Finisar®

APPLICATION NOTE

Controlling the WaveAnalyzer using the Web API

1. Scope and Overview

The Wave Analyzer High Resolution Optical Spectrum Analyzer provides a rich set of tools to allow instrument control and data acquisition from within other programs. The WaveAnalyzer hosts a Web server on the instrument and can interact with software applications using HTTP (HyperText Transfer Protocol). The software commands and control interface are based on RESTful style Web services that provide the ability to configure the device and access to measurement data using HTTP commands from a Web browser or from a software programming language.

This Application Note provides details on how to connect to the WaveAnalyzer and the principles underlying the WaveAnalyzer WEB API. Details of each command are explained and finally programming examples for the major programming and instrument control languages are given.

2. Connecting to the WaveAnalyzer

The link layer connection to the WaveAnalyzer is Ethernet, which is carried over standard Cat 6 cable or USB 2.0 cable. This enables user interaction from a Web browser or the WaveAnalyzer software application GUI, as well as automated control from most modern programming languages.

The user can connect to the WaveAnalyzer in two ways:

- Point-to-point mode from a computer, using Ethernet or USB connection, or
- Remote network device mode over an IP network.

In point-to-point mode, the WaveAnalyzer is connected directly to the user's computer via a GbE (Gigabit Ethernet) or a USB 2.0 port. GbE point-to-point connection over Cat 6 cable is recommended, to enable the fastest available WaveAnalyzer measurement rates. Other connection methods might result in slower measurement display refresh rates, due to insufficient performance of the networks or connections.

Connection	Default IP address
mDNS	WA000xxx.local
GbE (point-to-point or DHCP)	169.254.3.8
USB (Ethernet over USB)	192.168.2.8

Table 2-1 Default IP addresses

2.1 GbE point-to-point connection

The WaveAnalyzer has a GbE (Gigabit Ethernet) interface, and it is recommended to use GbE on the user's computer to enable the fastest available WaveAnalyzer measurement rates.

The default IP address of the point-to-point Ethernet interface is 169.254.3.8

The IP address of the WaveAnalyzer Ethernet interface is set to the subnet 169.254.xxx.xxx, enabling automatic private IP address allocation in Windows 7 and higher.

A computer with a free GbE port can connect directly to a WaveAnalyzer, while simultaneously using another Ethernet connection to a network.

A computer with a single GbE port can swap a single Cat 6 cable between a network and a WaveAnalyzer without requiring re-configuration. However, for convenience, the computer can usually be expanded with an additional NIC (network interface controller) or a USB Ethernet adapter.

2.2 USB point-to-point connection

The WaveAnalyzer communicates over USB by creating an Ethernet link over USB. This functionality is provided by a driver called RNDIS/Ethernet Gadget, which is installed along with the WaveAnalyzer software package. Standard installations of Windows 7 and higher include this driver.

The default IP address of the USB Ethernet interface is 192.168.2.8

The speed and stability of the WaveAnalyzer connection is affected by the USB connection. The cable quality and length, as well as the USB port and driver on the computer may slow the transfer rate or cause connection instability.

For this reason, GbE is the recommended connection method, while USB operation is offered for convenience.

2.3 IP name resolution

Identifying the IP address of a network device can be tedious and difficult. To avoid this, the WaveAnalyzer can use its serial number as its IP address, e.g. WaveAnalyzer serial no. WA000123 can be addressed as "WA000123.local". This alias is resolved by a Windows service called multicast Domain Name System (mDNS), which is installed on the user's computer along with the WaveAnalyzer software package.

The alias works regardless of the physical connection; both GbE and USB connectivity work with mDNS. The initial automated connection can take time to establish, while the mDNS service discovers and translates the alias IP address. Once the connection is established, normal connection speed is available.

2.4 Remote network device connection

The WaveAnalyzer can connect to a network and can be operated from remote locations on the network. This requires the WaveAnalyzer to be configured to use DHCP (Dynamic Host Configuration Protocol). This is achieved by connecting to the WaveAnalyzer using a web browser pointed at the IP address of the WaveAnalyzer.

The Web interface allows the user to set a fixed IP address for the WaveAnalyzer 1500S, or configure it to use DHCP (Dynamic Host Configuration Protocol). In the Web interface, click *Tools* and then *Change IP configuration* to configure the WaveAnalyzer 1500S IP address.

Once the WaveAnalyzer has accepted a network IP address, multiple users can access the instrument to view measurements, albeit with a reduced refresh rate in some instances.

