
NetXMS Administrator Guide

Release 5.2.0

Raden Solutions, SIA

Apr 28, 2025

CONTENTS

1	Introduction	1
2	Concepts	3
3	Installation	13
4	Upgrade	39
5	Quick start	45
6	Agent management	59
7	Server management	81
8	SNMP	89
9	User management	115
10	Object management	127
11	Network discovery	157
12	Data collection	161
13	Event processing	189
14	Data and Network visualisation	217
15	Grafana integration	257
16	Operating System Monitoring	261
17	File System Monitoring	273
18	Log monitoring	279
19	Windows Event Log Synchronization	291
20	SSH monitoring	295
21	Network Service Monitoring	299
22	Data Collection from Web Services	305

23	Modbus	309
24	Database monitoring	311
25	Application monitoring	331
26	ICMP ping	335
27	Hardware(sensor) monitoring	341
28	UPS monitoring	347
29	Cluster monitoring	349
30	JVM monitoring	351
31	Hypervisor monitoring	353
32	Asterisk monitoring	355
33	Network topology	361
34	Hardware Asset Management	365
35	Business services	369
36	Remote file management	375
37	Package management	379
38	Reporting	381
39	Image library	385
40	Mobile Client	387
41	Web API/Rest API	399
42	Advanced topics	423
43	Scheduled tasks	437
44	Scripting	441
45	High Availability Setup	447
46	Appendix	449
47	Glossary	561
	Index	567

INTRODUCTION

This document covers the installation, configuration, and use of NetXMS.

NetXMS is an enterprise grade multi-platform modular open source network management and monitoring system. It provides comprehensive event management, performance monitoring, alerting, reporting and graphing for all layers of IT infrastructure — from network devices to the business application layer. Having been designed with flexibility and scalability in mind, NetXMS features a wide range of supported platforms. It is licensed under the GNU General Public License version 2 as published by the Free Software Foundation.

1.1 Product Support

Contact us if you run into a problem or found a bug.

- [Forum](#)
- [Telegram](#)
- [Issue tracker](#)
- [Facebook](#)
- [Twitter](#)

Priority support for NetXMS is provided by [Raden Solutions](#)

1.2 Conventions

The following typographical conventions are used in this manual.

Sample	Description
<i>Button</i>	Any GUI element: Button, Menu item
<i>Another Guide</i>	Reference to external manual or man page
Control-M	Keyboard shortcut
<i>DCI</i>	Term which can be found in the glossary
<i>File</i> ▶ <i>Exit</i>	Menu selection path. You must click on <i>File</i> , then <i>Exit</i>

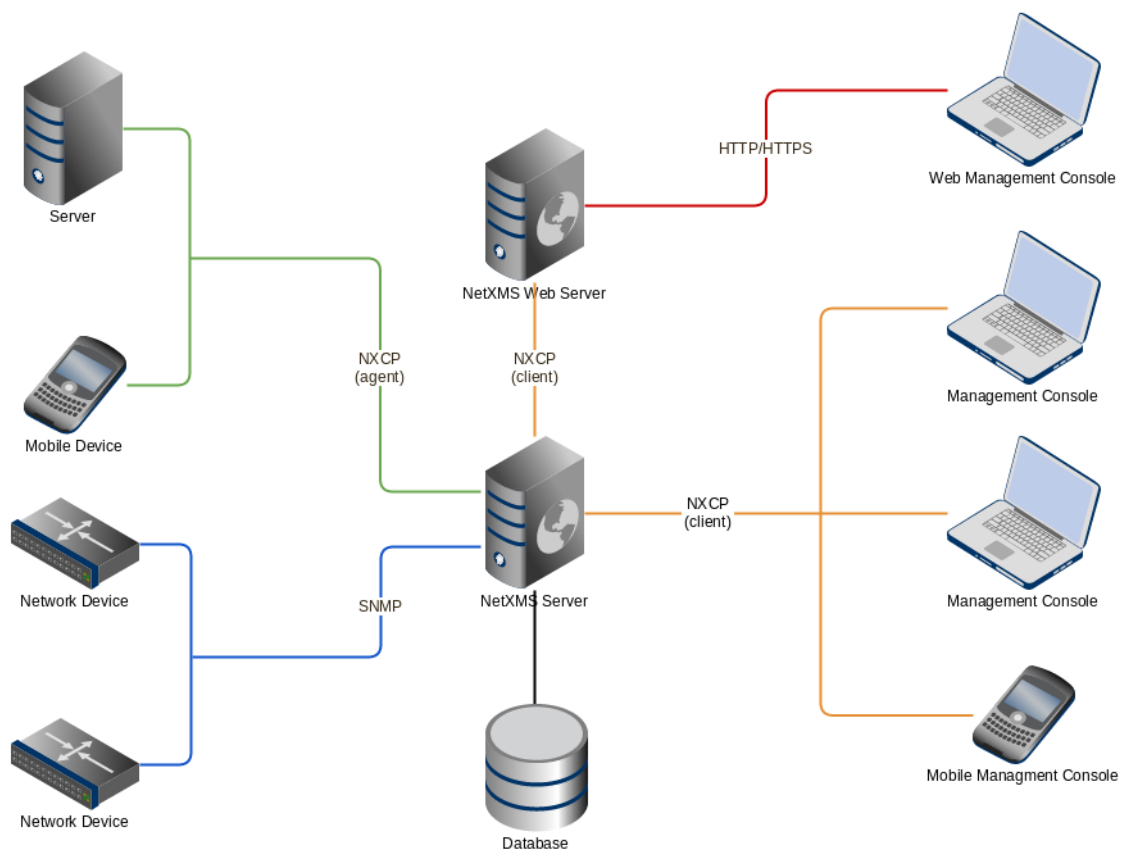
1.2.1 Changelog

Complete change log for each product release is available at <https://github.com/netxms/changelog/blob/master/ChangeLog>.

CONCEPTS

2.1 Architecture overview

The system has three-tier architecture: the information is collected by monitoring agents (either our own high-performance agents or SNMP agents) and delivered to monitoring server for processing and storage. Network administrator can access collected data using cross-platform Desktop Management Client, Web Management Client or Management application for Android. Desktop and Web clients have almost the same functionality and the same user interface.



2.2 Objects

All monitored network infrastructure is represented as a set of *objects* in NetXMS monitoring system. Each object represents one physical or logical entity (e.g. host or network interface), or group of them (e.g. subnet, container). Objects are organized into hierarchical structure. An object can have several parents, e.g. a node can belong to multiple containers, subnets and templates. Structure can be modified either manually or automatically with the help of Auto-bind scripts.

Each object has its own access rights. Access rights are applied hierarchically on all children of object. For example if *Read* access right is granted to a user on a *Container*, then user has *Read* right on all objects that this *Container* contains.

Every object has set of attributes; some of them exist for all objects (like *id* and *name* or *status*), while other depend on object class - for example, only *Node* objects have attribute *SNMP community string*. In addition to the above mentioned attributes, it's possible to define custom attributes. This can be done by user in the Management Client, from NXSL script or by external application via NetXMS API.

NetXMS has seven top level objects - Entire Network, Service Root (named "Infrastructure Services" after system installation), Template Root, Asset Root, Network Map Root, Dashboard Root and Business Service Root. These objects serve as an abstract root for an appropriate object tree. All top level objects have only one editable attribute - name.

Object Class	Description	Valid Child Objects
Entire Network	Abstract object representing root of IP topology tree. All zone are located under it. System can have only one object of this class.	<ul style="list-style-type: none"> • Zone
Zone	Object representing group of (usually interconnected) IP networks without overlapping addresses. Contains appropriate subnet objects.	<ul style="list-style-type: none"> • Subnet
Subnet	Object representing IP subnet. Typically objects of this class are created automatically by the system to reflect system's knowledge of IP topology. The system places Node objects inside an appropriate Subnet object based on an interface configuration. Subnet objects have only one editable attribute - <i>Name</i> .	<ul style="list-style-type: none"> • Node
Service Root	Abstract object representing root of your infrastructure service tree. System can have only one object of this class. After system installation it is named "Infrastructure Services".	<ul style="list-style-type: none"> • Circuit • Chassis • Cluster • Condition • Collector • Container • Mobile Device • Node • Rack • Sensor • Wireless Domain

continues on next page

Table 1 – continued from previous page

Object Class	Description	Valid Child Objects
Collector	Object similar to container, but with data collection capabilities.	<ul style="list-style-type: none"> • Circuit • Cluster • Chassis • Condition • Collector • Container • Mobile Device • Node • Rack • Sensor • Wireless Domain
Container	Grouping object which can contain any type of objects that Service Root can contain. With help of container objects you can build object's tree which represents logical hierarchy of IT services in your organization.	<ul style="list-style-type: none"> • Circuit • Cluster • Chassis • Condition • Collector • Container • Mobile Device • Node • Rack • Sensor • Wireless Domain
Cluster	Pseudo-object defining any process: technological or logical that aggregates information from several separate nodes. See Cluster monitoring for more information.	<ul style="list-style-type: none"> • Node
Circuit	Reference of multiple interfaces will allow to use this object to represent different types of network services beyond - multilink interfaces, links between sites, virtual circuits, etc.	<ul style="list-style-type: none"> • Interface
Rack	Object representing a rack. It has the same purpose as container, but allows to configure visual representation of equipment installed in a rack.	<ul style="list-style-type: none"> • Node • Chassis
Chassis	Object representing a chassis, e.g. a blade server enclosure. Chassis can be configured as a part of a rack.	<ul style="list-style-type: none"> • Node
Condition	Object representing complicated condition - like "cpu on node1 is overloaded and node2 is down for more than 10 minutes". Conditions may represent more complicated status checks because each condition can have a script attached. Interval for evaluation of condition status is configured in Server Configuration Variables as Condition-PollingInterval with default value 60 seconds.	

continues on next page

Table 1 – continued from previous page

Object Class	Description	Valid Child Objects
Node	Object representing physical host or network device (such as a router or network switch). These objects can be created either manually by administrator or automatically during network discovery process. They have a lot of attributes controlling all aspects of interaction between NetXMS server and managed node. For example, the attributes specify what data must be collected, how node status must be checked, which protocol versions to use, etc. Node objects contain one or more interface objects. The system creates interface objects automatically during configuration polls.	<ul style="list-style-type: none"> • Interface • Network Service • VPN Connector
Interface	Interface objects represent network interfaces of managed computers and devices. These objects created automatically by the system during configuration polls or can be created manually by user.	
Network Service	Object representing network service running on a node (like http or ssh), which is accessible online (via TCP IP). Network Service objects are always created manually. Currently, the system works with the following protocols - HTTP, POP3, SMTP, Telnet, SSH and Custom protocol type.	
VPN Connector	Object representing VPN tunnel endpoint, is used for interfaceless tunnels (like ipsec). Such objects can be created to add VPN tunnels to network topology known to NetXMS server. VPN Connector objects are created manually. In case if there is a VPN connection linking two different networks open between two firewalls that are added to the system as objects, a user can create a VPN Connector object on each of the firewall objects and link one to another. The network topology will now show that those two networks are connected and the system will take this condition into account during problem analysis and event correlation.	
Sensor	Logical object with data collection capabilities. NetXMS does not perform direct network communication with sensor, but data is collected by some other means, e.g. using MQTT protocol.	
Wireless Domain	Object representing wireless network, made up from one or several wireless controllers (represented by nodes with Wireless Controller capability) and thin access points.	<ul style="list-style-type: none"> • Access point • Node
Access point	Object representing thin wireless access point managed by a central controller. These objects are created automatically by the system.	
Template Root	Abstract object representing root of your template tree.	<ul style="list-style-type: none"> • Template • Template Group
Template Group	Grouping object which can contain templates or other template groups.	<ul style="list-style-type: none"> • Template • Template Group

continues on next page

Table 1 – continued from previous page

Object Class	Description	Valid Child Objects
Template	Data collection and agent policy template. See Data collection section for more information about templates. If an object is a child of a template, this means that template is applied to that object.	<ul style="list-style-type: none"> • Access point • Collector • Cluster • Mobile Device • Node • Sensor
Asset Root	Abstract object representing root of hardware asset management tree.	<ul style="list-style-type: none"> • Asset • Asset group
Asset Group	Grouping object which can contain assets or other asset group.	<ul style="list-style-type: none"> • Asset • Asset group
Asset	Hardware management asset	
Network Map Root	Abstract object representing root of your network map tree.	<ul style="list-style-type: none"> • Network Map • Network Map Group
Network Map Group	Grouping object which can contain network maps or other network map groups groups.	<ul style="list-style-type: none"> • Network Map • Network Map Group
Network Map	Network map.	
Dashboard Root	Abstract object representing root of your dashboard tree.	<ul style="list-style-type: none"> • Dashboard • Dashboard Group
Dashboard Group	Grouping object which can contain dashboards or other dashboard group	<ul style="list-style-type: none"> • Dashboard • Dashboard Group
Dashboard	Dashboard. Can contain other dashboards.	<ul style="list-style-type: none"> • Dashboard
Business Service Root	Abstract object representing root of your business service tree. System can have only one object of this class.	<ul style="list-style-type: none"> • Business Service • Business Service Prototype
Business Service	Object representing single business service. Can contain other business services or business service prototypes.	<ul style="list-style-type: none"> • Business Service • Business Service Prototype
Business Service Prototype	Prototype from which business service objects are automatically populated.	










2.2.1 Object status

Each object has a status. Status of an object calculated based on:

- Polling results
- Status of child objects (e.g. interfaces of node, nodes under container)
- Active alarms, associated with the object (after an alarm is resolved or terminated, it no longer affects object status)
- Value of status *DCIs* (DCI that has `Use this DCI for node status calculation` property enabled)

There are multiple options for status calculation, see [Status calculation](#) for more information.

For some object classes, like Report or *Template*, status is irrelevant. Status for such objects is always *Normal*. Object's status can be one of the following:

Nr.	Status	Description
0	 Normal	Object is in normal state.
1	 Warning	Warning(s) exist for the object.
2	 Minor	Minor problem(s) exist for the object.
3	 Major	Major problem(s) exist for the object.
4	 Critical	Critical problem(s) exist for the object.
5	 Unknown	Object's status is unknown to the management server.
6	 Unmanaged	Object is set to "unmanaged" state.
7	 Disabled	Object is administratively disabled (only applicable to interface objects).
8	 Testing	Object is in testing state (only applicable to interface objects).

2.2.2 Unmanaged status

Objects can be unmanaged. In this status object is not polled, DCIs are not collected, no data is updated about object. This status can be used to store data about an object that is temporary or permanently unavailable or not managed.

2.2.3 Maintenance mode

This is special status, that's why it is not included in above status list. This status prevents event processing for specific node. While this node in maintenance mode is still polled and DCI data is still collected, but no event is generated.

2.3 Event Processing

NetXMS is event based monitoring system. Events can come from different sources (polling processes (status, configuration, discovery, and data collection), *SNMP* traps, and directly from external applications via client library). All events all are forwarded to NetXMS Event Queue.

NetXMS Event Processor can process events from Event Queue in either sequential or parallel mode. In sequential mode events are processed one-by-one. Parallel processing mode allows to process events in several parallel threads, thus increasing processing performance. See [Event processing](#) for more information.

Events in the Event Queue are processed according to rules defined in [Event Processing Policy](#). As a result of event processing, preconfigured actions can be executed, and/or event can be shown up as *alarm*.

Usually alarm represents something that needs attention of network administrators or network control center operators, for example low free disk space on a server. NetXMS provides one centralized location, the Alarm Browser, where alarms are visible. It can be configured which events should be considered important enough to show up as alarm.

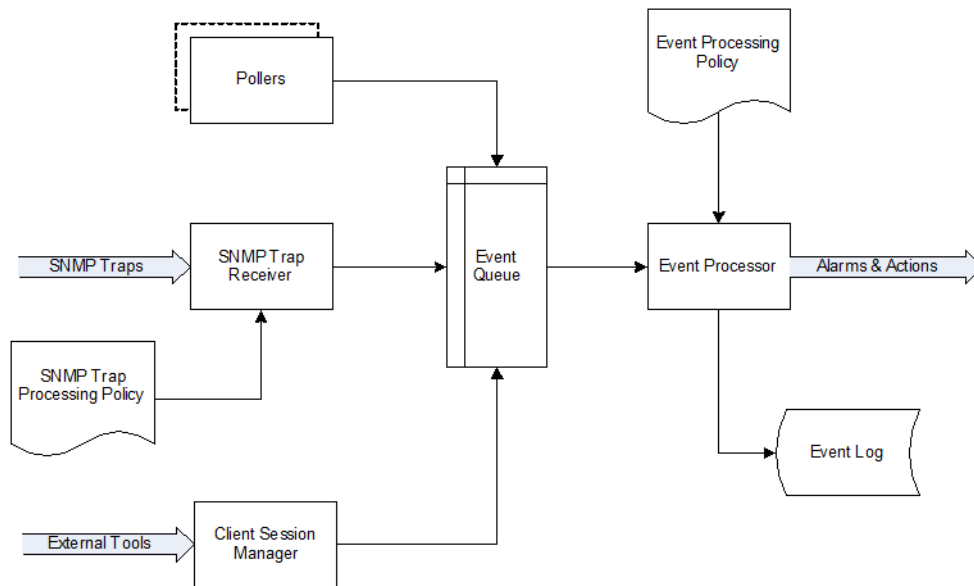


Fig. 1: Event flow inside the monitoring system

2.4 Polling

For some type of objects NetXMS server start gathering status and configuration information as soon as they are added to the system. These object types are: nodes, access points, conditions, clusters, business services, zones (if a zone has more then one proxy, proxy health check is being performed). This process called *polling*. There are multiple polling types, each having specific execution intervals (set by server configuration variables). In the end of polling process hook script is being executed.

Type	Purpose	Interval server configuration variable	Hook script
Status	Determine current status of an object	Objects.StatusPollingInterval	Hook::StatusPoll
Configuration	Determine current configuration of an object (list of interfaces, supported protocols, etc.) By default executes auto bind scripts for templates and containers, use “Objects.AutobindOnConfigurationPoll” server configuration variable to disable.	Objects.ConfigurationPollingInterval	Hook::ConfigurationPoll
Configuration (full)	Same as usual configuration poll but resets previously detected capabilities and detects them again. (can only be executed manually)		
Interface	Updates names of the interfaces. This operation also happens during Configuration Poll. (can only be executed manually)		
Topology	Gather information related to network link layer topology	Topology.PollingInterval	Hook::TopologyPoll
Routing	Gather information about IP routing (cannot be executed manually)	Topology.RoutingTableUpdateInterval	
ICMP	Ping nodes and gather response time statistics (cannot be executed manually)	ICMP.PollingInterval	
Instance Discovery	Perform Instance Discovery to add/remove DCIs	DataCollection.InstancePollingInterval	Hook::InstancePoll
Automatic Binding	Checks and bind or unbind Containers, Templates and Context Dashboards to nodes according to auto-bind script.	Objects.AutobindPollingInterval	
Network Discovery	Searches for new nodes by polling information about neighbor IP addresses from known nodes. Accessible from Configuration perspective.	NetworkDiscovery.PassiveDiscovery.Interval	Hook::DiscoveryPoll

Polling intervals can be set for specific objects by adding a custom attribute named `SysConfig:nnn`, where `nnn` is the name of server configuration variable e.g.: `SysConfig:Objects.ConfigurationPollingInterval`.

2.5 Data Collection

From each node NetXMS can collect one or more *metrics* which can be either single-value (e.g. “CPU.Usage”), list (e.g. “FileSystem.MountPoints”) or table (e.g. “FileSystem.Volumes”). When new data sample is collected, it's value is checked against configured thresholds. This documentation use term *Data Collection Item* (DCI) to describe configuration of metric collection schedule, retention, and thresholds.

Metrics can be collected from multiple data sources:

Source	Description
Internal	Data generated inside NetXMS server process (server statistics, etc.)
NetXMS Agent	Data is collected from NetXMS agent, which should be installed on target node. Server collect data from agent based on schedule.
SNMP	SNMP transport will be used. Server collect data based on schedule.
Web service	Data is obtained from JSON, XML, or plain text retrieved via HTTP
Push	Values are pushed by external system (using <i>npush</i> or API) or from NXSL script.
Windows Performance counters	Data is collected via NetXMS agent running on Windows machine.
SM-CLP	Data is collected via Server Management Command Line Protocol
Script	Value is generated by NXSL script. Script should be stored in <i>Script Library</i> .
SSH	Data is obtained from output of ssh command executed through SSH connection.
MQTT	Data is obtained by subscribing to MQTT broker topics.
Network Device Driver	Some SNMP drivers (NET-SNMP, RITTAL as of NetXMS v. 3.8) provide parameters for data collection. E.g. NET-SNMP provides information about storage this way.
Modbus	Data is collected via Modbus-TCP industrial protocol. See <i>Modbus</i> for more information.
EtherNet/IP	

2.6 Discovery

2.6.1 Network discovery

NetXMS can detect new devices and servers on the network and automatically create node objects for them. Two modes are available - passive and active.

In passive mode server will use only non-intrusive methods by querying ARP and routing tables from known nodes. Tables from the server running NetXMS are used as seed for passive discovery.

In active mode in addition to passive scan methods configured address ranges are periodically scanned using ICMP echo requests.

NetXMS can also use SNMP trap and syslog messages as seed for discovery. Network discovery is available from Configuration perspective.

2.6.2 Instance discovery

NetXMS can create metrics names for *Data Collection Item* automatically. Instance discovery collects information about node instances like disk mountpoints, device list, etc. and automatically creates or removes *DCIs* with obtained data. To run instance discovery manually and check it's results select in nodes menu *Poll -> Instance discovery*

2.7 Security

All communications are encrypted using either AES-256, AES-128, or Blowfish and authenticated. As additional security measure, administrator can restrict list of allowed ciphers.

Agent authenticate incoming connections using IP white list and optional preshared key.

User passwords (if internal database is used) as hashed with salt with SHA-256.

All shared secrets and passwords stored in the system can be obfuscated to prevent snooping.

INSTALLATION

3.1 Major changes between releases

3.1.1 5.1.4

IP v4 addresses are now supported only in a.b.c.d format with decimal numbers

3.1.2 5.1

NXSL changes: node attribute 'ipAddr' is deprecated. The newly added 'ipAddress' attribute should be used instead.

3.1.3 5.0

Additionally loaded MIB files will not work. They should be uploaded again in the *Configuration* → *SNMP MIB files* configuration view. Starting with version 5.0, the MIB compilation file extension changed to ".mib" and the already compiled MIB file extension is now ".cmib". The default MIB file location has changed to \$HOME/share/netxms/mibs/ and user additional MIB files should be loaded in *Configuration* → *SNMP MIB files*.

The default format of SNMP OID changes to a format without leading dot. Potentially this can break some scripts that use SNMP OID string comparisons.

The NXSL syntax has changed. During upgrade, existing scripts get converted automatically. If you need to manually convert a script, this could be done via the nxscript command line utility (`nxscript -5 script-file.nxsl`). NXSL syntax major changes:

Description	Old example	New example
String concatenation changes from '.' to '..'	variable = "Text first part " . "text second part";	variable = "Text first part " .. "text second part";
Dereference changed form '->' to '.'	equals = \$node- >getInterface(\$5) == variable- >interfaceAttribute;	equals = \$node.getInterface(\$5) == vari- able.interfaceAttribute;
Use '[]' to initialize an array instead of '%()'	a = %(1,2,3);	a = [1,2,3];
Use safe dereference '?' instead of '@'	customAttribute- Value = test@\$node;	customAttribute- Value = \$node?.test;
Use 'import' keyword instead of 'use' for library import	use ToolBox;	import ToolBox;
Use 'function' keyword instead of 'sub' for function definition	sub EnumerateN- odes(obj, level)	function Enumer- ateNodes(obj, level)

Class 'TIME' is now renamed as 'DateTime'. Created Math, Base64, Crypto, Net, and IO modules, and functions moved

under them. The most used functions are left as deprecated, but others were just renamed. The table below shows the full renamed list containing functions that were just renamed and do not have deprecated versions:

Old name	New name	Type
TIME	DateTime	class
asin	Math::Asin	function
acos	Math::Acos	function
atan	Math::Atan	function
atan2	Math::Atan2	function
cosh	Math::Cosh	function
exp	Math::Exp	function
gethostbyaddr	Net::ResolveAddress	function
gethostbyname	Net::ResolveHostname	function
log	Math::Log	function
log10	Math::Log10	function
md5	Crypto::MD5	function
md5	Crypto::MD5	function
sha1	Crypto::SHA1	function
sha256	Crypto::SHA256	function
sinh	Math::Sinh	function
tanh	Math::Tanh	function
weierstrass	Math::Weierstrass	function
decode	Base64::Decode	function
encode	Base64::Encode	function
CopyFile	IO::CopyFile	function
CreateDirectory	IO::CreateDirectory	function
DeleteFile	IO::DeleteFile	function
FileAccess	IO::FileAccess	function
OpenFile	IO::OpenFile	function
RemoveDirectory	IO::RemoveDirectory	function
RenameFile	IO::RenameFile	function

Abort and other runtime errors in the script DCI will set DCI to an error state. Before version 5.0, DCI changed state to unsupported.

Importing the dashboard configuration exported from the previous version of NetXMS will not upgrade the script syntax to the 5.0 format.

3.1.4 4.4

The minimal JRE (Java Runtime Environment) version for both web and management client is now Java 17.

3.1.5 4.2

The NXSL functions ‘AgentExecuteAction’ and ‘AgentExecuteActionWithOutput’ are renamed to ‘AgentExecuteCommand’ and ‘AgentExecuteCommandWithOutput’.

3.1.6 4.1

The CreateDCI NXSL method changed. In the new version the last two parameters “polling interval” and “retention time” should be set to null instead of 0 to have a default value in the DCI configuration.

NXSL decimal numbers written with leading zeros will NOT be interpreted as octal.

3.1.7 4.0

Incompatible attributes in NXSL DCI class: instance now refers to an instance value (as in {instance} macro), not instance name as before. The instance name can be accessed via the attribute “instanceName”.

Several WEB API endpoints were renamed, e.g. *API_HOME/summaryTable/adHoc* became *API_HOME/summary-table/ad-hoc*.

3.1.8 3.8

The minimal JRE (Java Runtime Environment) version for the management client is Java 11. A Desktop Management Client with bundled JRE is provided for Windows.

3.1.9 3.7

Introduced boolean type in NXSL. Comparisons like “func() == 1”, where ‘func’ is a function that returns a boolean type, will always result as false as the boolean value ‘true’ is not equal to 1. This might require fixes in some NXSL scripts.

Regex matching operation in NXSL returns an array with capture groups or false as a result.

Clusters now have configuration poll. If you have a configuration poll hook script that is referring to the *\$node* object, this will produce an error message in the server log each time a configuration poll runs on a cluster. Replace *\$node* with *\$object* or use the condition `if (classof($object) == "Node")` or `if ($node != null)` prior to accessing attributes or methods of *\$node*.

3.1.10 3.6

In this version the “Certificate manager” was removed from server. All CA certificates configuration should be manually moved to the “TrustedCertificate” configuration parameter in the server configuration file.

3.1.11 3.5

External Metrics (ExternalMetric, etc...) expect UTF-8 encoding on Windows. It might be needed to adjust scripts called by external metrics if non-ASCII characters are returned.

3.1.12 3.1

Regex matching operation in NXSL returns array with capture groups or NULL as result. NXSL objects and arrays in logical expressions are evaluated to TRUE. This might require some NXSL script adjustments.

3.1.13 3.0

Notification channels are introduced as new functionality. SMS configuration automatically moved from server configuration to notification channel depending on old driver with one of the next names: AnySMS, DBTable, Dummy, GSM, Kannel, MyMobile, Nexmo, NXAgent, Portech, Slack, SMSEagle, Text2Reach, WebSMS. No manual actions are required.

Flags and dynamic flags are moved to the NetObject class. Separated node flags set by user and capability flags set by system to flags and capabilities. Numeric values for flags, capabilities and dynamic flags were changed. This affects only NXSL scripts that checked those flags directly.

The 32 bit version of management client is not available any more.

The Agent always requires encryption unless the RequireEncryption parameter explicitly set to off. It might be required to manually add the “RequireEncryption” configuration parameter where required to disable encryption.

Agent policies were merged with templates. Each policy was converted to a template. No changes required.

3.2 Planning

3.2.1 Operating system

Both NetXMS server and agent work fine on most operating systems, including Windows, Linux, and commercial UNIXes. However, we test and officially support only some of them.

Supported platforms for NetXMS server and agent:

- Debian 10 (Buster), 11 (Bullseye), 12 (Bookworm)
- Ubuntu 18.04 LTS (Bionic), 20.04 LTS (Focal Fossa), 22.04 LTS (Jammy Jellyfish), 24.04 (Noble)
- Linux Mint 19.3 (Tricia), 20.3 (Una), 21.2 (Victoria)
- Linux Mint Debian Edition 4
- Devuan ASCII
- Red Hat Enterprise Linux 8, 9
- CentOS 8
- Windows 11, Windows 10, Windows Server 2016, 2019, 2022
- FreeBSD 12
- ArchLinux (Latest)
- AlpineLinux 3.8+
- Raspbian Buster

Support for the following platforms is provided only to customers with an active support contract:

- Debian 8 (Jessie)
- Ubuntu 16.04 LTS (Xenial)
- Devuan Jessie
- Red Hat Enterprise Linux 6, 7
- CentOS 6, CentOS 7
- FreeBSD 11, FreeBSD 11.3
- Windows 7, Windows 8.1, Windows Server 2008 R2, 2012, 2012 R2
- AIX 6.1, AIX 7.x
- SUSE Linux Enterprise Server 11, 12, 15
- Solaris 11 (agent only)
- HP-UX 11.31 (agent only)

3.2.2 Server hardware

Minimal requirements: Core 2 duo 1GHz, 1024MB RAM, 1GB disk space.

3.2.3 Linux kernel tuning

An important requirement on large systems might be the need to tune Linux network buffer size. Default values may not be enough if the system is sending many ICMP pings, for example. The following kernel parameters should be changed:

- `net.core.rmem_default`

- `net.core.wmem_default`
- `net.core.rmem_max`
- `net.core.wmem_max`

In our test lab, value 1703936 seems to be working well (default was 212992).

Example:

- `sudo sysctl -w net.core.rmem_default=1703936`
- `sudo sysctl -w net.core.wmem_default=1703936`
- `sudo sysctl -w net.core.rmem_max=1703936`
- `sudo sysctl -w net.core.wmem_max=1703936`

Kernel changes will not be preserved after reboot unless `sysctl` commands are applied in the system configuration file, which is typically located at `/etc/sysctl.conf`. Increasing these kernel values also increases kernel memory space in use and may impact other applications.

3.2.4 Database

Database engines supported by NetXMS server:

- PostgreSQL 9.5, 9.6, 10, 11, 12, 13, 14, 15, 16, 17
- PostgreSQL with TimescaleDB 11, 12, 13, 14, 15, 16, 17
- MySQL 5.6, 5.7, 8.0
- MariaDB 10.1, 10.2, 10.3, 10.4
- Oracle 12c, 18c, 19c
- Microsoft SQL Server 2012, 2014, 2016, 2017, 2022
- SQLite (only for test purposes)

PostgreSQL database tuning might be required depending on database size. Increasing `shared_buffers` might be needed. A rough recommendation is 25% of available RAM. Increasing `max_locks_per_transaction` is needed if using TimescaleDB. A rough recommendation is 512.

Database size and load is very hard to predict, because it is depending on the number of monitored nodes and collected metrics. If you plan to install a database engine on the same machine as NetXMS server, increase your hardware requirements accordingly.

3.2.5 Java

A Java Runtime Environment (JRE) is needed for the Desktop Management Client (nxmc) and for the Web Management Client. The Supported Java version is 17 and higher.

Since version 3.8 the Desktop Management Client with a bundled JRE is provided for Windows.

3.2.6 Agent

Agent resource usage is negligible and can be ignored.

3.3 Installing from DEB repository

We host a public APT repository at <http://packages.netxms.org/> for most deb-based distributions (Debian, Ubuntu, Mint, Raspbian, etc.). Packages are signed, and you'll need to install an additional encryption key for signature verification.

Supported URLs (*CODENAME* should be replaced with output of *lsb_release -sc*):

- Debian, LMDE - “deb <http://packages.netxms.org/debian> *CODENAME* main”
- Ubuntu, Mint - “deb <http://packages.netxms.org/ubuntu> *CODENAME* main”
- Raspbian - “deb <http://packages.netxms.org/raspbian> *CODENAME* main”

3.3.1 Add APT repository

There are two options to add an APT repository: by hand or by using the netxms-release package. Use of the release package is strongly encouraged because it allows easy change in repository configuration and encryption keys will be updated in the future.

Using the netxms-release package

Download and install the netxms-release-latest.deb package, which contain a source list file of the repository as well as a signing key.

```
wget http://packages.netxms.org/netxms-release-latest.deb
sudo dpkg -i netxms-release-latest.deb
sudo apt-get update
```

Manually

Add the repository to your sources.list:

```
echo "deb http://packages.netxms.org/${lsb_release -si | tr A-Z a-z} ${lsb_release -
↪sc | tr A-Z a-z} main" > /etc/apt/sources.list.d/netxms.list
wget -q -O - https://packages.netxms.org/netxms-keyring.gpg | gpg --dearmor -o /etc/
↪apt/trusted.gpg.d/netxms-keyring.gpg
sudo apt-get update
```

3.3.2 Installing packages

Server

The server requires two components to function: the server itself (package “netxms-server”) and at least one database abstraction layer driver (multiple can be installed at the same time, e.g. for migration purposes). These database drivers are also used by the agent for database monitoring (performing queries to databases).

Provided driver packages:

- netxms-dbdrv-pgsql - PostgreSQL driver
- netxms-dbdrv-mariadb - Mariadb driver
- netxms-dbdrv-mysql - MySQL driver (not built for Ubuntu 20 / Mint 20)
- netxms-dbdrv-odbc - unixODBC driver (can be used with DB/2 and Microsoft SQL)
- netxms-dbdrv-oracle - Oracle driver (requires Oracle client installation)

1. Install required packages (adjust command to match your environment):

```
apt-get install netxms-server netxms-dbdrv-pgsql
```

2. Create user and database (*examples*).
3. Modify server configuration file (“/etc/netxmsd.conf” to match your environment.
4. Load database schema and default configuration:

```
nxdbmgr init
```

5. Start server:

```
systemctl start netxms-server
```

6. Enable automatic startup of server:

```
systemctl enable netxms-server
```

7. If the database engine is running on the same system, add ordering dependency for database in the netxmsd systemd unit override file. This will ensure database shutdown only after netxmsd process completion on system shutdown/restart. To add the dependency e.g. for the PostgreSQL database, run:

```
systemctl edit netxms-server
```

and add the following lines:

```
[Unit]
After=network.target postgresql.service
```

After editing run `systemctl daemon-reload` to reload systemd configuration.

Note

Default credentials - user “admin” with password “netxms”.

Agent

Install the core agent package (“netxms-agent”) and optional subagent packages, if required:

```
apt-get install netxms-agent
```

Start agent

```
systemctl start netxms-agent
```

Enable automatic startup of agent

```
systemctl enable netxms-agent
```

Management Client

Desktop Management Client

Due to a limitation of the Eclipse platform used to build the Management Client, only a x64 build is provided.

1. Make sure you have 64-bit Java version 17 installed on your system.

2. Download the latest .jar file from <http://www.netxms.org/download/>, for example nxmc-5.1.0-standalone.jar.
3. Run the .jar file using java, for example `java -jar nxmc-xxx.jar`.

The desktop management client produces a log file named `.nxmc/data/.metadata/.log` in the home folder of the currently logged in user. Inspect this log file if you encounter errors when running the client.

Web Management Client

The NetXMS web interface is java based and should be deployed into a servlet container to run. Minimal supported versions are: Jetty 10, Tomcat 9. The supported Java version is 17 or later.

1. Install one of the servlet containers that support servlet-api version 4.
2. Download the latest version of WAR file from the Web Interface Binaries section <https://www.netxms.org/download/> named nxmc-VERSION.war, for example nxmc-5.1.0.war.
3. Copy nxmc.war to the webapps directory. In a few seconds it will be autodeployed and available at http://SERVER_IP:SERVER_PORT/nxmc/

Tomcat default folder: `/var/lib/tomcat9/webapps`

Jetty default folder: `$JETTY_HOME/webapps/`

The web management client produces a log file. For Tomcat it is located at `/var/lib/tomcat9/work/Catalina/localhost/nxmc/eclipse/workspace/.metadata/.log`. Inspect this log file if you encounter errors when running the web client.

3.4 Installing from RPM repository

We provide RPM packages for RHEL and Fedora, both amd64 and aarch64. If you need a build for another system, please contact us for support or check this section: *Installing from source*.

The RHEL repository is at <https://packages.netxms.org/epel/>.

The Fedora repository is at <https://packages.netxms.org/fedora/>.

A complete repository file and signing key is available in each corresponding root.

3.4.1 Add repository

DNF provides a simple way to add a repository. Please note that you may need to install the EPEL repository first. See details):

```
# RHEL and compatible
dnf config-manager --add-repo https://packages.netxms.org/epel/netxms.repo
# Fedora
dnf config-manager --add-repo https://packages.netxms.org/fedora/netxms.repo
```

Once added, you can install any package with `dnf install` (e.g. `dnf install netxms-agent`).

3.4.2 Installing packages

Server

The server requires two components to function - the server itself (package “netxms-server”) and at least one database abstraction layer driver (multiple can be installed at the same time, e.g. for migration purposes). These database drivers are also used by the agent for database monitoring (performing queries to databases).

Provided driver packages:

- netxms-dbdrv-pgsql - PostgreSQL driver
- netxms-dbdrv-mariadb - Mariadb driver
- netxms-dbdrv-mysql - MySQL driver, currently under development (not built for Ubuntu 20 / Mint 20)
- netxms-dbdrv-odbc - unixODBC driver (can be used with DB/2 and Microsoft SQL)
- netxms-dbdrv-oracle - Oracle driver (requires Oracle client installation)

1. Instal required packages (adjust command to match your environment):

```
dnf install netxms-server netxms-dbdrv-pgsql
```

2. Create user and database (*examples*).

3. Modify the server configuration file (“/etc/netxmsd.conf” to match your environment.

4. Load database schema and default configuration:

```
nxdbmgr init
```

5. Start server:

```
systemctl start netxms-server.service
```

6. Enable automatic startup of server:

```
systemctl enable netxms-server.service
```

7. If the database engine is running on the same system, add ordering dependency for database into netxmsd systemd unit override file. This will ensure database shutdown only after netxmsd process completion on system shutdown/restart. To add the dependency e.g. for the PostgreSQL database, run:

```
systemctl edit netxmsd
```

and add the following lines:

```
[Unit]
After=network.target postgresql.service
```

After editing, run `systemctl daemon-reload` to reload systemd configuration.

Note

Default credentials - user “admin” with password “netxms”.

Agent

Install the core agent package (“netxms-agent”) and optional subagent packages, if required:

```
dnf install netxms-agent
```

Start agent

```
systemctl start netxms-agent
```

Enable automatic startup of agent

```
systemctl enable netxms-agent
```

Management Client

Desktop Management Client

Due to a limitation of the Eclipse platform used to build the Management Client, only a x64 build is provided.

1. Make sure you have 64-bit Java version 17 installed on your system.
2. Download the latest .jar file from <https://www.netxms.org/download/>, for example nxmc-5.1.0-standalone.jar.
3. Run the .jar file using java, for example `java -jar nxmc-xxx.jar`.

The desktop management client produces a log file named `.nxmc/data/.metadata/.log` in the home folder of the currently logged in user. Inspect this log file if you encounter errors when running the client.

Web Management Client

The NetXMS web interface is java based and should be deployed into a servlet container to run. Minimal supported versions are: Jetty 10, Tomcat 9. The supported Java version is 17, but is found to be working with later versions, for example 21.

1. Install one of the servlet containers that support servlet-api version 4.
2. Download the latest version of WAR file from Web Interface Binaries section <https://www.netxms.org/download/> named nxmc-VERSION.war, for example nxmc-5.0.6.war.
3. Copy nxmc.war to the webapps directory. In a few seconds it will be autodeployed and available at http://SERVER_IP:SERVER_PORT/nxmc/

Tomcat default folder: `/var/lib/tomcat9/webapps`

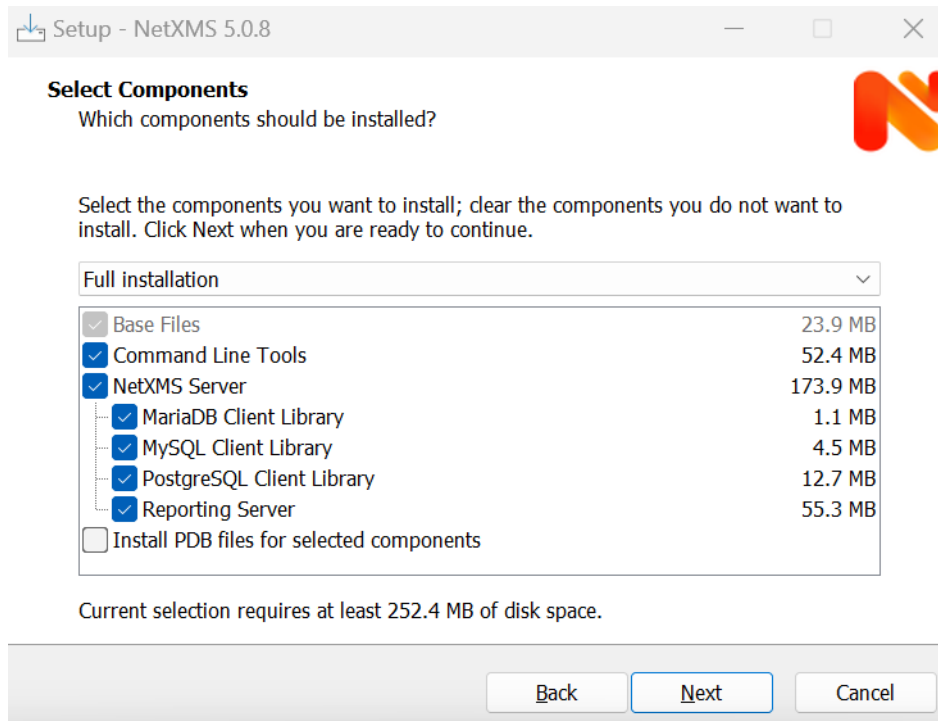
Jetty default folder: `$JETTY_HOME/webapps/`

The web management client produces a log file. For Tomcat it is located at `/var/lib/tomcat9/work/Catalina/localhost/nxmc/eclipse/workspace/.metadata/.log`. Inspect this log file if you encounter errors when running the web client.

