

# Chapter 24



## External Data Feed (EDF) Reference

### 24.1 Overview

The External Data Feed (EDF) allows external data to be sent to and processed by NetVigil as though it had been collected by NetVigil itself. Any external tool can send results and events for any existing test, and the result/event will be processed as if a NetVigil monitor had polled the result.

The EDF process is accessed via a text based protocol over a TCP socket. Protocol messages can be sent from programs written in C, Java, Perl or any other language.

Typically, you should provision the test with a type of 'external' (using the Web interface or the BVE Server) before inserting test results via the EDF server, but you can also use this process to enter data for any existing test using the test's serial number.

It is recommended that the NetVigil Perl API described in Chapter 25, "NetVigil Perl API", be used to access the EDF instead of a direct telnet connection for consistency.

#### 24.1.1 Connecting To The Server

Communication with the EDF server consists of two phases - a connection establishment phase and a command-execution phase. The connection establishment phase is where remote client provides authentication information to the server in the form of a login id, and the corresponding password. Once the authentication information has been verified, all subsequent commands sent to the server will be executed with the permissions and privileges of the specified user.

Note that the login information provided to the EDF Server is the username and password specified in the `dge.xml` configuration file and *not* the web user login and password. On login, the user can insert data for all the devices and tests in NetVigil.

Once the connection establishment phase has been completed, the client application may send one command at a time and wait to receive a reply (possibly consisting of multiple lines of output) from the server.

A client application establishes a connection to the EDF Server by connecting to a TCP/IP socket, using the hostname/ip of the server that is running the monitor, and a pre-defined port number (default port number is 7657). Upon establishment of the TCP session, the server will greet the client with a welcome message following the rules outlined below. If the server is ready to accept data, it will respond with

```
OK NetVigil External Data Feed Server Ready
```

at which point the remote client can send authentication information. If the server is unavailable, an error message would be printed in the form

```
ERR [reason]
```

and the server will disconnect the client.

### 24.1.2 Disconnecting From the Server

When the client application would like to disconnect from the EDF Server, it is recommended that the client issue a disconnect request (see below for appropriate command) instead of simply closing the socket connection. This will allow the server to perform proper cleanup before disconnecting the session.

Also if the EDF Server does not receive anything from the client for an extended period of time, the session will timeout and disconnect the client. The default timeout is currently 2 minutes and can be changed by editing `dge.xml`

### 24.1.3 Command-reply Formatting Rules

The commands sent by a client and responses sent back by the server must adhere to the following formatting conventions:

## Client Command Format

1. Each client command is composed of a single line of text terminated by a newline character. A carriage return followed by a newline (`\r\n`) is considered to be the same as a newline character (`\n`) alone.
2. Client commands may or may not require additional parameters. Each parameter consists of values, separated by 'pipe' symbol (`|`). Example `command_name value1 [ | value2 | value3 .. ]`.
3. Pipe symbol (`|`) is not permitted as part of the value.
4. For each client command, the server will respond with a response code indicating success or failure, and optionally some descriptive text indicating actions taken.
5. Command names are NOT case sensitive.
6. Parameters/values for any command must appear in exact order following the command. If a value is not applicable or existent for a particular command, an empty value (`|` ) should be provided

## Server Response Format

The server always responds (to client initiated commands/requests) with text of the following format:

```
<status code> [optional informative text]
```

where `status code` is one of:

**OK** which indicates the command/request was successful.

**ERR** which is indicative of failure to execute the request.

## 24.2 Client Commands

### Login

Provide authentication information to the server. This username and password are specified in the `dge.xml` configuration file.

```
Login <login_id> | <password>
```

