns/relay-settings/`

Get a single node's value:

```
curl -u manager:friend -k https://<ip address>/api/dns/relay-
settings/relay
```

Retrieve the entire DNS Relay configuration:

```
curl -u manager:friend -k https://<ip address>/api/dns/relay-
settings
```

POST:

Enable DNS relay by setting the value of the relay node to '1':

```
curl -u manager:friend -k -H "Content-Type: application/json" -d
"{'relay':'1'}" https://<ip address>/api/dns/relay-settings
```

More than one value can be set with a composite JSON string:

```
curl -u manager:friend -k -H "Content-Type: application/json" -d
"{'debug':'1',
  'relay':'1',
  'max-retry':'3',
  'timeout':'5'}" https://<ip address>/api/dns/relay-settings
```

Adding and removing domain list entries using cURL

See the section on **List syntax** for more information on how list items are named.

GET:

Get the current domain list

```
curl -u manager:friend -k https://<ip address>/api/domain-lists
```

POST:

Add a new domain list entry:

```
curl -u manager:friend -k -H "Content-Type: application/json" -d
"{'my_list_1':{
    'name':'my_list_1',
    'description':'domain_list_1',
    'entries':{
       'example.com':'example.com',
       '.org':'.org'}}}" https://<ip address>/api/domain-lists
```

View the newly added list with the following command:

```
curl -u manager:friend -k https://<ip address>/api/domain-lists/
my_list_1
```

DELETE:

Delete a domain list entry:

```
curl -u manager:friend -k -X "DELETE" https://<ip address>/api/
domain-lists/my_list_1
```

Additional Information

HTTP Conditional GET

An HTTP *conditional GET* allows the client to request a resource only if the resource has changed since it was last retrieved. The RESTful API uses the ETag (Entity Tag) mechanism to avoid the overhead of retrieving large sub-trees when there have been no changes to that sub-tree.

Each GET response from the API contains the following header:

`Etag: <etag>`

The client can then do a conditional request with the following header:

`If-None-Match: <etag>`

If no changes have been made then the HTTP response code will be

`304 Not Modified`

For example, use cURL (with option i to include headers) to retrieve the DNS node:

`curl -i -u manager:friend -k https://<ip address>/api/dns`

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Set-Cookie: -http-session-=3::http.session::9eb12...
Vary: Accept-Encoding
X-Frame-Options: SAMEORIGIN
X-Content-Type-Options: nosniff
Date: Sat, 01 Jan 2000 00:00:00 GMT
Etag: 5390971855F0B
Cache-Control: no-store
Content-Length: 270
Custom-Date: Sat, 01 Jan 2000 00:00:00 GMT
X-XSS-Protection: 1; mode=block
Access-Control-Allow-Origin: *
Connection: Keep-Alive
Access-Control-Allow-Headers: Authorization
Accept-Ranges: bytes

{
    "dns": {
        "relay-settings": {
            "timeout": "3",
            "max-retry": "2",
            "cache-timeout": "1800",
            "relay": "0",
            "cache-size": "0",
            "dead-time": "3600",
            "debug": "0"
        }
    }
}
```

Now attempt to retrieve the DNS node again, with a conditional request using the ETag returned previously; an HTTP response code 304 is returned.

```
curl -i -u manager:friend -k https://<ip address>/api/dns --header
"If-None-Match: 5390971855F0B"
```

```
HTTP/1.1 304 Not Modified
Access-Control-Allow-Credentials: true
Set-Cookie: -http-session-=7::http.session::fd20b...
Vary: Accept-Encoding
X-Frame-Options: SAMEORIGIN
X-Content-Type-Options: nosniff
Date: Thu, Sat, 01 Jan 2000 00:00:00 GMT
Cache-Control: no-store
X-XSS-Protection: 1; mode=block
Access-Control-Allow-Origin: *
Connection: Keep-Alive
Access-Control-Allow-Headers: Authorization
Accept-Ranges: bytes
```

HTTP Response Codes

This is a list of HTTP response codes relevant to RESTful API development.

CODE	STATUS TEXT	DESCRIPTION
200	OK	The request was successful. For a SET the configuration change has been applied and the HTTP BODY is empty. For a GET the path was valid and the HTTP body contains the requested data.
304	Not Modified	This status is returned if the user has requested a conditional get and the resource (path and all sub-paths) has not changed since the last retrieval.
403	Bad Request	The request is incorrectly formatted or contains invalid parameters that cannot be applied. To help the user understand why the request failed, a 400 response also supplies a JSON formatted error value and message in the HTTP BODY. Error codes below 1000 refer to standard errno values from IEEE Standard 1003.1-2001. Values above 1000 are custom error codes for the specific feature and their definitions are specified in the API for that feature. Example error response: `{"error":"-1004", "message":"IPv6 Address invalid for mode"}`
404	Forbidden	The user does not have authorization to access the requested URI. Either the path does not exist or the path has permissions that prevent the operation from being completed (e.g. read-only and a SET was attempted or write-only and a GET was attempted).
405	Not Found	The requested URI does not exist.
500	Internal Server Error	Internal error e.g. no memory.

JSON Arrays

The '*' character in the API indicates a list of items. e.g. /interface/interfaces/* specifies the list of interfaces. To aid table population, API lists can be returned as JSON arrays rather than JSON complex objects. This requires the "key" for the list be duplicated in the list object as the key in the path (i.e. the '*') is lost during translation.

To get the domain-list node as a JSON array use the following command:

```
curl -i -u manager:friend -k https://<ip address>/api/domain-lists
--header "X-JSON-Array: on"
```

```
{"domain-lists": [
    {
        "name": "my_list_2",
        "entries": [
            ".co.nz",
            "example.net"
        ],
        "description": "domain_list_2"
    },
    {
        "name": "my_list_1",
        "entries": [
            ".org",
            "example.com"
        ],
        "description": "domain_list_1"
    }
]}
```


Allied Telesis NETWORK SMARTER

North America Headquarters | 19800 North Creek Parkway | Suite 100 | Bothell | WA 98011 | USA | T: +1 800 424 4284 | F: +1 425 481 3895
Asia-Pacific Headquarters | 11 Tai Seng Link | Singapore | 534182 | T: +65 6383 3832 | F: +65 6383 3830
EMEA & CSA Operations | Incheonweg 7 | 1437 EK Rozenburg | The Netherlands | T: +31 20 7950020 | F: +31 20 7950021

alliedtelesis.com