How To Use the Experimental InfluxDB Performance Data Module in Nagios XI 2026R1.2+

Purpose

This document describes how to use the experimental InfluxDB Performance Data Nagios Event Broker (NEB) module for Nagios XI 2026R1.2. This module automatically captures Nagios host and service check performance data and writes it to an InfluxDB v1.x time-series database using the InfluxDB line protocol format over HTTP.

WARNING: This module is EXPERIMENTAL and is not intended for use in production environments. Deploy only on non-production test instances. Use at your own risk.

Overview

The **influx_perf_api_v1** module is a NEB module that intercepts host and service check events from the Nagios Core engine. When a check completes, the module extracts performance data from the check result, formats it in InfluxDB line protocol, and sends it to InfluxDB via HTTP POST using libcurl.

Experimental Status

This module is currently in an experimental phase. It has not undergone the full testing and validation cycle required for production deployment. Certain functionality may not be complete or behave as expected. Before using this module, be aware of the following:

- This module should only be deployed on non-production test instances.
- Create regular backups of your Nagios configuration and InfluxDB data before and during use.
- Closely monitor system performance (CPU, memory) and disk space on both the Nagios server and the InfluxDB server.
- InfluxDB may consume a significant amount of disk space as performance data accumulates over time.
- The module may impact Nagios Core performance under high check volumes.
- Features, configuration options, and data formats may change in future releases.

Prerequisites

- Nagios XI 2024R1.2+ installed and running on a non-production test instance
- InfluxDB 1.x installed and running (the setup script below handles installation)
- libcurl development headers (installed during setup)
- Network connectivity between the Nagios server and InfluxDB
- Sufficient disk space for InfluxDB data storage (monitor regularly)

Installation

WARNING: Only install this experimental module on non-production test instances. Before proceeding, create a full backup of your Nagios configuration files and verify that you have adequate disk space available for InfluxDB data storage.

The module is compiled and installed as part of the Nagios XI build process. After installation, the setup script is used to install InfluxDB and configure the broker module in Nagios.

Pre-Installation Checklist

1. Confirm this is a *non-production* test instance of Nagios XI
2. Create a full backup of `/usr/local/nagios/etc/` and any other critical configuration
3. Verify available disk space (recommended: at least 10 GB free for InfluxDB data)
4. Document your current Nagios performance baseline for comparison

Running the Setup Script

The setup script is located in the Nagios XI scripts directory after installation. Run the following command as root to install InfluxDB and configure the module:

```
/usr/local/nagiosxi/scripts/setup_influx.sh
```

The setup script performs the following steps:

1. Detects the operating system (RHEL/CentOS/Oracle Linux or Ubuntu/Debian)
2. Installs InfluxDB 1.x from the official InfluxData repository
3. Starts the InfluxDB service
4. Creates the perfdata database in InfluxDB
5. Creates a timestamped backup of `/usr/local/nagios/etc/nagios.cfg`
6. Adds the broker_module configuration line to nagios.cfg

Supported Operating Systems

- RHEL / CentOS / Oracle Linux (uses yum)
- Ubuntu / Debian (uses apt-get)

You can review the OS Compatibility Matrix for full details on supported versions:

[Nagios Operating System Compatibility Matrix]

Post-Installation

After the setup script completes, restart Nagios to activate the module:

```
systemctl restart nagios
```

Verify the module loaded successfully by checking the Nagios log:

```
tail -f /usr/local/nagios/var/nagios.log
```

You should see log entries similar to:

```
influxdb: Module initialized - URL=http://localhost:8086/write?db=perfdata,
timeout=10s, batch=10, ssl_verify=1
influxdb: Registered for 25 callback types
```

WARNING: After activating the module, monitor your Nagios process, CPU and memory usage, overall system load, and InfluxDB disk space consumption. If you observe problems or the performance is unacceptable, you should be able to deactivate the module using the `--deactivate` flag and restarting the Nagios process.