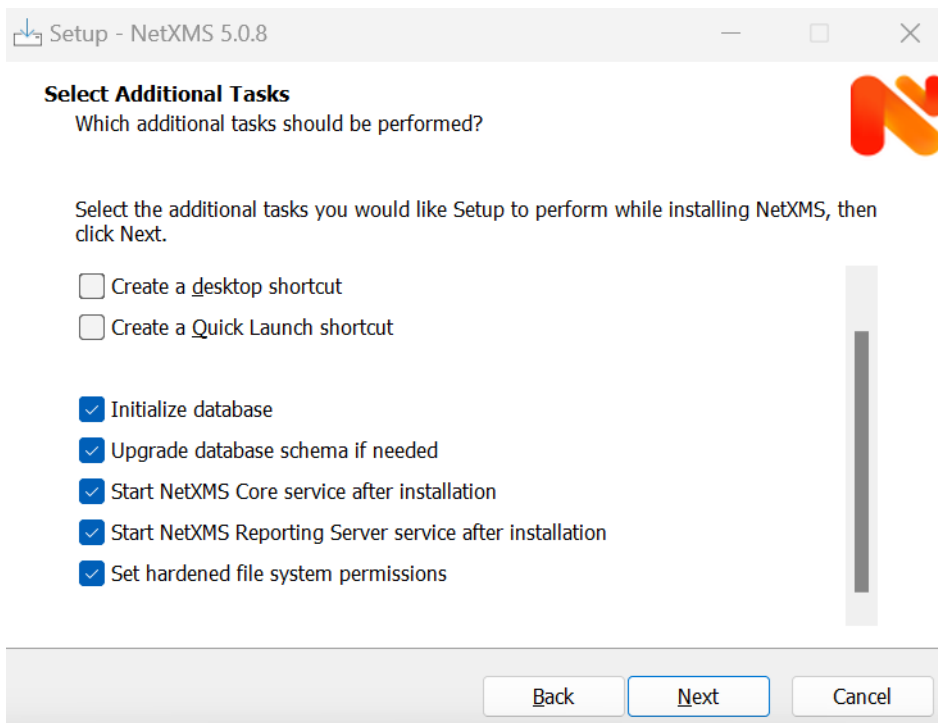
3.5 Installing on Windows

3.5.1 Server

1. Download the latest version from <http://www.netxms.org/download/>. You will need Windows the installer named netxms-VERSION-x64.exe, e.g. netxms-server-5.0.8-x64.exe. Please note that in the following steps VERSION will be used as a substitution for an actual version number.
2. Run the installer package on your server. The installation wizard will be displayed. Follow the prompts until the Select Components window opens.
3. On the Select Components window, select the NetXMS Server option and an appropriate database client library. You do not have to install a database client library from NetXMS package if it is already installed on the machine (however, it might be required to add the folder where the client library is installed to system path).



4. For a typical installation, keep default settings in the Select Additional Tasks window. *Set hardened file system permissions* makes the installation folder accessible only to members of the Administrators group and the SYSTEM user.



5. The Database selection window will open:

Initialize Database
Initialize database for use by NetXMS server

Database type
PostgreSQL

Database server
localhost

Database name
netxms_db

Login name
netxms

Password
•••••

☒ Create database and database user before initialization

DBA login name
postgres

DBA password
•••••

Back Next Cancel

- Select the desired database type. Enter the name of database server.
- In the DBA login name and DBA password fields, enter the database administrator login name and password. You have to fill these fields only if you have chosen the *Create database and database user before initialization* option.
- Enter the desired database name, database user name and password.

Note for MySQL:

The bundled MySQL database driver does not support caching_sha2_password authentication which is the default for MySQL starting from version 8. Either select Legacy Authentication Method when installing MySQL, or use the database driver installed along with MySQL. The database driver gets installed when installing MySQL with Server-only option, however these two folders should be included into system path: C:\Program Files\MySQL\MySQL Server 8.0\lib C:\Program Files\MySQL\MySQL Server 8.0\bin.

Note for Microsoft SQL Server:

Please refer to the Appendix for detailed Windows/MSSQL setup installation instructions [instructions](#)

Note for Oracle:

We recommend to use the native database driver (oracle.ldr).

6. On the Ready to Install window, check whether everything is correct, then press the Install button.
7. After installation, start the Netxms client and connect with the following credentials

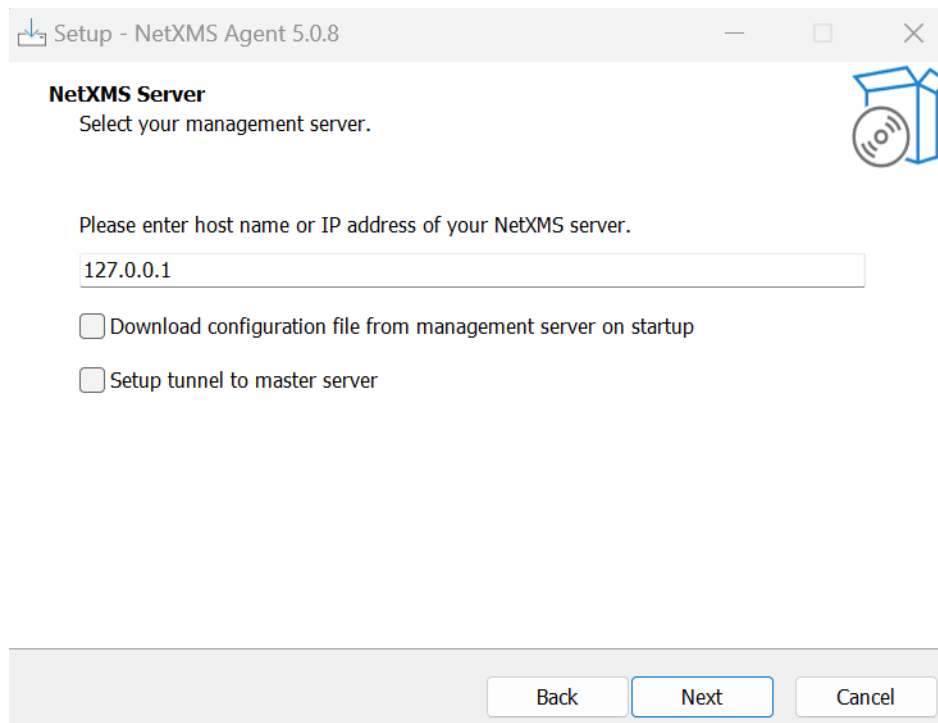
Server default credentials:

Login: admin

Password: netxms

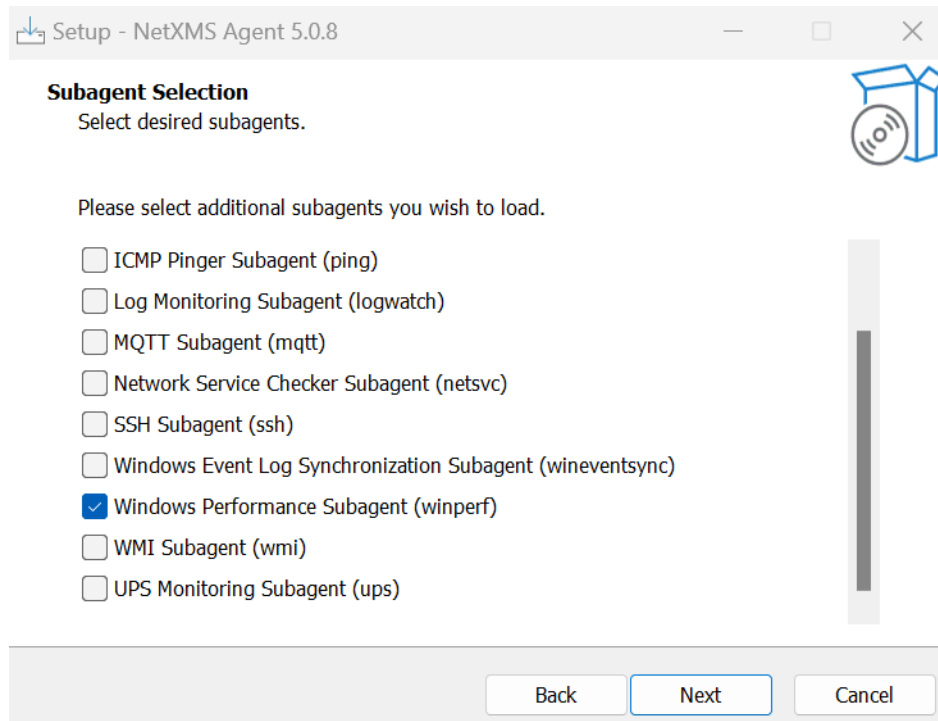
3.5.2 Agent

1. Download the latest version from <http://www.netxms.org/download/>. You will need Windows Agent installer (named nxagent-VERSION.exe or nxagent-VERSION-x64.exe, for example nxagent-5.0.8-x64.exe).
2. Run the installer package on the target server. The installation wizard will be displayed. Follow the prompts until the NetXMS Server window opens:



Enter the IP address or host name of your NetXMS server. You can specify multiple management servers, separating them by commas. Press the Next button to continue.

3. The subagent selection window will open:



In this window you can select which subagents you wish to load. Each subagent extends the agents functionality, e.g.:

Subagent	Description
filemgr.nsm	Provides access to specified folders on the monitored host from the NetXMS Management Client File Manager. This is also used for distributing Agent Policy configuration files (see Agent Policies .)
logwatch	Allows monitoring log files and Windows Event Log and sending matched events to NetXMS server.
ping.nsm	Adds the possibility to send ICMP pings from the monitored host. Ping round-trip times can be collected by management server.
netsvc.nsm, portcheck.nsm	Adds the possibility to check network services (like FTP or HTTP) from the monitored host.
winperf.nsm	Provides access to Windows performance counters. This subagent is required if you need to collect CPU utilization from monitored host.
wmi.nsm	Provides access to WMI data.
ups.nsm	Adds support for UPS monitoring. The UPS can be attached to host via a serial cable or USB.

For more information on subagents, please refer to [Subagents](#).

1. Follow the prompts to complete the installation.

3.5.3 Management Client

Desktop Management Client:

1. Download the latest version from <https://www.netxms.org/download/>. Since version 3.8 there are three options - archive (e.g. nxmc-5.0.8-win32-x64.zip), archive with bundled JRE (nxmc-5.0.8-win32-x64-bundled-jre.zip) and installer, which also has JRE bundled (e.g. netxms-client-5.0.8-x64.exe). If using the archive without JRE,

make sure you have JRE version 11 or 15 installed. Due to a limitation of the Eclipse platform used to build the Management Client, only an x64 build is currently provided.

2. If using the archive version, extract the zip in the preferred directory. If using the installer, launch it and follow the instructions.
3. Run the nxmc file from the extracted catalog, or launch from the Windows Start Menu, if you used the installer.

Web Management Client:

On the Windows platform there are two options: one is to manually install the .war file into a servlet container and the second one is to use the netxms-webui-VERSION-x64.exe installer. The installer will install Jetty and copy the .war file into required folder. Here the installation via the installer is described:

1. Download the latest version from <https://www.netxms.org/download>. You will need Windows installer netxms-webui-VERSION-x64.exe (e.g.: netxms-webui-5.0.8-x64.exe). Due to a limitation of the Eclipse platform used to build the Management Client, only an x64 build is currently provided.
2. Run the installer package on your server. The Installation wizard will be displayed. Follow the prompts. The installer allows to change the installation path and port.
3. After the installation procedure is finished, check that the WEB GUI is available at http://SERVER_IP:SERVER_PORT/nxmc/

3.5.4 Unattended installation of the NetXMS Agent

The Windows Agent installer, named nxagent-VERSION.exe, for example nxagent-5.0.8-x64.exe, has various command line options for unattended installation. Installation will ignore any configuration file options (/CONFIGENTRY, /NO-SUBAGENT, /SERVER, /SUBAGENT, etc) if a config file already exists or if the /CENTRALCONFIG option is used. However, it is possible to delete and recreate the configuration file using the /FORCECREATECONFIG command line option.

The options are the following:

Option	Description
/CENTRALCONFIG	Enable read configuration from server on startup. See Agent configuration files on server for more information.
/CONFIGENTRY=value	It can be used to add any parameter to the configuration file during initial install. You can specify it multiple times to add multiple lines. Section names can be added as well.
/CONFIGINCLUDEDIR=path	Set folder containing additional configuration files (will be set in configuration file as <code>ConfigIncludeDir</code>).
/DIR=path	Set installation directory (default is <code>C:\NetXMS</code>).
/FILESTORE=path	Sets directory to be used for storing files uploaded by management server(s) (will be set in configuration file as <code>FileStore</code>).
/FORCECREATECONFIG	Delete existing agent configuration file and recreate it. However, settings stored by installer in Windows registry will be used, if not explicitly specified by command line parameters. See <code>/IGNOREPREVIOUSDATA</code> .
/IGNOREPREVIOUSDATA	Ignore any settings from previous install that are not explicitly specified in current run. This is related to settings that can be changed when installer is run in GUI mode, e.g. list of selected sub-agents. These settings are stored in Windows registry.
/LOCALCONFIG	Use local configuration file (it is the default).
/LOG	Causes Setup to create a log file in the TEMP directory of the user detailing file installation and [Run] actions taken during the installation process.
/LOG=filename	Same as /LOG, except it allows to specify a fixed path/filename to use for the log file. If a file with the specified name already exists it will be overwritten. If the file cannot be created, Setup will abort with an error message.
/LOGFILE=filename	Set agent log file (will be set in configuration file as <code>LogFile</code>).
/MERGETASKS="tasknames"	Comma-separated list of tasks for installation. If a task is specified with ! character prior to its name, it will be deselected. Possible values are <code>fspermissions</code> - set hardened file system permissions, <code>sessionagent</code> - Install session agent, <code>useragent</code> - Install user support application. e.g. <code>/MERGETASKS="!fspermissions, useragent"</code>
/NOSUBAGENT=name	Disable subagent name
/NOTUNNEL	Disable tunnel operation (it is the default)
/REINSTALLSERVICE	Reinstalls Windows service
/SERVER=IP	Set server IP address or host name (will be set in the configuration file as <code>MasterServers</code>).
/SILENT	Don't show installation wizard, only a progress bar
/SUBAGENT=name	Add sub-agent loading directive to configuration file. You can specify this parameter multiple times to add more than one sub-agent. List of possible subagents: Subagents .
/SUPPRESSMSGBOXES	Don't ask user anything. Only has an effect when combined with <code>/SILENT</code> and <code>/VERYSILENT</code> .
/TUNNEL	Enable tunnel operation to IP address specified with <code>/SERVER=</code> .
/VERYSILENT	Don't show anything

Example:

```
nxagent-5.0.8-x64.exe /VERYSILENT /SUPPRESSMSGBOXES /SERVER=10.0.0.1 /
SUBAGENT=UPS /SUBAGENT=FILEMGR /CONFIGENTRY=ZoneUIN=15 /CONFIGENTRY=[FILEMGR] /
CONFIGENTRY=RootFolder=C:\
```

This command will add 3 lines at the end of generated config file:

```
ZoneUIN=15
[FILEMGR]
```

(continues on next page)

(continued from previous page)

`RootFolder=C:\`

3.5.5 Unattended uninstallation of NetXMS Agent

The uninstaller application is named `unins???.exe` and is located in the agent folder (`C:\NetXMS` by default). The following options are supported:

Option	Description
<code>/SILENT</code>	Don't show uninstallation wizard, only a progress bar
<code>/VERYSILENT</code>	Don't show anything
<code>/LOG</code>	Causes to create a log file in the TEMP directory of the user.
<code>/LOG=filename</code>	Same as <code>/LOG</code> , except it allows to specify a fixed path/filename to use for the log file.
<code>/SUPPRESSMSGBOXES</code>	Don't ask user anything. Only has an effect when combined with <code>/SILENT</code> and <code>/VERYSILENT</code> .
<code>/NORESTART</code>	Instructs the uninstaller not to reboot even if it would be necessary.

Example:

```
unins000.exe /SUPPRESSMSGBOXES /VERYSILENT /NORESTART
```

3.6 Install on Android

3.6.1 Management Client

To install Android management client download `netxms-console-VERSION.apk` (example: `netxms-console-3.4.178.apk`) file from the <http://www.netxms.org> page. Check that installation of applications from unknown sources is allowed in security settings of your phone. Run this installer on required device.

After the agent is installed, go to settings and in the main menu, connection part, set all required connection credentials: server address, port, user name, password.

Note

The user configured for the connection should have the *Login as mobile device* user permission.

3.7 Installing from sources

3.7.1 Server

1. Download the source archive (`netxms-VERSION.tar.gz`) from <https://www.netxms.org/download/>. *VERSION* is used in names instead of an actual version number.
2. Unpack the archive:

```
tar zxvf netxms-VERSION.tar.gz
```

3. Since version 3.8, the reporting server is being built along with the sources. This requires maven to be installed on the system. You need Oracle and MS SQL JDBC drivers in your local maven repository.

The Oracle JDBC driver library can be obtained here: https://oracle.com/otn-pub/otn_software/jdbc/199/ojdbc8.jar

the Microsoft SQL JDBC driver library can be obtained from here: <https://www.microsoft.com/en-us/details.aspx?id=54671> You will need sqljdbc_4.2/enu/jre8/sqljdbc42.jar file from this archive.

To install these libraries: `mvn install:install-file -DgroupId=com.microsoft.sqlserver -DartifactId=sqljdbc4 -Dversion=4.2 -Dpackaging=jar -Dfile=sqljdbc42.jar mvn install:install-file -DgroupId=com.oracle -DartifactId=ojdbc8 -Dversion=12.2.0.1 -Dpackaging=jar -Dfile=ojdbc8.jar`

4. Change directory to netxms-VERSION and run the configure script:

```
cd netxms-VERSION
```

```
./configure --enable-release-build --with-server --with-pgsql --with-agent
```

Most commonly used options (check full list with `./configure --help`):

Name	Description
<code>--prefix=DIRECTORY</code>	Installation prefix, all files go to the specified directory (e.g. <code>--prefix=/opt/netxms</code>)
<code>--with-server</code>	Build server binaries. You will need to select at least one DB driver as well
<code>--with-agent</code>	Build monitoring agent. It is strongly recommended to install agent on a server box
<code>--with-pgsql</code>	Build PostgreSQL DB Driver (if you plan to use PostgreSQL as backend database)
<code>--with-mysql</code>	Build MySQL DB Driver (if you plan to use MySQL as backend database)
<code>--with-odbc</code>	Build ODBC DB driver (if you plan to connect to your backend database via unixODBC)
<code>--with-sqlite</code>	Build SQLite DB driver (if you plan to use embedded SQLite database as backend database)

5. Run build binaries and install them into /usr/local (unless changed with configure flag `--prefix`)

```
make
```

```
make install
```

6. Copy sample config file:

```
cp contrib/netxmsd.conf-dist /usr/local/etc/netxmsd.conf
```

By default, server load configuration file PREFIX/etc/netxmsd.conf (where PREFIX is installation prefix set by configure), unless different file is specified with command line switch `“-c”`.

7. Create database user and adjust configuration file (netxmsd.conf) accordingly. Database creation examples can be found [there](#).

8. Further adjust server configuration file if required.

Detailed information about each configuration parameter can be found in section [Server configuration file \(netxmsd.conf\)](#).

9. Create required tables and load initial configuration using nxdbmgr utility:

```
/usr/local/bin/nxdbmgr init
```

10. Run server:

```
/usr/local/bin/netxmsd -d
```

3.7.2 Agent

1. Download the source archive (netxms-VERSION.tar.gz) from <https://www.netxms.org/download/>. *VERSION* is used in names instead of an actual version number.
2. Unpack the archive:

```
tar zxvf netxms-VERSION.tar.gz
```

3. Change directory to netxms-VERSION and run the configure script:

```
cd netxms-VERSION
```

```
./configure --enable-release-build --with-agent
```

Most commonly used options (check full list with `./configure --list`):

Name	Description
<code>--prefix=DIRECTORY</code>	Installation prefix, all files go to the specified directory
<code>--with-agent</code>	Build monitoring agent. It is strongly recommended to install the agent on a server

4. Run build binaries and install them into /usr/local (unless changed with configure flag `--prefix`)

```
make
```

```
make install
```

5. Copy sample config file:

```
cp contrib/nxagentd.conf-dist /usr/local/etc/nxagentd.conf
```

By default the agent load configuration file is PREFIX/etc/netxmsd.conf (where PREFIX is installation prefix set by configure), unless a different file is specified with the command line switch “-c”.

6. Adjust the agent configuration file if required.

Detailed information about each configuration parameter can be found in section *Agent configuration file (nxagentd.conf)*.

Minimal required configuration:

```
MasterServers = 172.16.1.1 # server IP - agent will drop connections unless
↪address is provided here
LogFile = /var/log/nxagentd
```

7. Run agent:

```
/usr/local/bin/nxagentd -d
```

3.8 Customizing the compilation process

3.8.1 Adding additional compiler or linker flags

(e.g. fixing atomics)

3.9 WebUI additional configuration

There are a few settings available for configuration of the WebUI.

- autoLoginOnReload - autologin on page reload in browser (default: true)
- enableCompression - enable protocol compression between Web UI and server process (default: true)
- loginFormImage - path to custom login image
- loginFormImageBackground - colour of background around custom login image
- loginFormImageMargins - margins in px around custom login image (default: 10)
- server - server DNS name or IP (default: 127.0.0.1)

There are multiple ways to set the connection configuration from WebUI to NetXMS server. Configuration is checked in this order:

1. Using JNDI. Environment should be set like nxmc/NAME for example: nxmc/server
2. nxmc.properties properties file in the class path of your application server. This file should be created in ini format: NAME=VALUE. For example:

```
server = 127.0.0.1
```

Default locations:

Jetty

Tomcat

The default location of this file on Debian and Ubuntu is in /usr/share/tomcat9/lib. Other Linux distributions may use a different location.

Oracle Weblogic

\$WEBLOGIC_HOME/user_projects/domains/YOURDOMAIN

3. jvm parameter in format -Dnxmc.NAME=VALUE. For example: -Dnxmc.server=127.0.0.1
4. Environment variable NXMC_NAME=VALUE. For example NXMC_server=127.0.0.1
5. If none of the above configurations exist, the Web UI tries to resolve the “NETXMS_SERVER” DNS name as server connection.
6. If none of above configurations exist, the Web UI uses “127.0.0.1” as a server address.

3.9.1 Custom logo on login screen

It is possible to change the default logo on the login screen to a custom image by setting the loginFormImage property in nxmc.properties file. The image file must be located within the application server class path and the file name must be given relative to the class path root with a leading slash. For example, if the custom image is in a file logo.jpg located in the same directory as nxmc.properties, the correct entry will be:

```
loginFormImage = /logo.jpg
```

3.9.2 How to configure the NetXMS web client with jetty in Linux

1. Download the latest version of Jetty (12.0.13 at the moment of writing).

```
curl -O https://repo1.maven.org/maven2/org/eclipse/jetty/jetty-home/12.0.13/jetty-  
home-12.0.13.tar.gz
```


2. Create directories and extract Jetty, then create the initial configuration by running start.jar.

```
tar -xvf jetty-home-12.0.13.tar.gz -C /opt

ln -s /opt/jetty-home-12.0.13 /opt/jetty-home-12

mkdir -p /opt/netxms-webui/{etc,logs} && cd /opt/netxms-webui

java -jar /opt/jetty-home-12/start.jar --add-modules=ee8-deploy,gzip,http,http2,https,
↳ logging-logback,plus,server,ssl,work
```

3. Download the war file (version 5.1.2 at the moment of writing) and place it in the webapps directory.

```
curl -o webapps/ROOT.war https://netxms.com/releases/5.1/nxmc-5.1.2.war
```

4. Generate ssl key (for testing purposes) and adjust the ssl.ini file. A reverse proxy with proper certificate should be used in production. Adjust DN, keyStorePassword and keyStorePath as per requirements.

```
keytool -genkeypair -alias jetty -keyalg RSA -keysize 2048 -keystore /opt/netxms-
↳ webui/etc/keystore.p12 -storetype PKCS12 -storepass password -keypass password -
↳ validity 3650 -dname "CN=netxms-webui, OU=netxms, O=netxms, L=netxms, ST=netxms,
↳ C=netxms"

sed 's,# jetty.sslContext.keyStorePassword=,jetty.sslContext.
↳ keyStorePassword=password,-i' start.d/ssl.ini
```

5. Run Jetty to verify the configuration. Once verified, stop with Ctrl+C.

```
java -Dnxmc.logfile=/opt/netxms-webui/logs/nxmc.log -jar /opt/jetty-home-12/start.jar
```

6. Create a systemd service file for Jetty (sample is bellow).

```
systemctl edit --force --full netxms-webui.service
```

```
[Unit]
Description=NetXMS WebUI
StartLimitIntervalSec=0

[Service]
Type=simple
WorkingDirectory=/opt/netxms-webui
Environment=JETTY_HOME=/opt/jetty-home-12
Environment=JETTY_BASE=/opt/netxms-webui
User=jetty
Group=jetty
ExecStart=java -Dnxmc.logfile=/opt/netxms-webui/logs/nxmc.log -jar /opt/jetty-home-
↳ 12/start.jar
Restart=on-failure
RestartSec=30
TimeoutSec=900

[Install]
WantedBy=multi-user.target
EnableDefaultCounters = yes
```

7. Enable netxms-web.service and start it.

```
systemctl enable --now netxms-web.service
```

3.10 Default login credentials

The default login is “admin” with password “netxms”. On first login, the user will be prompted to change their password immediately.

If required, the password can be reset back to default using *nxdbmgr utility*.

3.11 Database creation examples

This chapter provides some database creation SQL examples. Please consult the relevant database documentation for the initial install.

3.11.1 PostgreSQL

```
createuser -P netxms
createdb -O netxms netxms
```

If the TimescaleDB extension is to be used, it should be added to the newly created database:

```
psql netxms
CREATE EXTENSION IF NOT EXISTS timescaledb CASCADE;
\q
```

Configuration file example:

```
DBDriver = pgsqldriver
DBServer = localhost
DBName = netxms
DBLogin = netxms
DBPassword = PaSsWd
```

3.11.2 MariaDB

```
echo "CREATE DATABASE netxms CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;" | \
mysql -u root -p
echo "CREATE USER 'netxms'@'localhost' IDENTIFIED BY 'PaSsWd';" | mysql -u root -p
echo "GRANT ALL on netxms.* to 'netxms'@'localhost';" | mysql -u root -p
```

Configuration file example:

```
DBDriver = mariadb.driver
DBServer = localhost
DBName = netxms
DBLogin = netxms
DBPassword = PaSsWd
```

3.11.3 MySQL

```
echo "CREATE DATABASE netxms CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;" | \
↪mysql -u root -p
echo "CREATE USER 'netxms'@'localhost' IDENTIFIED BY 'PaSsWd';" | mysql -u root -p
echo "GRANT ALL on netxms.* to 'netxms'@'localhost';" | mysql -u root -p
```

Configuration file example:

```
DBDriver = mysql.ldr
DBServer = localhost
DBName = netxms
DBLogin = netxms
DBPassword = PaSsWd
```

3.11.4 Oracle

```
-- USER SQL
CREATE USER netxms IDENTIFIED BY PaSsWd
DEFAULT TABLESPACE USERS
TEMPORARY TABLESPACE TEMP;
-- QUOTAS
ALTER USER netxms QUOTA UNLIMITED ON USERS;
-- ROLES
GRANT CREATE SESSION, CREATE TABLE, CREATE PROCEDURE TO netxms;
```

Configuration file example:

```
DBDriver = oracle.ldr
DBServer = //127.0.0.1/XE # instant client compatible connection string
DBLogin = netxms
DBPassword = PaSsWd
```

3.11.5 How to install NetXMS server on Windows Server with local Microsoft SQL Server Express

1. Login as administrator
2. Install Microsoft SQL Server Express with default options.

If enabling mixed authentication mode:

3. Enable mixed authentication mode as per <https://learn.microsoft.com/en-us/sql/database-engine/configure-windows/change-server-authentication-mode> Don't forget to restart SQL Server after changing authentication mode.
4. Run NetXMS Server installer. When prompted for database information, use the following answers:
 - Server type: MS SQL
 - Server name: localhostSQLEXPRESS
 - Database name: (any valid name, we use "netxms")
 - Login name: (any valid account name, we use "netxms")
 - Password: (any password complex enough to match OS password policy)

- Create database and database user: check
- DBA login name: *
- DBA password: (left empty)

This assumes the currently logged in user has DBA access to the SQL Server instance. This should be the case if SQL Server was just installed by the same user. An alternative approach is to enable the “sa” user in SQL server and use sa login and password as DBA login name and password.

The installer should create database, database user, assign the user as database owner, and the NetXMS Core service should start successfully.

If mixed authentication is not an option:

Currently the installer does not support automatic database creation for Windows authentication mode, so there will be more manual steps.

3. Login to SQL Server Management Studio
4. Create a new database with the default owner (owner should be set to currently logged in administrator user)
5. Run the NetXMS Server installer. On “Select additional tasks” page uncheck “Start NetXMS Core service”.
6. When prompted for database information, use the following answers:
 - Server type: MS SQL
 - Server name: localhostSQLEXPRESS
 - Database name: (database name from step 4)
 - Login name: *
 - Password: (left empty)
 - Create database and database user: uncheck
7. After installation is complete, go to “Services”, find the “NetXMS Core” service, and set it to login as administrator user (same user used for installation)
8. Start NetXMS Core service

3.11.6 How to install NetXMS server on Windows Server with remote Microsoft SQL Server Express

Assumptions:

- Both the SQL Express Server and the NetXMS Server are in the same domain
- TCP/IP is enabled in SQL Server network properties
- TCP/IP is configured to use a fixed port
- A firewall rule is added to allow incoming connections on the SQL Server TCP port (it may be needed to add this manually)
- Mixed authentication mode is already enabled on SQL Server (only for scenario 1 below)

If using a SQL account for NetXMS services is acceptable

1. Log in to the NetXMS Server machine with a domain account that has local administrator rights as well as sysadmin rights on SQL Server
2. Install ODBC Driver for SQL Server
3. Run the NetXMS Server installer. When prompted for database information, use the following answers:

- Server type: MS SQL
- Server name: SQL server domain computer name or fully qualified DNS name (if the TCP port is not 1433, then use the form server_name,port)
- Database name: (any valid name, we use “netxms”)
- Login name: (any valid account name, we use “netxms”)
- Password: (any password complex enough to match OS password policy)
- Create database and database user: check
- DBA login name: *
- DBA password: (left empty)

The installer should create database, database user, assign user as database owner, and the NetXMS Core service should start successfully.

In this scenario the server will use login and password on SQL server, so the service can continue to run under Local System account, or you can change it to any domain account.

If the server has to use domain account for accessing the database

1. Install ODBC Driver for SQL Server
2. If not already done, create a new login on SQL Server for the domain user to be used by NetXMS Core service
3. Create the new database, assign login from step 2 as owner
4. Log in to the NetXMS Server machine with the same domain user
5. Run the NetXMS Server installer. On “Select additional tasks” page, uncheck “Start NetXMS Core service”.
6. When prompted for database information, use the following answers:
 - Server type: MS SQL
 - Server name: SQL server domain computer name or fully qualified DNS name (if the TCP port is not 1433, then use the form server_name,port)
 - Database name: (database name from step 4)
 - Login name: *
 - Password: (left empty)
 - Create database and database user: uncheck
7. After installation is complete, go to “Services”, find the “NetXMS Core” service, and set it to login as administrator user (same user used for installation)
8. Start the NetXMS Core service

UPGRADE

4.1 Upgrading on Debian or Ubuntu

4.1.1 Upgrading server and agent

1. It's recommended to check database for possible inconsistencies prior to the upgrade. To do this, stop the server and run command:

```
nxdbmgr check
```

Proceed to the next step only if database checker does not report any errors!

2. To update NetXMS server and agent packages run command:

```
apt-get update && apt-get upgrade
```

During package upgrade database schema should be upgraded as well and NetXMS server would start automatically. However, in some cases (e.g. if database engine packages were also upgraded) automatic database upgrade may not happen. If this is the case, NetXMS server won't get started and it's log would show, e.g.: Your database has format version 41.07, but server is compiled for version 41.18. To upgrade the database, run command:

```
nxdbmgr upgrade
```

Once database upgrade is complete, start the server.

Management client

Desktop Management Client:

1. Download the latest version from <http://www.netxms.org/download>. You will need Linux installer (named nxmc-VERSION-linux-gtk-x86.tar.gz or nxmc-VERSION-linux-gtk-x64.tar.gz, for example nxmc-5.1.0-linux-gtk-x64.tar.gz).
2. Extract and replace old management client with the new one.

```
tar xzvf nxmc-VERSION-linux-gtk-x86.tar.gz -C /DIRECTORY
```

3. Run nxmc file from extracted catalog.

Web Management Client:

1. Download latest version of WAR file from Web Interface Binaries section <http://www.netxms.org/download/> (named nxmc-VERSION.war, for example nxmc-5.1.0.war).

2. Replace old WAR file with the new one.

Sometimes it's possible that new WAR file is not detected and previous version of WAR continues to run. In this case stop servlet container, delete the WAR file. Then start servlet container and copy the war file to webapps directory.

4.2 Upgrading on Red Hat, Fedora, CentOS or ScientificLinux

4.2.1 Upgrading

Server

1. Download the latest version from <http://www.netxms.org/download>, if you don't have it. You will need source archive (named netxms-VERSION.tar.gz, for example netxms-1.2.15.tar.gz). Please note that in the following steps VERSION will be used as a substitution for an actual version number.
2. Unpack the archive:

```
tar zxvf netxms-5.1.0.tar.gz
```

3. Change directory to netxms-version and run configure script and make:

```
cd netxms-5.1.0

sh ./configure --enable-release-build --with-server --with-mysql

make
```

Be sure to include all configuration options that were used at installation time.

4. Stop NetXMS server.
5. Stop NetXMS agent.
6. Check database for possible inconsistencies:

```
nxdbmgr check
```

Proceed to the next step only if database checker does not report any errors!

7. Run make install:

```
make install
```

8. Upgrade database:

```
nxdbmgr upgrade
```

9. Start NetXMS agent.
10. Start NetXMS server.

Agent

1. Download the latest version from <http://www.netxms.org/download>, if you don't have it. You will need source archive (named netxms-VERSION.tar.gz, for example netxms-5.1.0.tar.gz). Please note that in the following steps VERSION will be used as a substitution for an actual version number.
2. Unpack the archive:


```
tar zxvf netxms-5.1.0.tar.gz
```

3. Change directory to netxms-version and run configure script and make:

```
cd netxms-5.1.0`

sh ./configure --enable-release-build --with-agent

make
```

Be sure to include all configuration options that were used at installation time.

4. Stop NetXMS agent.
5. Run make install:

```
make install
```

6. Run agent:

```
/usr/local/bin/nxagentd -d
```

Management Client

Desktop Management Client:

1. Download the latest version from <http://www.netxms.org/download>. You will need Linux installer(named nxmc-VERSION-linux-gtk-x86.tar.gz or nxmc-VERSION-linux-gtk-x64.tar.gz, for example nxmc-5.1.0-linux-gtk-x64.tar.gz).
2. Extract and replace old management client with the new one.

```
tar zxvf nxmc-VERSION-linux-gtk-x86.tar.gz -C /DIRECTORY
```

3. Run nxmc file from extracted catalog.

```
cd /<path_to_nxmc>

./nxmc &
```

Web Management Client:

1. Download latest version of WAR file from Web Interface Binaries section <http://www.netxms.org/download/> (named nxmc-VERSION.war, for example nxmc-5.1.0.war).
2. Replace old WAR file with the new one.

Sometimes it's possible that new WAR file is not detected and previous version of WAR continues to run. In this case stop servlet container, delete the WAR file. Then start servlet container and copy the war file to webapps directory.

4.3 Upgrading on Windows

4.3.1 Upgrade

Server

1. Download the latest version from <http://www.netxms.org/download>, if you don't have it. You will need Windows installer (named netxms-VERSION.exe, for example netxms-5.1.0.exe).
2. Stop NetXMS server.
3. Check database for possible inconsistencies:

```
C:\NetXMS\bin> nxdbmgr check
```

Proceed to the next step only if database checker does not report any errors!

4. Run NetXMS installer and follow the prompts. Normally, you will not need to change any settings on installation wizard windows. Alternatively, you can run the installer with /SILENT option to disable any prompts:

```
C:\Download> netxms-5.1.0.exe /SILENT
```

5. Check whether NetXMS Server service is running again. If it's not, most likely you have to upgrade your database to newer version. To upgrade database, use nxdbmgr utility:

```
C:\NetXMS\bin> nxdbmgr upgrade
```

6. Start NetXMS server, if it is not already started.

Agent

We highly recommend using centralized agent upgrade feature for agent upgrades. However, if you decide to upgrade agent manually, it can be done in just a few steps:

1. Download the latest version from <http://www.netxms.org/download>, if you don't have it. You will need Windows Agent installer (named nxagent-VERSION.exe or nxagent-VERSION-x64.exe, for example nxagent-5.1.0.exe).
2. Run NetXMS agent installer and follow the prompts. Normally, you will not need to change any settings on installation wizard dialog windows. Alternatively, you can run installer with /SILENT option to disable any prompts:

```
C:\Download> nxagent-5.1.0.exe /SILENT
```

Management Client

Desktop Management Client:

1. Download the latest version from <http://www.netxms.org/download>. You will need Windows installer (named nxmc-VERSION-win32-x86.zip or nxmc-VERSION-win32-x64.zip, for example nxmc-5.1.0-win32-x64.zip).
2. Replace old folder with content of the zip.
3. Run nxmc.exe file from extracted catalog.

Web Management Client:

1. Download latest version of WAR file from Web Interface Binaries section <http://www.netxms.org/download/> (named nxmc-VERSION.war, for example nxmc-5.1.0.war).
2. Replace old WAR file with the new one. Default path: `INSTALLATION_DIR\webapps`.

Sometimes it's possible that new WAR file is not detected and previous version of WAR continues to run. In this case stop servlet container, delete the WAR file. Then start servlet container and copy the war file to webapps directory.

4.4 Generic upgrade using source tarball

4.4.1 Server

1. Download the latest version from <http://www.netxms.org/download>, if you don't have it. You will need source archive (named netxms-VERSION.tar.gz, for example netxms-5.1.0.tar.gz). Please note that in the following steps VERSION will be used as a substitution for an actual version number.

2. Unpack the archive:

```
tar zxvf netxms-5.1.0.tar.gz
```

3. Change directory to netxms-version and run configure script and make:

```
cd netxms-5.1.0

sh ./configure --enable-release-build --with-server --with-mysql

make
```

Be sure to include all configuration options that were used at installation time.

4. Stop NetXMS server.
5. Stop NetXMS agent.
6. Check database for possible inconsistencies:

```
nxdbmgr check
```

Proceed to the next step only if database checker does not report any errors!

7. Run make install:

```
make install
```

8. Upgrade database:

```
nxdbmgr upgrade
```

9. Start NetXMS agent.
10. Start NetXMS server.

4.4.2 Agent

1. Download the latest version from <http://www.netxms.org/download>, if you don't have it. You will need source archive (named netxms-VERSION.tar.gz, for example netxms-5.1.0.tar.gz). Please note that in the following steps VERSION will be used as a substitution for an actual version number.

2. Unpack the archive:

```
tar zxvf netxms-5.1.0.tar.gz
```

3. Change directory to netxms-version and run configure script and make:

```
cd netxms-5.1.0

sh ./configure --enable-release-build --with-agent

make
```

Be sure to include all configuration options that were used at installation time.

4. Stop NetXMS agent.
5. Run make install:

```
make install
```

6. Run agent:

```
/usr/local/bin/nxagentd -d
```

4.5 Centralized agent upgrade

You can use *Package management* functionality to perform centralized upgrade of NetXMS agents.

QUICK START

In this section will describe basic configuration to be performed after server and agent clean install. Configuration for monitoring some common metrics like CPU usage of file system free space will also be shown.

5.1 Default Credentials

Server login default credentials

Login: admin

Password: netxms

5.2 Basic agent configuration

Minimal configuration that should be set for agent is server address and path to log file. Action differ depending on a platform where agent is installed. On Windows systems configuration file is automatically generated and populated by installer, on UNIX systems it should be created/edited manually.

See below for editing agent configuration file on Windows and UNIX/Linux platforms.

5.2.1 Windows

In case if while installation MasterServer was set correctly no action is required from user.

Automatically generated configuration file can be found there: `installation directory\etc\nxagentd.conf` (by default `C:\NetXMS\etc\nxagentd.conf`.)

Configuration file for Windows should look like this:

```
#  
# Sample agent's configuration file  
#  
MasterServers = 127.0.0.1  
LogFile = {syslog}
```

5.2.2 UNIX/Linux

After agent is installed on a UNIX/Linux system it is required to create/edit file `/etc/nxagentd.conf`. This file should contain at least this information:

```
#  
# Sample agent's configuration file  
#  
MasterServers = 127.0.0.1  
LogFile = /var/log/nxagentd
```

5.3 Basic server tuning

Server has two types of configuration: configuration file parameters and server configuration variables.

For server configuration file minimal requirements are path to log file, database driver name and all required credentials depending on database. Location and required actions depends on what OS is used. More about OS specific configuration search in OS subsections of this chapter.

List of possible database drivers:

- `mssql` Driver for Microsoft SQL database.
- `mysql` Driver for MySQL database.
- `odbc` ODBC connectivity driver (you can connect to MySQL, PostgreSQL, MS SQL, and Oracle via ODBC).
- `oracle` Driver for Oracle database.
- `pgsql` Driver for PostgreSQL database.
- `sqlite` Driver for embedded SQLite database.

See below for editing server configuration file on Windows and UNIX/Linux platforms.

5.3.1 Windows

For Windows systems this information is added to configuration file while installation procedure. It can be check that all data was set correctly in this file: 'installation directory'\etc\netxmsd.conf (by default C:\NetXMS\etc\netxmsd.conf.)