Note: Any user can change settings on the WaveAnalyzer; this may cause unexpected results for other users.

3. Web API Basics

The WaveAnalyzer uses HTTP methods, mainly GET/PUT, to configure the instrument and retrieve measurement data. The WaveAnalyzer Web server can be accessed via all available connection methods, with different approaches to defining its IP address.

The most common use case is retrieving measurement data from the WaveAnalyzer. A Web browser can send an HTTP GET request to the WaveAnalyzer Web server, to retrieve measurement data. Once the WaveAnalyzer is connected, which can be verified using a "ping" command, open a Web browser and enter the following URL into the address bar.

http://<waveanalyzer IP address>/wanl/data/text

E.g., for a WaveAnalyzer connected via GbE, the following URL will return the current measurement data in text format (see Section 4.2 for more details on the syntax of the command and the returned data)

http://169.254.3.8/wanl/data/text

For more complex automation, modern programming languages support HTTP requests through common libraries and these are described in Section 5.

4. WaveAnalyzer Commands

The WaveAnalyzer Web service API supports the following functions:

4.1 Scan configuration

Description: Set the WaveAnalyzer scan range, with center frequency and span specified in MHz. The port parameter selects either the *Normal* or *HighSens* optical input port on the WaveAnalyzer.

HTTP method: GET/PUT

URL: http://<ip>/wanl/scan/ /[/<port>]

Parameter	Description
center	Scan center frequency (MHz).
span	Scan range span (MHz).
port (optional)	Select the optical input connection in use. *Normal* (default) or *HighSens*

Response: JSON formatted response includes the following fields.

JSON field	Description
rc	Result code. 0: success. Non-zero for errors (see result code table).

Example:
http://169.254.3.8/wanl/scan/194000000/1000000/Normal

The above example sets the scan range from 193.0 to 195.0 THz using the *Normal* optical input connection for a WaveAnalyzer connected via GbE.

4.2 Download measurement data

Description: Retrieve the most recent WaveAnalyzer measurement data. The <format> tag is used to specify how the data is returned, either in tab-delimited text, JSON or binary. The [?triggerin=on] parameter includes the trigger mask along with the measurement data, identifying the portion of the measurement data when the Trigger In port was on.

HTTP method: GET

URL: http://<ip>/wanl/data/<format>[?triggerin=on]

Parameter	Description
Format	Response format text|json|bin where text: raw text format, json: JSON format, bin: binary format
triggerin (optional)	Include input trigger flag along with measurement data.

Text response: The text response contains header information followed by formatted measurement data, as shown below.

```
Model Type: <model type>
Model Number: < model number>
Serial Number: <serial>
Firmware Version: <version>
Software Version: N.A.
Scan ID: <scan id>
Creation Time: N.A.

Data:
Frequency [MHz] Absolute Power [mdBm]  Power X-
Polarization [mdBm]  Power Y-Polarization [mdBm]
193299000       -90000  -90000  -90000
193299020       -90000  -90000  -90000
193299040       -90000  -90000  -90000
193299060       -90000  -90000  -90000
193299080       -90000  -90000  -90000
```

An extra *Flag* column is included when the "triggerin=on" parameter is used, with "0" indicating trigger low and "1" indicating trigger high.

```
Model Type: <model type>
Model Number: < model number>
Serial Number: <serial>
Firmware Version: <version>
Software Version: N.A.
Scan ID: <scan id>
Creation Time: N.A.

Data:
Frequency [MHz] Absolute Power [mdBm]  Power X-
Polarization [mdBm]  Power Y-Polarization [mdBm] Flag
193299000       -90000  -90000  -90000  0
193299020       -90000  -90000  -90000  0
193299040       -90000  -90000  -90000  1
193299060       -90000  -90000  -90000  1
193299080       -90000  -90000  -90000  0
```

JSON response: The JSON formatted response includes the following fields.

JSON field	Description
Id	Scan sequence ID.
data	Array of scan records. Each record is a JSON array of integer values in the following order: [Frequency (MHz), Absolute Power (mdBm), Power X-Polarization (mdBm), Power Y-Polarization (mdBm), Flag (optional, included with "triggerin=on" parameter)].