Configuration

Broker Module Syntax

While the setup_influx should install the module for you it can be manually configured or adjusted in your nagios.cfg file (`/usr/local/nagios/etc/nagios.cfg`). The module is configured via a broker_module line similar to NDO and should follow the following syntax:

```
broker_module=<path_to_module> <influxdb_url> [timeout=N] [batch=N]
[ssl_verify=0|1] [ca_cert=PATH] [client_cert=PATH] [client_key=PATH]
```

Configuration Options

Argument	Description	Default	Example
URL (first argument)	InfluxDB write endpoint URL (required)	None	`http://localhost:8086/write?db=perfdata`
timeout=N	HTTP request timeout in seconds	5	`timeout=10`
batch=N	Batch N events before sending (max 100)	10	`batch=20`
ssl_verify=0|1	Verify SSL certificates (1=yes, 0=no)	1	`ssl_verify=0`
ca_cert=PATH	Path to CA certificate bundle	None	`ca_cert=/etc/ssl/ca.pem`
client_cert=PATH	Path to client certificate file	None	`client_cert=/etc/ssl/client.crt`
client_key=PATH	Path to client private key file	None	`client_key=/etc/ssl/client.key`

Configuration Examples

Basic Configuration (Default)

This is the default configuration added by the setup script:

```
broker_module=/usr/local/nagios/bin/influx_perf_api_v1.o
http://localhost:8086/write?db=perfdata timeout=10 batch=10
```

Custom Timeout and Batch Size

Increase the timeout to 15 seconds and batch 25 events before sending:

```
broker_module=/usr/local/nagios/bin/influx_perf_api_v1.o
http://localhost:8086/write?db=perfdata timeout=15 batch=25
```

HTTPS with SSL Verification Disabled

Connect to InfluxDB over HTTPS without verifying the SSL certificate:

```
broker_module=/usr/local/nagios/bin/influx_perf_api_v1.o
https://influxdb.example.com:8086/write?db=perfdata ssl_verify=0
```

HTTPS with CA Certificate

Connect to InfluxDB over HTTPS using a custom CA certificate:

```
broker_module=/usr/local/nagios/bin/influx_perf_api_v1.o
https://influxdb.example.com:8086/write?db=perfdata ca_cert=/etc/ssl/certs/ca-
bundle.pem
```

HTTPS with Client Certificate Authentication

Connect to InfluxDB over HTTPS using mutual TLS (client certificate authentication):

```
broker_module=/usr/local/nagios/bin/influx_perf_api_v1.o
https://influxdb.example.com:8086/write?db=perfdata ca_cert=/etc/ssl/ca.pem
client_cert=/etc/ssl/client.crt client_key=/etc/ssl/client.key
```

Remote InfluxDB Server

Send data to an InfluxDB instance on a different server:

```
broker_module=/usr/local/nagios/bin/influx_perf_api_v1.o http://influxdb-
server.example.com:8086/write?db=nagios_metrics timeout=10 batch=10
```

Data Format

The module writes data to InfluxDB using the line protocol format. All data is written to the database specified in the URL query parameter (default: perfdata). Two measurements are created for performance data, a host check measurement and a service check measurement.

Database

The target database is specified in the `db=` query parameter of the InfluxDB URL in the broker_module line. The setup script creates a database named perfdata by default. The database name can be changed by modifying the `db=` parameter in the URL.

Host Check Measurement: nagios_host_check

This measurement stores data from Nagios host check results.