Example of sample Windows configuration for mysql:

```
#  
# Sample server configuration file  
#  
  
DBDriver = mysql.ldr  
DBServer = localhost  
DBName = netxms_db  
DBLogin = netxms  
DBPassword = password  
LogFile = {syslog}
```

5.3.2 UNIX/Linux

For UNIX based systems `/etc/netxmsd.conf` file should be created/populated manually.

Configuration file example for oracle database:

```

DBDriver = oracle.ldr
DBServer = ServerIP/Hostname.DomainName #Here is service (full database name), not SID
DBName = netxms
DBLogin = netxms
DBPassword = PaSwD
LogFile = /var/log/netxmsd

```

5.3.3 Server configuration variables

There are quite a few important server parameters to be set right after installation. These parameters are accessible through the *Server Configuration* window in the management client. To open it, click on *Configuration* ▶ *Server Configuration*. To edit a setting, double click on the row in the table or right-click and select *Edit*. The following parameters may need to be changed:

Parameter	Description
ThreadPool.Poller.MaxSize	This parameter represents maximum thread pool size. This pool provides threads for all types of polls: Status poll, Configuration poll, etc. In case of big load on a server number of threads will be increased up to this size. When load come back to normal, number of threads will be automatically decreased down to base size. If you plan to monitor large number of hosts, increase this parameter from the default value to approximately 1/5 of host count.
ThreadPool.Poller.BaseSize	This parameter represents base thread pool size. This is minimum number of threads that will always run. If you plan to monitor large number of hosts increase this parameter from the default value to approximately 1/10 of host count.
ThreadPool.DataCollector.MaxSize	Maximum number of threads that perform data collection. If you plan to monitor large number of hosts, increase this number to approximately 1/5 of host count. Use larger value if you plan to gather many DCIs from each host.
ThreadPool.DataCollector.BaseSize	Minimum number of data collection threads what will always run. For large number of hosts increase to approximately 1/10 of host count.
Syslog.EnableListener	Set this parameter to <code>True</code> if you want to enable NetXMS built-in syslog server.

5.4 Notification channels

Various ways how to send notifications - email, messengers, SMS, etc are configured via Notification Channels. This allows to create actions that will send notification on defined events.

Notification channels are configured on *Configuration* ▶ *Notification Channels*. Each channel has textual configuration, e.g. for SNMP driver configuration may look like this:

```

Server=smtp.example.com
FromAddr=netxms@example.com
FromName=NetXMS Server
IsHTML=no
TLSmode=TLS
Login=smtp-username
Password=password

```

Information about notification channel configuration parameters is available here: [Notification channels](#).

5.5 Actions and Alarms

In this section we will configure alarm automatic creation and termination and message sending via a notification channel on predefined SYS_THRESHOLD_REACHED and SYS_THRESHOLD_REARMED events.

Given that a notification channel is configured, we can create an action in *Configuration* ▶ *Actions*. Recipient address is specified in action's properties, it's possible to set several recipients separated by semicolon (;). Subject and message fields support *Macros for Event Processing* - in below example when message will be sent, macros "%n" will be substituted with name of the node and "%m" will be substituted with event message. Value of event message is specific for each event and can be found in event template (*Configuration* ▶ *Event Templates*).

Create action

Name
Send email

Type
☐ Execute command on management server
☐ Execute command on remote node via agent
☐ Execute command on remote node via SSH
☐ Execute NXSL script
☒ Send notification
☐ Forward event to other NetXMS server

Options
☐ Action is disabled

Channel name
SMTP-Text

Recipient's address
admin@example.com

Subject
Something happened on node %n

Message text
%m for node %n

Cancel OK

Next step is to configure event processing policies. It is done in *Configuration* ▶ *Event Processing Policy*. A number of rules is included out-of-the-box, including rules that react to SYS_THRESHOLD_REACHED and SYS_THRESHOLD_REARMED events. In these rules we will add email sending action that we have configured above.

Alarm created by the rule for SYS_THRESHOLD_REACHED has a key which is composed from "SYS_THRESHOLD_REACHED_" text, id of DCI and ID of node. This allows to resolve or terminate alarms automatically - for example rule for SYS_THRESHOLD_REARMED automatically terminates alarm using the key.

After all configuration is done *Event Processing Policy* should be saved.

Event Processing Policy					
20	Terminate NetXMS server network connection loss alarm when connection restored				
21	Show alarm when DCI status changes to DISABLED or UNSUPPORTED				
22	Terminate DCI status alarms when DCI status returns to ACTIVE				
Generate alarm on threshold violation					
23	<table border="1"> <thead> <tr> <th>Filter</th><th>Action</th></tr> </thead> <tbody> <tr> <td>IF event code is one of the following: ▲ SYS_THRESHOLD_REACHED</td><td> Generate alarm %m with key "DC_THRESHOLD_%i_%<dcid>" Execute the following predefined actions: Send email </td></tr> </tbody> </table>	Filter	Action	IF event code is one of the following: ▲ SYS_THRESHOLD_REACHED	Generate alarm %m with key "DC_THRESHOLD_%i_%<dcid>" Execute the following predefined actions: Send email
Filter	Action				
IF event code is one of the following: ▲ SYS_THRESHOLD_REACHED	Generate alarm %m with key "DC_THRESHOLD_%i_%<dcid>" Execute the following predefined actions: Send email				
Terminate threshold violation alarms					
24	<table border="1"> <thead> <tr> <th>Filter</th><th>Action</th></tr> </thead> <tbody> <tr> <td>IF event code is one of the following: ● SYS_THRESHOLD_REARMED</td><td> Terminate alarms with key "DC_THRESHOLD_%i_%<dcid>" Execute the following predefined actions: Send email </td></tr> </tbody> </table>	Filter	Action	IF event code is one of the following: ● SYS_THRESHOLD_REARMED	Terminate alarms with key "DC_THRESHOLD_%i_%<dcid>" Execute the following predefined actions: Send email
Filter	Action				
IF event code is one of the following: ● SYS_THRESHOLD_REARMED	Terminate alarms with key "DC_THRESHOLD_%i_%<dcid>" Execute the following predefined actions: Send email				
25	Generate alarm on table threshold violation				
26	Terminate table threshold violation alarms				
27	Generate an alarm when one of the system threads hangs or stops unexpectedly				
28	Terminate the alarm when one of the system threads which previously hanged or stoped unexpectedly returned to the running state				
29	Terminate alarms for hanged or unexpectedly stopped system threads that could have been created prior to server restart				
30	Generate an alarm when the object enters the maintainance mode				
31	Terminate the alarm when the object leaves the maintainance mode				
32	Generate an alarm if the NetXMS agent on the node stops responding				
33	Terminate the alarm if the NetXMS agent on the node start responding again				
34	Generate an alarm if the SNMP agent on the node stops responding				
35	Terminate the alarm if the SNMP agent on the node start responding again				
36	Generate an alarm when error occurred during LDAP synchronization				
37	Generate an alarm when there is problem with agent log				

5.6 SNMP Defaults

If you have a number of *SNMP* devices with same credentials on your network, you can configure default community strings and authorization credentials. This information is set in *Configuration -> Network Credentials*.

When performing configuration poll, provided community strings, USM credentials and network ports will be tried sequentially until a combination that allows communication with a device is found.

5.7 Passive discovery

It is recommended to enable passive discovery when it is required to add all nodes in local network. In case if NetXMS server has access to switches and routers via SNMP, all devices in network will be added automatically by discovery process.

To enable passive network discovery open *Configuration -> Network Discovery*. There in *General* section select *Passive only* option. Network discovery will be using default SNMP credentials that were discussed above in *SNMP Defaults* section. Other options that can be set depending on requirements:

- Option to use SNMP trap source for further network discovery
- Option to set filter that will define rules for not adding nodes to NetXMS server

In our configuration we will not use filter to add all node available on our network and turn on option to use SNMP trap source address for discovery. After all configuration is done remember to save it.

5.7.1 Notes

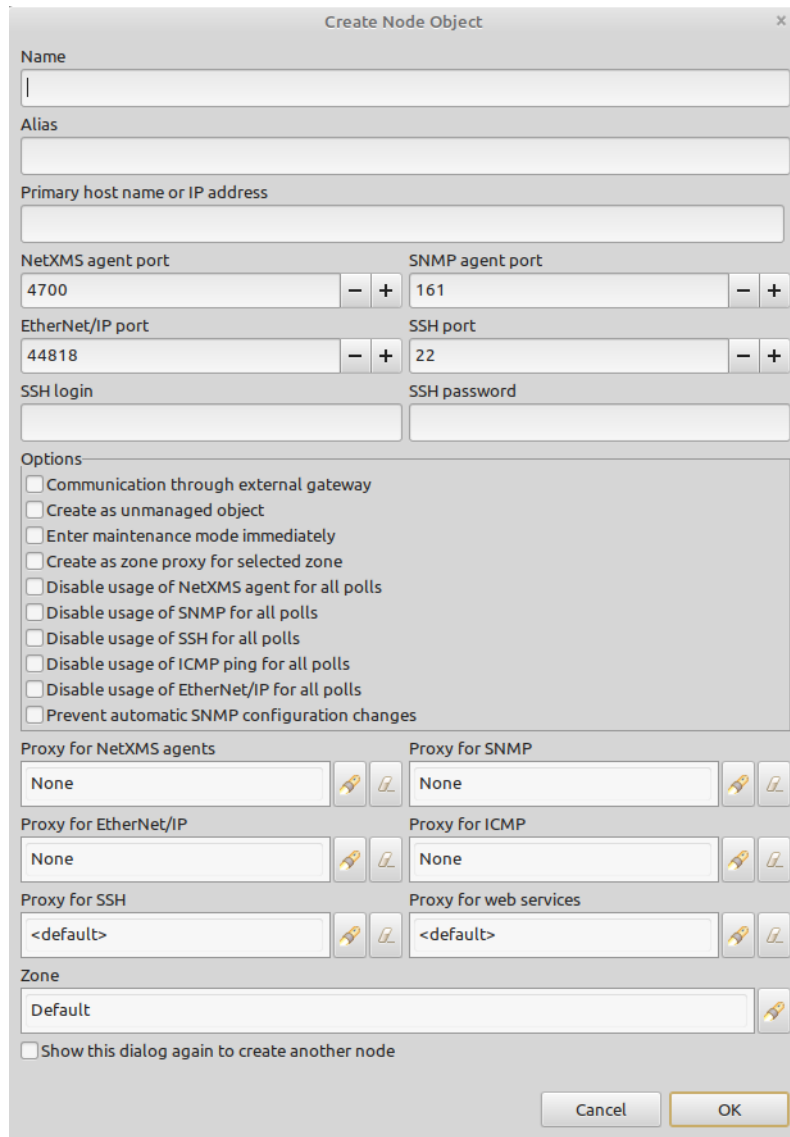
If you have enabled automatic network discovery, wait for initial network discovery completion. This process can take time, depending on size and complexity of your network. For large networks, we recommend that you let NetXMS run over night to gather the majority of network information available. Once devices are discovered, they appear under appropriate subnets in the *Network* perspective.

Please note that for successful network discovery your network must meet the following requirements:

- NetXMS server must have access to switches and routers via SNMP.
- All your network devices credentials (community string and credentials for SNMP v3) should be added to default credential list in *Network Credentials*.

5.8 Manually add node

If the automatic network discovery does not detect all of your hosts or devices, or you decide not to use network discovery at all, you may need to manually add monitored nodes to the system. The easiest way to accomplish this is to right-click on *Infrastructure Services* in the *Infrastructure* perspective and select *Create node*. You will be presented with the following dialog window:



The 'Create Node Object' dialog box contains the following fields and options:

- Name:** A text input field.
- Alias:** A text input field.
- Primary host name or IP address:** A text input field.
- NetXMS agent port:** A numeric input field with a value of 4700 and minus/plus buttons.
- SNMP agent port:** A numeric input field with a value of 161 and minus/plus buttons.
- EtherNet/IP port:** A numeric input field with a value of 44818 and minus/plus buttons.
- SSH port:** A numeric input field with a value of 22 and minus/plus buttons.
- SSH login:** A text input field.
- SSH password:** A text input field.
- Options:** A group box containing several checkboxes:
 - ☐ Communication through external gateway
 - ☐ Create as unmanaged object
 - ☐ Enter maintenance mode immediately
 - ☐ Create as zone proxy for selected zone
 - ☐ Disable usage of NetXMS agent for all polls
 - ☐ Disable usage of SNMP for all polls
 - ☐ Disable usage of SSH for all polls
 - ☐ Disable usage of ICMP ping for all polls
 - ☐ Disable usage of EtherNet/IP for all polls
 - ☐ Prevent automatic SNMP configuration changes
- Proxy for NetXMS agents:** A dropdown menu set to 'None' with a lock icon.
- Proxy for SNMP:** A dropdown menu set to 'None' with a lock icon.
- Proxy for EtherNet/IP:** A dropdown menu set to 'None' with a lock icon.
- Proxy for ICMP:** A dropdown menu set to 'None' with a lock icon.
- Proxy for SSH:** A dropdown menu set to '<default>' with a lock icon.
- Proxy for web services:** A dropdown menu set to '<default>' with a lock icon.
- Zone:** A dropdown menu set to 'Default' with a lock icon.
- ☐ Show this dialog again to create another node
- Buttons:** 'Cancel' and 'OK' buttons at the bottom right.

Fig. 1: Create Node window

Please note that adding a new node object may take some time, especially if a node is down or behind a firewall. After successful creation, a new node object will be placed into appropriate subnets automatically. As soon as you add a new node to the system, NetXMS server will start regular polling to determine the node status.

5.9 Data Collection items

In this section we will add data collection items (DCIs) for CPU usage monitoring and interface incoming traffic via NetXMS agent or SNMP. Threshold configuration for these DCIs will be shown. This threshold will generate `SYS_THRESHOLD_REACHED` event when defined condition is met and `SYS_THRESHOLD_REARMED` when collected data value returns to normal.

Earlier we already described how to configure notification sending and alarm generation and termination based on events. This chapter describes data collection and event generation based on collected data.

To add DCI for a node select the node, open *Data Collection* tab and click + button on the toolbar.

5.9.1 CPU usage

Add CPU usage metric from agent metrics:

1. Check that as origin is selected NetXMS Agent.
2. Click on *Select* button - list of available agent metrics will open. Note: this list is populated on configuration poll.
3. Type in the input box “CPU”

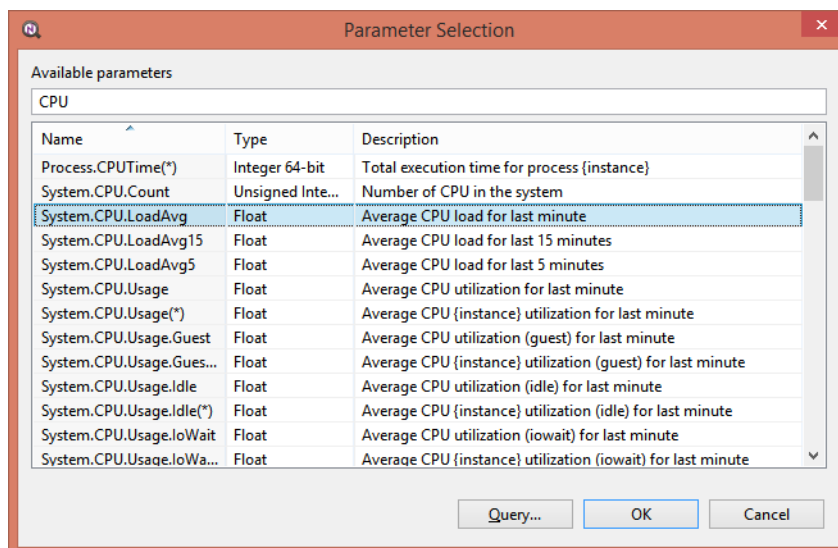


Fig. 2: Metric Selection

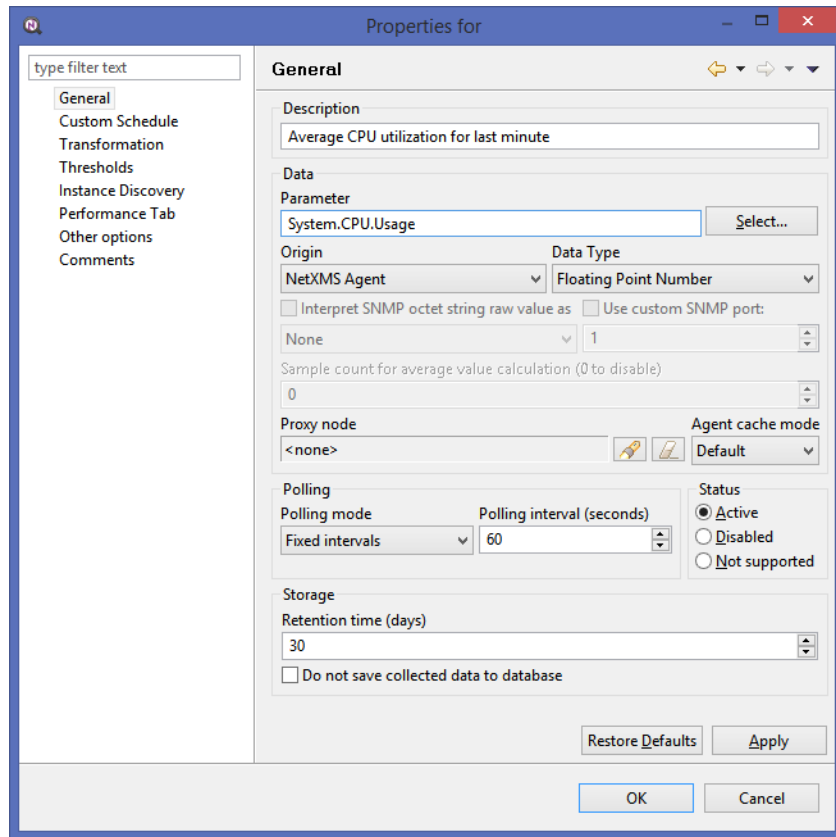


Fig. 3: Properties

4. Select *System.CPU.Usage*
5. Go to *Threshold* tab
6. Click *Add*
7. Set that if last one polled value is gather than 85, then generate SYS_THRESHOLD_REACHED event, when value is back to normal generate SYS_THRESHOLD_REARMED event.

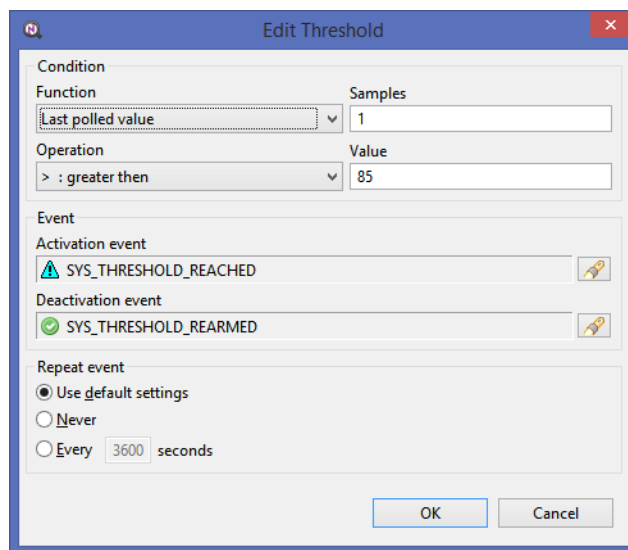


Fig. 4: Threshold

8. Click *OK*

Add CPU usage metric from SNMP metrics:

1. Check that as origin is selected NetXMS Agent.
2. Click on *Select* button
3. Type in the input box “.1.3.6.1.4.1.9.9.109.1.1.1.4” (this OID can may be not available for some devices)
4. Click *Walk*

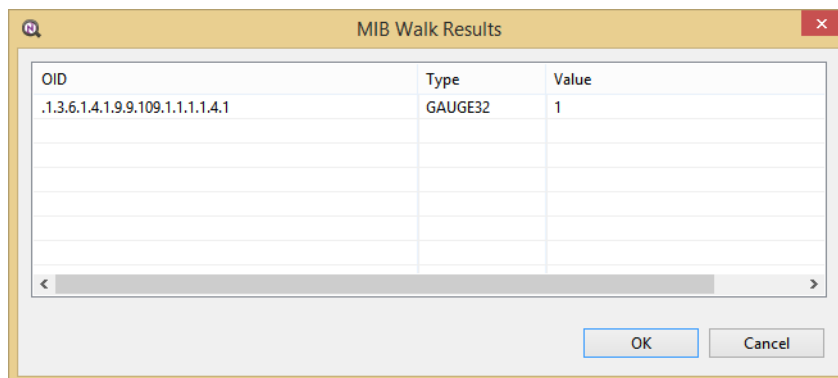


Fig. 5: Mib Walk Result

5. Select CPU that should be monitored in our case it is “.1.3.6.1.4.1.9.9.109.1.1.1.4.1”

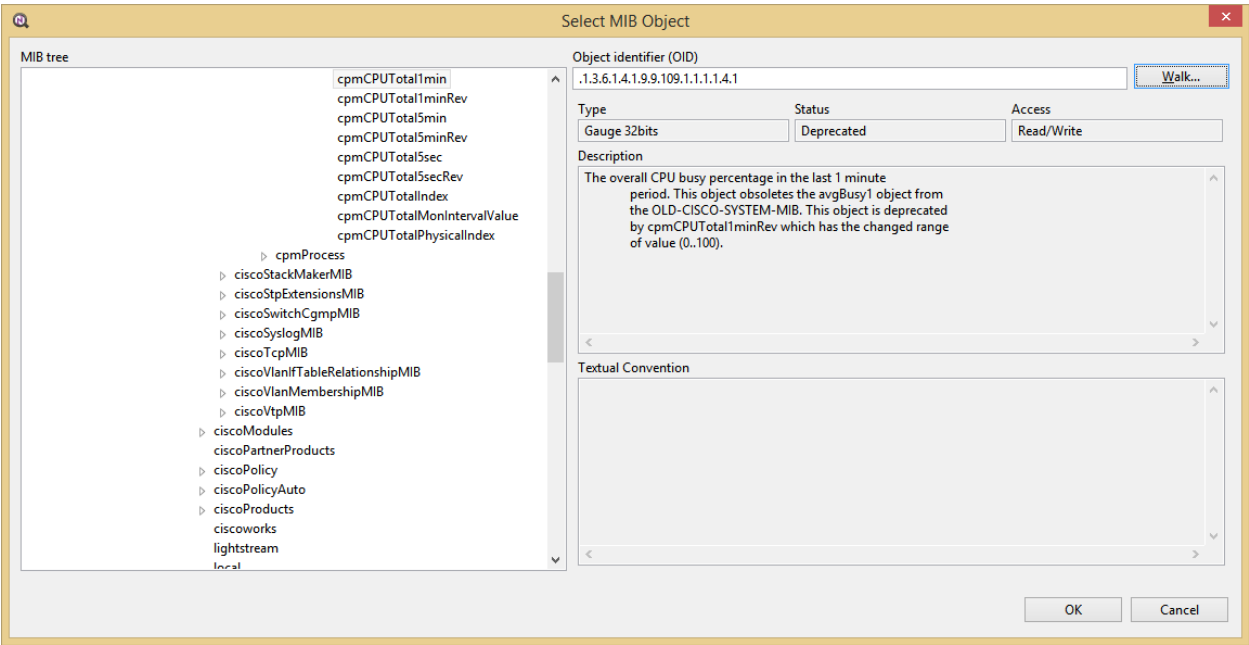


Fig. 6: Select Window For SNMP DCI

6. Click *OK*

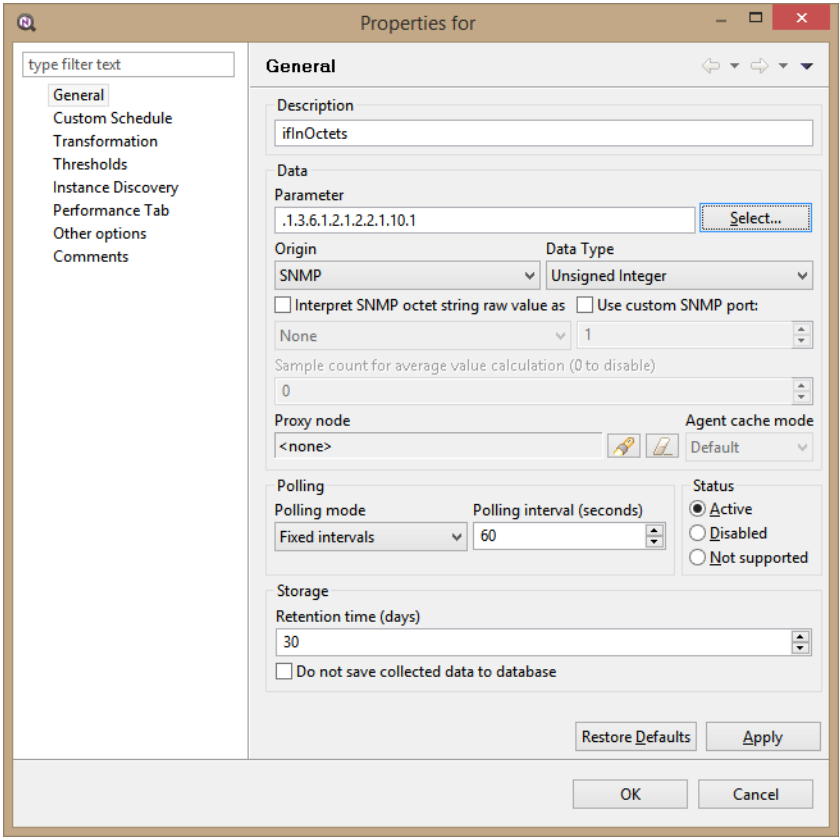


Fig. 7: Properties

7. Go to *Threshold* tab
8. Click *Add*
9. Set that if last one polled value is greater than 85, then generate SYS_THRESHOLD_REACHED event, when value is back to normal generate SYS_THRESHOLD_REARMED event.

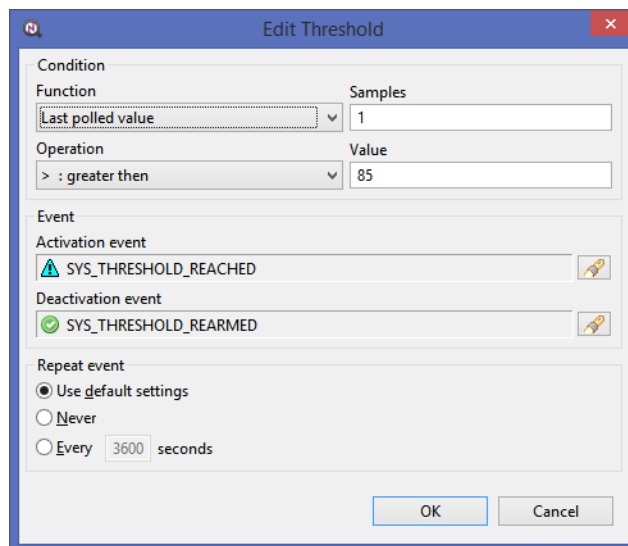


Fig. 8: Threshold

10. Click *OK*

Now you configured data collection of metric for CPU usage that will be collected every 60 seconds, data will be stored for 30 days, with 1 threshold that will be activated when CPU usage is more than 85%.

5.9.2 Interface traffic

There is shortcut to create all required DCIs for interface traffic for nodes where you have either NetXMS agent or SNMP. Select interfaces for which should be created traffic collection DCIs and select *Create data collection items* from context menu. Select checkboxes for the metrics that you need - DCIs will be created automatically.

Create Interface DCI

Data

☒ Inbound traffic (bytes) ☒ Delta value (average per second)

Description: Inbound traffic on @@ifName@@ (bytes/sec)

☒ Outbound traffic (bytes) ☒ Delta value (average per second)

Description: Outbound traffic on @@ifName@@ (bytes/sec)

☐ Inbound traffic (bits) ☒ Delta value (average per second)

Description: Inbound traffic on @@ifName@@ (bits/sec)

☐ Outbound traffic (bits) ☒ Delta value (average per second)

Description: Outbound traffic on @@ifName@@ (bits/sec)

☐ Inbound traffic (packets) ☒ Delta value (average per second)

Description: Inbound traffic on @@ifName@@ (packets/sec)

☐ Outbound traffic (packets) ☒ Delta value (average per second)

Description: Outbound traffic on @@ifName@@ (packets/sec)

☐ Input errors ☒ Delta value (average per second)

Description: Inbound error rate on @@ifName@@ (errors/sec)

☐ Output errors ☒ Delta value (average per second)

Description: Outbound error rate on @@ifName@@ (errors/sec)

Options

Polling pollingInterval (seconds) Retention time (days)

6030

OK

Cancel

AGENT MANAGEMENT

6.1 Introduction

NetXMS agent is daemon or service that runs on a *node* to provide additional monitoring options. This is optional for installation, but it's installation gives following advantages:

- Centralized configuration - you can change configuration of agent from management client; if needed, you can even store agent configs on NetXMS server
- More secure: communications between NetXMS server and agent is encrypted by default, additional authentication on agent can be configured
- TCP instead of UDP is used for communications with agent - this can help in case of slow and poor quality links
- Remote command execution - agents can be used to execute commands on managed systems as a reaction to certain events
- Proxy functionality: agent can be used as a proxy to reach agents on hosts not directly accessible by NetXMS server
- *SNMP* proxy: agent can be used as a proxy to reach remote SNMP devices
- *SNMP Trap* proxy: agent can be used as a proxy to get messages from remote SNMP device
- *Syslog* proxy: agent can be used as a proxy to get syslog messages from remote devices
- Modbus TCP proxy: agent can be used as a proxy to reach remote devices via Modbus TCP protocol
- Web service proxy: agent can be used as a proxy to reach remote web services
- TCP proxy: agent can be used to establish connection to TCP port on remote devices, e.g. to access web UI on a device
- Extensible: you can add new metrics very easy using configuration option like `ExternalMetric` or by writing your own subagents
- Easy upgrade - you can upgrade all agents at once from management client
- Provides file management possibilities on agent.
- Provides log file monitoring functionality.

6.2 Agent configuration files

Agent have 3 types of configuration files: master configuration file, additional configuration files and Agent Policy configuration files. Master configuration file is the only mandatory file. Minimal configuration for master configuration file is server address. Address should be set as `MasterServers` to be able to apply other configuration settings from the server.

After configuration file change agent should be restarted to apply new changes.

Two formats are supported for configuration files and configuration file policies: XML and 'key = value' format.

In 'key = value' format configuration file can contain one or more parameters in *Parameter = Value* form, each parameter should be on its own line. Parameters are grouped into sections. Beginning of a section is denoted by section name in square brackets (example: "[sectionName]"). Section named "[Core]" contains parameters for agent itself. It's the default section, if a configuration file starts from parameter and not from section name, parameters are treated as belonging to "Core" section. Subagents' parameters should be placed in separate sections named by subagent name. Same section name can be present several times in the configuration file. Comments can be inserted after "#" sign

In XML format general tag should be <config>, second level tags contain section names and third level tags are agent or subagent configuration parameters.

'key = value' format example:

```
[Core]
MasterServers = 10.0.0.4
SubAgent = winperf.nsm
# Below is a configuration for winperf subagent, in separate section
[WinPerf]
EnableDefaultCounters = yes
```

Same example in XML format:

```
<config>
  <Core>
    <MasterServers>10.0.0.4</MasterServers>
    <SubAgent>winperf.nsm</SubAgent>
  </Core>
  <!-- Below is a configuration for winperf subagent, in separate section -->
  <WinPerf>
    <EnableDefaultCounters>yes</EnableDefaultCounters>
  </WinPerf>
</config>
```

Example of configuration sections:

```
[core]
SubAgent=ping.nsm
SubAgent=filemgr.nsm
SubAgent = logwatch.nsm
```

```
LogFile=/opt/netxms/log/nxagentd
MasterServers=netxms.demo
ServerConnection=netxms.demo
DebugTags=logwatch:9,logwatch.*:9
EnableProxy=yes
```

```
ActionShellExec=echo:echo "$1"
```

```
[filemgr]
RootFolder=/home/zev/Downloads/tmp
```

```
[logwatch]
Parser = /opt/netxms/etc/parser.xml
```

```
[filemgr]
RootFolder=/
```

```
[ping]
Target=8.8.8.8
Target=127.0.0.1
```

```
ActionShellExec=netstat:netstat
SubAgent=portcheck.nsm
```

Core section

File manager section 1

Log watch section

File manager section 2

Ping section

Will not work, as it should be defined in core section, but is given in ping section. These lines should either be moved up to core section or another core section can be specified above these lines.

Detailed list of parameters can be found here: [Agent configuration file \(nxagentd.conf\)](#). The following parameters can be specified in master configuration file only (and will be ignored if found in other configuration files): `DataDirectory` and `ConfigIncludeDir`.

6.2.1 Master configuration file

File `nxagentd.conf` is a master configuration file for NetXMS agent. Depending on OS there are different locations, where agent tries to find master configuration file.

UNIX-like systems

On UNIX systems master configuration file is searched in the following order:

1. If `$NETXMS_HOME` environment variable is set: `$NETXMS_HOME/etc/nxagentd.conf`
2. `'prefix'/etc/nxagentd.conf`. 'prefix' is set during build configuration with `--prefix='prefix'` parameter. If that parameter was not specified during build, `/usr/local` is used.
3. `/Database/etc/nxagentd.conf`
4. `/usr/etc/nxagentd.conf`
5. `/etc/nxagentd.conf`

If configuration file is placed in a different location or named in a different way, then it's location and file name can be given to agent with `-c` parameter or by specifying `$NXAGENTD_CONFIG` environment variable. In this cause search in the locations mentioned above is not performed.

Windows

On Windows location of NetXMS config is stored in the registry. Alternatively, location of configuration file can be provided to agent with `-c` command line parameter. If there is no record in the registry and `-c` parameter is not specified, then agent tries to find configuration files in the following locations:

1. `'installation directory'\etc\nxagentd.conf`

2. `C:\nxagentd.conf`

6.2.2 Additional configuration files

To increase maintainability, configuration can be stored in multiple additional configuration files located in a specific folder. Additional configuration files override (if a parameter supports only one value) or supplement (if parameter supports multiple values, e.g. list of servers or root folders for filemgr subagent) configuration parameters from master file. Depending on OS there are different locations, where agent tries to find master configuration file.

UNIX-like systems

On UNIX systems it is searched in the following order (search is performed until first existing folder is found):

1. If `$NETXMS_HOME` environment variable is set: `$NETXMS_HOME/etc/nxagentd.conf.d`
2. `'prefix'/etc/nxagentd.conf.d`. 'prefix' is set during build configuration with `--prefix='prefix'` parameter. If that parameter was not specified during build, `/usr/local` is used.
3. `/Database/etc/nxagentd.conf.d`
4. `/etc/nxagentd.conf.d`
5. `/usr/etc/nxagentd.conf.d`

A different configuration file folder name can be given by specifying `$NXAGENTD_CONFIG_D` environment variable. In this cause search in the locations mentioned above is not performed.

Windows

On Windows location of configuration file folder is stored in the registry. If there is no record in the registry, then agent tries to find configuration file folder in the following locations (search is performed until first existing folder is found):

1. `'installation directory'\etc\nxagentd.conf.d`
2. `C:\nxagentd.conf.d`

6.2.3 Agent policy configuration files

Agent policies allow to store agent configuration on server and deliver it to the agents. More information about Policies can be read there: [Agent Policies](#).

On agent configuration policy files are stored in a separate folder named `config_ap` under `DataDirectory` folder. Every policy is saved into a separate file named by policy GUID.

6.3 Agent configuration options from server

6.3.1 Edit configuration file remotely

Right click on node, select *Edit agent's configuration file* from menu. When closing the editor, a dialog will be presented. New configuration apply is performed on agent restart. So to immediately apply new configuration select *Save and Apply*. This option will save configuration file and automatically restart the agent. If just *Save* is selected, then agent should be manually restarted to apply new configuration.

6.3.2 Agent configuration files on server

Agent master configuration files can be stored on server side and requested by agent, if it is launched with `-M <serverAddress>` command line parameter. Each configuration file on server is stored along with filter script. When server receives configuration request from agent, it goes through available configs and executes filter scripts to find an appropriate configuration.

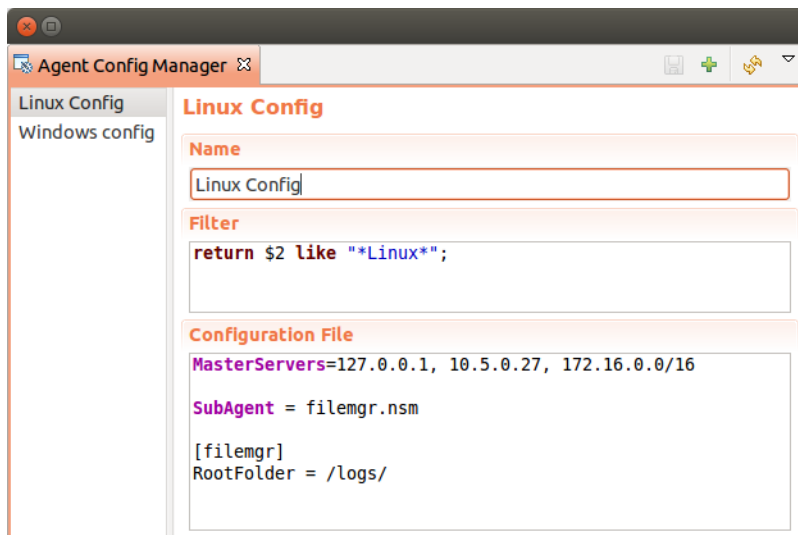
If appropriate configuration file is found, it is sent to agent and old `nxagentd.conf` file is overwritten (or a new `nxagentd.conf` file is created, if it did not exist). When agent can't connect to server or server hasn't found right configuration, the agent is started with old configuration file. In case if agent configuration file does not exist and it is not possible to get new one from the server - agent fails to start.

Doesn't work with tunnel agent connection

Configuration

Each configuration has a name, filter script and the configuration file text.

- Name just identifies the configuration.
- Filter script is executed on configuration request to define which configuration file to send to the agent. Filter is defined with help of *NXSL* scripting language. The following parameters are available in the filter script:
 - \$1 - IP address
 - \$2 - platform
 - \$3 - major version number
 - \$4 - minor version number
 - \$5 - release number
- Configuration file is the text of returned configuration file.



6.3.3 Agent configuration policy

Another option to store and distribute agent configuration are agent policies. In this case agent configuration is stored on the server side as a policy belonging to template and deployed to the agent when corresponding template is applied to a node. More information about policies and their types can be found in *Agent Policies* chapter.

6.3.4 Agent Configuration Policies vs. Agent Configuration Files on Server

A short lists of main points to compare both options:

Agent Configuration Files on Server:

- Assignment is based on rules described in filter scripts

- When configuration is changed, agent restart is needed to activate new configuration
- Config download from server is each time the agent starts (if option '-M servername')
- When config is found on server, local Master config is overwritten, if not - existing Master config is used
- Works with master configuration file
- Does not required initial config (agent can be started without config), but in this case agent would fail if nothing was returned from server
- Server provides configuration file without authorization which can be a security issue, if sensitive information is present in configuration file.
- Doesn't work via proxy
- Doesn't work via tunnel agent connection

Agent Policies:

- Not possible for bootstrap agent
- After policy is deployed to agent, the agent should be restarted to activate new configuration.
- At minimum the server connection parameters must be in master config to be able to start agent
- Each policy is saved in a separate configuration file
- If policy and master config have same parameter that can be set only once (e.g. LogFile), then policy will overwrite master config configuration
- If policy and master config have same parameter that can be set multiple times (e.g. Target for PING subagent or Query for DBQUERY), then policy will merge lists of configs
- Can work via proxy
- Can work with tunnel agent connection

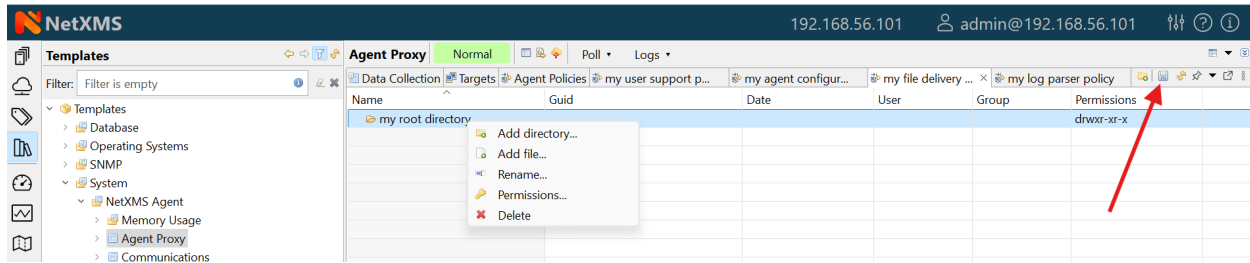
6.4 Agent Policies

Agent policies are additional configuration created by user (agent configuration or files) that are uploaded and updated on agent when template is manually or automatically applied on the node. Agent policies belong to templates, so they are applied to nodes to which a corresponding template is applied.

The following policy types are available:

- Agent configuration policy
- File delivery policy
- Log parser policy
- User support application policy

To create policy, select a template and click *Agent policies* tab. Click plus icon to create a new policy, give it a name, choose correct policy type and click *OK*. Newly created policy will open for editing in a new tab. For example, for File Delivery policy, right click and *Add root directory...* option will prompt you to create directory. Then, right click on newly created directory and more options, like *Add directory...*, *Add file...*, *Rename...*, *Permissions...* and *Delete...* will be available. Existing policy can be modified by right clicking it and selecting *Edit* from the menu or by double clicking on it. Use *Save* button after configuration changes.



Policies are automatically deployed to nodes after creation/modification or when a template is applied to a node. When configuration policy is deleted or template is removed from a node, the policy is automatically undeployed from node.

Policies get deployed / undeployed:

- On node configuration poll.
- When list of Agent Policies is closed in the management client. If a node is down at that moment, next attempt will happen on configuration poll.
- When template is applied or removed from a node. If a node is down at that moment, next attempt will happen on configuration poll.

Installed policy configurations are stored as additional files under agent *DataDirectory*. List of applied policies is stored in agent local database.

If agent discovers for a record in local database, that policy file is missing, it will delete the record from database.

When performing deployment, server checks information in agent's database with it's database and issues necessary commands.