An example JSON formatted response is shown below.

```
{
"id":20255,
"data":[[193299000,-90000,-90000,-90000,0],
[193299020,-90000,-90000,-90000,0],
[193299040,-90000,-90000,-90000,0],
[193299060,-90000,-90000,-90000,0],
[193299080,-90000,-90000,-90000,0],
[193299100,-90000,-90000,-90000,0]... ]
}
```

Binary response: Binary encoded response starts with a 1000 bytes fixed size header followed by binary integer encoded records. Each record has 20 bytes that include five 32-bit integers (little endian). The format is detailed below.

Offset (bytes)	Description

0	JSON formatted header information with the following fields:
	`{ "ver": <version>,`
	`  "id": <scan sequence id> }`
	(Zero-padded to 1000 bytes.)

1000	Frequency (MHz)
1004	Absolute power (mdBm)
1008	Power x-polarization (mdBm)
1012	Power y-polarization (mdBm)
1016	Flag
…	Additional records with the same format as above.

Example:

http://169.254.3.8/wanl/data/text?triggerin=on

The above example retrieves data from a WaveAnalyzer connected via GbE, and returns the data as formatted text.

4.3 Download measurement data – linear scale

Description: Retrieve the most recent WaveAnalyzer 1500S measurement data with the power levels represented in mW. The [?triggerin=on] parameter includes the trigger mask along with the measurement data, identifying the portion of the measurement data when the *Trigger In* port was active.
Note: This feature is introduced in WaveAnalyzer 1500S firmware version 1.02.

HTTP method: GET

URL: http://<ip>/wanl/lineardata/bin[?triggerin=on]

Parameter	Description
triggerin (optional)	Include input trigger flag along with measurement data.

Binary response: Binary encoded response starts with a 1000 bytes fixed size header followed by binary integer encoded records. Each record has 20 bytes that include five 32-bit integers (little endian). The format is detailed below.

Offset (bytes)	Description

0	JSON formatted header information with the following fields:
	`{ "ver": <version>,`
	`  "id": <scan sequence id> }`
	(Zero-padded to 1000 bytes.)

1000	Frequency (MHz)
1004	Absolute power (mW)
1008	Power in x-polarization (mW)
1012	Power in y-polarization (mW)
1016	Flag
…	Additional records with the same format as above.

Example:

http://169.254.3.8/wanl/lineardata/bin?triggerin=on

The above example retrieves data from a WaveAnalyzer 1500S connected via GbE, and returns the data in binary format.

4.4 Retrieve WaveAnalyzer device information

Description: Retrieve the status of the latest measurement scan.

HTTP method: GET

URL: http://<ip>/wanl/info

Response: The JSON formatted response includes the following fields.

JSON field	Description
model	WaveAnalyzer model
sno	Serial number
version	Firmware version
vendo	Manufacture company

4.5 Retrieve scan information

Description: Retrieve the configuration of the latest measurement scan.

HTTP method: GET

URL: http://<ip>/wanl/scan/info

Response: The JSON formatted response includes the following fields.

JSON field	Description
scanid	Scan count in the current power-on session.
center	Center frequency of the current measurement (MHz).
span	Frequency span (range) of the current measurement (MHz).
startfreq	Start frequency of the current measurement range (MHz).
stopfreq	Stop frequency of the current measurement range (MHz).
port	Selected optical input port (*Normal* or *HighSens*) for the current measurement.

5. Integration guide

Most programming languages include libraries to perform HTTP transactions. Some examples are given below.

5.1 Python

Python has several libraries for creating HTTP requests. The requests package has an extremely simple API and is well suited to interacting with the WaveAnalyzer. An advantage of using requests is the ability of the request object to hold the returned data as a JSON object, allowing easy parsing in a Python script.

Example Python script to retrieve WaveAnalyzer data:

```
import requests
import numpy as np
m =
requests.get('http://WA000001.local/wanl/data/json')
measData = m.json()['data']
foo = m.json()['data']
measData = np.array(foo)
freq = measData[:,0]
freq = measData[:,0]/1e6
Pwr = measData[:,3]/1e3
```

5.2 C#

Example C# function to retrieve WaveAnalyzer data:

```
using System;
using System.IO;
using System.Net;

namespace WaveAnalyzerExample1
{
    class Program
    {
        static void Main(string[] args)
        {
            // Create a request for the URL.
            WebRequest request = WebRequest.Create(
                "http://192.168.2.8/wanl/data/text");
            // Get the response.
            WebResponse response =
request.GetResponse();
            // Get the stream containing content
returned by the server.
            // Open the stream using a StreamReader for
easy access.
            StreamReader reader = new

StreamReader(response.GetResponseStream());
            // Read the content.
            string data = reader.ReadToEnd();
            // Process data
            System.Console.Write(data);
            // Clean up the streams and the response.
            reader.Close();
            response.Close();
        }
    }
}
```