Tags (Indexed)

Tag	Description	Example Values
host_name	Name of the monitored host	webserver01, db-master
state	Host state (numeric)	0=UP, 1=DOWN, 2=UNREACHABLE
state_type	State type (numeric)	0=SOFT, 1=HARD

Fields (Values)

Field	Type	Description
current_attempt	Integer	Current check attempt number
max_attempts	Integer	Maximum check attempts before hard state
return_code	Integer	Plugin return code
latency	Float	Check latency in seconds
execution_time	Float	Check execution time in seconds
output	String	Check output message
(perfdata fields)	Float	Parsed from plugin performance data (e.g., rta, pl)

Example Line Protocol

```
nagios_host_check,host_name=webserver01,state=0,state_type=1
current_attempt=1i,max_attempts=3i,return_code=0i,latency=0.052000,execution_ti
me=4.012000,output="PING OK - Packet loss = 0%",rta=0.045000,pl=0.000000
1706900000000000000
```

Service Check Measurement: nagios_service_check

This measurement stores data from Nagios service check results. Only service checks that return performance data are recorded.

Tags (Indexed)

Tag	Description	Example Values
host_name	Name of the monitored host	webserver01, db-master
service_description	Nagios service description	HTTP, SSH, CPU Load

Fields (Values)

Field	Type	Description
(perfdata fields)	Float	Parsed from plugin performance data (varies by service)

Example Line Protocol

```
nagios_service_check,host_name=webserver01,service_description=HTTP
time=0.004000,size=1234.000000 1706900000000000000
```

Querying Data from InfluxDB

Once the module is active and Nagios is running, performance data is written to InfluxDB in real time. You can query the data using the InfluxDB CLI or HTTP API.

Accessing the InfluxDB CLI

```
influx
```

Then switch to the perfdata database:

```
USE perfdata
```

Example Queries

View All Host Check Data

```
printf "SELECT * FROM nagios_host_check" | influx -database 'perfdata'
```

View All Service Check Data

```
printf "SELECT * FROM nagios_service_check" | influx -database 'perfdata'
```

Query Recent Host Checks for a Specific Host

```
printf "SELECT * FROM nagios_host_check WHERE host_name='localhost' AND time >
now() - 1h" | influx -database 'perfdata'
```

Query Service Performance Metrics

```
printf "SELECT * FROM nagios_service_check WHERE host_name='webserver' AND
service_description='HTTP' ORDER BY time DESC LIMIT 10" | influx -database
'perfdata'
```

Get Average Execution Time by Host

```
SELECT MEAN(execution_time) FROM nagios_host_check WHERE time > now() - 24h
GROUP BY host_name
```

View Available Measurements

```
printf "SHOW MEASUREMENTS" | influx -database 'perfdata'
```

View Tags and Fields for a Measurement

```
printf "SHOW TAG KEYS FROM nagios_host_check" | influx -database 'perfdata'
printf "SHOW FIELD KEYS FROM nagios_host_check" | influx -database 'perfdata'
```

Exiting the InfluxDB CLI

To exit the InfluxDB CLI, type:

```
exit
```

Monitoring and Maintenance

WARNING: Because this module is experimental, ongoing monitoring and maintenance are critical. Failure to monitor disk space and system performance may result in system instability or data loss.

Disk Space Monitoring

InfluxDB stores all performance data on disk and may consume significant space over time, especially in environments with a large number of hosts and services. It is recommended that you regularly monitor disk usage on the InfluxDB data directory.

Check current disk usage:

```
df -h /var/lib/influxdb
```

Check the size of the InfluxDB data directory:

```
du -sh /var/lib/influxdb/data
```

Consider configuring an InfluxDB retention policy to automatically expire old data. For example, to keep only 30 days of data:

```
influx -execute 'CREATE RETENTION POLICY "30_days" ON "perfdata" DURATION 30d REPLICATION 1 DEFAULT'
```

System Performance Monitoring

Monitor the Nagios server for any performance degradation after enabling the module. Key metrics to watch:

- Nagios process CPU and memory usage
- Check execution latency (increased latency may indicate module overhead)
- System load average
- InfluxDB process CPU and memory usage
- Network traffic between Nagios and InfluxDB

Monitor Nagios process resource usage:

```
ps aux | grep nagios
```

Monitor InfluxDB process resource usage:

```
ps aux | grep influxdb
```

Check system load:

```
uptime
```

Regular Backups

Create regular backups of your system as well as your Nagios configuration and InfluxDB data while testing this experimental module.