6.4.1 Agent configuration policy

Agent configuration policy provides option to populate agent configuration with additional parts. Main agent configuration is merged with additional rules from policy. Using policy for configuration file maintenance has advantages that configuration is edited in centralized way and gives granular control on the configuration that each node gets. More information about different agent configuration options can be found in above chapters.

It is possible to use the same parameters and format as in any NetXMS agent configuration file (key=value format or XML format).

Example:

```
MasterServer=127.0.0.1
SubAgent=netsvc.nsm
SubAgent=dbquery.nsm
SubAgent=filemgr.nsm

[DBQUERY]
Database=id=myDB;driver=mysql.ldr;server=127.0.0.1;login=netxms;password=xxxxx;
↪ dbname=netxms
Query=dbquery1:myDB:60:SELECT name FROM images
ConfigurableQuery=dbquery2:myDB:Comment in param :SELECT name FROM images WHERE name_
↪ like ?
ConfigurableQuery=byID:myDB:Comment in param :SELECT name FROM users WHERE id=?

[filemgr]
RootFolder=/
```

```

<config>
  <core>
    <!-- there can be added comment -->
    <MasterServers>127.0.0.1</MasterServers>
    <SubAgent>netsvc.nsm</SubAgent>
    <SubAgent>dbquery.nsm</SubAgent>
    <SubAgent>filemgr.nsm</SubAgent>
  </core>
  <DBQUERY>
    <Database>id=myDB;driver=mysql.ldr;server=127.0.0.1;login=netxms;password=xxxxx;
    ↪ dbname=netxms</Database>
    <Query>dbquery1:myDB:60:SELECT name FROM images</Query>
    <ConfigurableQuery>dbquery2:myDB:Comment in param :SELECT name FROM images WHERE ↪
    ↪ name like ?</ConfigurableQuery>
    <ConfigurableQuery>byID:myDB:Comment in param :SELECT name FROM users WHERE id=?</
    ↪ ConfigurableQuery>
  </DBQUERY>
  <filemgr>
    <RootFolder>/</RootFolder>
  </filemgr>
</config>

```

Example:



Agent should be manually restarted to apply the configuration after the configuration policy is deployed or undeployed to node.

6.4.2 Log parser policy

Information about log parser format and usage available in [Log monitoring](#) chapter.

Log parser configuration is applied right after log parser policy is deployed or undeployed to node - no agent restart is required.

6.4.3 File delivery policy

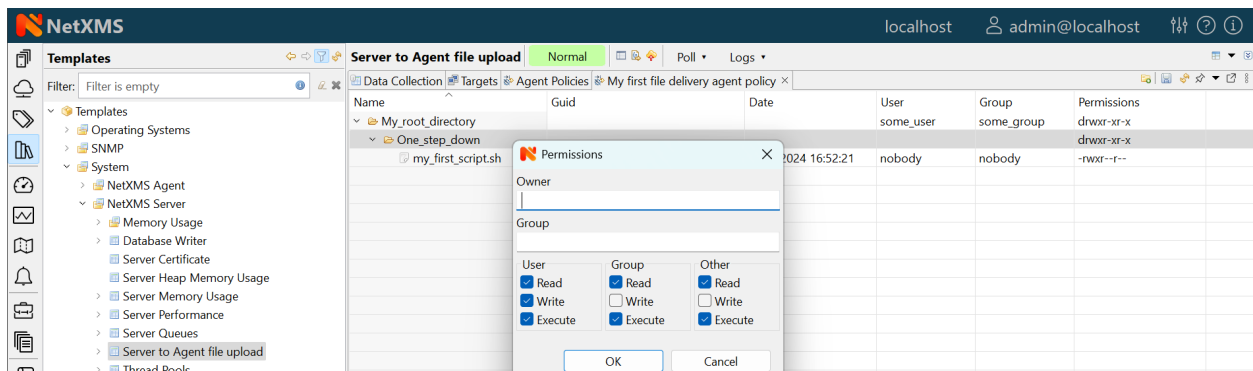
File delivery policy is created to automatically upload files from server to agents.

Firstly, *root folder* or folders should be created - folders with the full path to location where uploadable file(s) and folder structure should be placed. After folder structure is created, files can be added to this structure. On policy apply folders will be created, if possible, and files will be uploaded.

In file and folder names the following macros can be used:

- Environment variables as `%{ENV_VAR_NAME}`
- `strftime(3C)` macros
- Text inside ``` braces will be executed as a command and first line of output will be taken

Example:



Note

File delivery policy uses *File manager* to upload files so *filemgr* subagent should be loaded and root folders should be defined to provide write access to folders.

For Windows there is the following access rights conversion: Read is translated to FILE_GENERIC_READ, write to FILE_GENERIC_WRITE and execute to FILE_GENERIC_EXECUTE. *Other* are translated as Windows group *Everyone* access rights.

6.4.4 User support application policy

6.5 Agent registration

Two ways of agent-server communication are available. Standard one is when server initializes connection to agent, the second one is when tunnel is used and agent initialize connection to server.

6.5.1 Server to agent connection

There are few ways to register agent:

1. To enter it manually by creating a node
2. Run the network discovery and enter the range of IP addresses.

3. Register agent on management server `nxagentd -r <addr>`, where `<addr>` is the IP address of server. To register agents using this option `EnableAgentRegistration` server configuration parameter should be set to 1.

6.5.2 Agent to server connection

This connection requires certificate configuration on server side. More about required actions can be found in [Server configuration for Agent to Server connection / Tunnel connection](#). Server address to which the agent should connect is specified in agent configuration file. There are two options:

ServerConnection parameter

`ServerConnection` parameter set in `agentd.conf` file to server [DNS](#) or server IP address. It's also possible to specify port number separated by colon, e.g.:

```
ServerConnection=monitoring.example.com
ServerConnection=192.168.77.77:1234
```

ServerConnection section

[`ServerConnection`] section is set in `agentd.conf`. This allows to specify additional parameters, e.g.:

```
[ServerConnection]
Hostname=192.168.77.77
Port=4703
CertificateFile=/etc/cert/agent_certificate.crt
ServerCertificateFingerprint=E6:5A:5D:37:22:.....FC:EF:EA:4B:22
```

The following parameters are supported in `ServerConnection` section:

Parameter	Description
Hostname	Server DNS or server IP address
Port	Port number
CertificateId	Id of Certificate in Certificate Store (Windows only). E.g.: <code>template:1.5.3.76.23.45.6.23.4235.56234.234</code>
CertificateFile	Agent certificate file.
Password	Certificate password
ServerCertificateFingerprint	Fingerprint to verify server certificate. Setting this parameter forces verification of server certificate.

Using `CertificateId` or `CertificateFile` allows to provide agent certificate manually, not by auto-generation by NetXMS server.

It is possible to have several `ServerConnection` parameters or sections in the config, in this case agent will establish tunnel connection to multiple servers.

In addition to `ServerConnection` it's necessary to set `MasterServers`, `ControlServers` or `Servers` parameter to configure what access rights server has to this agent.

Agent can validate certificate chain, when connecting to server. This is configured in agent configuration file, e.g.:

```
TrustedRootCertificate=/etc/cert/root_cert.crt
TrustedRootCertificate=/etc/cert/root_certs
VerifyServerCertificate=yes
```

`TrustedRootCertificate` can point to either certificate file or a folder with certificates. Several `TrustedRootCertificate` parameters can be specified. For Windows system agent loads certificates from Certificate Store. For non-Windows systems a number of default certificate locations are automatically loaded by agent:

Path	OS where this path is used
<code>/etc/ssl/certs</code>	Ubuntu, Debian, and many other Linux distros
<code>/usr/local/share/certs</code>	FreeBSD
<code>/etc/pki/tls/certs</code>	Fedora/RHEL
<code>/etc/openssl/certs</code>	NetBSD
<code>/var/ssl/certs</code>	AIX

If `ServerCertificateFingerprint` is specified for a server, server certificate is always verified, disregarding the `VerifyServerCertificate` value.

Agent registration on server

Right after agent start it will try to connect to the server. On first connect node will be shown in *Agent Tunnels*.

There are few ways to register agent:

1. To enter it manually by creating a node and then binding tunnel to already created node.
2. Create node from *Agent Tunnels* view by selecting one or more tunnels and selecting *Create node and bind...* menu item.

Debugging

In case of errors enable server debug for “agent.tunnel” and “crypto.cert” to level 4 and agent log debug for “tunnel” and “crypto.cert” to level 4. Check for “SYS_TUNNEL_SETUP_ERROR” events on management node.

6.6 Security

6.6.1 Message encryption in server to agent communication

Server encryption policy is configured in *Server Configuration* view by selecting one of 4 options for *DefaultEncryptionPolicy* parameter. Default Policy is 2.

Policy types:

- 0 - Forbid encryption. Will communicate with agents only using unencrypted messages. If agent force encryption (*RequireEncryption* agent configuration parameter is set to *yes*), server will not accept connection with this agent.
- 1 - Allow encryption. Will communicate with agents using unencrypted messages if encryption is not enforced by setting *RequireEncryption* agent configuration parameter to *yes* or by selecting *Force encryption* option in Communication properties of node object.
- 2 - Encryption preferred. Will communicate with agents using encryption. In case if agent does not support encryption will use unencrypted communication.
- 3 - Encryption required. Will communicate with agent using encryption. In case if agent does not support encryption will not establish connection.

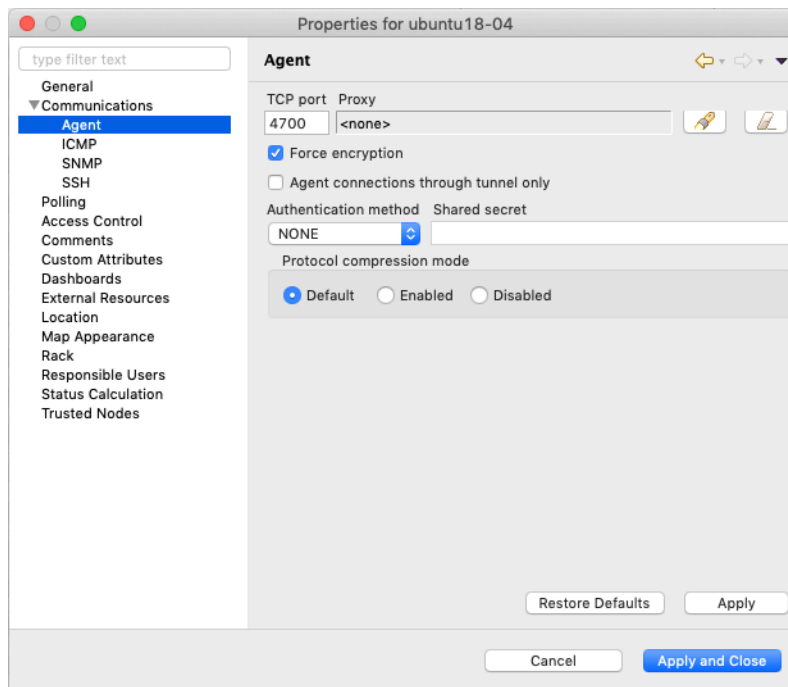


Fig. 1: Force encryption option for node.

6.6.2 Security in agent to server connection

Agent to server connection uses *TLS* protocol to ensure communication security. Server has root certificate, that is used to issue public certificate for agent. Server issues certificate to node when user manually binds tunnel to a node in *Agent Tunnels*, or node is bind automatically (when *AgentTunnels.UnboundTunnelTimeoutAction* server configuration parameter is set to *Bind tunnel to existing node* or *Bind tunnel to existing node or create a new node*). If required, this process can also be automated by NXShell. More information: [NXShell examples](#), [Latest Javadoc](#).

6.6.3 Server access levels

Depending on how server's IP address (or domain name) is added to in *nxagentd.conf*, it will have different access level. It is preferred to use *MasterServers*. There are 3 levels of access for an agent:

1. *MasterServers* - full access.
2. *ControlServers* - can read data and execute predefined actions and make screenshots
3. *Servers* - read only access. (Is default for tunneled agent connection if other server level is not defined)

In case if server IP is not listed in one of this parameters agent will not enable connection with server in server to agent connection or will set access level to *Servers* if tunnel connection is used.

Detailed list of functionality available to above mentioned access levels is the following:

Functionality	Mas- terServer	Con- trolServer	Servers
Read metrics, lists and table metrics	X	X	X
Web service, modbus, SNMP trap, syslog, tftp proxy operation (also requires enabling specific proxy type in agent configuration file)	X	X	X
Execute actions defined in agent configuration files or configuration policies	X	X	
Take screenshots	X	X	
Edit agent main configuration file	X		
Remote agent upgrade	X		
Install software packages	X		
Deploy/undeploy agent policies	X		
File manager – all write operations, e.g. file or folder creation, deletion, etc. (also requires enabling file manager and specifying root folder in agent configuration file)	X		
Sending notifications via user support application	X		
Running commands inside ` braces for File.* metrics and in log file monitoring	X		
Use of File.Content() metric	X		
SNMP.ScanAddressRange() and TCP.ScanAddressRange() lists (also requires EnableProxy = yes in agent configuration file)	X		
Agent, SNMP and TCP proxy operation (also requires enabling specific proxy type in agent configuration file)	X		

6.6.4 Shared secret

Shared secret is another level of server verification. By default authentication is disabled.

To enable *Shared Secret* verification on agent set *RequireAuthentication* agent configuration parameter to *yes*. In *Shared-Secret* agent configuration parameter set password what should be used for authentication.

If authentication for agent is enabled, then while connection agent requested shared secret from the server. Server check if password was set for this specific node in *Shared secret* field in communication properties of node. In case if there is no shared secret server sends content of *AgentDefaultSharedSecret* server configuration variable as shared secret.

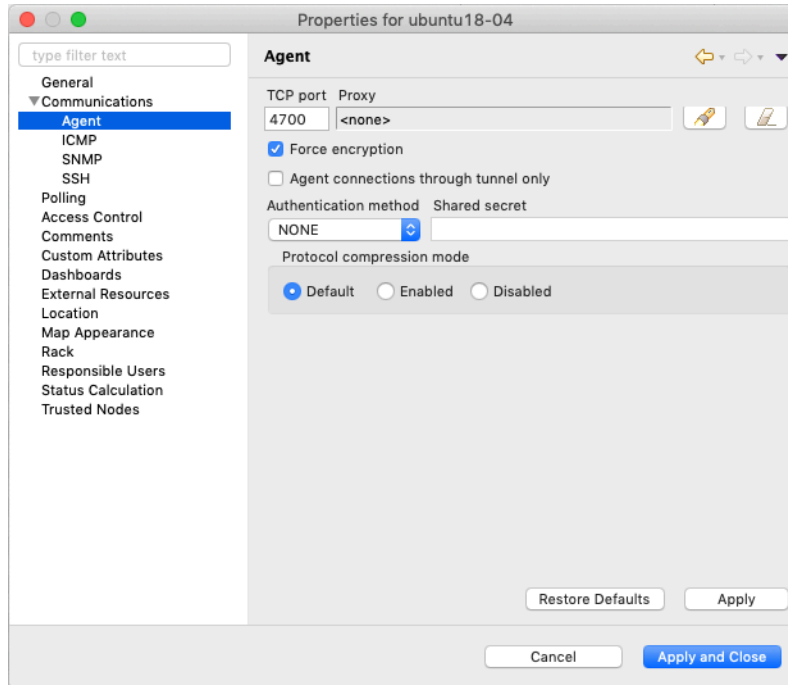


Fig. 2: Shared secret field in node communication properties.

In case shared secrets are not identical connection is not established.

6.6.5 Password encryption

When it is required to write password or *Shared Secret* in agent configuration file, there is possibility to encrypt it. All passwords can be encrypted with help of *nxencpasswd* command line tool and added in configuration file in encrypted way.

6.7 Subagents

Subagents are used to extend agent functionality. NetXMS subagent are libraries that are loaded by agent.

On Linux systems, where agent is installed from packages, some subagents are provided in separate packages (e.g. *netxms-agent-mqtt*) to avoid pulling unnecessary dependencies on systems where specific functionality is not needed. Subagents that do not require dependencies are shipped in *netxms-agent* package.

On Windows all available subagents are shipped in agent installer.

Subagents are enabled by adding corresponding line in agent configuration file (for example: *SubAgent=dbquery*).

Below is list of available NetXMS subagents:

- Bind9
- *Asterisk*
- *DB2*
- *Database Query (dbquery)*
- *DS18x20*
- File Manager (filemgr)

- `gps`
- *Informix*
- *Java*
- Linux (automatically loaded on Linux systems)
- *Log file and Windows event log monitoring (logwatch)*
- *lm-sensors*
- *MongoDB*
- *MQTT*
- *MySQL*
- *Network Service Check (netSVC)*
- *Oracle*
- ICMP Ping (`ping`)
- *Postgres*
- *Raspberry Pi*
- `sms`
- *ssh*
- *UPS*
- *Windows event log synchronization (wineventsync)*
- WinNT (Automatically loaded on Windows systems)
- Windows Performance (`winperf`)
- *WMI*
- XEN

6.7.1 Java subagent

This is a special type of subagent, that allows to load Java plugins (subagents written using Java language). Java subagent does not provide any functionality by itself.

There are several configuration parameters that are supported by Java subagent. None of them is mandatory.

Parameter	Description
<code>Jvm</code>	Path to JVM. System default is used if not set.
<code>Classpath</code>	This parameter is added to java CLASSPATH.
<code>Plugin</code>	This parameter defines plugin that should be loaded. Can be used multiple times.

Configuration example:

```
MasterServers = netxms.demo
SubAgent=java.nsm

[JAVA]
Jvm = /path/to/jvm
```

(continues on next page)

(continued from previous page)

```
Classpath = /path/to/user/classes  
Plugin = bind9.jar
```

Java plugins

List of available java plugins:

- JMX
- Bind9

6.7.2 Load of subagent as separate process

Load of subagent as separate process can be used in case it is necessary to load agent and subagent under different users. It can be done by adding `ExternalSubagent` parameter with unique ID that will represent connection name between agent and subagent. Create second configuration file for this subagent and add there `ExternalMasterAgent` parameter with same ID and run instance of `nxagentd` with this config. Now external subagent will communicate with master agent using Named Pipe. Only master agent will communicate with server.

6.8 Agent Proxy node configuration

In case it is required to monitor nodes behind firewall, it can be configured access to one of subnet nodes and used this node as a proxy node for others.

Proxy node can be set during node creation or in *Communications* tab of node properties. To configure proxy node select node in object selector *NetXMS Agent Proxy*.

Create Node Object

Name
|

Alias

Primary host name or IP address

NetXMS agent port
4700 - +

SNMP agent port
161 - +

EtherNet/IP port
44818 - +



SSH port
22 - +



SSH login



SSH password



Options



- ☐ Communication through external gateway
- ☐ Create as unmanaged object
- ☐ Enter maintenance mode immediately
- ☐ Create as zone proxy for selected zone
- ☐ Disable usage of NetXMS agent for all polls
- ☐ Disable usage of SNMP for all polls
- ☐ Disable usage of SSH for all polls
- ☐ Disable usage of ICMP ping for all polls
- ☐ Disable usage of EtherNet/IP for all polls
- ☐ Prevent automatic SNMP configuration changes



Proxy for NetXMS agents
None  


Proxy for SNMP
None  

Proxy for EtherNet/IP
None  

Proxy for ICMP
None  

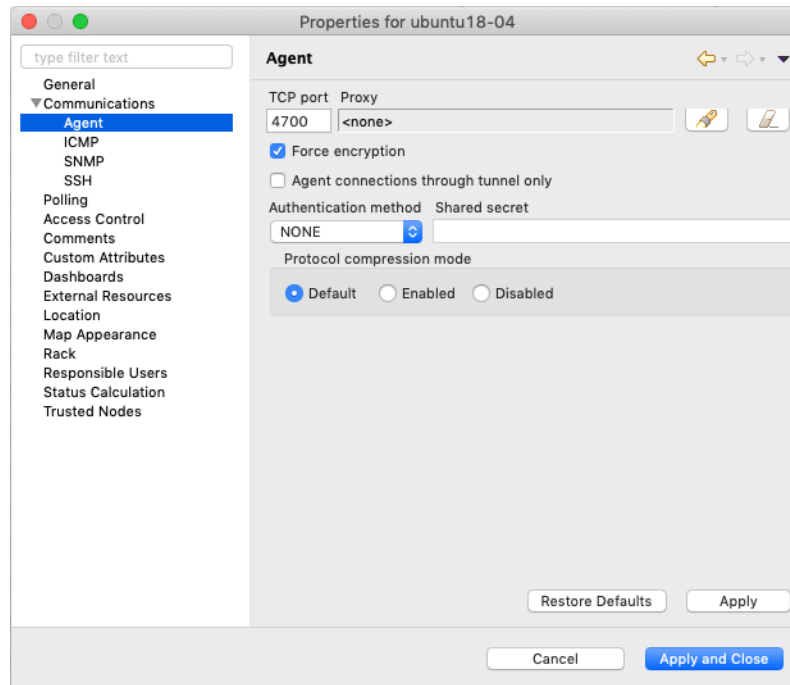
Proxy for SSH
<default>  

Proxy for web services
<default>  

Zone
Default 

☐ Show this dialog again to create another node

Cancel OK



6.8.1 Agent configuration

To enable NetXMS Agent proxy “EnableProxy” agent configuration parameter should be set to yes.

6.9 Agent External Metrics

Other option to define new metric that can be collected from node is to use `ExternalMetric/ExternalMetricShellExec`, or `ExternalList`, or `ExternalMetricProvider` configuration parameters to define a command that will be executed on a node and its output will be provided as a metric. This functionality provides flexibility to create your own metrics, lists or table metrics.

New metrics will be visible in the *Available metrics* list only after agent restart (agent reads its configuration files only once on start) and subsequent configuration poll, so to force its appearance run *Configuration poll* manually after agent restart.

Note

On Windows platforms UTF-8 encoding should be returned in External Metrics.

6.9.1 ExternalMetric/ExternalMetricShellExec

`ExternalMetric` defines name of the metric and command that is executed synchronously when this metric is requested by the server. Parameters from DCI configuration can be provided, these will be available as \$1, \$2, \$3..., \$9 variables. To accept parameters metric name should contain “(*)” symbols after name. Only first line of command output will be given as a result of execution (metric’s value).

`ExternalMetricShellExec` has same meaning as `ExternalMetric` and behaves identically on non-Windows systems. On Windows systems `ExternalMetric` executes specified command using system process execution API’s `CreateProcess()` function. It will search in PATH, but the command should be with file extension, e.g. `command.exe`. `ExternalMetricShellExec` will use shell to execute specified command on Windows.

To add multiple metrics, you should use multiple `ExternalMetric/ExternalMetricShellExec` entries.

As these commands are executed synchronously, long-executing commands may cause timeout. There are two timeouts - one on the agent side (controlled by `ExternalMetricTimeout` in agent's configuration file) and generic timeout for all requests to agent (controlled by `AgentCommandTimeout` in server's configuration file). It's strongly not recommended to increase server timeout to more than a few seconds because this may lead to performance issues due to poller threads spending too much time on timeouts. `ExternalMetricProvider` can be used to handle long-executing commands.

```
# Example

# Without DCI parameters
ExternalMetric=Name:command
ExternalMetricShellExec=Name:command

# With DCI parameters
ExternalMetric=Name(*):command $1 $2
ExternalMetricShellExec=Name(*):command $1 $2
```

For each metric configured two agent metrics are provided - one is `Name` as specified in `ExternalMetric/ExternalMetricShellExec` which provides output of the command (first line only), the other is `Name.ExitCode` that provides exit code of the executed command.

```
# Real example
ExternalMetric = Test:echo test
ExternalMetric = LineCount(*):cat $1 | wc -l
```

```
> nxget localhost Test
test
> nxget localhost 'LineCount(somefile.txt)'
42
> nxget localhost 'LineCount(somefile.txt).ExitCode'
0
```

6.9.2 ExternalList

`ExternalList` defines name of the list metric and command that is executed synchronously when this metric is requested by server. Parameters from DCI configuration can be provided, these will be available as `$1`, `$2`, `$3...`, `$9` variables. To accept parameters metric name should contain “(*)” symbols after name. Lines of the list are separated by new line character.

```
# Example

# Without DCI parameters
ExternalList=Name:command

# With DCI parameters
ExternalList=Name(*):command $1 $2
```

6.9.3 ExternalMetricProvider

`ExternalMetricProvider` defines command (script) and execution interval in seconds. Defined script will be executed regularly and agent will cache list of metrics along with their values. When server will request one of provided metrics, it's value will be read from the agent cache. Main purpose is to provide data from long-running processes, or retrieve multiple values by running a command only once.

Timeout in milliseconds for command execution is defined by *ExternalMetricProviderTimeout* parameter in agent configuration file.

Script should print one or more “Metric=Value” pairs to standard output. Multiple pairs should be separated by new line. If metric takes a parameter, it should be included in “Metric(...)”.

Example of the script:

```
#!/bin/sh
echo 'Metric1=Value1'
echo 'Metric2=Value2'
echo 'MetricWithParams(parameter)=Value3'
echo 'MetricWithParams(another_parameter)=Value4'
```

Example of agent configuration:

```
#Example
ExternalMetricProvider=PATH_TO_PROVIDER_SCRIPT:EXECUTION_INTERVAL_IN_SECONDS

#Example (run /tmp/test.sh every 5 seconds)
ExternalMetricProvider=/tmp/test.sh:5
```

6.9.4 ExternalTable

ExternalTable defines table that is provided by agent and how it can be obtained. Table can be collected synchronously when requested by the server or regularly in the background (in this case server gets cached data). Second option is useful when command for table creation is taking a long time to avoid timeout. To collect table in the background “PollingInterval” configuration option is required.

Timeout in milliseconds for background operation is defined by *ExternalMetricProviderTimeout* parameter in agent configuration file.

Each table line is separated with new line symbol. First line in returned text should contain name of columns, subsequent lines contain table data. Parameters from DCI configuration can be provided, these will be available like \$1, \$2, \$3..., \$9 variables. To accept parameters metric name should contain (*) symbols after name.

Name	Re- quired	Description
Command	Yes	Result of this command execution will be used as a value for table DCI. First row is used as column names.
Separator	No	Symbol that will be used as a separator for columns. If separator is not specified, default value of , is used.
<div style="border: 1px solid #ccc; padding: 10px; margin: 10px 0;"> <p>Note</p> <p>Separator supports special macros for separator:</p> <ul style="list-style-type: none"> \n - \n \r - \r \s - space \t - tab \u115 - unicode character number 115 </div>		
InstanceColumns	No	Comma separated instance column list.
<div style="border: 1px solid #ccc; padding: 10px; margin: 10px 0;"> <p>Note</p> <p>Instance column should contain unique identifier for each table row. If several instance columns are used, then combination of these columns should be unique. This is necessary for building graphs and for correct threshold violation event generation. Row number is used if instance column is not set.</p> </div>		
Description	No	Table DCI description that will be shown in table DCI selector.
PollingInterval	No	Interval that is used to poll table in the background. Table will be collected synchronously (per request) if this parameter is omitted.
ColumnType	No	Data type of the column. Is set in format columnName:datatypeName. If column does not have type int32 is used by default.
<p>Possible options:</p> <ul style="list-style-type: none"> int32 uint32 int64 uint64 string float counter32 counter64 		

```
# Example

# Simple example
[ExternalTable/test]
Command = echo 'col1;col2;col3\na;b;c'
Separator = ;

# Without DCI parameters
[ExternalTable/dciName]
Command = command
Separator = ;
```

(continues on next page)

(continued from previous page)

```

InstanceColumns = columnName,columnName2
Description = description
PollingInterval = 60
ColumnType = columnName:string
ColumnType = columnName3:string

# With DCI parameters
[ExternalTable/dciName(*)]
Command = cat /folder/with/my/files/$1

# Old configuration format
ExternalTable=dciName::command
ExternalTable=dciName:instanceColumns=columnName;description=description;
↪separator=:command
ExternalTable=dciName(*) :instanceColumns=columnName;description=description;
↪separator=:command $1 $2
#Old configuration format with background polling
ExternalTable=dciName:instanceColumns=columnName;description=description;
↪separator=:command;backgroundPolling=yes;pollingInterval=60

```

Note

backgroundPolling configuration should be set to true or yes in order to use polling interval with old configuration format.

6.10 Agent Actions

For security reasons actions that can be executed on agent first are defined in agent configuration file and only then can be used by users. This excludes that an unauthorized user can access system data through an arbitrary entered command. Only users with access to the agent configuration file editing can define executed commands.

There are 2 options to define action:

1. Action - usual action definition. On Windows platform system process execution API's CreateProcess() is used to run the command, it will search in PATH, but the command should be with file extension, e.g. command.exe.
2. ActionShellExec - Same as Action, but on the Windows platform agent will use shell to execute command instead of normal process creation. There is no difference between Action and ActionShellExec on UNIX platforms.

Both versions accept parameters that will be available like \$1, \$2, \$3..., \$9 variables.

After action is defined it can be used in the *object tools - agent action* or in *actions - action execution on remote node*. Action should be defined in main section of agent configuration file.

```

# Example
Action=Name:command
Action=Name:command $1 $2
Action=cleanLogs:rm /opt/netxms/log/*
Action=ping:ping $1
ActionShellExec=listFiles:dir $1

```


SERVER MANAGEMENT

7.1 Configuration file

File `netxmsd.conf` is a configuration file for NetXMS server. It contains information necessary for establishing database connection, and some optional server parameters. Default location for this file is `/etc/netxmsd.conf` on UNIX systems and `InstallationPath\etc\netxmsd.conf` on Windows.

The file can contain one or more parameters in *Parameter = Value* form, each parameter should be on its own line. Comments can be inserted after “#” sign.

Detailed list of parameters can be found there: *Server configuration file (netxmsd.conf)*.

Configuration file example:

```
#  
# Sample server configuration file  
#  
  
DBDriver = mysql.ldr  
DBServer = localhost  
DBName = netxms_db  
DBLogin = netxms  
DBPassword = password  
LogFile = {syslog}
```

7.2 Server configuration for Agent to Server connection / Tunnel connection

NetXMS provides option to establish connection from agent to server. This requires additional configuration on server and on agent sides. This chapter describes server side configuration. Agent side configuration can be found in *Agent to server connection*. Agent to server connection is a *TLS* tunnel carrying virtual server to agent connections.

Server configuration can be separated into two parts: initial configuration (certificate generation and configuration) and node binding.

Server provide option to configure automatic options on new unbound tunnel connection. Once new unbound tunnel connection comes to server - idle timeout counter starts for this connection. If nothing done while *AgentTunnels.UnboundTunnelTimeout* time, automatic action selected in *AgentTunnels.UnboundTunnelTimeoutAction* will be executed.

There are 4 types of actions, that can be done automatically:

1. Reset tunnel - close tunnel. It will be automatically reopened again by agent. This process will update information on server in case of change on agent.

2. Generate event - generates event *SYS_UNBOUND_TUNNEL*, that later can be used for administrator notification or any other automatic action (see *Event processing*).
3. Bind tunnel to existing node - will try to find correct node and bind tunnel to it. Node matching rules will be described further.
4. Bind tunnel to existing node or create new node - will try to find correct node and bind tunnel to it. If node is not found new node will be created under container mentioned in *AgentTunnels.NewNodesContainer* server configuration parameter. Node matching rules will be described further.

Node is matched for binding if:

1. Zone UIN given by agent (is configured in agent configuration under *ZoneUIN*) match to node zone id
2. IP given by agent match to node's IP address
3. Hostname or FQDN match with node name

7.2.1 Initial configuration

Certificate should be issued and added to the server configuration. This certificate will be used to issue public certificates for agents. Certificate usage should allow certificate signing. Certificates should be in PEM format. Server key should be added to the certificate file or should be provided as a separate configuration parameter.

Certificate can be obtained in two ways:

1. By sending *CSR* request to your *CA*
2. Create self signed certificate

Settings in server configuration file:

Parameter	Description	Required
TrustedCertificate	Certificate issued by certificate authority or self-signed <i>CA</i> certificate. If certificate chain for server certificate is longer, all upper level certificates should be added to configuration file by adding multiple TrustedCertificate entries.	Yes
ServerCertificate	Certificate issued by certificate authority. This certificate is used to issue agent certificates. ServerCertificate parameter also implies that this certificate is trusted by the server when checking agent certificate validity.	Yes
ServerCertificatePassword	Server certificate password.	Can be omitted if certificate does not use password.
ServerCertificateKey	Server certificate private key.	Can be omitted if key is included in server certificate file.

There are additional option to configure separate certificates for agent certificate issuing and for connection. If there is no need to issue certificates (they are externally provisioned) only connection certificate is required.

Connection certificate settings: TunnelCertificate, TunnelCertificateKey, TunnelCertificatePassword
Issuing certificate settings: InternalCACertificate, InternalCACertificateKey, InternalCACertificatePassword

Note

If ServerCertificate settings are set it will be fall back option for TunnelCertificate and InternalCACertificate

Server configuration variable settings:

Parameter	Description	Default
AgentTunnels.UnboundTunnelTimeoutAction	Action that will be executed after idle timeout. Actions are described here: <i>Server configuration for Agent to Server connection / Tunnel connection</i>	Reset tunnel
AgentTunnels.UnboundTunnelTimeout	Tunnel idle timeout in seconds, that will be waited till automatic action execution.	3600
AgentTunnels.NewNodesContainer	Container name where newly created nodes will accrue. You can use -> character pair to create subtree (like Office->Tunnel). If no container is set nodes will appear under <i>Entire Network</i>	

Self signed certificate sample

This manual describes only simplest option: self signed certificate creation. It does not contain any information about file access right assignment.

1. Create private root key (add `-aes256` parameter to use password):
`openssl genrsa -out rootCA.key 2048`
2. Create self signed root certificate:
`openssl req -x509 -new -key rootCA.key -days 10000 -out rootCA.crt`
3. Create server key (add `-aes256` parameter to use password)
`openssl genrsa -out server.key 2048`
4. Create `openssl.conf` file. Content of file (dn section should be changed accordingly):

```
[req]
distinguished_name = dn
req_extensions = v3_ca
prompt = no

[dn]
countryName = LV
stateOrProvinceName = Riga
localityName = Riga
organizationName = netxms.org
commonName = Monitoring Server

[v3_ca]
basicConstraints = CA:TRUE
```

5. Create server certificate request
`openssl req -new -key server.key -out server.csr -config openssl.conf`

6. Sign server certificate with root CA certificate

```
openssl x509 -req -in server.csr -CA rootCA.crt -CAkey rootCA.key -CAcreateserial
-out server.crt -days 5000 -extfile openssl.conf -extensions v3_ca
```

Add newly created certificates to server configuration (netxmsd.conf file).

```
TrustedCertificate = /opt/netxms/key/rootCA.crt
ServerCertificate = /opt/netxms/key/server.crt
ServerCertificateKey = /opt/netxms/key/server.key
```

7.2.2 Reissuing server certificate

When server certificate validity term is coming to an end or there are some security considerations, server certificate can be reissued. There are two options - server certificate can be reissued using same root CA or, if you use self-signed root CA, it can also be reissued.

To perform a smooth transition from old to new server certificate, old certificates can be specified as TrustedCertificate in server configuration file. In this case agents with certificates issued based on the old server certificate would still be able to connect, but new agent certificates will be issued based on the new server certificate.

After all agents will receive agent certificate signed by the new server certificate, old certificates can be removed from server configuration file.

Server configuration example if self-signed root CA was reissued:

```
# ~~~ Old root certificate ~~~
TrustedCertificate = /opt/netxms/key/old_rootCA.crt

# ~~~ Old server certificate ~~~
TrustedCertificate = /opt/netxms/key/old_server_certificate.crt

# ~~~ New root certificate ~~~
TrustedCertificate = /opt/netxms/key/rootCA.crt

# ~~~ New server certificate ~~~
ServerCertificate = /opt/netxms/key/server.crt
ServerCertificateKey = /opt/netxms/key/server.key
```

7.2.3 Node binding

Once server certificates are configured and agent is correctly configured (*ServerConnection* parameter set in *agentd.conf*) requests for agent to server connection will be shown in *Agent Tunnel Manager* view.

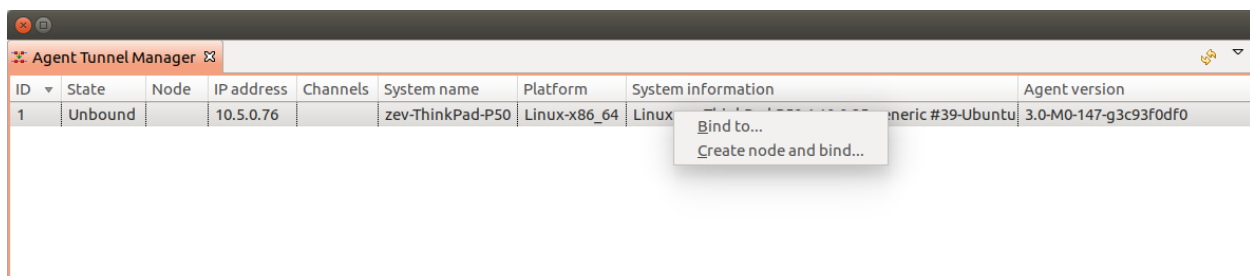
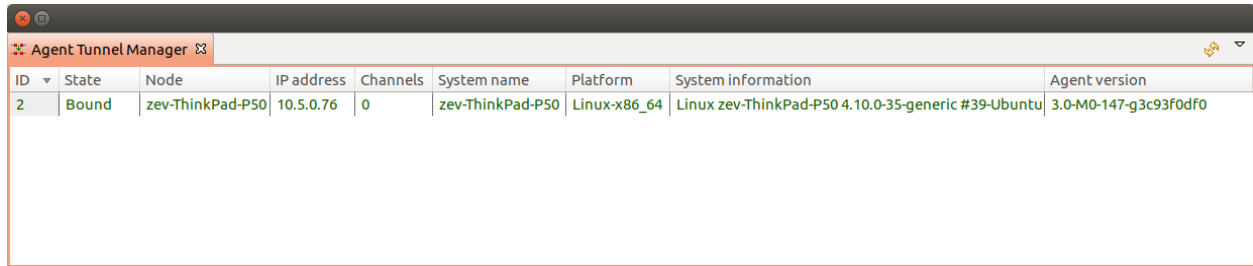


Fig. 1: Agent Tunnel Manager

User should manually accept them by binding to existing node *Bind...* or by creating new one *Create node and bind...*. Once node will be bound - it's state in *Agent Tunnel Manager* view will be changed to *Bound*.

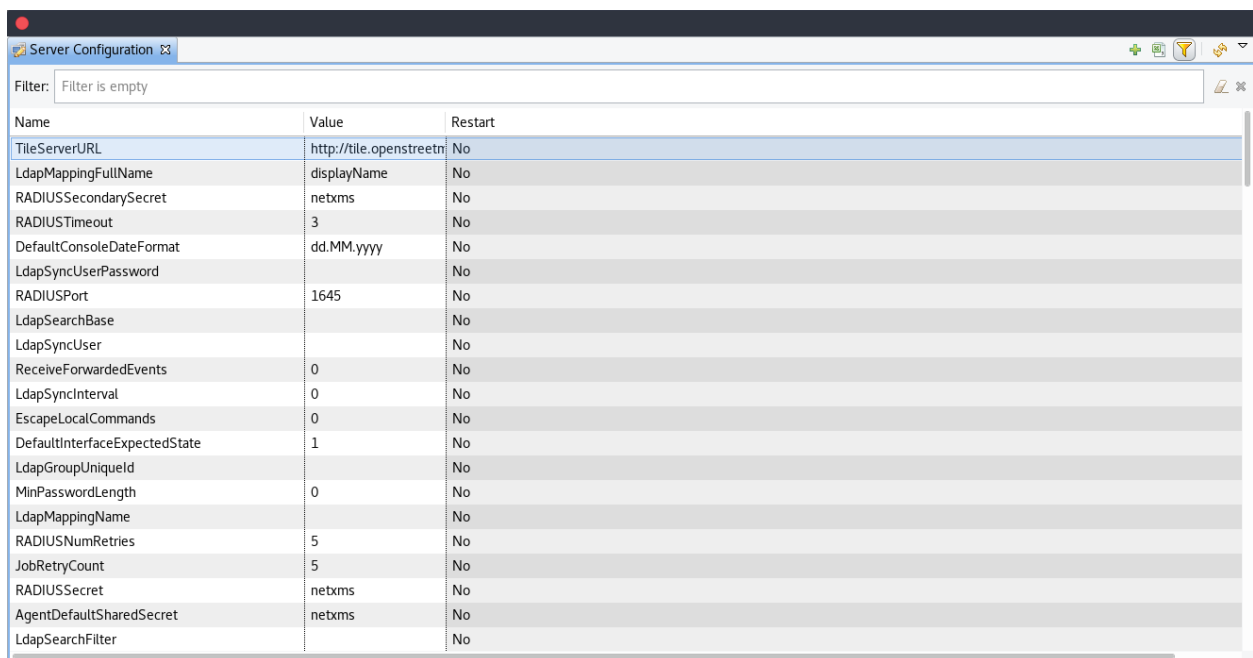


ID	State	Node	IP address	Channels	System name	Platform	System information	Agent version
2	Bound	zev-ThinkPad-P50	10.5.0.76	0	zev-ThinkPad-P50	Linux-x86_64	Linux zev-ThinkPad-P50 4.10.0-35-generic #39-Ubuntu	3.0-M0-147-g3c93f0df0

Fig. 2: Agent Tunnel Manager

7.3 Configuration variables

These variables are stored in database and can be changed using *Server Configuration Editor* [view](#) accessing it *Configuration*•*Server Configuration* or with help of `nxdbmgr`` (example: `:code:`nxdbmgr set <name> <value>``).



Name	Value	Restart
TileServerURL	http://tile.openstreetn	No
LdapMappingFullName	displayName	No
RADIUSSecondarySecret	netxms	No
RADIUSTimeout	3	No
DefaultConsoleDateFormat	dd.MM.yyyy	No
LdapSyncUserPassword		No
RADIUSPort	1645	No
LdapSearchBase		No
LdapSyncUser		No
ReceiveForwardedEvents	0	No
LdapSyncInterval	0	No
EscapeLocalCommands	0	No
DefaultInterfaceExpectedState	1	No
LdapGroupUniqueld		No
MinPasswordLength	0	No
LdapMappingName		No
RADIUSNumRetries	5	No
JobRetryCount	5	No
RADIUSSecret	netxms	No
AgentDefaultSharedSecret	netxms	No
LdapSearchFilter		No

Fig. 3: Server Configuration

Detailed description of each configuration can be found there: [Server configuration parameters](#). Please note that changes to most of the settings will take effect only after server restart.