### Logout | Quit

End a login session.

```
Logout
```

## Result.insert

Insert a result value for an existing test into the database.

```
Result.insert device_name | device_addr | test_name | test_serial  
| date_time | result_value
```

where

- `device_name` is the descriptive name that was used when the device was provisioned.
- `device_ip` is the fully qualified address or ip address that was used when provisioning the device.
- `test_name`, along with `device_name` and `device_ip` are used to obtain the unique serial number for the test if `test_serial` is not provided. This is the descriptive test name that was used during provisioning.
- `test_serial` is the unique serial number of the test, which should be already provisioned. If no serial number is provided, the device name, address and test name (if provided) will be used to obtain the test serial number. If no test matching the serial number can be found, the result value will be ignored.
- `date_time` is provided either in `yyyy.mm.dd-hh:mm`, or `nnnnnnnnnn` format where `nnnnnnnnnn` is number of seconds since 1970. If the date and time are not provided, or a value of 0 is used, current system time in GMT will be used. Because of the real-time aggregation, you must provide a timestamp newer than the last data value for the test.
- `result_value` is the value which should be inserted into the database. The provided result will be multiplied by the result multiplier, and processed in the manner set via process-directive, both set during the creation of the test.

## 24.3 Templates for EDF Tests

You can setup templates for EDF tests. If you create an xml configuration file under the “plugin/monitors” directory (e.g. “my\_edf\_test.xml”) and restart the Web Application and DGE components, you will see the defined tests under `manage >devices >tests >advanced tests` (under external tests section). You can create additional tests with other names with same sub-type

```
<monitor type="external">
<testtype>
  <displayName>Sample EDF test</displayName>
  <displayCategory>application</displayCategory>
  <subType>edf_1</subType>
  [...]
</testtype>

</monitor>
```

Note that the monitor type is set to “external”.

## 24.4 EDF Monitors vs. Plugin Monitors

Tests from a plugin monitor are executed at the specified interval by the DGE. In contrast, the DGE does not perform any tasks for EDF tests. The DGE expects to receive test results from an external data source (script, application) at specific intervals via a TCP socket. The connecting application will need to following the EDF API protocol to communicate with the DGE. The EDF monitor is useful when the metric to be monitored is on a different host that is not accessible from the DGE via standard (SNMP, WMI) or proprietary (IP based) methods. The EDF API is also scalable to a larger extent compared to plugin monitors since the remote host can insert results for multiple tests over a single TCP session.

## 24.5 Example

**STEP 1:** The device and test need to be created in NetVigil using either the Web Interface (under the Advanced Tests section) or the BVE Server. The testType should be set to `external` for EDF tests.

```
% telnet bve_host 7661
OK 200 NetVigil BVE TCP Server v3.5 ready

LOGIN localuser/localpassword
OK 201 request accepted and processed, ready for next request

DEVICE.CREATE "devicename=my_device", "address=192.168.123.25",
"devicetype=unix", "snmpcid=public", "comment=my workstation",
"locationName=Denver Office"
OK 201 request accepted and processed, ready for next request
```

```
TEST.CREATE "devicename=my_device", "testname=my_test",
"testtype=external", "subtype=external", "interval=15m",
"units=xyz", "warningThreshold=55", "criticalThreshold=85",
"maxvalue=100", "resultProcessDirective=0", "resultMultiplier=1"
OK 201 request accepted and processed, ready for next request
```

```
QUIT
```

**STEP 2:** Now connect to the EDF server on port 7657 (using the username and password in the dge.xml configuration file, which is different from the one we used to access the BVE Server in the first step. Note that we are not using the test serial number and are also specifying the timestamp as 0 which indicates use current date and time.

```
% telnet dge_host 7657
OK NetVigil External Data Feed Server Ready

login edfuser|fixme
OK

result.insert my_device | 192.168.123.25 | my_test | | 0 | 25
OK

QUIT
OK Received logout - bye
```

**NOTE:** To view the newly inserted test result via BVE Server:

```
% telnet bve_host 7661
OK 200 NetVigil BVE TCP Server v3.5 ready

login localuser/localpassword
OK 201 request accepted and processed, ready for next request

RESULT.SEARCH "devicename=my_device", "testname=my_test",
"starttime=NOW"
OK 203 request accepted, records returned: 1
```

```
my_device|470000|my_test|470003|external|external|20030506100124|1  
|25|25|25|Ok|55|85
```

QUIT

OK 299 Logging out.