Nagios Configuration Backup

```
cp -r /usr/local/nagios/etc/ /usr/local/nagios/etc.backup.$(date +%Y%m%d)
```

InfluxDB Data Backup

```
influxd backup -database perfdata /path/to/backup/directory
```

Deactivating the Module

To disable the InfluxDB module without uninstalling InfluxDB, run the setup script with the –deactivate flag:

```
sudo /usr/local/nagiosxi/scripts/setup_influx.sh --deactivate
```

This will:

1. Create a timestamped backup of nagios.cfg in /usr/local/nagiosxi/tmp/
2. Comment out the broker_module line in nagios.cfg with a # (`#broker_module`)
3. Display a confirmation message.

After deactivating, restart Nagios:

```
systemctl restart nagios
```

WARNING: If you experience any system instability, performance degradation, or unacceptable levels of disk space consumption, deactivate the module using the command above and restart Nagios.

Troubleshooting

Module Not Loading

Check the Nagios and system logs:

```
tail -100 /usr/local/nagios/var/nagios.log | grep influxdb
tail -100 /var/log/messages /var/log/syslog | grep influxdb
```

Common causes:

- Incorrect InfluxDB URL in nagios.cfg
- InfluxDB service not running
- Network connectivity issues between Nagios and InfluxDB
- Missing libcurl shared library

No Data in InfluxDB

Verify the following:

1. InfluxDB is running: `systemctl status influxdb`
2. The perfdata database exists: `influx -execute 'SHOW DATABASES'`
3. The module is loaded: check nagios.log for "`influxdb: Module initialized`"
4. Nagios checks are returning performance data in their output

Creating the InfluxDB Database Manually

If the perfdata database does not exist, create it manually:

```
influx -execute 'CREATE DATABASE perfdata'
```

Checking InfluxDB Service Status

```
systemctl status influxdb
```

Start and enable InfluxDB if it is not running:

```
systemctl start influxdb
systemctl enable influxdb
```

Verifying InfluxDB Connectivity

Test connectivity to the InfluxDB write endpoint:

```
curl -i http://localhost:8086/ping
```

Disk Space Issues

If InfluxDB is consuming too much disk space:

1. Check current usage: df -h /var/lib/influxdb
2. Set a retention policy to limit data age (see Monitoring and Maintenance section)
3. If urgent, drop old data manually:
   ```
   influx -execute 'DROP SERIES FROM /.*/ WHERE time < now() - 7d' -database perfdata
   ```
4. Consider deactivating the module until disk space is resolved

Performance Considerations

WARNING: As an experimental module, performance characteristics are not fully validated. Monitor all of the following closely and be prepared to deactivate the module if system performance is impacted.

- **Batching:** The batch parameter (default 10) batches events before sending to reduce network overhead. Higher values reduce HTTP requests but increase memory usage and data delivery latency.
- **Timeout:** Adjust the timeout parameter based on network latency between Nagios and InfluxDB. The default is 5 seconds.
- **Maximum Batch Size:** The maximum batch size is 100 events. Values above 100 are capped automatically.
- **Memory:** Monitor Nagios process memory usage if using large batch sizes, as events are held in memory until the batch is full.
- **Disk Space:** InfluxDB data will grow continuously as performance data accumulates. Configure a retention policy and monitor disk usage regularly to prevent the filesystem from filling up.
- **CPU:** The module adds processing overhead to each host and service check. In environments with a high number of checks, this may increase Nagios CPU usage.

Finishing Up

This completes the documentation for the experimental InfluxDB Performance Data Module for Nagios XI. Remember that this module is experimental and should only be used on non-production test instances.

We appreciate your feedback and if you have additional questions or other support-related questions, please visit us at the Nagios Support Forum, Nagios Documentation Hub, or Nagios Library:

Visit Nagios Support Forum Visit Documentation Hub Visit Nagios Library