7.4 Synchronization between servers

NetXMS does not provide horizontal scalability for server. But there is option to exchange with events between servers. Information about configuration can be found there: [Forward event](#). Event forward does not work with zones.

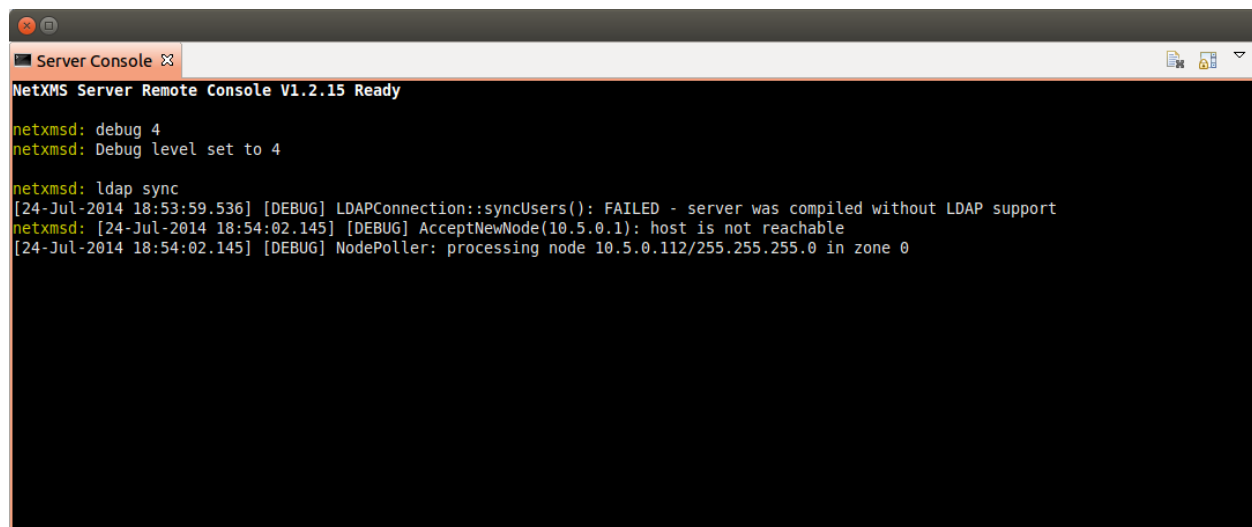
7.5 netxmsd commandline options

Command	Description
-e	Run database check on startup
-c <file>	Set non-default configuration file Default is {search}
-d	Run as daemon/service
-D <level>	Set debug level (valid levels are 0..9)
-h	Display help and exit
-p <file>	Specify pid file.
-q	Disable interactive console
-v	Display version and exit

7.6 Server debug console

Server debug console can be opened in Management Client. It can be found in *Tools -> Server Console*.

It can be used to check debug messages or to execute one of server commands like “ldap sync”.



```
NetXMS Server Remote Console V1.2.15 Ready

netxmsd: debug 4
netxmsd: Debug level set to 4

netxmsd: ldap sync
[24-Jul-2014 18:53:59.536] [DEBUG] LDAPConnection::syncUsers(): FAILED - server was compiled without LDAP support
netxmsd: [24-Jul-2014 18:54:02.145] [DEBUG] AcceptNewNode(10.5.0.1): host is not reachable
[24-Jul-2014 18:54:02.145] [DEBUG] NodePoller: processing node 10.5.0.112/255.255.255.0 in zone 0
```

7.6.1 Server commands

Command	Description
debug [<level> off]	Set debug level (valid range is 0..9)
down	Shutdown NetXMS server
exec <script> [<params>]	Executes NXSL script from script library
exit	Exit from remote session
kill <session>	Kill client session
get <variable>	Get value of server configuration variable
help	Display this help
ldapsync	Synchronize ldap users with local user database
poll <type> <node>	Initiate node poll
raise <exception>	Raise exception
set <variable> <value>	Set value of server configuration variable
show components <node>	Show physical components of given node
show dbcp	Show active sessions in database connection pool
show fdb <node>	Show forwarding database for node
show flags	Show internal server flags
show index <index>	Show internal index
show modules	Show loaded server modules
show objects	Dump network objects to screen
show pollers	Show poller threads state information
show queues	Show internal queues statistics
show routing-table <node>	Show cached routing table for node
show sessions	Show active client sessions
show stats	Show server statistics
show topology <node>	Collect and show link layer topology for node
show users	Show users
show vlans <node>	Show cached VLAN information for node
show watchdog	Display watchdog information
trace <node1> <node2>	Show network path trace between two nodes

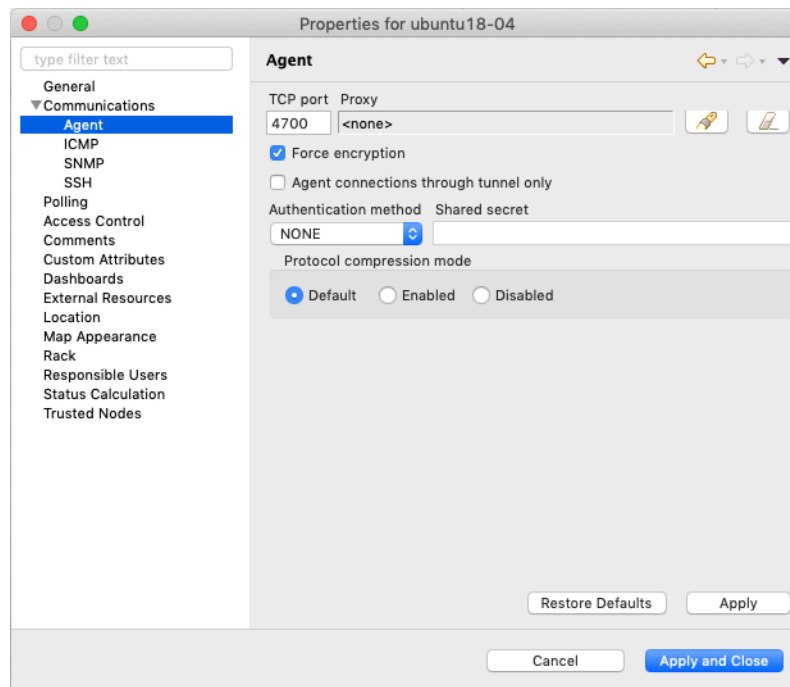
7.7 Configuring self-monitoring

7.8 Database connection pool

7.9 ICMP proxy

To used ICMP proxy Ping subagent should be loaded for ICMP proxy node.

This proxy is used to check node availability when *Zones* are used.



8.1 SNMP Drivers

Various SNMP devices might require special measures to get information, e.g. some devices provide additional information for interfaces only under vendor OIDs, etc. To address this, NetXMS provides a concept of SNMP drivers. SNMP driver is detected automatically.

If SNMP driver was not automatically detected, it's possible to set it manually by specifying driver name in custom attribute `snmp.driver` on a node.

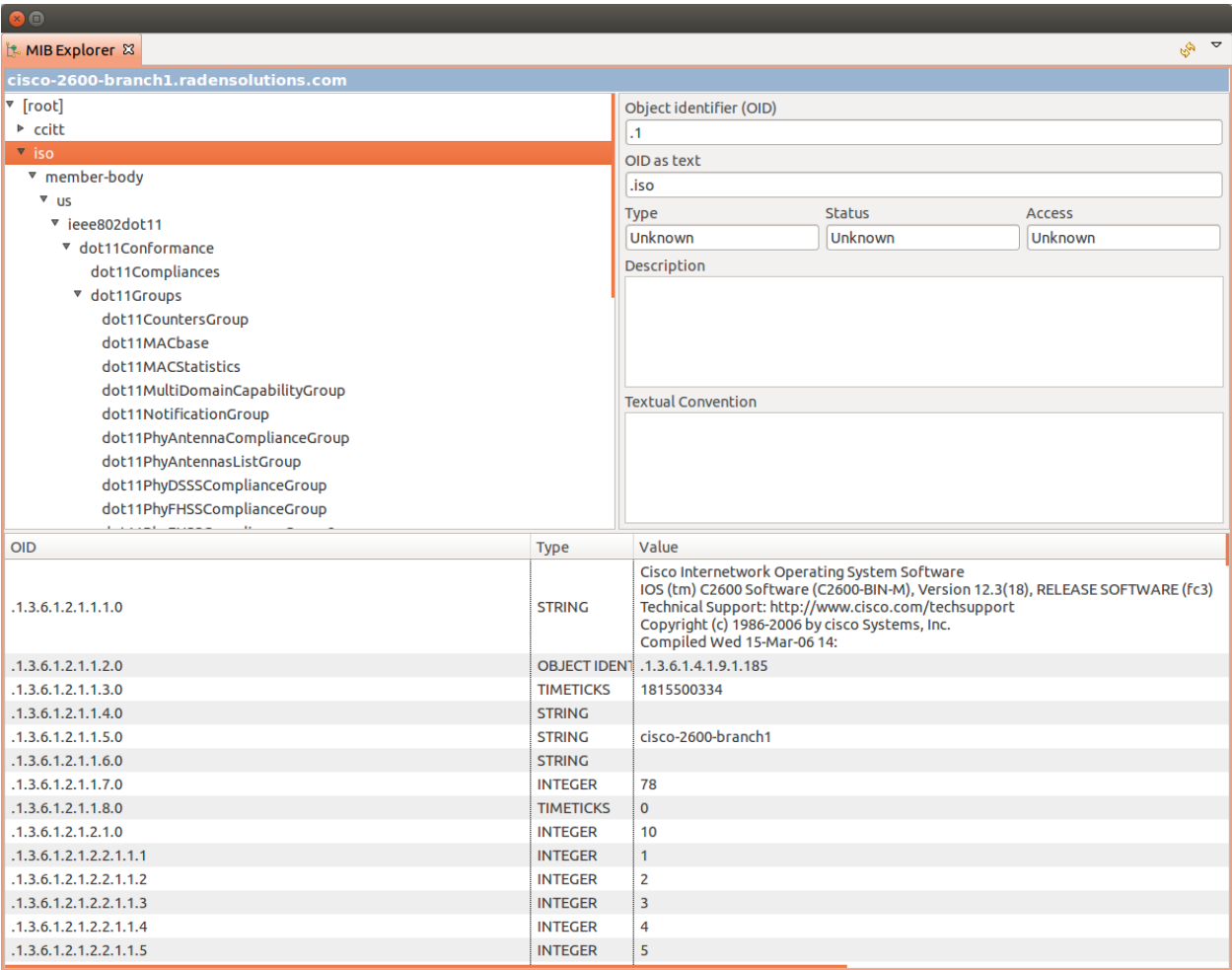
Possible SNMP driver names are:

- ARUBA-SW
- AT
- BAYSTACK
- CAMBIUM-CNPILOT
- CAMBIUM-EPMP
- CATALYST-2900XL
- CATALYST-GENERIC
- CISCO-ESW
- CISCO-GENERIC
- CISCO-NEXUS
- CISCO-SB
- CISCO-WLC
- DELL-PWC
- DLINK
- EDGECORE-ESW
- ELTEX
- ERS8000
- ETHERWAN
- EXTREME
- FORTIGATE
- H3C

- HIRSCHMANN-CLASSIC
- HIRSCHMANN-HIOS
- HPE-ILO
- HPSW
- HUAWEI-SW
- IGNITENET
- JUNIPER
- MDS-ORBIT
- MIKROTIK
- MOXA-EDR
- NET-SNMP
- NETONIX
- NETSCREEN
- NTWS
- OPTIX
- PING3
- PROCURVE
- QTECH-OLT
- QTECH-SW
- RITTAL
- RUGGEDCOM
- SAF-INTEGRA-B
- SYMBOL-WS
- TB
- TELTONIKA
- TPLINK
- UBNT-AIRMAX
- UBNT-EDGESW
- WESTERSTRAND

8.2 MIB Explorer

MIB browser shows all loaded MIB configurations, and allows to run *SNMP* walk on a selected node *nodes*. Node can be selected in browser by selecting *Set node object...* option in view menu or by opening *MIB Explorer* from node menu.



To run walk user should select line of tree from were will be requested all data. By walk will be requested all OID subtree of selected item.

After walk is done it's results will shown in the table below.

OID	Type	Value
.1.3.6.1.2.1.1.1.0	STRING	Cisco Internetwork Operating System Software IOS (tm) C2600 Software (C2600-BIN-M), Version 12.3(18), RELEASE SO Technical Support: http://www.cisco.com/techsupport Copyright (c) 1986-2006 by cisco Systems, Inc. ed 15-Mar-06 14: .1.185
.1.3.6.1.2.1.1.2.0		
.1.3.6.1.2.1.1.3.0		
.1.3.6.1.2.1.1.4.0		
.1.3.6.1.2.1.1.5.0		
.1.3.6.1.2.1.1.6.0		
.1.3.6.1.2.1.1.7.0		
.1.3.6.1.2.1.1.8.0		
.1.3.6.1.2.1.2.1.0		
.1.3.6.1.2.1.2.2.1.1		
.1.3.6.1.2.1.2.2.1.2		

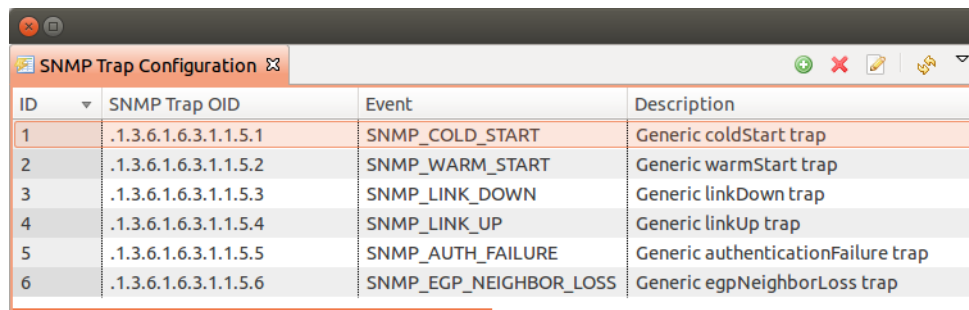
There are next options available for results:

- Copy result line to clipboard
- Copy name of selected line to clipboard

- Copy type of selected line to clipboard
- Copy value of selected line to clipboard
- Export selected lines to CSV
- Show selection in MIB tree
- Create DCI from selected item

8.3 SNMP Trap Configuration

In this view is configured which event will be generated on exact trap OID and which OID data will be used as event parameter data.



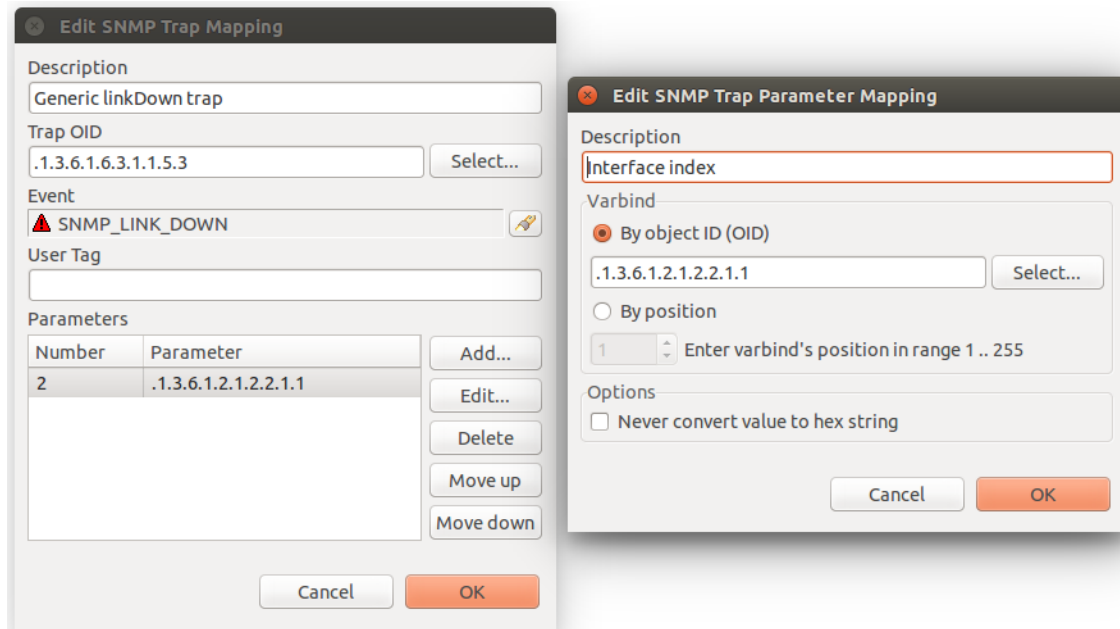
ID	SNMP Trap OID	Event	Description
1	.1.3.6.1.6.3.1.1.5.1	SNMP_COLD_START	Generic coldStart trap
2	.1.3.6.1.6.3.1.1.5.2	SNMP_WARM_START	Generic warmStart trap
3	.1.3.6.1.6.3.1.1.5.3	SNMP_LINK_DOWN	Generic linkDown trap
4	.1.3.6.1.6.3.1.1.5.4	SNMP_LINK_UP	Generic linkUp trap
5	.1.3.6.1.6.3.1.1.5.5	SNMP_AUTH_FAILURE	Generic authenticationFailure trap
6	.1.3.6.1.6.3.1.1.5.6	SNMP_EGP_NEIGHBOR_LOSS	Generic egpNeighborLoss trap

In SNMP Trap mapping configuration window can be set next parameters:

- Description of mapping rule
- Trap OID or trap OID group with many subtree OIDs, matching OID will be given to event as \$1 parameter
- Event that will be generated on selected Trap OID
- User Tag is special event attribute, that can be got by %u macros or as attribute of event class. This attribute can be set there or by script.
- Parameters - OID values that will be passed to event as \$2, \$3, \$4... parameters

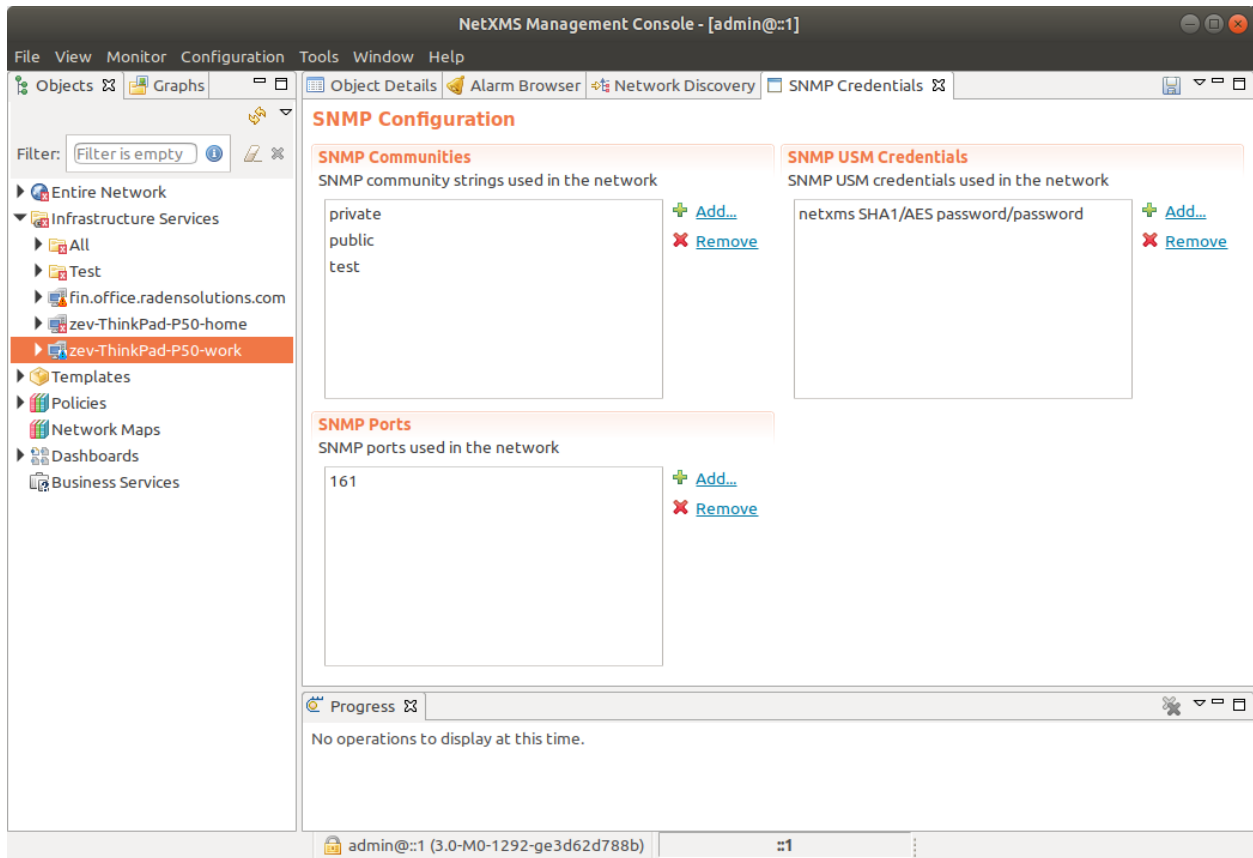
In parameter configuration(*Edit SNMP Trap Parameter Mapping*) can be configured next things:

- Description of a parameter
- Select if parameter should be found by OID or by position in the message
- Option not to convert value to hex string. If string contains not readable symbols(symbol number less than space symbol number) it will be automatically converted to hex string, this option is required to prevent auto conversion.



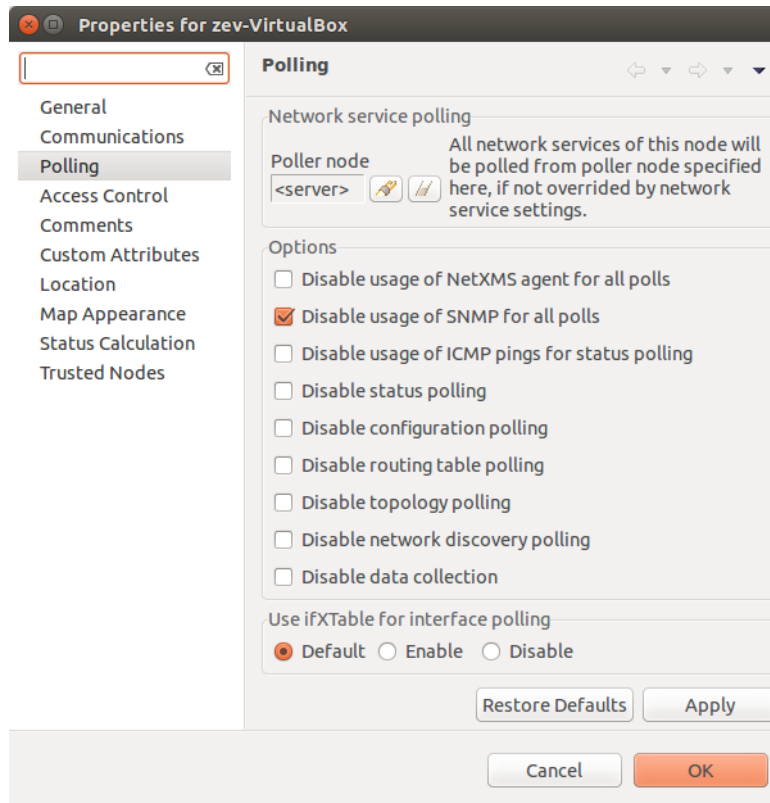
8.4 Default SNMP credentials

Default SNMP credentials can be set in *Configuration* ▶ *SNMP Credentials*. It does not matter if credentials are used for adding nodes manually, through network discovery or with the help of agent registration - in each case *SNMP Credentials* configuration value will be checked.



8.5 Using ifTable and ifXTable

There are 2 types of subtree that provides information about interfaces: old one ifTable and new one ifXTable. Sometimes usage of new one creates error situations. In this situation ifXTable can be disabled. This can be done in Properties of *node* in *Polling*. Or this configuration can be set globally by changing UseIfXTable server configuration parameter.



8.6 Configure SNMP Proxy

If there is need to monitor nodes behind firewall using SNMP, there is option to install on one of the nodes NetXMS agent, open all required ports for this node and send SNMP request to other nodes in this subnet through installed agent.

Proxy configuration can be done while creation of node or for already created node can be change in *Communications* tab of node properties. To configure proxy node select node in object selector *SNMP Proxy*.

Create Node Object

Name

Alias

Primary host name or IP address

NetXMS agent port

4700

-

+

SNMP agent port

161

-

+

EtherNet/IP port

44818

-

+

SSH port

22

-

+

SSH login

SSH password

Options

☐ Communication through external gateway
☐ Create as unmanaged object
☐ Enter maintenance mode immediately
☐ Create as zone proxy for selected zone
☐ Disable usage of NetXMS agent for all polls
☐ Disable usage of SNMP for all polls
☐ Disable usage of SSH for all polls
☐ Disable usage of ICMP ping for all polls
☐ Disable usage of EtherNet/IP for all polls
☐ Prevent automatic SNMP configuration changes

Proxy for NetXMS agents

None

Proxy for SNMP

None

Proxy for EtherNet/IP

None

Proxy for ICMP

None

Proxy for SSH

<default>

Proxy for web services

<default>

Zone

Default

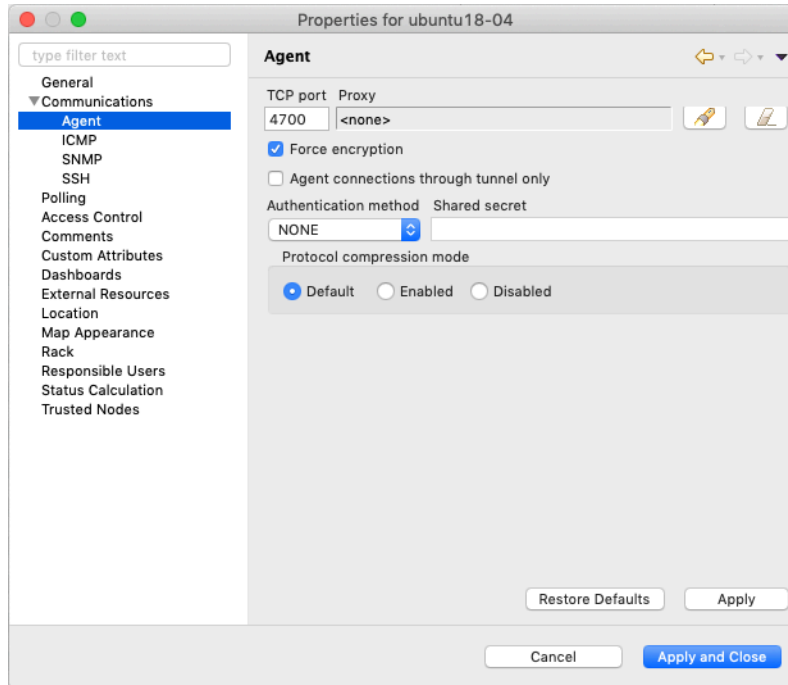
☐ Show this dialog again to create another node

Cancel

OK

96

Chapter 8. SNMP



8.6.1 Agent configuration

To enable SNMP proxy “EnableSNMPProxy” parameter should be set to “yes”.

8.7 Configure SNMP Trap Proxy

It is possible to proxy SNMP traps.

In this case as a destination of traps should be set the proxy node.

8.7.1 Agent configuration

To enable trap proxy “EnableSNMPTrapProxy” parameter should be set to “yes”.

Optionally can be configured also “SNMPTrapListenAddress” and “SNMPTrapPort”. Default values can be checked there: [Master configuration file](#)

8.7.2 Server configuration

By default traps are accepted only from known nodes. To accept all traps set “LogAllSNMPTraps” server configuration variable to 1.

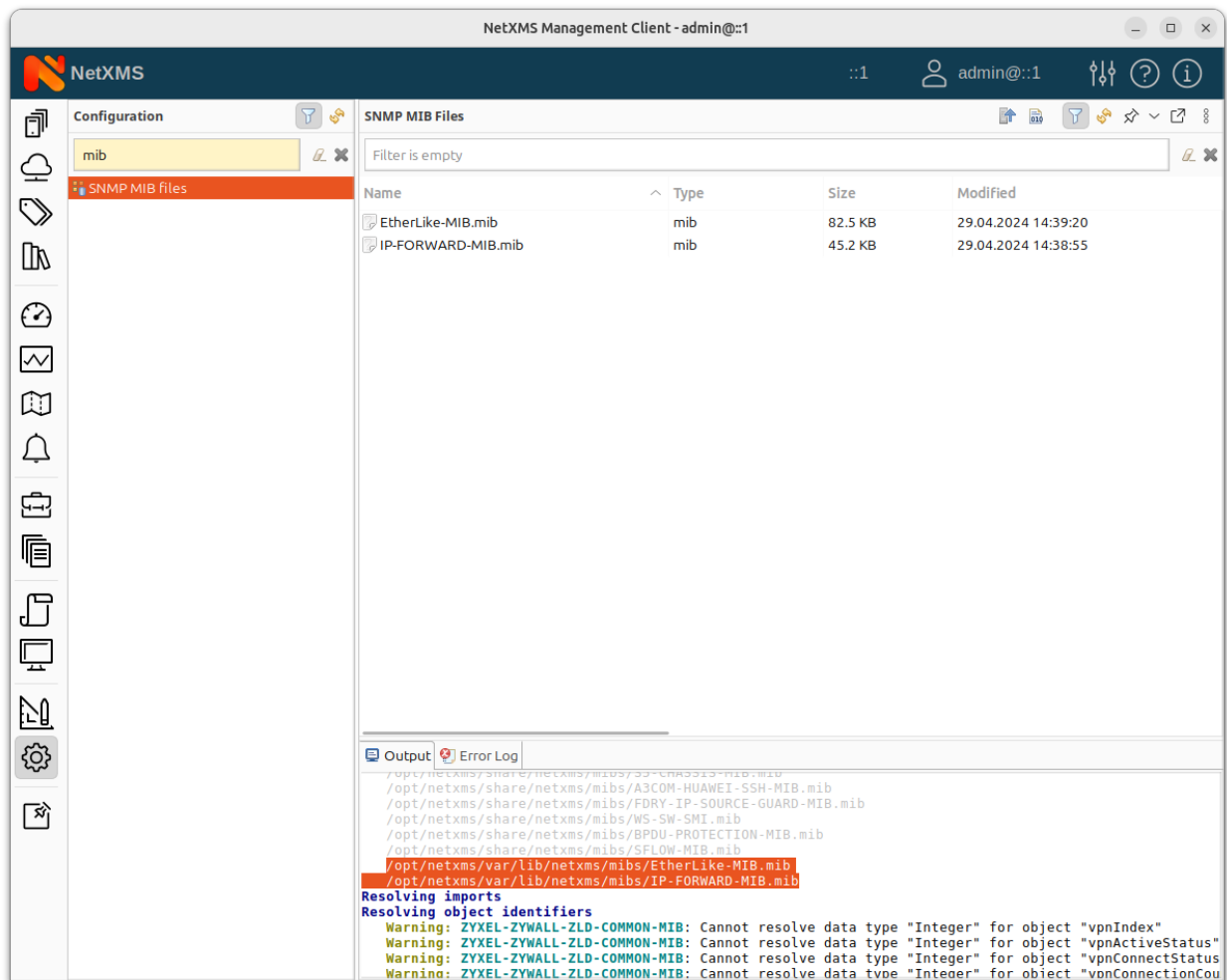
To correctly send response for SNMPv3, it should be also configured the proxy node for the sender node. It is done in sender node properties in “Communications” tab, SNMP section.

8.8 Import MIB

MIB files (MIBs) describe structure of information transferred via SNMP. Every device can support multiple MIBs, some of them are standard and public, other can be proprietary and vendor specific. NetXMS uses compiled MIBs to allow you to select OID and see its description (for example when selecting SNMP data for DCI collection). You do not need to compile new MIBs if you are OK with direct input of OID.

8.8.1 Manage User MIBs

To add additional MIBs go to *Configuration* → *SNMP MIB files*. Upload mib files with extension **.mib** and hit compile button. MIB compilation log will be visible in *Output* tab and warnings/errors will be added to *Error Log* tab. After MIB files are successfully compiled all opened clients automatically download new version from server.



8.8.2 For versions older 5.0

Compiling MIBs

- Change suffix of your new MIB file to .txt
- Copy your MIB file to /usr/share/netxms/mibs
- Use nxmibc binary to create a new compiled MIB file from all MIBs in directory. Add parameter -z for compressed output file.

```
nxmibc -d /usr/share/netxms/mibs -o /var/lib/netxms/netxms.mib
```

Parameters recognized by nxmibc:

```
nxmibc [options] source1 ... sourceN
```

Valid options:

- d <dir> : Include all MIB files from given directory to compilation
- o <file> : Set output file name (default is netxms.mib)
- P : Pause before `exit`
- s : Strip descriptions from MIB objects
- z : Compress output file

Troubleshooting

If nxmibc fails, it may be caused by syntax or import errors in your MIB. Try to check it with smilint (part of net-snmp package) and correct any errors on level 3.

8.9 Working with the SNMP Tables

When we do SNMP walk the resulting SNMP table item OIDs consist of three parts. For the sake of our explanation, we will mark these parts with the letters:

XXXXYYNNN, where

XXX is part that does not change — we can call it a Table base OID; YY is part that represents different columns; NNN is the instance part. The instance part represents rows in the table.

Now, as an example, we can imagine the table with base “1.3.6.1.2.1.2.2.1” like the one below:

1.3.6.1.2.1.2.2 .1	.2	.3	.4	.5	.6
.1	1	lo	24	65536	10000000
.2	2	VMware VMXNET3 Ether- net Controller	6	1500	4294967295 005056A5BA4D

In this table the columns are YY numbers (that are usually single numbers in ascending order), and the rows are the NNN number.

In this table the columns are YY numbers (that are usually single numbers in ascending order), and the rows are the NNN number.

Example

So, in order to get the “lo” value we should request “1.3.6.1.2.1.2.2.1.1”, where “1.3.6.1.2.1.2.2.1” represent XXX, “.2” (the value in the column where “lo” is situated) represents the YY and “.1” (the value in the row where “lo” is situated) represents the NNN.

8.9.1 How to Create a Table

To create a table, use the table base and the column part OID (XXXXYY).

In this way, taking as the example the SNMP table shown above, “1.3.6.1.2.1.2.2.1.1” can be used as the metric for the DCI cofniguration.

Properties for .1.3.6.1.2.1.2.2.1.1

General

Custom Schedule
Table Columns
Transformation
Table Thresholds
Instance Discovery
Access Control
SNMP
Other Options
Comments

General

Metric to collect—

Origin: SNMP Source node override: None

Metric: .1.3.6.1.2.1.2.2.1.1

Display name: **Interface table**

Collection schedule—

☒ Server default interval (60 seconds)
☐ Custom interval
☐ Advanced schedule

History retention period—

☒ Server default (30 days)
☐ Custom
☐ Do not save to the database

Restore Defaults Apply

Cancel Apply and Close

Fig. 1: General Page

Moreover, we can use any table column for configuraiton (in the example in the sentence above, we used the “.1” column, as you rightly understood), that returns non-empty results in MIB Explorer, as they will be used to make the SNMP walk to get all the instances.

As for the columns — each of those you’d like to monitor should then be added to the *Table Columns* property page.

In our case they could be:

1. Add index column 1.3.6.1.2.1.2.2.1.1
2. Add description 1.3.6.1.2.1.2.2.1.2
3. Add Physical address 1.3.6.1.2.1.2.2.1.6
4. Add MTU 1.3.6.1.2.1.2.2.1.4...

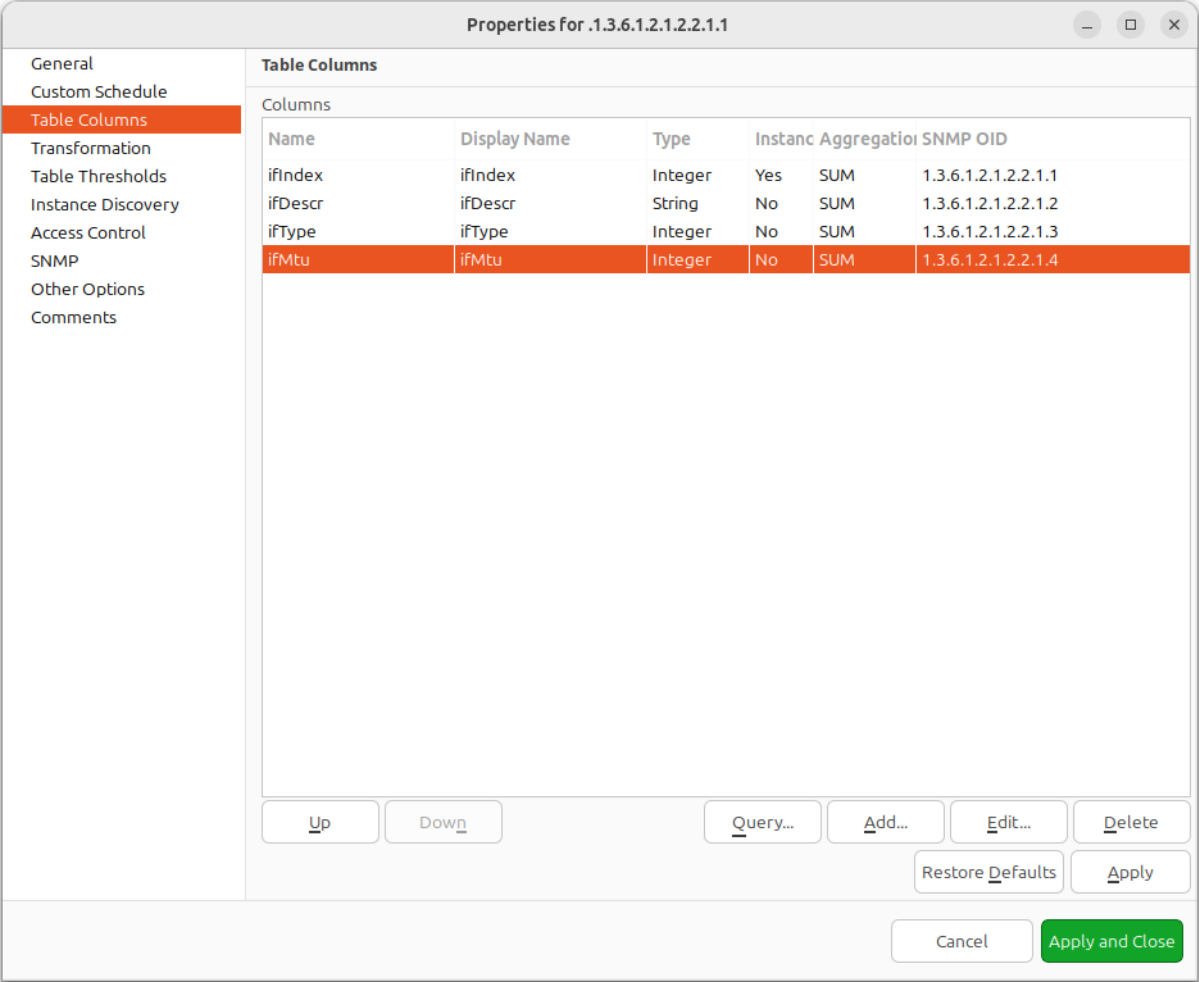


Fig. 2: Table Columns configuration

Another option to add columns is to click *Query...* button. Automatic table columns query is done by SNMP Walk on Metric OID where column part is cut out.