5.3 VB

Example VB.NET function to retrieve WaveAnalyzer data:

```
Imports System
Imports System.IO
Imports System.Net

Module WaveAnalyzerExample1

    Sub Main()
        ' Create a request for the URL.
        Dim request As WebRequest = _

WebRequest.Create("http://192.168.2.8/wanl/data/text")
        ' Get the response.
        Dim response As WebResponse =
request.GetResponse()
        ' Open the stream using a StreamReader for easy
access.
        Dim reader As New
StreamReader(response.GetResponseStream())
        ' Read the content.
        Dim responseFromServer As String =
reader.ReadToEnd()
        ' Display the content.
        Console.WriteLine(responseFromServer)
        ' Clean up the streams and the response.
        reader.Close()
        response.Close()
    End Sub

End Module
```

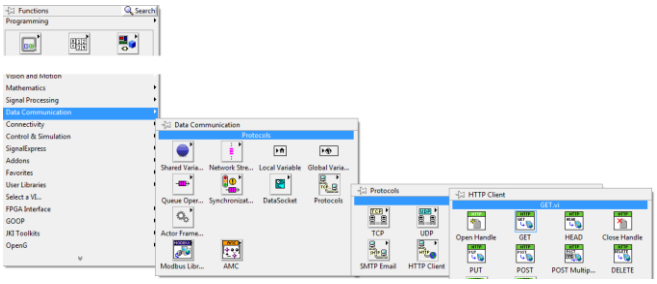
5.4 Java

Java has many HTTP libraries, e.g. Apache HttpClient library. Example Java code to retrieve WaveAnalyzer data:

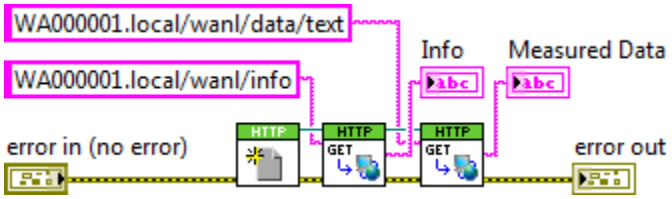
```
import org.apache.http.client.fluent.Request;
import org.apache.http.entity.ContentType;

...

String data =
Request.Get("http://192.168.2.8/wanl/data/json")
    .connectTimeout(120000)
    .socketTimeout(120000)
    .execute().returnContent().asString();

//Process the result.
```

5.5 LabVIEW

LabVIEW contains SubVIs that handle HTTP requests. These are collected in the Data Communication > Protocols > HTTP Client palette shown below.



Example LabVIEW VI to retrieve WaveAnalyzer information and data:



5.6 MATLAB

In MATLAB, the *webread* and *webwrite* commands are recommended for accessing RESTful services; these commands were introduced in MATLAB version 2014b. Older versions of MATLAB can use *urlread* and *urlwrite*, as described in the GNU Octave section.

5.7 GNU Octave

RESTful services are accessed in GNU Octave through the libcurl library, which is wrapped by the commands *urlread* and *urlwrite*.

urlread retrieves the response to an HTTP request into a string, whereas *urlwrite* dumps the response to a file.

```
textdata = urlread('192.168.2.8/wanl/data/text')

//Process the result.

Filepath = urlwrite('192.168.2.8/wanl/data/text',
'text.txt')
```

5.8 C/C++

In C/C++, *libcurl* is recommended for communicating with the WaveAnalzyer.

Example code to retrieve WaveAnalyzer measurement data:

```
#include <curl/curl.h>
#include <stdio.h>

int main() {
    curl_global_init(CURL_GLOBAL_ALL);
    CURL*  curl_ = curl_easy_init();
    CURLcode curlrc;
    curl_easy_setopt(curl_, CURLOPT_URL,

"http://192.168.2.8/wanl/data/text");
    curlrc = curl_easy_perform(curl_);
    curl_easy_cleanup(curl_);
    return 0;
}
```

For more information on the WaveAnalyzer please contact your local WaveAnalyzer sales representative or visit http://www.finisar.com/instruments.

FINISAR

1389 Moffett Park Drive
Sunnyvale, CA 94089
Tel.: +1-408-548-1000
Fax: +1-408-541-6138
waveanalyzer@finisar.com
http://www.finisar.com/instruments