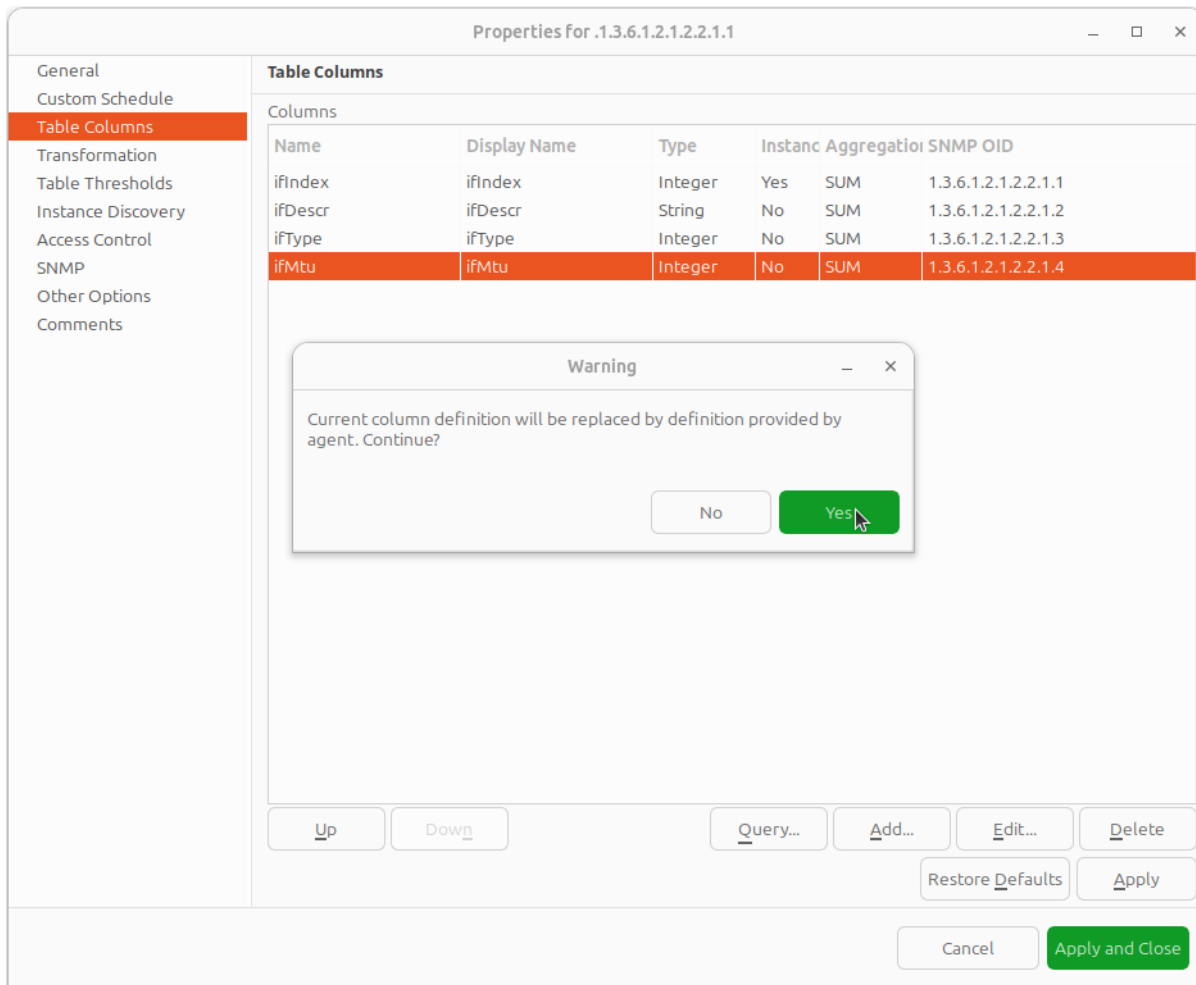
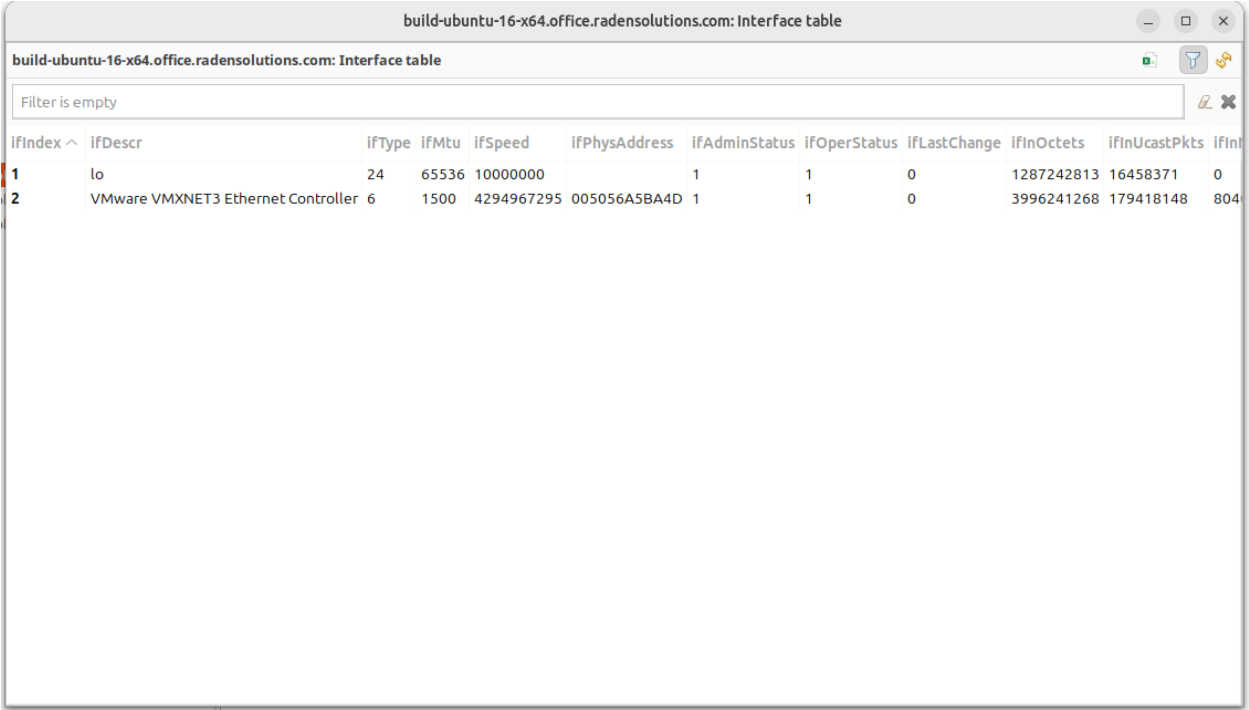


Fig. 3: Query warning



build-ubuntu-16-x64.office.radensolutions.com: Interface table

Filter is empty

ifIndex ^	ifDescr	ifType	ifMtu	ifSpeed	ifPhysAddress	ifAdminStatus	ifOperStatus	ifLastChange	ifInOctets	ifInUcastPkts	ifIn...
1	lo	24	65536	10000000		1	1	0	1287242813	16458371	0
2	VMware VMXNET3 Ethernet Controller	6	1500	4294967295	005056A5BA4D	1	1	0	3996241268	179418148	804

Fig. 4: Configured table

1.3.6.1.2.1.4.35.1	.4	.5	.6	.7
.2.1.4.10.5.5.1	00 23 7D 5F 27 BB	428943151	3	1
.2.1.4.10.5.5.20	00 50 56 A5 3D 86	428943151	3	1

We can see in the table above that the instance OID can also be a string of multiple numbers with dots. In the case of a physical address map instance, OID part will contain IP address.

OID	OID as text	Type	Value	Raw value
1.3.6.1.2.1.4.35.1.4.2.1.4.10.5.30	ipNetToPhysicalPhysAddress	STRING	?PV?b?	00 50 56 A5 62 11
1.3.6.1.2.1.4.35.1.4.2.1.4.10.5.37	ipNetToPhysicalPhysAddress	STRING	?PV??-	00 50 56 A5 AF 2D
1.3.6.1.2.1.4.35.1.4.2.1.4.10.5.111	ipNetToPhysicalPhysAddress	STRING	?PV???	00 50 56 A5 93 AD
1.3.6.1.2.1.4.35.1.4.2.1.4.10.5.7.1	ipNetToPhysicalPhysAddress	STRING	?#?_?	00 23 7D 5F 27 BB
1.3.6.1.2.1.4.35.1.4.2.2.16.32.1.4.112.223.51.0.5.0.0.0.0.0.0.1	ipNetToPhysicalPhysAddress	STRING	?#?_?	00 23 7D 5F 27 BB
1.3.6.1.2.1.4.35.1.4.2.2.16.254.128.0.0.0.0.0.0.2.35.125.255.254.95.39.187	ipNetToPhysicalPhysAddress	STRING	?#?_?	00 23 7D 5F 27 BB
1.3.6.1.2.1.4.35.1.5.2.1.4.10.5.5.1	ipNetToPhysicalLastUpdated	TIMETICKS	1124810588	5C 3F 0B 43
1.3.6.1.2.1.4.35.1.5.2.1.4.10.5.5.20	ipNetToPhysicalLastUpdated	TIMETICKS	1124810588	5C 3F 0B 43
1.3.6.1.2.1.4.35.1.5.2.1.4.10.5.5.25	ipNetToPhysicalLastUpdated	TIMETICKS	1124810588	5C 3F 0B 43
1.3.6.1.2.1.4.35.1.5.2.1.4.10.5.5.27	ipNetToPhysicalLastUpdated	TIMETICKS	1124810588	5C 3F 0B 43
1.3.6.1.2.1.4.35.1.5.2.1.4.10.5.5.30	ipNetToPhysicalLastUpdated	TIMETICKS	1124810588	5C 3F 0B 43
1.3.6.1.2.1.4.35.1.5.2.1.4.10.5.5.37	ipNetToPhysicalLastUpdated	TIMETICKS	1124810588	5C 3F 0B 43
1.3.6.1.2.1.4.35.1.5.2.1.4.10.5.5.111	ipNetToPhysicalLastUpdated	TIMETICKS	1124810588	5C 3F 0B 43
1.3.6.1.2.1.4.35.1.5.2.1.4.10.5.7.1	ipNetToPhysicalLastUpdated	TIMETICKS	1124810588	5C 3F 0B 43
1.3.6.1.2.1.4.35.1.5.2.2.16.32.1.4.112.223.51.0.5.0.0.0.0.0.0.1	ipNetToPhysicalLastUpdated	TIMETICKS	1124810588	5C 3F 0B 43
1.3.6.1.2.1.4.35.1.5.2.2.16.254.128.0.0.0.0.0.0.2.35.125.255.254.95.39.187	ipNetToPhysicalLastUpdated	TIMETICKS	1124810588	5C 3F 0B 43
1.3.6.1.2.1.4.35.1.6.2.1.4.10.5.5.1	ipNetToPhysicalType	INTEGER	3	03 00 00 00
1.3.6.1.2.1.4.35.1.6.2.1.4.10.5.5.20	ipNetToPhysicalType	INTEGER	3	03 00 00 00
1.3.6.1.2.1.4.35.1.6.2.1.4.10.5.5.25	ipNetToPhysicalType	INTEGER	3	03 00 00 00
1.3.6.1.2.1.4.35.1.6.2.1.4.10.5.5.27	ipNetToPhysicalType	INTEGER	3	03 00 00 00
1.3.6.1.2.1.4.35.1.6.2.1.4.10.5.5.30	ipNetToPhysicalType	INTEGER	3	03 00 00 00
1.3.6.1.2.1.4.35.1.6.2.1.4.10.5.5.37	ipNetToPhysicalType	INTEGER	3	03 00 00 00
1.3.6.1.2.1.4.35.1.6.2.1.4.10.5.5.111	ipNetToPhysicalType	INTEGER	3	03 00 00 00
1.3.6.1.2.1.4.35.1.6.2.1.4.10.5.7.1	ipNetToPhysicalType	INTEGER	3	03 00 00 00
1.3.6.1.2.1.4.35.1.6.2.2.16.32.1.4.112.223.51.0.5.0.0.0.0.0.0.1	ipNetToPhysicalType	INTEGER	3	03 00 00 00
1.3.6.1.2.1.4.35.1.6.2.2.16.254.128.0.0.0.0.0.0.2.35.125.255.254.95.39.187	ipNetToPhysicalType	INTEGER	3	03 00 00 00
1.3.6.1.2.1.4.35.1.7.2.1.4.10.5.5.1	ipNetToPhysicalState	INTEGER	2	02 00 00 00
1.3.6.1.2.1.4.35.1.7.2.1.4.10.5.5.20	ipNetToPhysicalState	INTEGER	1	01 00 00 00
1.3.6.1.2.1.4.35.1.7.2.1.4.10.5.5.25	ipNetToPhysicalState	INTEGER	2	02 00 00 00
1.3.6.1.2.1.4.35.1.7.2.1.4.10.5.5.27	ipNetToPhysicalState	INTEGER	1	01 00 00 00
1.3.6.1.2.1.4.35.1.7.2.1.4.10.5.5.30	ipNetToPhysicalState	INTEGER	2	02 00 00 00
1.3.6.1.2.1.4.35.1.7.2.1.4.10.5.5.37	ipNetToPhysicalState	INTEGER	2	02 00 00 00
1.3.6.1.2.1.4.35.1.7.2.1.4.10.5.5.111	ipNetToPhysicalState	INTEGER	2	02 00 00 00
1.3.6.1.2.1.4.35.1.7.2.1.4.10.5.7.1	ipNetToPhysicalState	INTEGER	2	02 00 00 00
1.3.6.1.2.1.4.35.1.7.2.2.16.32.1.4.112.223.51.0.5.0.0.0.0.0.0.1	ipNetToPhysicalState	INTEGER	2	02 00 00 00
1.3.6.1.2.1.4.35.1.7.2.2.16.254.128.0.0.0.0.0.0.2.35.125.255.254.95.39.187	ipNetToPhysicalState	INTEGER	2	02 00 00 00
1.3.6.1.2.1.4.35.1.8.2.1.4.10.5.5.1	ipNetToPhysicalRowStatus	INTEGER	1	01 00 00 00
1.3.6.1.2.1.4.35.1.8.2.1.4.10.5.5.20	ipNetToPhysicalRowStatus	INTEGER	1	01 00 00 00

Fig. 5: Physical Address MIB Explorer

Another difference with the first example can be determined by executing the SNMP walk for the table above. The device returns values only for the columns with the OIDs “.4”, “.5”, “.6”, “.7”, “.8”.

If we do walk for the “1.3.6.1.2.1.4.35.1.1” table column, it will return us empty result. This also should be taken into consideration when we create a table with physical addresses - only columns that return indexes can be used for the Metric field in the DCI Table creation property page.

8.9.2 Table Thresholds and Instance Columns

When setting up table thresholds, it's helpful to understand instance columns. An instance column is similar to a primary key in a database — it's the unique ID. In NetXMS, this is known as an instance- or key column. It is possible to set multiple columns as instance columns, similar to composite keys in databases. However, if instance columns aren't defined, and rows change order between polling periods, it can trigger false threshold alerts. The system might register that a different row is exceeding a threshold when, in fact, the same data is present, just in a different row. Specifying an instance column can mitigate this confusion.

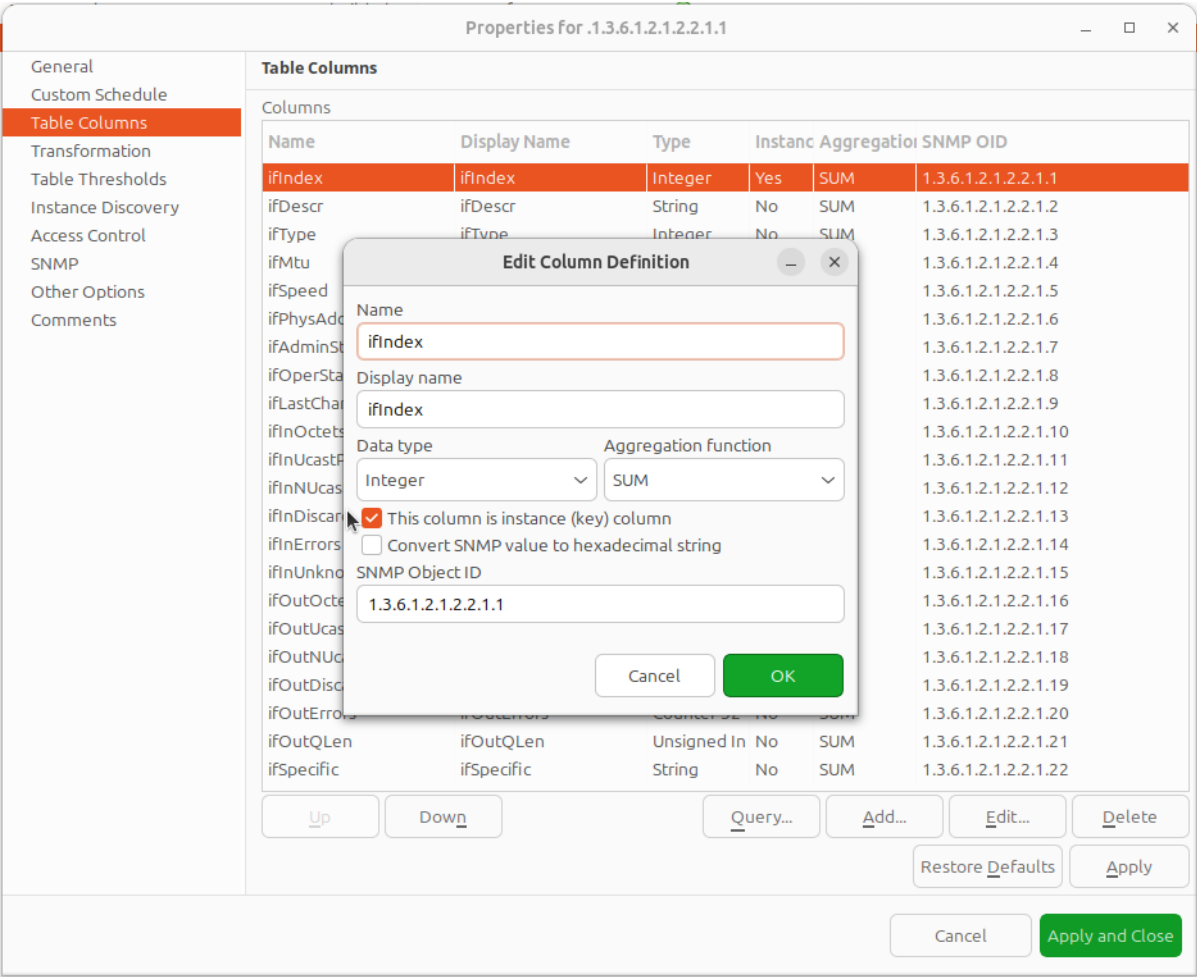


Fig. 6: Table columns configuration — editing column definition

As you see, the NetXMS table metrics are a powerful tool for collecting and managing a wealth of network data. While they can be more complex to set up and require more storage than single with similar content, they present a great possibility to view more complex sets of data.

8.9.3 Configuration example

In order to show how table metrics are configured in NetXMS, and how to distinguish what each part of it represents, we will go to the MIB explorer and use one of the tables in the system.

The screenshot shows the MIB Explorer application window. The title bar reads "MIB Explorer - build-ubuntu-16-x64.office.radensolutions.com". The main window is divided into three main sections:

- Left Pane (Tree View):** Displays a hierarchy of MIB objects. The path "inetAddressMIB > interfaces > ifTable > ifEntry > ifIndex" is selected, and "ifIndex" is highlighted in orange.
- Right Pane (Details):**
 - Object identifier (OID):** A text field containing "1.3.6.1.2.1.2.2.1.1".
 - OID as text:** A text field containing "iso.org.dod.internet.mgmt.mib-2.interfaces.ifTable.ifEntry.ifIndex".
 - Type:** A dropdown menu showing "Integer 32bits".
 - Status:** A dropdown menu showing "Current".
 - Access:** A dropdown menu showing "Read/Write".
 - Index:** A text field.
 - Description:** A text area containing: "A unique value, greater than zero, for each interface. It is recommended that values are assigned contiguously starting from 1. The value for each interface sub-layer must remain constant at least from one re-initialization of the entity's network management system to the next re-initialization."
 - Textual Convention:** A text area containing: "A unique value, greater than zero, for each interface or interface sub-layer in the managed system. It is recommended that values are assigned contiguously starting from 1. The value for each interface sub-layer must remain constant at least from one re-initialization of the entity's network management system to the next re-initialization."
- Bottom Pane (Table):** A table with the following columns: "OID", "OID as text", "Type", "Value", and "Raw value". It contains two rows of data:

OID	OID as text	Type	Value	Raw value
1.3.6.1.2.1.2.2.1.1.1	ifIndex	INTEGER	1	01 00 00 00
1.3.6.1.2.1.2.2.1.1.2	ifIndex	INTEGER	2	02 00 00 00

In this picture we can see the table OID “1.3.6.1.2.1.2.2.1”. After the “1.3.6.1.2.1.2.2.1” goes “.1”, that represents the column OID. So in OID search field we have “1.3.6.1.2.1.2.2.1.1” — the table column OID. And as a result of the MIB walk for the given OID we get 2 instances “1.3.6.1.2.1.2.2.1.1.1” and “1.3.6.1.2.1.2.2.1.1.2”.

The screenshot shows the MIB Explorer application window. The left pane displays a tree view of MIB objects. The right pane shows details for the selected object, including its OID, type, status, access, and description. The bottom pane shows a table of instances for the selected OID.

MIB Explorer - build-ubuntu-16-x64.office.radensolutions.com

Object identifier (OID)
1.3.6.1.2.1.2.2.1.2

OID as text
iso.org.dod.internet.mgmt.mib-2.interfaces.ifTable.ifEntry.ifDescr

Type Octet String **Status** Current **Access** Read/Write

Description
A textual string containing information about the interface. This string should include the name of the manufacturer, the product name and the version of the interface hardware/software.

Textual Convention
Represents textual information taken from the NVT ASCII character set, as defined in pages 4, 10-11 of RFC 854. To summarize RFC 854, the NVT ASCII repertoire specifies:
- the use of character codes 0-127 (decimal)
- the graphics characters (32-126) are interpreted as US ASCII
- NUL, LF, CR, BEL, BS, HT, VT and FF have the special meanings specified in RFC 854.

Filter: Filter is empty

OID	OID as text	Type	Value	Raw value
1.3.6.1.2.1.2.2.1.2.1	ifDescr	STRING	lo	6C 6F
1.3.6.1.2.1.2.2.1.2.2	ifDescr	STRING	VMware VMXNET3 Ethernet Controller	56 4D 77 61 72 65 20 56 4D 58 4E

We can make the MIB walk for another table column “1.3.6.1.2.1.2.2.1.2” and get the same two instances, just for another column: “1.3.6.1.2.1.2.2.1.2.1” and “1.3.6.1.2.1.2.2.1.2.2”. In this way we know now, that the table base id is “1.3.6.1.2.1.2.2.1.2”.

To configure this table we can use any table column, that via a MIB walk will return the instances like: “1.3.6.1.2.1.2.2.1.1” or “1.3.6.1.2.1.2.2.1.2”. Let’s use “1.3.6.1.2.1.2.2.1.1”.

The screenshot shows a window titled "Properties for .1.3.6.1.2.1.2.2.1.1". On the left is a sidebar with a list of tabs: General (selected), Custom Schedule, Table Columns, Transformation, Table Thresholds, Instance Discovery, Access Control, SNMP, Other Options, and Comments. The main area is titled "General" and contains the following fields and options:

- Metric to collect:**
 - Origin:** A dropdown menu showing "SNMP".
 - Source node override:** A text field showing "None" with edit and delete icons.
- Metric:** A text field showing ".1.3.6.1.2.1.2.2.1.1" with an edit icon.
- Display name:** A text field showing "Interface table" with a red border.
- Collection schedule:**
 - ☒ Server default interval (60 seconds)
 - ☐ Custom interval
 - ☐ Advanced schedule
- History retention period:**
 - ☒ Server default (30 days)
 - ☐ Custom
 - ☐ Do not save to the database

At the bottom right of the main area are two buttons: "Restore Defaults" and "Apply". At the bottom of the window are three buttons: "Cancel", "Apply and Close" (highlighted in green), and "Apply".

Fig. 7: General Page

Press *Apply and Close* button to apply changes and open configuration again (To update DCI configuration). Then let's go to the Table Column configuration property page and do query. It will add all the columns to the table list.

Properties for .1.3.6.1.2.1.2.2.1.1

General
Custom Schedule
Table Columns
Transformation
Table Thresholds
Instance Discovery
Access Control
SNMP
Other Options
Comments

Table Columns

Columns

Name	Display Name	Type	Instance	Aggregation	SNMP OID
ifIndex	ifIndex	Integer	No	SUM	1.3.6.1.2.1.2.2.1.1
ifDescr	ifDescr	String	No	SUM	1.3.6.1.2.1.2.2.1.2
ifType	ifType	Integer	No	SUM	1.3.6.1.2.1.2.2.1.3
ifMtu	ifMtu	Integer	No	SUM	1.3.6.1.2.1.2.2.1.4
ifSpeed	ifSpeed	Unsigned In	No	SUM	1.3.6.1.2.1.2.2.1.5
ifPhysAddress	ifPhysAddress	String	No	SUM	1.3.6.1.2.1.2.2.1.6
ifAdminStatus	ifAdminStatus	Integer	No	SUM	1.3.6.1.2.1.2.2.1.7
ifOperStatus	ifOperStatus	Integer	No	SUM	1.3.6.1.2.1.2.2.1.8
ifLastChange	ifLastChange	Unsigned In	No	SUM	1.3.6.1.2.1.2.2.1.9
ifInOctets	ifInOctets	Counter 32-	No	SUM	1.3.6.1.2.1.2.2.1.10
ifInUcastPkts	ifInUcastPkts	Counter 32-	No	SUM	1.3.6.1.2.1.2.2.1.11
ifInNUcastPkts	ifInNUcastPkts	Counter 32-	No	SUM	1.3.6.1.2.1.2.2.1.12
ifInDiscards	ifInDiscards	Counter 32-	No	SUM	1.3.6.1.2.1.2.2.1.13
ifInErrors	ifInErrors	Counter 32-	No	SUM	1.3.6.1.2.1.2.2.1.14
ifInUnknownProtos	ifInUnknownProtos	Counter 32-	No	SUM	1.3.6.1.2.1.2.2.1.15
ifOutOctets	ifOutOctets	Counter 32-	No	SUM	1.3.6.1.2.1.2.2.1.16
ifOutUcastPkts	ifOutUcastPkts	Counter 32-	No	SUM	1.3.6.1.2.1.2.2.1.17
ifOutNUcastPkts	ifOutNUcastPkts	Counter 32-	No	SUM	1.3.6.1.2.1.2.2.1.18
ifOutDiscards	ifOutDiscards	Counter 32-	No	SUM	1.3.6.1.2.1.2.2.1.19
ifOutErrors	ifOutErrors	Counter 32-	No	SUM	1.3.6.1.2.1.2.2.1.20
ifOutQLen	ifOutQLen	Unsigned In	No	SUM	1.3.6.1.2.1.2.2.1.21
ifSpecific	ifSpecific	String	No	SUM	1.3.6.1.2.1.2.2.1.22

Up Down Query... Add... Edit... Delete

Restore Defaults Apply

Cancel Apply and Close

Fig. 8: The query result of the table columns

Now we have table with all the columns. Columns can be renamed by a user afterwards, as necessary. What we are missing here is an instance column. Our instance column will be the ifIndex column.

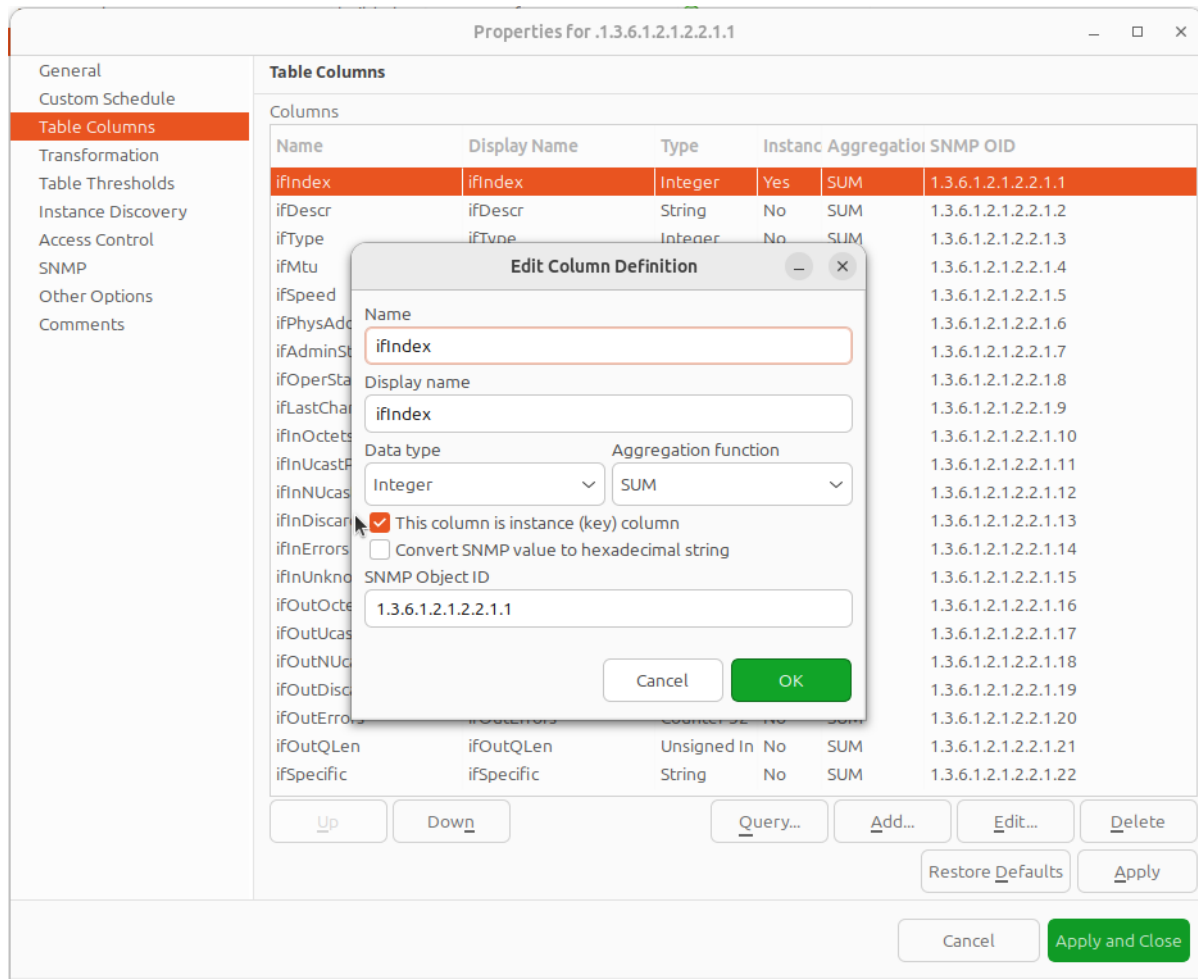


Fig. 9: Table column configuration — renaming columns and editing their definition

As a result we will get the table below:

As we can see, the column ipPhysAddress shows nonsense. The column contains the hexadecimal string, but we try showing it as a regular string.

Let's go back to the table configuration and adjust it by setting "Convert SNMP value to hexadecimal string" option for a column.

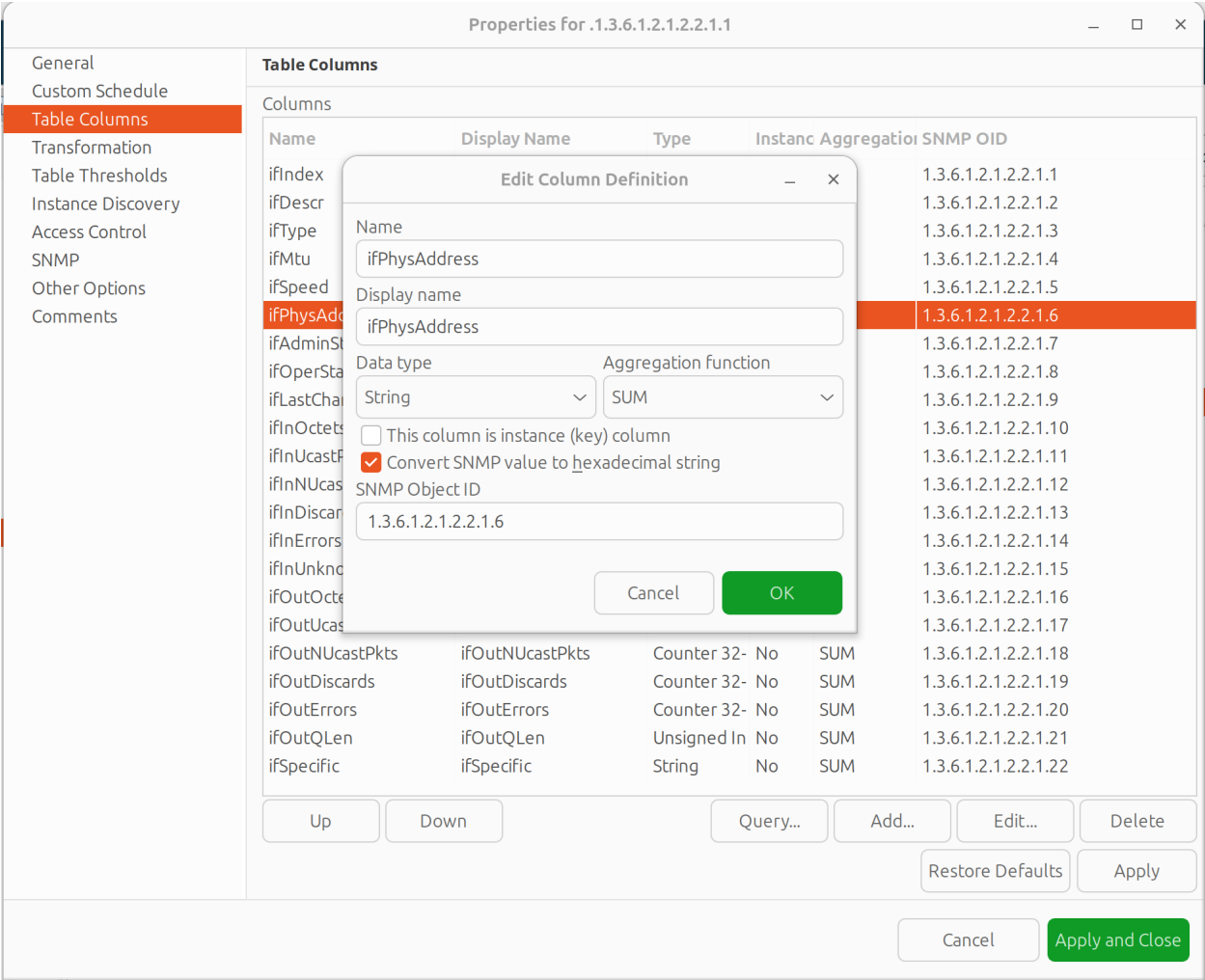


Fig. 10: Table column configuration — renaming columns and editing their definition

You can also adjust some column names for more clarity.

Properties for .1.3.6.1.2.1.2.2.1.1

General
Custom Schedule
Table Columns
Transformation
Table Thresholds
Instance Discovery
Access Control
SNMP
Other Options
Comments

Table Columns

Columns

Name	Display Name	Type	Instance	Aggregation	SNMP OID
ifIndex	Index	Integer	Yes	SUM	1.3.6.1.2.1.2.2.1.1
ifDescr	Description	String	No	SUM	1.3.6.1.2.1.2.2.1.2
ifType	Type	Integer	No	SUM	1.3.6.1.2.1.2.2.1.3
ifMtu	Mtu	Integer	No	SUM	1.3.6.1.2.1.2.2.1.4
ifSpeed	Speed	Unsigned In	No	SUM	1.3.6.1.2.1.2.2.1.5
ifPhysAddress	Physical Address	String	No	SUM	1.3.6.1.2.1.2.2.1.6
ifAdminStatus	Administrative State	Integer	No	SUM	1.3.6.1.2.1.2.2.1.7
ifOperStatus	Operational State	Integer	No	SUM	1.3.6.1.2.1.2.2.1.8
ifLastChange	Last Change	Unsigned In	No	SUM	1.3.6.1.2.1.2.2.1.9
ifInOctets	Incomming Octets	Counter 32-	No	SUM	1.3.6.1.2.1.2.2.1.10
ifInUcastPkts	ifInUcastPkts	Counter 32-	No	SUM	1.3.6.1.2.1.2.2.1.11
ifInNUcastPkts	ifInNUcastPkts	Counter 32-	No	SUM	1.3.6.1.2.1.2.2.1.12
ifInDiscards	ifInDiscards	Counter 32-	No	SUM	1.3.6.1.2.1.2.2.1.13
ifInErrors	ifInErrors	Counter 32-	No	SUM	1.3.6.1.2.1.2.2.1.14
ifInUnknownProtos	ifInUnknownProtos	Counter 32-	No	SUM	1.3.6.1.2.1.2.2.1.15
ifOutOctets	ifOutOctets	Counter 32-	No	SUM	1.3.6.1.2.1.2.2.1.16
ifOutUcastPkts	ifOutUcastPkts	Counter 32-	No	SUM	1.3.6.1.2.1.2.2.1.17
ifOutNUcastPkts	ifOutNUcastPkts	Counter 32-	No	SUM	1.3.6.1.2.1.2.2.1.18
ifOutDiscards	ifOutDiscards	Counter 32-	No	SUM	1.3.6.1.2.1.2.2.1.19
ifOutErrors	ifOutErrors	Counter 32-	No	SUM	1.3.6.1.2.1.2.2.1.20
ifOutQLen	ifOutQLen	Unsigned In	No	SUM	1.3.6.1.2.1.2.2.1.21
ifSpecific	ifSpecific	String	No	SUM	1.3.6.1.2.1.2.2.1.22

Up Down Query... Add... Edit... Delete

Restore Defaults Apply

Cancel Apply and Close

The end result will look like the table below:

build-ubuntu-16-x64.office.radensolutions.com: Interface table

Filter is empty

ifIndex ^	ifDescr	ifType	ifMtu	ifSpeed	ifPhysAddress	ifAdminStatus	ifOperStatus	ifLastChange	ifInOctets	ifInUcastPkts	ifIn...
1	lo	24	65536	10000000		1	1	0	1287242813	16458371	0
2	VMware VMXNET3 Ethernet Controller	6	1500	4294967295	005056A5BA4D	1	1	0	3996241268	179418148	804

8.9.4 Additional tips

If two tables share the same instances, they can be shown in one table — as a process table for ESX:

Properties for .1.3.6.1.2.1.25.4.2.1.4

General
Cluster Options
Custom Schedule
Table Columns
Transformation
Table Thresholds
Instance Discovery
Access Control
SNMP
Other Options
Comments

Table Columns

This DCI was added by template "ESX"
All local changes can be overwritten at any moment

Columns

Name	Display Name	Type	Instance	Aggregation	SNMP OID
Index	Index	Integer	Yes	SUM	.1.3.6.1.2.1.25.4.2.1.1
Name	Name	String	No	SUM	.1.3.6.1.2.1.25.4.2.1.2
Path	Path	String	No	SUM	.1.3.6.1.2.1.25.4.2.1.4
Parameters	Parameters	String	No	SUM	.1.3.6.1.2.1.25.4.2.1.5
Type	Type	Integer	No	SUM	.1.3.6.1.2.1.25.4.2.1.6
Status	Status	Integer	No	SUM	.1.3.6.1.2.1.25.4.2.1.7
CPU Consumption	CPU Consumption	Integer	No	SUM	.1.3.6.1.2.1.25.5.1.1.1
Memory usage	Memory usage	Integer	No	SUM	.1.3.6.1.2.1.25.5.1.1.2

Up Down Query... Add... Edit... Delete

Restore Defaults Apply

Cancel Apply and Close

esx1: Process table							
Filter is empty							
Index	Name	Path	Parameters	Type	Status	CPU Consumption	Memory usage
2097622	init	/bin/init		4	2	2	2336
2097676	python	/bin/python	/usr/lib/vmware/vmsyslog/bin/vmsyslogd.pyc -i	4	2	0	12712
2097677	python	/bin/python	/usr/lib/vmware/vmsyslog/bin/vmsyslogd.pyc -i	4	2	117417	18016
2097688	sh	/bin/sh	/sbin/watchdog.sh -s vobd /usr/lib/vmware/vob/bin/vobd	4	2	0	804
2097698	vobd	/usr/lib/vmware/vob/bin/vobd		4	2	4	14412
2097726	sh	/bin/sh	/sbin/watchdog.sh -s vmkeventd -u 10 -q 5 -t 9999999 /usr/l	4	2	0	804
2097733	vmkeventd	/usr/lib/vmware/vmkeventd/bin/vmkeventd		4	2	11	3852
2097903	vmkdevmgr	/bin/vmkdevmgr		4	2	125	6152
2098061	sh	/bin/sh	/sbin/watchdog.sh -s net-lacp -u 1000 -q 100 -t 100 /usr/sbin	4	2	0	804
2098071	net-lacp	/usr/sbin/net-lacp		4	2	52556	3408
2098110	imaShim32d	/usr/sbin/imaShim32d		4	2	8378	2508
2098139	vmkiscsid	/usr/sbin/vmkiscsid		4	2	68852	3360
2098634	busybox	/usr/lib/vmware/busybox/bin/busybox	crond	4	2	4965	840
2098742	busybox	/usr/lib/vmware/busybox/bin/busybox	inetd /var/run/inetd.conf	4	2	1089	840
2098878	sh	/bin/sh	/sbin/watchdog.sh -t 100 -s ntpd /sbin/ntpd -g -n -c /etc/ntp.	4	2	0	804
2098888	ntpd	/sbin/ntpd	-g -n -c /etc/ntp.conf -f /etc/ntp.drift	4	2	41740	1436
2098935	sh	/bin/sh	/sbin/watchdog.sh -s usbarbitrator -t 5 /usr/lib/vmware/bin/	4	2	0	804
2098945	vmware-usbarbitrator	/usr/lib/vmware/bin/vmware-usbarbitrator	-t --max-clients=542	4	2	28579	3604
2098982	sh	/bin/sh	/sbin/watchdog.sh -s iofiltervpd /usr/lib/vmware/iofilter/bir	4	2	0	804
2098992	ioFilterVPServer	/usr/lib/vmware/iofilter/bin/ioFilterVPServer		4	2	647863	9100
2099027	sh	/bin/sh	/sbin/watchdog.sh -s swapobjd /usr/lib/vmware/swapobj/bii	4	2	0	420
2099037	swapobjd	/usr/lib/vmware/swapobj/bin/swapobjd		4	2	18486	3748
2099151	sh	/bin/sh	/sbin/watchdog.sh -s hostdCgiServer hostdCgiServer	4	2	0	804
2099161	hostdCgiServer	hostdCgiServer		4	2	10	15784
2099206	sh	/bin/sh	/sbin/watchdog.sh -s hostd -e LD_PRELOAD=/lib64/libMallc	4	2	0	804
2099216	hostd	hostd	/etc/vmware/hostd/config.xml	4	2	28355	110932
2099227	sh	/bin/sh	/sbin/watchdog.sh -s rhttpproxy rhttpproxy ++min=0,swapsi	4	2	0	804
2099237	rhttpproxy	rhttpproxy	-r /etc/vmware/rhttpproxy/config.xml	4	2	1356	13076

USER MANAGEMENT

9.1 Introduction

NetXMS has its own user database. All NetXMS user accounts are stored in the backend SQL database. Each account has its own unique login name and identifier. The account may also have a password.

9.2 Terms and Definitions

9.2.1 Users

NetXMS has the following attributes for users:

- Unique identifier
- Unique login name
- Full name
- Email
- Phone number
- Description
- System Access Rights configuration
- Authentication method configuration
- TOTP configuration
- Password
- Certificate

Not all attributes are mandatory.

Superuser

NetXMS has a built-in superuser account with ID 0, which always has full access to the system. The default login name for the superuser account is `system`. By default this account is disabled. The superuser account can be renamed or disabled/enabled, but cannot be deleted.

The system user can be used to correct access rights to objects that exists, but to which no other users have access to.

9.2.2 Groups

Each user can be a member of several groups. Groups are the preferred way to organize access permissions. You should always grant permission to groups instead of using individual users. That way you will get a much shorter access control list which is easier to handle. Access rights from multiple groups are summarized to calculate effective user access rights.

Other groups can also be added as group members, in this case, the user access rights will be calculated by summarizing the access rights from all the groups in the path to the user.

Everyone Group

NetXMS has a built-in virtual group called *Everyone*. This group always contains all users in the system. It cannot be deleted, and its member list cannot be edited.

9.2.3 System Access Rights

NetXMS has two types of access rights: system access rights as described in this chapter and *object access rights*.

System access rights used to grant access to system-wide configuration (like *Event processing*) and functions (like agent registration).

The following system access rights can be granted:

Access Right	Description
Access server console	Allow user to access the server debug console. <i>Server debug console</i>
Configure event templates	Allow user to add, edit and delete event templates. <i>Event processing</i>
Configure object tools	Allow user to configure object tools. <i>Object Tools</i>
Configure server actions	Allow user to configure server actions. <i>Event processing</i>
Configure SNMP traps	Allow user to configure SNMP trap mapping.
Control user sessions	Allow user to see active user sessions and forcefully terminate them. (Not yet implemented)
Edit event processing policy	Allow user to edit Event Processing Policy. <i>Event processing</i>
Edit server configuration variables	Allow user to edit server configuration variables.
External tool integration account	Allow external software user authentication using NetXMS user accounts via <i>Web API/Rest API</i> .
Import configuration	Allow user to import configuration from file. Dashboard import is not restricted by this access right.
Initiate TCP proxy sessions	Allow to use functionality that allows to forward TCP connections inside the connection between NetXMS server and agent.
Login as mobile device	Allows user to login via mobile application.
Manage agent configurations	Allow user to create, edit and delete agent configurations stored on the server. <i>Agent configuration options from server</i>
Manage all scheduled tasks	Allow user to create, edit and delete all <i>Scheduled tasks</i> , including system ones.
Manage DCI summary table	Allows user to manage DCI summary table. <i>Summary table</i>
Manage geographical areas	Allows user to manage geographical areas
Manage image library	Allows user to manage image library. <i>Image library</i>
Manage mapping tables	Allows user to create, edit and delete mapping tables.
Manage object categories	Allows user to create, edit and delete object categories.
Manage object queries	Allows user to create, edit and delete saved object queries.
Manage own scheduled tasks	Allow user to create new and modify <i>Scheduled tasks</i> created by the user.

continues on next page

Table 1 – continued from previous page

Access Right	Description
Manage packages	Allow user to install, remove, and deploy server agent packages. <i>Centralized agent upgrade</i>
Manage persistent storage	Allows user to create, edit and delete persistent storage records
Manage script library	Allows user to add, edit, rename and delete scripts in script library.
Manage server files	Allow user to upload files to server and delete files stored on server. <i>Server File Management</i>
Manage SSH keys	Allows user to generate, import, edit and delete SSH keys.
Manage two-factor authentication methods	Allows user to configure system-wide two-factor authentication settings.
Manage user support application notifications	Allows to send, list and delete notifications that are being sent via user support application.
Manage user scheduled tasks	Allow user to create, edit and delete user-created <i>Scheduled tasks</i> (not system scheduled tasks).
Manage users	Allow user to manage user accounts. Please note that user having this access right granted can modify own account to get any other system right granted.
Manage web service definitions	Allow user to manage system-wide definitions of web services.
Read server files	Allow user to read files stored on the server and upload to agents (user still needs appropriate object rights for upload). <i>Server File Management</i>
Manage agent tunnels	Allow user to list, bind and unbind agent tunnels.
Reporting server access	Allow user to execute report generation, view generated reports, schedule report generation. <i>Reporting</i>
Schedule file upload	Allow user to schedule server file upload to an agent. <i>Scheduled tasks</i>
Schedule object maintenance	Allow user to schedule maintenance for an object. <i>Scheduled tasks</i>
Schedule script execution	Allow user to schedule script execution. <i>Scheduled tasks</i>
Send notifications	Allow user to send manual notifications via NetXMS server.
Unlink helpdesk tickets	Allow user to unlink alarms from external helpdesk system <i>Integration with external HelpDesk</i> .
View all alarm categories	Allow user to view all alarms generated by Event Processing Policy rules. If this is off, user will only see alarms for categories he/she has access to.
View audit log	Allow user to view audit log.
View event log	Allow user to view event log, alarm log.
View event templates configuration	Allow user to view configured event templates.
View SNMP trap log	Allow user to view SNMP trap log.
View syslog	Allow user to view syslog.

By granting the *View all alarms* access right, the user (or members of the group) will have access to view all generated alarms. Should it be required to configure alarm viewing access for specific users or groups, please refer to *Alarm Category Configurator*.

9.2.4 UI Access Rules

UI access rules allow to hide specific UI elements from user. This does not securely blocks access - hiding is only implemented in NetXMS Management Client, so e.g. nxshell is not affected by UI access rules.

UI access rules are stored in textual format, one UI element per line. UI elements have `category:name` format, * GLOB wildcard can be used to match multiple elements. E.g. `perspective:objects.maps` refers to Maps perspective, `perspective:*` refers to all perspectives, `view:objects.fdb` is FDB view (tab) on an object and * means all UI elements.

Adding UI element means that it should be included. Adding ! prefix means exclusion. ^ prefix means priority inclusion.

Rules are checked in the following order, until a matching rule is found:

1. Priority inclusion rules (rules with ^ prefix). If a rule is matched, UI element is enabled.
2. Exclusion rules (rules with ! prefix). If rule is matched, UI element is disabled.
3. Inclusion rules (without any prefix). If a rule is matched, UI element is enabled.
4. If no matching rules found, UI element is disabled.

Default configuration has * inclusion rule for user `Everyone` and `Admins` groups, thus enabling all UI elements. Based on that exclusion rules can be added, or it's possible to remove * rule and configure specific set of inclusion and, if needed, exclusion rules.

9.3 User Authentication

9.3.1 Internal Password

This is the default method for user authentication. The password provided by the user when authenticating is compared against the password stored in the NetXMS database.

Password Policy

Various restrictions can be put on internal passwords to force users to choose stronger passwords. The following server configuration variables controls password policy:

Variable	Description	Default
MinPasswordLength	Default minimum password length for a NetXMS user. The default applies only if a per-user setting is not defined.	0
PasswordComplexity	Required password complexity. See table below for details.	0
PasswordExpiration	Password expiration time in days. If set to 0, password expiration is disabled. This variable has no effect on users with the <i>Password never expires</i> flag set.	0
PasswordHistoryLength	Number of previous passwords to keep. Users are not allowed to set password if it matches one from their previous passwords list.	0

Possible flags for PasswordComplexity:

Value	Description
1	Password must contain digits
2	Password must contain uppercase letters
4	Password must contain lowercase letters
8	Password must contain special characters
16	Forbid alphabetical sequences (a password is considered invalid if it contains an alphabetical sequence of 3 or more letters of the same case).
32	Forbid keyboard sequences (a password is considered invalid if it contains a sequence of 3 or more characters that are located on keyboard next to each other, like <code>ASDF</code>).

Complexity flags can be combined to get the desired restrictions. For example, to force passwords to contain uppercase and lowercase letters, PasswordComplexity variable must be set to 6 (2 + 4).

Changes to these configuration variables become effective immediately and do not require an NetXMS server restart.

9.3.2 RADIUS

If *RADIUS* authentication method is selected, the password provided by the user is sent to a RADIUS server for validation. The user is granted access if the RADIUS server responds with `Access-Accept`. Communication between NetXMS server and RADIUS server is controlled by the following server configuration variables:

Variable	Description	Default value
RADIUS.AuthMethod	RADIUS authentication method to be used (PAP, CHAP, MS-CHAPv1, MS-CHAPv2).	PAP
RADIUS.NASIdentifier	Value for NAS-Identifier attribute in RADIUS request (will not be sent if empty)	none
RADIUS.NumRetries	The number of retries for RADIUS authentication.	5
RADIUS.Port	Port number used for connection to primary RADIUS server.	1645
RADIUS.SecondaryPort	Port number used for connection to secondary RADIUS server.	1645
RADIUS.SecondarySecret	Shared secret used for communication with secondary RADIUS server.	netxms
RADIUS.SecondaryServer	Host name or IP address of secondary RADIUS server.	none
RADIUS.Secret	Shared secret used for communication with primary RADIUS server.	netxms
RADIUS.Server	Host name or IP address of primary RADIUS server.	none
RADIUS.ServiceType	Value for Service-Type attribute in RADIUS request. Value of 0 will exclude service type from request attributes.	8
RADIUS.Timeout	Timeout in seconds for requests to RADIUS server	3

Changes to these configuration variables become effective immediately and do not require an NetXMS server restart.

9.3.3 Certificate Authentication

This type of authentication can be selected manually in user preferences.

Login process using a certificate works as follows:

1. The server sends a random challenge to the client
2. The client signs the servers challenge with their certificates' private key and send a signed challenge along with the public part of their certificate to the server
3. The server validates the certificate using its CA certificate
4. If the certificate is valid, the server validates the challenge signature using the certificates' public key
5. If the signature is valid, the server compares the certificate subject with mapping data from the user record
6. If the mapping data matches with the certificate subject, access is granted

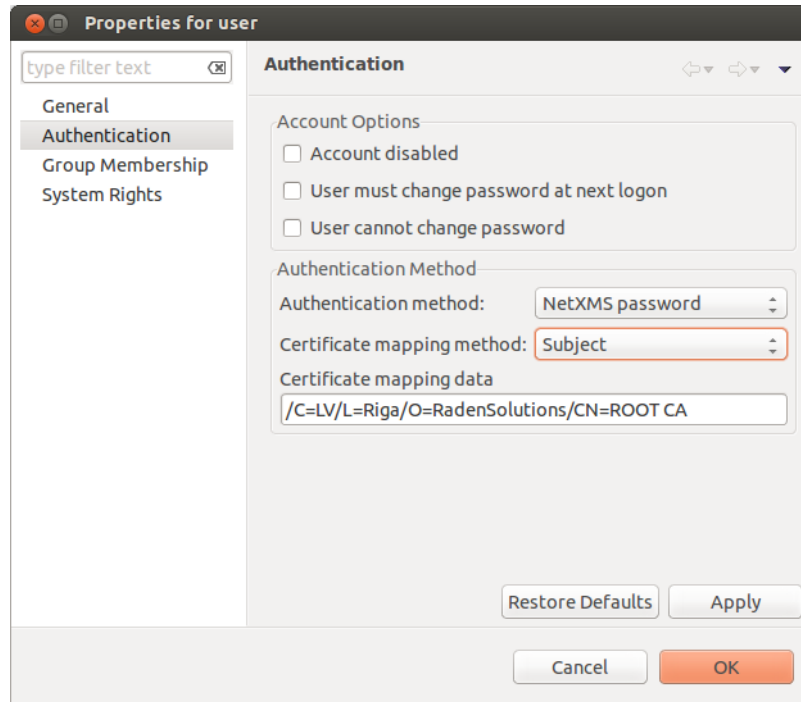
So, to login successfully, the user must posses a valid certificate with a private key. Authentication by certificate also allows smart card login - you just need to store the certificate used for login on a smart card instead of in a local certificate store.

Certificate management

CA certificates are looked up in the list configured by the "TrustedCertificate" configuration parameter in the server configuration file.

Link certificate and user

In the “User Manager” view select the user properties for the required user. Then go to the “Authentication” section.



In the “Authentication Method” section: “Certificate”, “Certificate or Password”, “Certificate or RADIUS”.

The next two fields in combination:

Certificate mapping method: “Subject”

Certificate mapping data: the subject of the CA.

Certificate mapping method: “Public key”

Certificate mapping data: the public key of the certificate

Certificate mapping method: “Common name”

Certificate mapping data: if no mapping data is set, then the linking certificate CN = user name, otherwise CN = mapping data

9.3.4 CAS authentication

Central Authentication Service (CAS) single sign-on is supported in the web interface only. The following server configuration parameters control CAS operation: CAS.AllowedProxies, CAS.Host, CAS.Port, CAS.Service, CAS.TrustedCACert, CAS.ValidateURL. See [Server configuration parameters](#) for the explanation of the meaning of the mentioned parameters.

Changes to these configuration variables become effective immediately and do not require a NetXMS server restart.

9.3.5 Two-factor authentication

In addition to the above authentication methods, two-factor authentication using **TOTP** or via a notification channel can be set up.

TOTP configuration is done in two places - in system-wide *Two-factor authentication methods* and in properties of specific users.

First of all it is necessary to configure a method in *Two-factor authentication methods*. For TOTP, select the driver name *TOTP*. No driver configuration is necessary. For using a notification channel, select the driver name *Message* and in driver configuration the name of notification channel should be specified, e.g.:

```
ChannelName=NotificationChannelName
```

The second step is to add the two-factor authentication method in properties of a user.

For message method it is necessary to specify the recipient for the message. This concludes the configuration - on login the user will receive a message with numeric code.

For the TOTP method no additional configuration is necessary. On the following login the user will be presented with a dialog containing a qr code and a secret as text. After entering the secret into the users TOTP application, it will generate a numeric code that should be entered to confirm TOTP initialization.

To repeat initialization it is possible to perform a reset for the TOTP method in the user properties. After that, on next login of the user the dialog with qr code and secret will be presented again.

It is possible to specify several two-factor authentication methods. In this case the user will be presented with a menu on login, allowing to choose which method to use.

9.4 Integration with LDAP

NetXMS can perform one-way synchronization of users and groups with an external LDAP server. The user list replica is refreshed automatically.

Already existing NetXMS users or groups will not be modified during initial synchronization (e.g. user “admin” or group “Everyone”).

9.4.1 LDAP synchronization configuration

Server parameters controlling LDAP synchronization:

Variable	Description	Default value
LdapConnection-String *	Comma- or whitespace-separated list of URIs in a format <i>schema://host:port</i> . Supported schemas: <i>ldap://</i> , <i>ldaps://</i> (LDAP over TLS), <i>ldapi://</i> (LDAP over IPC), and <i>cldap://</i> (connectionless LDAP). Windows specific: for servers based on Windows this parameter should be set according to these rules: empty string attempts to find the “default” LDAP server), a domain name, or a space-separated list of host names or dotted strings that represent the IP address of hosts running an LDAP server to which to connect. Each host name in the list can include an optional port number which is separated from the host itself with a colon (:). Note: most LDAP implementations except recent versions of OpenLDAP do not support mixed schema types in the single connection string.	<i>ldap://localhost:389</i>
LdapSyncUser *	User login for LDAP synchronization	
LdapSyncUserPassword *	User password for LDAP synchronization	
LdapSearchBase	The LdapSearchBase configuration parameter is the DN of the entry at which to start the search.	
LdapSearchFilter *	The LdapSearchFilter is a string representation of the filter to apply in the search.	
LdapUserDeleteAction *	This parameter specifies what should be done while synchronization with users deleted from the LDAP user/group. 0 - if user should be deleted from NetXMS DB. 1 - if the user should be disabled but kept in the database. If 1 is chosen, then on LDAP sync the user will be disabled and its description will be changed to “LDAP entry was deleted.” Afterwards this user/group can be detached from LDAP and enabled or deleted manually.	1
LdapUserMapping-Name *	The name of the attribute which value will be used as a users’ login name	
LdapGroupMappingName *	The name of the attribute which value will be used as a group’s identifier	
LdapMappingFull-Name	The name of the attribute which value will be used as the user full name	
LdapMappingDescription	The name of the attribute which value will be used as a user description	
LdapGroupClass	The object class which represents group objects. If the found entry is not of a user or group class, it will be simply ignored.	
LdapUserClass *	The object class that represents user objects. If the found entry is not of a user or group class, it will be simply ignored.	
Ldap-GroupUniqueId	Unique identifier for the LDAP group object. By default LDAP groups are identified by DN. If in your configuration the DN can be changed at any time it is useful to choose another attribute as a unique group identifier.	
LdapUserUniqueId	Unique identifier for the LDAP user object. By default LDAP users are identified by DN. If in your configuration the DN can be changed at any time it is useful to choose another attribute as a unique user identifier.	
LdapSyncInterval *	This parameter is for setting a synchronization interval in minutes between the NetXMS server and the LDAP server. If the synchronization parameter is set to 0 the synchronization will not be done.	0
LdapPageSize *	Limit of records that can be returned in one search page.	1000

* Required fields

Synchronization also can be done manually with *ldapsync* or the *ldap* command in the server debug console.

9.4.2 LDAP users/groups relationships with native NetXMS users/groups

LDAP users and groups are handled in exactly the same way as users from the internal database. The only difference is that for LDAP group membership is refreshed at each synchronisation and any non-LDAP user then will be removed from the group.

9.4.3 Login with help of LDAP user

The login process is completely transparent for the user - their user name should match the attribute set by *LdapMapping-Name* and their password should be the current LDAP password for that user.

9.4.4 LDAP configuration debugging

If users are not synchronized, the reason can be found by running *ldapsync* manually or by the *ldap* command in the server debug console on debug lever 4.

Log when LDAP sync passed correctly:

```
[11-Sep-2014 16:28:08.352] [DEBUG] LDAPConnection::initLDAP(): Connecting to LDAP
→server
[11-Sep-2014 16:28:08.353] [DEBUG] LDAPConnection::syncUsers(): Found entry count: 3
[11-Sep-2014 16:28:08.354] [DEBUG] LDAPConnection::syncUsers(): Found dn: CN=Users,
→CN=Customers,DC=Northwind,DC=Extranet
[11-Sep-2014 16:28:08.354] [DEBUG] LDAPConnection::syncUsers(): CN=Users,CN=Customers,
→DC=Northwind,DC=Extranet is not a user nor a group
[11-Sep-2014 16:28:08.354] [DEBUG] LDAPConnection::syncUsers(): Found dn: CN=zev333,
→CN=Users,CN=Customers,DC=Northwind,DC=Extranet
[11-Sep-2014 16:28:08.354] [DEBUG] LDAPConnection::syncUsers(): User added: dn:
→CN=zev333,CN=Users,CN=Customers,DC=Northwind,DC=Extranet, login name: zev333, full
→name: (null), description: (null)
[11-Sep-2014 16:28:08.354] [DEBUG] LDAPConnection::syncUsers(): Found dn: CN=user,
→CN=Users,CN=Customers,DC=Northwind,DC=Extranet
[11-Sep-2014 16:28:08.354] [DEBUG] LDAPConnection::syncUsers(): User added: dn:
→CN=user,CN=Users,CN=Customers,DC=Northwind,DC=Extranet, login name: user, full
→name: (null), description: (null)
[11-Sep-2014 16:28:08.354] [DEBUG] LDAPConnection::closeLDAPConnection(): Disconnect
→from ldap.
[11-Sep-2014 16:28:08.354] [DEBUG] UpdateLDAPUsers(): User added: dn: CN=zev333,
→CN=Users,CN=Customers,DC=Northwind,DC=Extranet, login name: zev333, full name:
→(null), description: (null)
[11-Sep-2014 16:28:08.354] [DEBUG] UpdateLDAPUsers(): User added: dn: CN=user,
→CN=Users,CN=Customers,DC=Northwind,DC=Extranet, login name: user, full name: (null),
→description: (null)
[11-Sep-2014 16:28:08.354] [DEBUG] RemoveDeletedLDAPEntry(): Ldap uid=john,ou=People,
→dc=nodomain entry was removed from DB.
[11-Sep-2014 16:28:08.354] [DEBUG] RemoveDeletedLDAPEntry(): Ldap uid=zev,ou=People,
→dc=nodomain entry was removed from DB.
[11-Sep-2014 16:28:08.354] [DEBUG] RemoveDeletedLDAPEntry(): Ldap uid=kasio,ou=People,
→dc=nodomain entry was removed from DB.
[11-Sep-2014 16:28:08.355] [DEBUG] RemoveDeletedLDAPEntry(): Ldap uid=usr1,ou=People,
→dc=nodomain entry was removed from DB.
```

Login credentials incorrect:

```
[11-Sep-2014 15:49:39.892] [DEBUG] LDAPConnection::initLDAP(): Connecting to LDAP_
↪server
[11-Sep-2014 15:49:39.896] [DEBUG] LDAPConnection::loginLDAP(): LDAP could not login.
↪Error code: Invalid credentials
[11-Sep-2014 15:49:39.896] [DEBUG] LDAPConnection::syncUsers(): Could not login.
```

Search base is set incorrectly or sync user does not have access:

```
[11-Sep-2014 15:54:03.138] [DEBUG] LDAPConnection::initLDAP(): Connecting to LDAP_
↪server
[11-Sep-2014 15:54:03.140] [DEBUG] LDAPConnection::syncUsers(): LDAP could not get_
↪search results. Error code: No such object
```

9.4.5 LDAP configuration examples

Active Directory

Variable	Value
LdapConnectionString	ldap://10.5.0.35:389
LdapSyncUser	CN=user,CN=Users,CN=Customers,DC=Domain,DC=Extranet
LdapSyncUserPass-word	xxxxxxx
LdapSearchBase	CN=Customers,DC=Domain,DC=Extranet
LdapSearchFilter	(objectClass=*)
LdapUserDeleteAction	1
LdapMappingName	sAMAccountName
LdapMappingFull-Name	displayName
LdapMappingDescrip-tion	description
LdapGroupClass	group
LdapUserClass	user
LdapGroupUniqueId	objectGUID
LdapUserUniqueId	objectGUID
LdapSyncInterval	1440

OpenLDAP

Variable	Value
LdapConnectionString	ldap://10.5.0.35:389
LdapSyncUser	cn=admin,dc=nodomain
LdapSyncUserPass-word	xxxxxxx
LdapSearchBase	dc=nodomain
LdapSearchFilter	(objectClass=*)
LdapUserDeleteAction	1
LdapMappingName	cn
LdapMappingFull-Name	displayName
LdapMappingDescrip-tion	description
LdapGroupClass	groupOfNames
LdapUserClass	inetOrgPerson
LdapGroupUniqueId	entryUUID
LdapUserUniqueId	entryUUID
LdapSyncInterval	1440

9.5 Managing User Accounts

All NetXMS user accounts can be managed from the *User Manager* view available at *Configuration ► User Manager* in NetXMS Management Client. Only users with granted system right *Manage users* can access *User Manager*.

- To create a new user account, select *Create new user* from the view menu or context menu.
- To create a new group, select *Create new group* from the view menu or context menu.
- To delete user account, select it in the list, right-click, and select *Delete* from pop-up menu. You can delete multiple accounts at a time.
- To modify properties of a user or group, select it in the list, right-click, and select *Properties* from the pop-up menu.
- To reset the password of a user, select the user account in the list, right-click, and select *Change password* from the pop-up menu.

9.6 Audit

All important user actions are written to the audit log. There are two audit logging modes: internal and external. Internal audit logging is on by default and writes audit records into a table in the NetXMS database. External audit logging allows sending audit records to an external system via the syslog protocol. External audit logging is off by default. Audit logging is controlled by the following server configuration variables:

Variable	Description	Default value
AuditLogRetention-Time	Retention time in days for the records in the internal audit log. All records older than specified will be deleted by the housekeeping process.	90
EnableAuditLog	Enable (1) or disable (0) audit logging.	1
ExternalAuditFacility	Syslog facility to be used in audit log records sent to external server.	13
ExternalAuditPort	UDP port of the external syslog server to send audit records to.	514
ExternalAuditServer	External syslog server to send audit records to. If set to none, external audit logging is disabled.	none
ExternalAuditSeverity	Syslog severity to be used in audit log records sent to the external server.	5
ExternalAuditTag	Syslog tag to be used in audit log records sent to the external server.	netxmsd-audit

OBJECT MANAGEMENT

10.1 Object browser

Object browser is a view in *Management Client*. It presents all existing *objects* as a hierarchical structure. Overall description of objects can be found in concepts part: *Objects*.

10.1.1 Object browser options

Object browser has a number of options that define how object tree is displayed.

Object browser has following options:

- Show filter `CTRL+F2`, that shows search line that has special syntaxes for search. Syntaxes description can be found there: *Filters*.
- Show status indicator `CTRL+F3`
- Hide unmanaged objects
- Hide check templates. This option will not show *Business Services* templates.

10.1.2 Filters

By default search is done by node name. In this type of search can be used '*' and '?' symbols for pattern search.

But there are few prefix that can be used for other search options:

- '/' - will search in comments
- '>' - will search by IP address

10.2 Objects

Detailed information about objects, it's usage, parents and children can be found in concept chapter, *Objects*. In this section will be described only actions and properties that can be applied on different object classes.

10.2.1 Subnet

Property pages:

Except common properties subnets has *Map Appearance* and *Trusted Nodes* tabs. *Map Appearance* tab defines images that will be used to display this object on a *Network Map* and drill-down object (object that will be opened when double click on this object on *Network Map*). *Trusted Nodes* is used to define object list that have access to this object from the script.

Menu items:

Full subnet can be managed or unmanaged. Management status will be applied to all subnet node. If subnet is deleted and is the only parent of a node, then node also will be deleted with the subnet. *Upload file* menu item will upload file from server to all nodes that have agent and have access to upload directory.

Under *Tools* menu are available predefined object tools that will be executed on each subnet node. More about object tool configuration can be found there: [Object Tools](#).

Execute server script will open [execute server script view](#) where arbitrary script can be executed. *Alarms* menu item will open view with all subnet nodes' alarms. And *802.1x port state* will open table with port authentication states, that can be exported to CSV.

10.2.2 Node

Property pages:

Except common properties node has *Communications* tab that is responsible for communication options with this node (like host name, agent proxy and authentication, SNMP proxy and authentication and ICMP proxy), *Polling* tab is responsible for disabling polls for specific node, *Location* is used to configure location of the node, *Map Appearance* tab defines images that will be used to display this object on a [Network Map](#) and drill-down object (object that will be opened when double click on this object on [Network Map](#)).

Menu items:

Usually interfaces for nodes are created automatically by Configuration poll results, but they can be also created manually with help of menu item *Create interface...* *This interface is a physical port* is used just for information purposes.

Information about service monitoring and *Create network service...* menu item can be found there: [Network Service Monitoring](#).

When node is unmanaged/managed - all it's children like interfaces and service monitoring are also unmanaged/managed. In unmanaged state [metrics](#) are not collected and no polls are scheduled.

Node can be deleted from NetXMS by *Delete* menu item. Node is not deleted synchronously, but it is scheduled node deletion. While node deletion all data about this node is also collected (like metrics).

If zones are enabled, then zone can be changed using *Change zone...* item. *File manager* will open agent file manager view. By default this view will be empty, to configure it refer to [Agent file management](#) chapter. *Upload file* can be used to upload file from server to node. This action can be applied simultaneously to all nodes.

Take screenshot for now halfway implemented functionality. For now screenshot can be taken only from Windows machines.

Remote control option will appear for nodes where [VNC](#) install is detected. In order to take advantage of this feature, one should add EnableTCPProxy = yes in agent configuration on remote node followed by agent restart. Run Configuration

Poll on the node you want to VNC to. Target VNC may require loopback connection to be enabled as well as firewall settings adjusted. In cases when there is no agent installed on remote node, but VNC is present, we can use agent on NetXMS server or agent serving as zone proxy. In this scenario, one would need to add `EnableTCPProxy = yes` in agent configuration on server or on agent that acts like proxy for zone. Your NetXMS user should have “Initiate TCP proxy sessions” system access right. In addition, in object tree user should have “Control” access rights to that node.

Description of *Edit agent's configuration* functionality can be found in [Edit configuration file remotely](#) chapter.

Poll options:

Poll Name	Description
Status	
Configuration	
Configuration (full)	
Instance discovery	
Instance names	
Topology	

Under *Tools* menu are available predefined object tools that will be executed on selected node. More about object tool configuration can be found there: [Object Tools](#).

Execute server script will open [execute server script view](#). Where arbitrary script can be executed. Node can be accessed with `$node` variable.

MIB Explorer will open [MIB explorer view](#). If geolocation of the node is set, then with help of *Geolocation* item can be opened map with shown on it object location. *Software Inventory* will show full software list for nodes with Windows systems or Linux systems(that used rpm or deb packages) and have NetXMS agent installed. *Service Dependency* will build tree from this node with all container where this node is included. *Alarms* will open alarm view with alarms only for this specific node.

Find switch port will open view with log of searches of switch port to which a node is connected. During search the interfaces will be checked one by one and first successful result will be shown.

802.1x port state will open table with port authentication states, that can be exported to CSV.

Topology menu item contains all options of predefined network maps for this node and some other options:

Routing table IP route from... will build network map with route from selected node to node that was selected in Object selector window. *IP route to...* will build network map with route to selected node from node that was selected in Object selector window. *IP Neighbors* will show all IP neighbors of this node.

Switch forwarding database(MAC address table) VLANs Layer 2 Topology

Radio interface Wireless stations

Last values will open [Last Values view](#). *Data Collection Configuration* will open [Data Collection Configuration view](#), that is used to configure collected *metrics* from node.

10.2.3 Rack

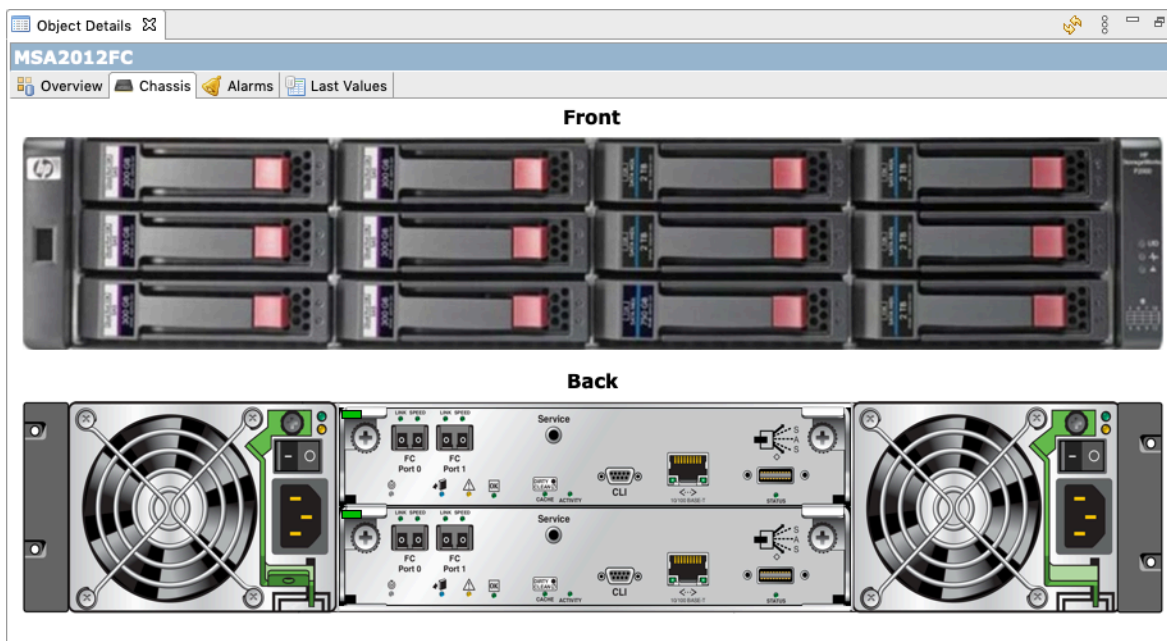
Rack is an object that visualizes server room organization in NetXMS. Node and chassis objects can be assigned to a rack in node properties, specifying position in the rack, height (number of occupied rack units), orientation (does it occupy full depth of the rack, or only present on front or back side of the rack). Front and/or rear images can be selected from [Image library](#).

Rack visualization is available in Object Detail -> Rack view. Left click on a rack unit display a pop-up with brief information about the node or chassis. Right click will display node or chassis context menu. Double click on a chassis will open Chassis View in a separate tab.

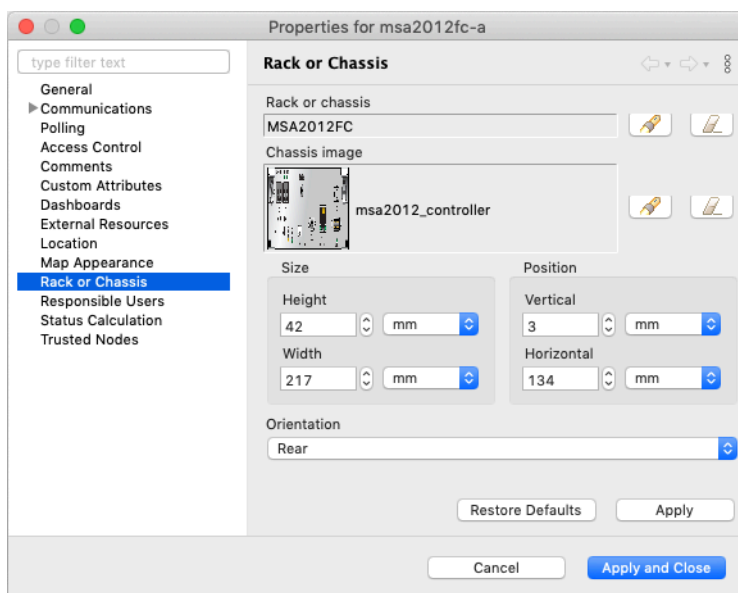
Status of rack units is denoted with color rectangle on the left edge of the rack.

10.2.4 Chassis

Chassis is an object visualizing a rack-mount chassis that have plug-in modules. Chassis visualization is available in Object Detail -> Chassis view.



Each node that represents chassis module can have an image that will be displayed atop of chassis image. Status of each node is denoted with color rectangle in the upper left corner of its image. Left click on node will display a pop-up with brief information about the node. Right click will display node context menu.



It is possible to configure the size of module's image and its position on chassis image. Vertical size and position could be specified in mm or rack units (RU), while horizontal - in mm or horizontal pitch units (HP). Size calculation assumes

that 1U chassis has 45mm height and 483mm width (including mounting brackets). Position (0, 0) is in the upper left corner.

You can use a graphic editor, e.g. Gimp to find position values in mm. Open chassis image in Gimp and set image width to 483 mm using Image -> Scale image. Now in the bottom left corner you can see current coordinates of mouse cursor in mm.

Chassis module images should be uploaded using Image Library [Image library](#).

10.2.5 Cluster

Is created to display nodes logical organization in cluster. Cluster nodes may have shared resources and networks, processes may move between nodes, so metric collection should be organized accordingly. Cluster object provides option to aggregate collected data from cluster nodes. More about data aggregation can be found there: [Data aggregation](#).

Besides default property pages cluster has also:

- *Cluster Resources* - there can be configured IP resources of the cluster. Further on *Cluster* view of *Object Details* will be shown current owner of resources
- *Cluster Networks*
- *Poling*
- *Dashboards* - there dashboard can be associated with object, so on right click associated dashboards will be displayed in the list
- *External Resources*
- *Location*
- *Map Appearance*
- *Trusted Nodes*

10.2.6 Interface

10.2.7 Network Service

10.2.8 VPN Connector

10.2.9 Condition

Conditions may represent more complicated status checks because each condition can have a script attached. Interval for evaluation of condition status is configured in Server Configuration Variables as ConditionPollingInterval with default value 60 seconds. Input values for the condition script can be set in object properties. Such values are accessible via \$1, \$2, ... variables inside the script. If the script returns 0, an activation event with the defined severity is created. If the script returns any other value, then a deactivation event is created.

Besides default property pages condition has also:

- *Events and Status*, were can be set activation and deactivation events, source of this objects and status of active and inactive condition.
- *Data*, were can be set DCI's that's data will be given to a script for condition status calculation.
- *Script* tab is used to write script that will calculate if condition should be activated or deactivated.
- *Map Appearance* tab defines images that will be used to display this object on a [Network Map](#) and drill-down object (object that will be opened when double click on this object on [Network Map](#)).
- ***Trusted Nodes* is used to define object list that**
have access to this object from the script.

Menu items:

Condition can be managed/unmanaged. If condition is unmanaged, evaluation of condition is not run. Condition can be deleted.

10.2.10 Container

Containers can be created in Infrastructure Services tree. Existing nodes and subnets can be added to containers by using Bind operation, and removed by using Unbind operation. New nodes, conditions, clusters, containers, mobile devices and racks can also be created. They can be created using required menu item of container under which this object should appear. Containers and nodes inside them can be moved by *Move to another container* menu item or using drag&drop.

Besides default property pages condition has also:

- *Automatic bind* about this functionality can be found [there](#)
- *Location* is used to configure location of the node
- *Map Appearance* tab defines images that will be used to display this object on a [Network Map](#) and drill-down object (object that will be opened when double click on this object on [Network Map](#)).
- ***Trusted Objects* is used to define object list that**
have access to this object from the script.

Menu items:

There are special menu item for each object that can be created in container. Objects like rack, container, mobile device, cluster are manually created objects. Node can be manually created or found by network discovery. In case if it is required to add already existing object to container use *Bind...* menu item. To remove node from container, but do not delete it use *Unbind...* menu item.

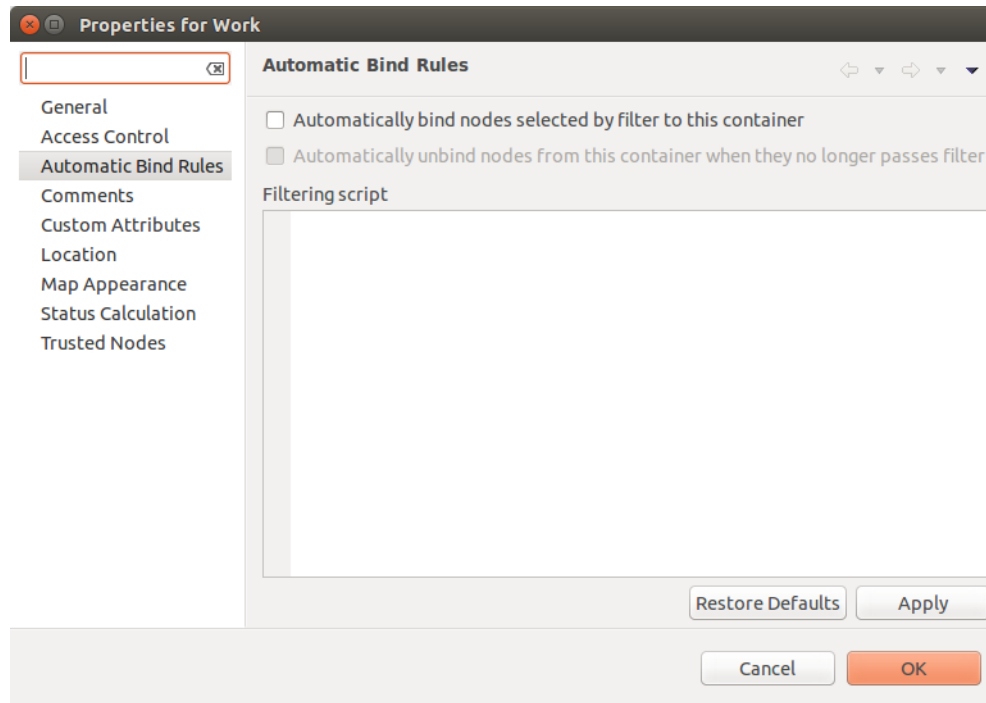
Using *Manage/Unmanage* all nodes will be managed/unmanaged under container. Container can be deleted. If deleted container was the only parent of an object, then this object will be also deleted. *Upload file...* will upload file from server to all nodes under container, same as each tool under *Tools* menu item will be executed on each node.

Execute script will open [execute server script view](#). Where an arbitrary script can be executed. *Geolocation* will show location of container on geographic map.

Logs will open alarm/event/trap view options with all active alarms for all children of this container.

Automatic bind option

For each container can be configured automatic binding rules. This can be done in *Automatic Bind Rules* tab of container properties.



Functionality would check and bind or unbind containers to nodes according to auto-bind script.

This script will be executed each configuration poll of each node.

10.2.11 Circuit

Circuits can be created in Infrastructure Services tree. Existing node interfaces can be added to circuit by using Bind operation, and removed by using Unbind. This object will generate events when state of underlying interface changes, and being an event source it will be able to have alarms on it. Reference of multiple interfaces will allow to use this object to represent different types of network services - multilink interfaces, links between sites, virtual circuits, etc. Circuits and interfaces inside them can be moved by *Move to another container* menu item or using drag&drop.

Besides default property pages circuit has also:

- *Automatic bind* functionality is described in more details [here](#)
- *Map Appearance* tab defines images that will be used to display this object on a *Network Map* and drill-down object (object that will be opened when double click on this object on *Network Map*).
- *Trusted Objects* is used to define object list that have access to this object from the script.

Menu items:

In case if it is required to add already existing interface to circuit use *Bind...* menu item. To remove nodeinterface from circuit, but do not delete it use *Unbind...* menu item.

Using *Manage/Unmanage* all interfaces will be managed/unmanaged under circuit.

Execute script will open *execute server script view*. Where an arbitrary script can be executed.

Logs will open alarm/event/trap view options with all active alarms for this circuit.

Automatic bind option

For each circuit one can configure automatic bind rules. It can be done in *Automatic Bind Rules* tab of circuit properties and it would check and bind or unbind circuit to interfaces according to auto-bind script.

Auto bind script will be executed while circuit auto bind is polled.

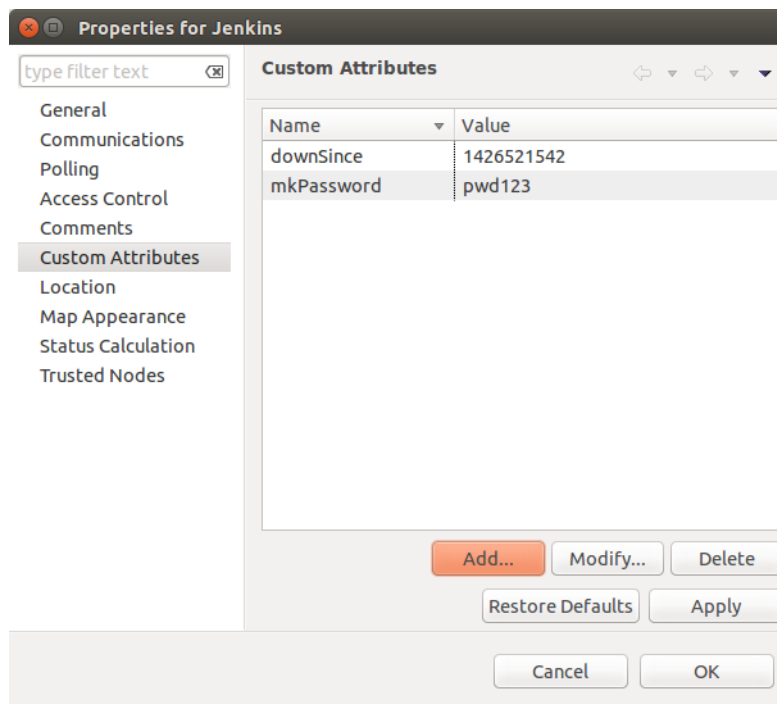
10.3 Common object properties

10.3.1 General

Each object has *General* tab in properties. There can be checked object class and ID, and changed object name. Each object has unique ID in the system. Object can be accessed by this ID.

10.3.2 Custom attributes

Every object can have custom attributes defined either by user or integrated application via NetXMS API. Custom attributes distinguished by names (an attribute name can contain up to 127 printable characters), and have string values of unlimited length. However, if you wish to access custom attributes in *NXSL* scripts as properties of node object, you should name them conforming to NXSL identifier naming constraints. To create or change value of custom attribute manually, right-click an object in NetXMS client, and select *Properties* ▶ *Custom Attributes* tab.



Custom attributes with name starting with \$ can be set from NXSL and read from NXSL (or macro), but never sent to management client and cannot be updated from management client. They can be used when it is required to store some information about node that should not be modified by users or seen by them.

Custom attributes with name starting with . are hidden, but can be seed and updated from management client if *Show hidden custom attributes* is enabled in it's properties.

10.3.3 Status calculation

Each object has it's own status calculation properties. Status of an object calculated based on:

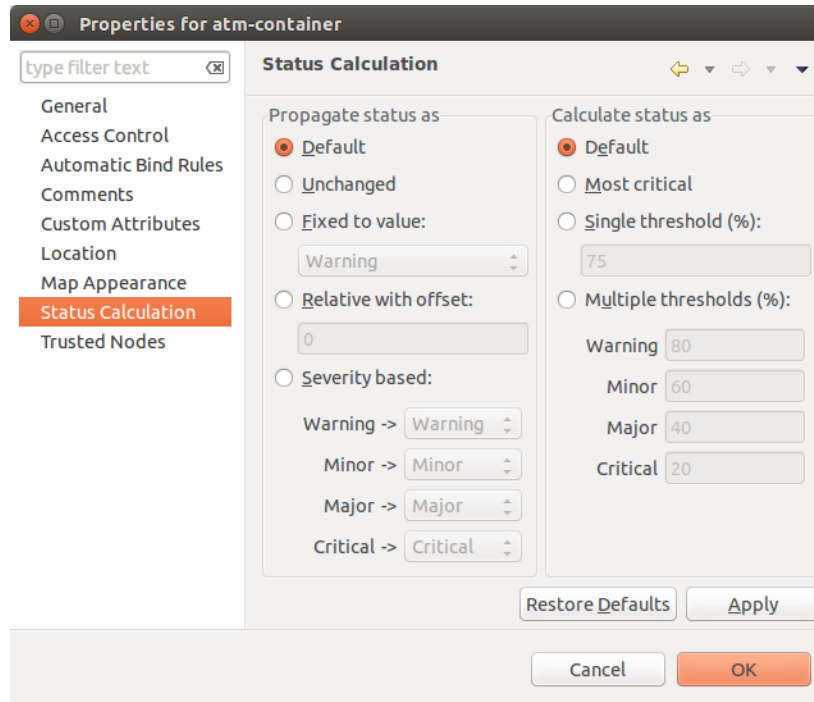
- Polling results
- Status of child objects (e.g. interfaces of node, nodes under container)

- Active alarms, associated with the object (after an alarm is resolved or terminated, it no longer affects object status)
- Value of status *DCIs* (DCI that has Use this DCI for node status calculation property enabled)

There are multiple options for status calculation that can be configured for specific objects or globally.

Status calculation has two configuration parts:

- status propagation - the way how status from object is pushed to upper objects;
- status calculation - the way how object is calculating it's status based on statuses propagated by children objects. Once child object status is calculated most critical status is taken from status of underlying objects, associated alarms and status *DCIs*.



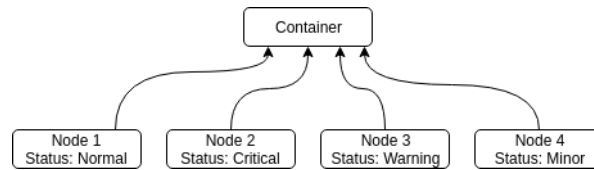
For status propagation the following options are available:

- Default - will take global configuration parameter (unchanged by default)
- Unchanged - will propagate status value without changes
- Fixed value: Normal, Warning, Minor, Major, Fixed - always will return fixed selected status
- Relative with offset - will add or remove some number for
- Severity based - will convert current status based on user configured status mapping table

For status calculation the following options are available:

- Default - will take global configuration parameter (most critical by default)
- Most critical - Most critical status will be taken
- Single threshold (%) - Percentage of objects that should be in status to change status of object
- Multiple thresholds - Same as previous but threshold is set for each status

Example of threshold status calculation



Statuses of nodes in table:

	Normal	Warning	Minor	Major	Critical
Node 1	1	0	0	0	0
Node 2	1	1	1	1	1
Node 3	1	1	0	0	0
Node 4	1	1	1	0	0

If “Single threshold (%)” option is selected and configuration is next:

- 75%

In this case status of container will be Warning, as 3/4 of nodes have Warning status or worse.

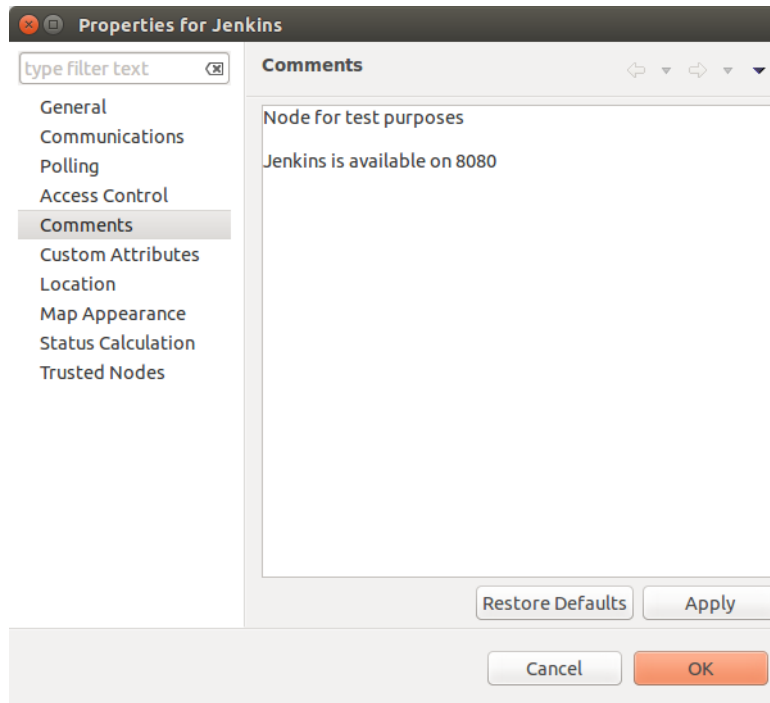
If “Multiple thresholds” is selected and configuration is next:

- Warning 80
- Minor 50
- Major 25
- Critical 35

In this case status of Container will be Major as bot thresholds for Minor and Major are reached and most critical from them is taken.

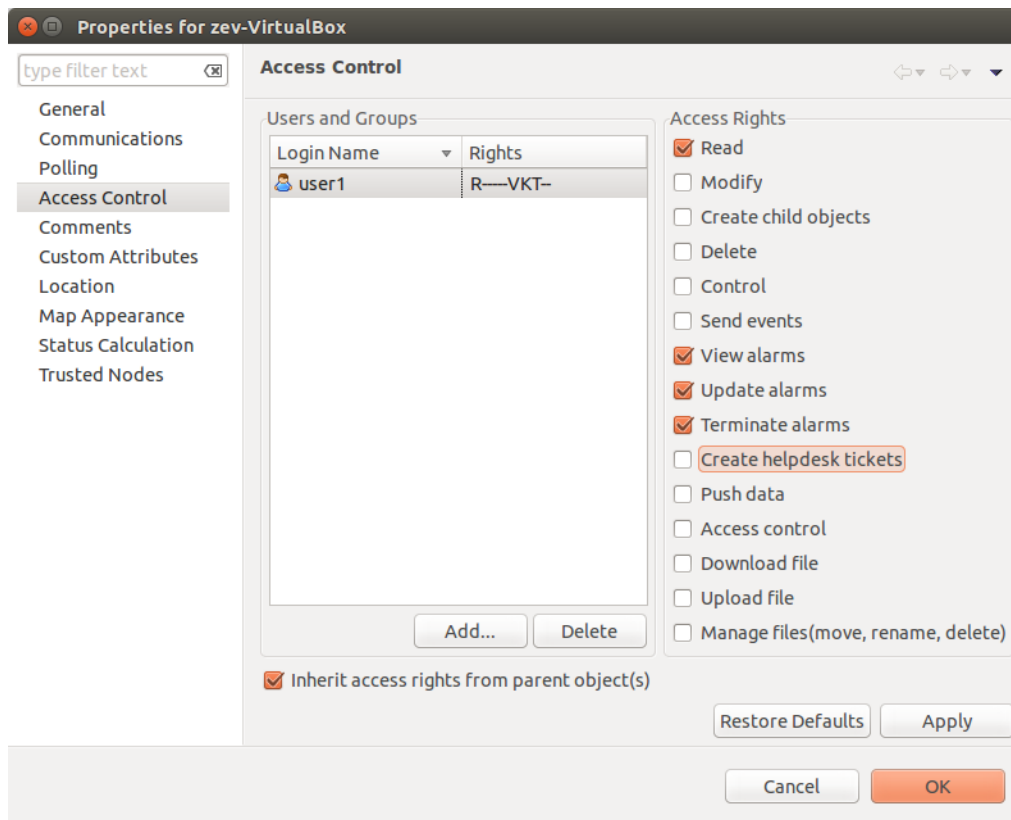
10.3.4 Comments

Each object in *Object Tree* can have comment. Comment can be set in Properties of the object. It is possible to use *macros for event processing* in the comments.



10.3.5 Access control

Object access rights controls access to NetXMS objects. Permissions given to an object inherited by all child objects, unless specifically blocked by turning off *Inherit access rights from parent object(s)* option in object's access control properties. Permissions given at different levels of the object tree summarize to form effective user rights for the object.



The following object access rights can be granted:

Access Right	Description
Read	View object in the tree and read it's information. For node objects, read access allows to view collected DCI data.
Read agent data	
Read SNMP data	
Modify	Modify object's properties (except access control).
Create child objects	Create child objects (or bind existing) under this object.
Delete	Delete this object.
Control	For node objects, execute object tools of type <i>Remote Command</i> .
Send events	Send events on behalf of this object.
View alarms	View alarms with this object as source.
Update alarms	Add comments to alarms, acknowledge alarms with this object as source.
Terminate alarms	Terminate alarms with this object as source.
Create helpdesk tickets	Create ticket in external helpdesk system
Push data	Push data for DCIs on this object.
Access control	Modify access control list for this object. Please note that user with this access right can grant any other access rights to own account.
Download files	Allow user to download files from this node (from paths defined by filemgr subagent settings in agent configuration file). This access right is also checked when downloading or tail of file is done from object tools.
Upload files	Allow user to upload files to this node (to paths defined by filemgr subagent settings in agent configuration file).
Manage files	Allow user to move, rename, delete files on this node (in paths defined by filemgr sub-agent settings in agent configuration file).
Control maintenance mode	
Take screenshot	Allow user to take screenshot of this node's screen (Windows only).

10.4 Object Details

Object details view provides main information about object. Each object has *Overview* tab that displays general information about object (like: ID, GUID, Class, and status of the object) and *Comments*.

10.4.1 Subnet

10.5 Object Tools

It is possible to create tools for execution on objects or alarms. Configured object tools are available under *Tools* in object browser's context menu or context menu of an alarm. A tool can run a command on NetXMS server or node, obtain data from SNMP or NetXMS agent, etc...

Object tools can be executed on Containers in object browser - depending on configuration of specific object tool it will be executed in context of that container or will be executed for all objects under that container.

Tools can be managed in *Configuration ▶ Object Tools*. There are some *predefined object tools* that are available after installation of the system.

If an object tool is not needed for some time it can be just disabled and then enabled when required. When object tool is disabled it is not shown under "Tools" item of context menu. If an image (16x16 px) is configured for an object tool, it will be displayed next to object tool name in "Tools" menu.

Tool can have *input fields*, *filter depending on execution object*, *macro substitution* and *personal access control configuration*.

10.5.1 Object tool types

Internal

The only operation available for now is `wakeup` that sends magic packet to wake up a node.

Agent Command

This tool will execute command on an agent node and will show it's output if *Command generates output* option is enabled.

Properties for Restart system

General

Access Control
Filter
Input Fields

Name: &Restart system

Description: Restart target node via NetXMS agent

Agent's command: System.Restart

Execution options

- ☐ Command generates output
- ☐ Suppress notification of successful execution

Confirmation

- ☒ This tool requires confirmation before execution

Confirmation message: Host %n (%a) will be restarted. Are you sure?

Show in commands

- ☒ Show this tool in node commands

Command name: Restart system Command short name: Restart

Other options

- ☐ Disabled

Cancel Apply and Close

Field name	Description
Name	Name that will be shown in node menu. Submenu can be created with “->” notation.
Description	Description is shown in “Object Tools” view. Should be used to describe tool purpose.
Command	Name of agent command that will be executed. There is a number of commands built into agent and additional commands can be added by defining them in agent’s config. If command accepts parameters they are supplied it the following format: <code>commandName param1 param2 param3...</code>
Command generates output	If this option is selected then command execution will open a window with it’s output.
This tool requires confirmation before execution	If chosen a Yes/No pop-up with text from “Confirmation message” field will be shown before execution of tool.
Confirmation message	Contains message that will be shown in confirmation pop-up.
Show this tool in node commands	If this option is selected, then this tool will be shown for applicable nodes on <i>Object Details</i> view as node command.
Command name	Name of the command
Command short name	Is used when <i>Command name</i> is too long for display.
Disable Object Tool	If chosen, tool is not shown in Object browser’s context menu and Commands in Object Details.

SNMP Table

SNMP Table is used to get SNMP table from node on which it is executed and then show results in the table form.

Properties for Routing table (SNMP)

General

Name: &Info->&Routing table (SNMP)

Description: Show IP routing table

Title: Routing Table

SNMP Table Options

Use as index for second and subsequent columns:

☒ OID suffix of first column

☐ Value of first column

Confirmation

☐ This tool requires confirmation before execution

Confirmation message:

Show in commands

☐ Show this tool in node commands

Command name: Command short name:

☐ Disable Object Tool

Cancel OK

Field name	Description
Name	Name that will be shown in node menu. Submenu can be created with “->” notation.
Description	Description is shown in “Object Tools” view. Should be used to describe tool purpose.
Title	Title of view where table will be shown.
Use as index for second and subsequent columns OID suffix of first column	This option defines that suffix of columns OID will be used as suffix for columns OID’s to match lines
Use as index for second and subsequent columns Value of first column	This option defines that value of columns OID will be used as suffix for columns OID’s to match lines
This tool requires confirmation before execution	If chosen, before execution of tool will be shown Yes/No pop-up with text from “Confirmation message” field.
Confirmation message	Can be set the message that will be shown in confirmation pop-up.
Show this tool in node commands	If this option is selected, then this tool will be shown for applicable nodes on <i>Object Details</i> view as node command.
Command name	This will be shown as a name of the command.
Command short name	Is used when usual name is too long for display.
Disable Object Tool	If chosen, tool is not shown in node menu.

Agent List

Agent List is used to get agent list from node on which it is executed and then show results in the table form. Regular expression is used to split received data to columns.

The screenshot shows a window titled "Properties for Supported actions". On the left is a sidebar with a search bar "type filter text" and a list of tabs: General, Access Control, Filter, Columns, and Input Fields. The "General" tab is active. The main area contains the following fields and options:

- Name:** &Info->&Agent->Supported &actions
- Icon:** Two icons representing a key and a document.
- Description:** Show list of actions supported by agent
- Title:** Supported actions
- Parameter:** Agent.ActionList
- Regular expression:** ^(*) .(*) ^(*)*
- Confirmation:**
 - ☐ This tool requires confirmation before execution
 - Confirmation message: (empty text field)
- Show in commands:**
 - ☐ Show this tool in node commands
 - Command name: (empty text field)
 - Command short name: (empty text field)
- ☐ Disable Object Tool

At the bottom are "Cancel" and "Apply and Close" buttons.

Field name	Description
Name	Name that will be shown in node menu. Submenu can be created with “->” notation.
Description	Description is shown in “Object Tools” view. Should be used to describe tool’s purpose.
Title	Title of view where table will be shown.
Parameter	Name of list
Regular expression	Regular expression that will parse each line of list to separate it on columns defined in <i>Columns</i> tab.
This tool requires confirmation before execution	If chosen, before execution of tool will be shown Yes/No pop-up with text from “Confirmation message” field.
Confirmation message	Can be set the message that will be shown in confirmation pop-up.
Show this tool in node commands	If this option is selected, then this tool will be shown for applicable nodes on <i>Object Details</i> view as node command.
Command name	This will be shown as a name of the command.
Command short name	Is used when usual name is too long for display.
Disable Object Tool	If chosen, tool is not shown in node menu.

Agent Table

Agent Table is used to get agent table from node on which it is executed and then show results in the table form.

The screenshot shows a dialog box titled "Properties for Process list". On the left is a sidebar with a search bar containing "type filter text" and a list of tabs: General, Access Control, Filter, and Input Fields. The "General" tab is active. The main area contains the following fields and options:

- Name:** &Info->&Process list
- Description:** Show list of currently running processes
- Title:** Process List
- Parameter:** System.Processes
- Confirmation:**
 - ☐ This tool requires confirmation before execution
 - Confirmation message: (empty text box)
- Show in commands:**
 - ☐ Show this tool in node commands
 - Command name: (empty text box)
 - Command short name: (empty text box)
- ☐ Disable Object Tool

At the bottom right are "Cancel" and "Apply and Close" buttons.

Field name	Description
Name	Name that will be shown in node menu. Submenu can be created with “->” notation.
Description	Description is shown in “Object Tools” view. Should be used to describe tool purpose.
Title	Title of view where table will be shown.
Parameter	Name of list
This tool requires confirmation before execution	If chosen, before execution of tool will be shown Yes/No pop-up with text from “Confirmation message” field.
Confirmation message	Can be set the message that will be shown in confirmation pop-up.
Show this tool in node commands	If this option is selected, then this tool will be shown for applicable nodes on <i>Object Details</i> view as node command.
Command name	This will be shown as a name of the command.
Command short name	Is used when usual name is too long for display.
Disable Object Tool	If chosen, tool is not shown in node menu.

URL

URL tool opens URL in web browser.

Properties for Open web browser

General

Name
 &Connect->Open &web browser

Description
 Open embedded web browser to node

URL
 http://%u

TCP tunnel
☐ Setup TCP tunnel to remote port 1

Confirmation
☐ This tool requires confirmation before execution
 Confirmation message

Show in commands
☐ Show this tool in node commands
 Command name Command short name

Other options
☐ Disabled
☐ Run in container context

Cancel Apply and Close

Field name	Description
Name	Name that will be shown in node menu. Submenu can be created with “->” notation.
Description	Description is shown in “Object Tools” view. Should be used to describe tool purpose.
URL	URL that should be passed to browser to be opened.
TCP tunnel	If enabled, on object tool execution management client will open a local port and establish tunnel via the server and via a proxy agent. Proxy should have <code>EnableTCPProxy=yes</code> in it's configuration file. The following macros can be used in URL field: <ul style="list-style-type: none"> • <code>\${local-address}</code> - local IP address • <code>\${local-port}</code> - local port number
This tool requires confirmation before execution	If chosen, before execution of tool will be shown Yes/No pop-up with text from “Confirmation message” field.
Confirmation message	Can be set the message that will be shown in confirmation pop-up.
Show this tool in node commands	If this option is selected, then this tool will be shown for applicable nodes on <i>Object Details</i> view as node command.
Command name	This will be shown as a name of the command.
Command short name	Is used when usual name is too long for display.
Disable Object Tool	If chosen, tool is not shown in node menu.
Run in container context	If this option is selected, then tool will run only for selected container, not affecting children nodes.

Local Command

Local Command tool will execute command on the node, where Desktop Management Client is running and will show it's output if *Command generates output* option is enabled.



This tool type is not visible from Web Client as it is not possible to execute command on web page receiver's machine.

Properties for SSH

General

Access Control
Filter
Input Fields

General

Name: &Connect->&SSH Icon:  

Description: Run SSH connection to node in Linux terminal

Command: x-terminal-emulator -e sh -c "ssh %(username)@%u || read tmp"

TCP tunnel

☐ Setup TCP tunnel to remote port 1 - +

Execution options

☐ Command generates output
☐ Suppress notification of successful execution

Confirmation

☐ This tool requires confirmation before execution

Confirmation message

Show in commands

☐ Show this tool in node commands

Command name: Command short name:

Other options

☐ Disabled
☐ Run in container context

Cancel Apply and Close

Field name	Description
Name	Name that will be shown in node menu. Submenu can be created with “->” notation.
Description	Description is shown in “Object Tools” view. Should be used to describe tool purpose.
Command	Command that should be executed on a local machine
TCP tunnel	If enabled, on object tool execution management client will open a local port and establish tunnel via the server and via a proxy agent. Proxy should have <code>EnableTCPProxy=yes</code> in it's configuration file. The following macros can be used in command field: <ul style="list-style-type: none"> <code>\${local-address}</code> - local IP address <code>\${local-port}</code> - local port number
Command generated output	If this option is selected, then command execution will open a window with output of the command.
This tool requires confirmation before execution	If chosen, before execution of tool will be shown Yes/No pop-up with text from “Confirmation message” field.
Confirmation message	Can be set the message that will be shown in confirmation pop-up.
Show this tool in node commands	If this option is selected, then this tool will be shown for applicable nodes on <i>Object Details</i> view as node command.
Command name	This will be shown as a name of the command.
Command short name	Is used when usual name is too long for display.
Disable Object Tool	If chosen, tool is not shown in node menu.
Run in container context	If this option is selected, then tool will run only for selected container, not affecting children nodes.

Server Command

Server command tool can be used to execute command on the server.

Properties for Print Hello

General

Name: Print Hello

Description: This command will

Command: echo "hello"

Execution options: ☐ Command generates output

Confirmation: ☐ This tool requires confirmation before execution

Confirmation message:

Show in commands: ☐ Show this tool in node commands

Command name: Command short name:

☐ Disable Object Tool

☐ Run in container context

Cancel Apply and Close

Field name	Description
Name	Name that will be shown in node menu. Submenu can be created with “->” notation.
Description	Description is shown in “Object Tools” view. Should be used to describe tool purpose.
Command	Command that should be executed on a server
Command generated output	If this option is selected, then command execution will open a window with output of the command.
This tool requires confirmation before execution	If chosen, before execution of tool will be shown Yes/No pop-up with text from “Confirmation message” field.
Confirmation message	Can be set the message that will be shown in confirmation pop-up.
Show this tool in node commands	If this option is selected, then this tool will be shown for applicable nodes on <i>Object Details</i> view as node command.
Command name	This will be shown as a name of the command.
Command short name	Is used when usual name is too long for display.
Disable Object Tool	If chosen, tool is not shown in node menu.
Run in container context	If this option is selected, then tool will run only for selected container, not affecting children nodes.

Download File

Download file tool can be used to monitor agent logs. This tool will retrieve the content of the file from agent.

Properties for

type filter text

General

Access Control

Filter

Columns

General

Name
Logs->Netxms log

Description
Description

Remote file name
/var/log/netxmsd

File Options
Limit initial download size (in bytes, 0 for unlimited)
500

☒ Follow file changes

Confirmation
☐ This tool requires confirmation before execution
Confirmation message

☐ Disable Object Tool

Cancel OK

Field name	Description
Name	Name that will be shown in node menu. Submenu can be created with “->” notation.
Description	Description is shown in “Object Tools” view. Should be used to describe tool purpose.
Remote File Name	Name of file that will be retrieved. In Windows systems should be with double back slash as a separator(C:\log\log.log). Can be used <code>strftime(3C)</code> macros
Limit initial download size	Limits the size of download file. If is set to 500, tool will retrieve last 500 bytes of requested file. If is set to 0, complete file will be retrieved.
Follow file changes	If chosen, “File View” will be updated when file will be populated with new data.
This tool requires confirmation before execution	If chosen, before execution of tool will be shown Yes/No pop-up with text from “Confirmation message” field.
Confirmation message	Can be set the message that will be shown in confirmation pop-up.
Show this tool in node commands	If this option is selected, then this tool will be shown for applicable nodes on <i>Object Details</i> view as node command.
Command name	This will be shown as a name of the command.
Command short name	Is used when usual name is too long for display.
Disable Object Tool	If chosen, tool is not shown in node menu.

Server Script

Server Script tool can be used to execute NXSL script from *Script Library*. This tool provide full range of capabilities that are available thought NXSL scripting.

Properties for ServerScript

General

Name: ServerScript

Description: Execute server script

Script: scriptName

Execution options: ☐ Command generates output

Confirmation: ☐ This tool requires confirmation before execution
Confirmation message:

Show in commands: ☐ Show this tool in node commands
Command name: Command short name:

☐ Disable Object Tool
☐ Run in container context

Cancel Apply and Close

Field name	Description
Name	Name that will be shown in node menu. Submenu can be created with “->” notation.
Description	Description is shown in “Object Tools” view. Should be used to describe tool purpose.
Script	Name of the script from the <i>Script Library</i>
Command generates output	If chosen, new window with script execution result will be opened.
This tool requires confirmation before execution	If chosen, before execution of tool will be shown Yes/No pop-up with text from “Confirmation message” field.
Confirmation message	Can be set the message that will be shown in confirmation pop-up.
Show this tool in node commands	If this option is selected, then this tool will be shown for applicable nodes on <i>Object Details</i> view as node command.
Command name	This will be shown as a name of the command.
Command short name	Is used when usual name is too long for display.
Disable Object Tool	If chosen, tool is not shown in node menu.
Run in container context	If this option is selected, then tool will run only for selected container, not affecting children nodes.

10.5.2 Properties

Filter

Filters are used to chose on which nodes to show object tool. There are 5 types of filtering. Show object tool:

1. if agent available on a node
2. if node supports SNMP
3. if node SNMP OID matches with provided string

4. if nodes OS matches provided comma separated regular expression list
5. if provided *template* name matches provided comma separated regular expression list

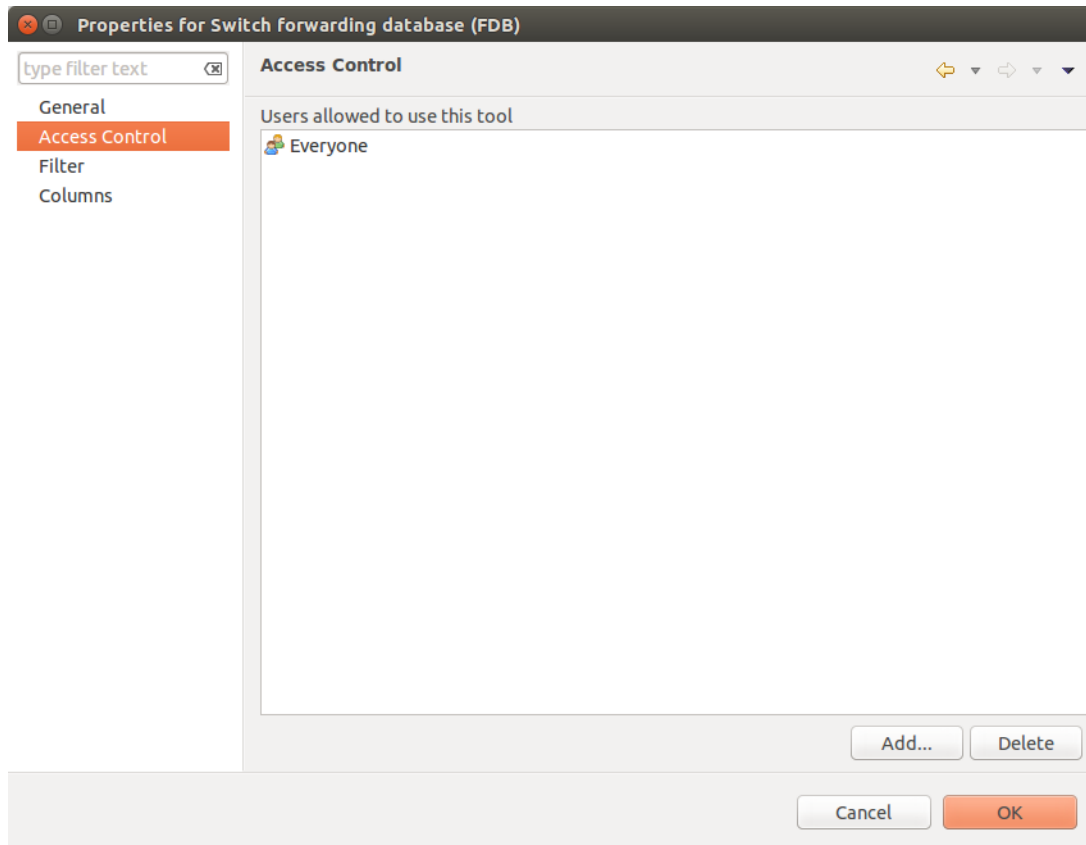
The screenshot shows a dialog box titled "Properties for Switch forwarding database (FDB)". On the left is a sidebar with four tabs: "General", "Access Control", "Filter" (which is selected and highlighted in orange), and "Columns". Above the sidebar is a text input field labeled "type filter text" with a small icon to its right. The main area of the dialog is titled "Filter" and contains several configuration options:

- ☐ NetXMS agent should be available
- ☒ Node should support SNMP
- ☐ Node SNMP OID should match with the following template:
text,text
- ☐ System OS name should match this template(coma separated regular expression list):
- ☐ Parent template name should match this template(coma separated regular expression list):

At the bottom right of the dialog are two buttons: "Cancel" and "OK".

Access Control

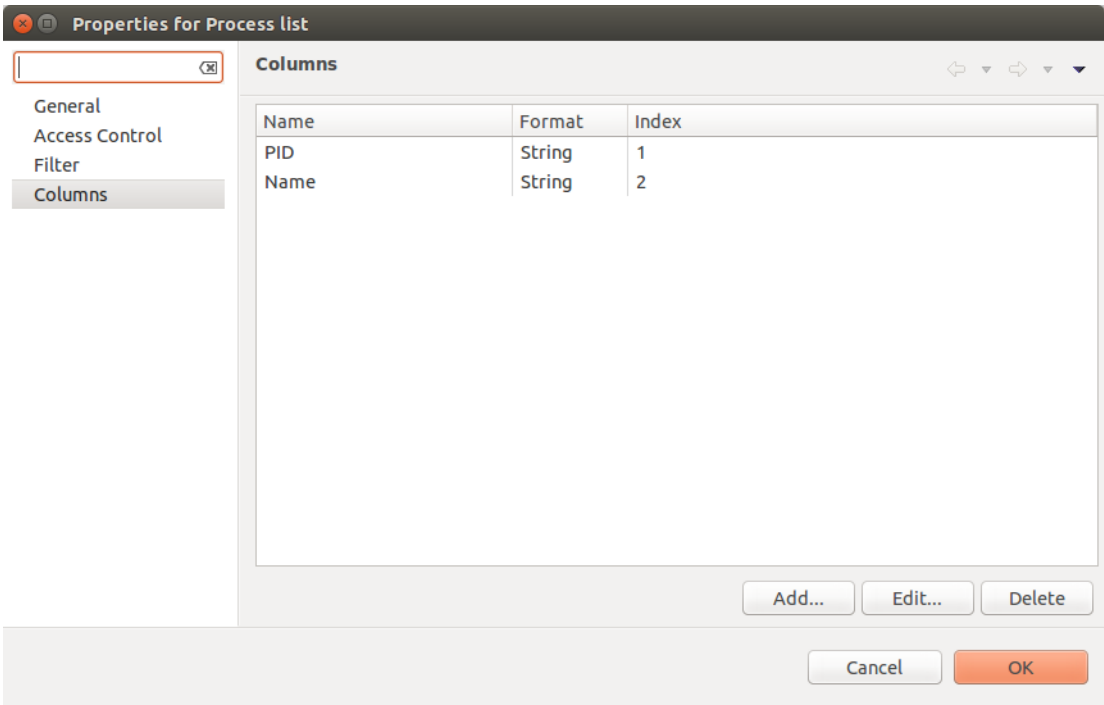
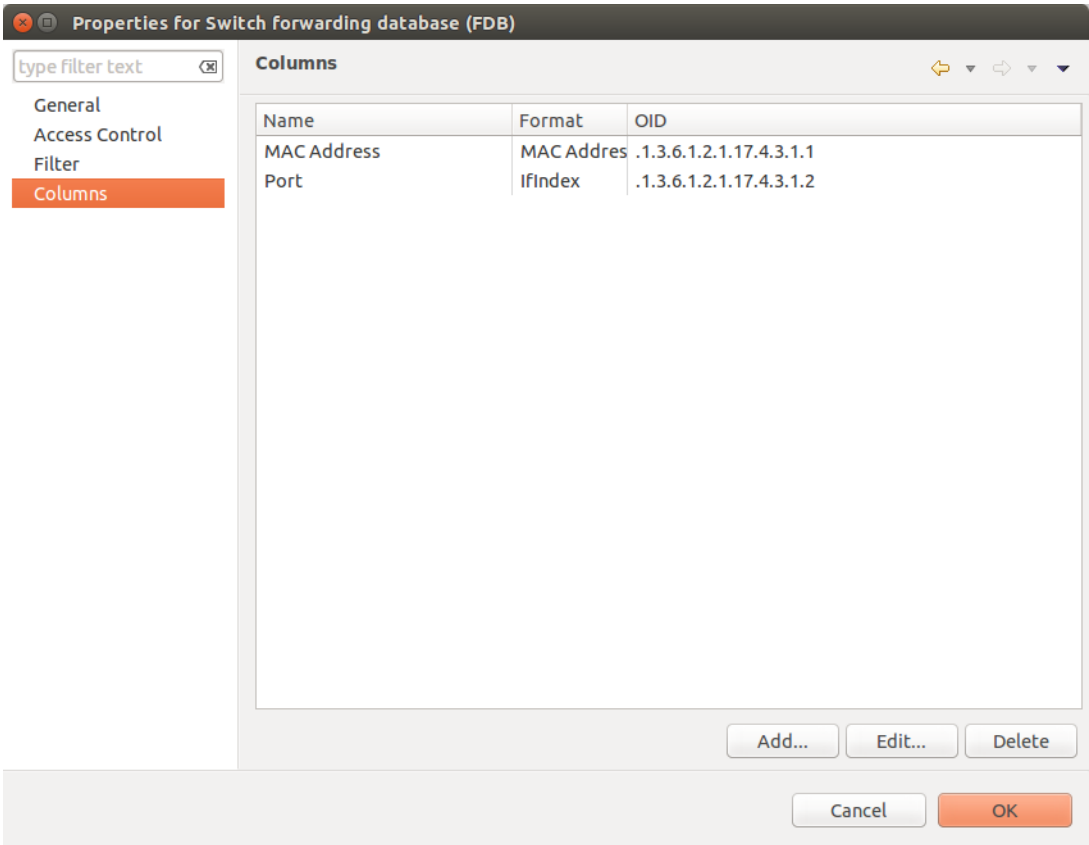
In *Access Control* tab can be defined which users or groups can execute this action. If the list is empty, only administrator will be able to execute this action.



Columns

Columns tab is used only for *Agent List* and *SNMP Table* object tool types.

For *SNMP Table* it describes name and type of matching OID from response message.

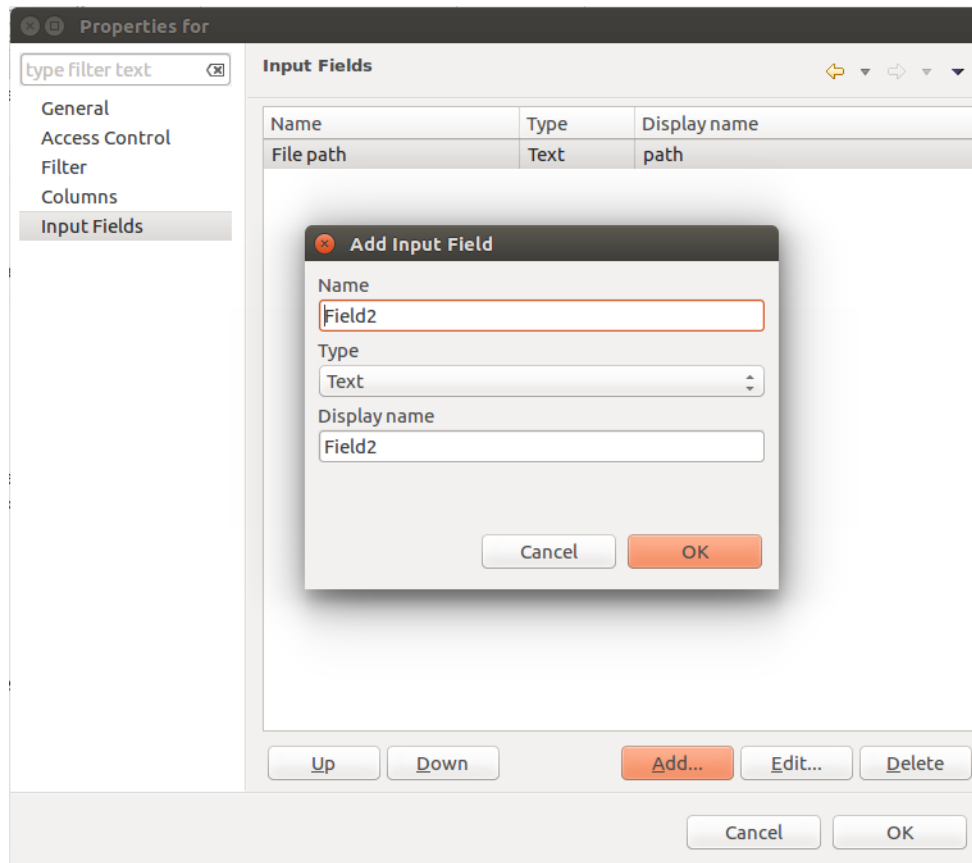


Input fields

There is option to add input fields for object tool commands. This fields are defined on the *Input fields* view and added to command in % (name) format. More about formats can be found in [Macro Substitution](#) chapter.

Input field can be one of this types:

- Text
- Password
- Number



10.5.3 Macro Substitution

Action, file download, local command, and URL tool types allows macro substitution. Any string starting with percent sign considered macro name and is expanded. The following macros are recognized:

Macro	Description
%a	IP address of event source object.
%g	Globally unique identifier (GUID) of event source object.
%i	Unique ID of event source object in hexadecimal form. Always prefixed with 0x and contains exactly 8 digits (for example 0x000029AC).
%I	Unique ID of event source object in decimal form.
%n	Name of event source object.
%u	IP address of event source object for use in URL. Expands into [addr] for IPv6 and addr for IPv4.
%U	User name of user that launched the object tool from user interface
%v	NetXMS server's version.
%[name]	Value returned by script. You should specify name of the script from script library. It's possible to specify script entry point separating it by /, e.g. to call a function named calculate: %[name/calculate]. Script parameters can be specified in brackets, e.g.: %[name(123, "A textual parameter")]
%{name}	Value of custom attribute.
%{name:default_value}	Value of custom attribute. If such custom attribute does not exist on a particular node, default_value is taken. If custom attribute exists, but has empty value, this empty value is taken.
%(name)	Value of input field.
%<name>	Parameter with given name.
\${local-address}	Local IP address for TCP tunnel
\${local-port}	local port number for TCP tunnel
%%	Insert % character.

If object tool called from alarm's pop-up menu the following additional macros are available:

Macro	Description
%A	Alarm's text (can be used only in actions to put text of alarm from the same event processing policy rule).
%c	Event's code.
%m	Event's message text (meaningless in event template).
%N	Event's name.
%S	Event's severity code as number. Possible values are: <ul style="list-style-type: none"> • 0 - <i>Normal</i> • 1 - <i>Warning</i> • 2 - <i>Minor</i> • 3 - <i>Major</i> • 4 - <i>Critical</i>
%S	Event's severity code as text.
%y	Alarm state as number. Possible values are: <ul style="list-style-type: none"> • 0 - <i>Outstanding</i> • 1 - <i>Acknowledged</i> • 2 - <i>Resolved</i> • 3 - <i>Terminated</i>
%Y	Alarm's id.

Internal object tool is special case of object tools. Macro expansions not performed for *Internal object tools*.

For any unknown macro name system will try to read custom attribute with given name (attribute search is case sensitive). If attribute with given name not found, empty string will be inserted.

10.5.4 Predefined Object Tools

NetXMS is delivered with a number of predefined Object Tools. Here is the list of them:

Name	Type	Description	Filter
<u>C</u> onnect• <u>O</u> pen browser	<u>w</u> eb URL	Open embedded web browser to node	
<u>C</u> onnect-> <u>O</u> pen browser (<u>H</u> TTPS)	<u>w</u> eb URL	Open embedded web browser to node using HTTPS	
<u>I</u> nfo-> <u>A</u> gent-> <u>L</u> oaded subagents	Agent Table	Show information about loaded subagents	NetXMS agent should be available
<u>I</u> nfo-> <u>A</u> gent-> <u>C</u> onfigured targets	Agent Table	Show information about ICMP targets configured on this agent	NetXMS agent and ping subagent should be available
<u>I</u> nfo-> <u>A</u> gent-> <u>S</u> upported actions	Agent List	Show information about actions supported by agent	NetXMS agent should be available
<u>I</u> nfo-> <u>A</u> gent-> <u>S</u> upported lists	Agent List	Show list of lists supported by agent	NetXMS agent should be available
<u>I</u> nfo-> <u>A</u> gent-> <u>S</u> upported metrics	Agent List	Show list of metrics supported by agent	NetXMS agent should be available
<u>I</u> nfo-> <u>A</u> gent-> <u>S</u> upported tables	Agent List	Show list of tables supported by agent	NetXMS agent should be available
<u>I</u> nfo-> <u>C</u> urrent processes	Agent Table	Show information about currently running processes	NetXMS agent should be available
<u>I</u> nfo-> <u>R</u> outing table (SNMP)	SNMP Table	Show IP routing table	NetXMS should support SNMP
<u>I</u> nfo-> <u>S</u> witch forwarding database (FDB)	SNMP Table	Show switch forwarding database	NetXMS should support SNMP
<u>I</u> nfo-> <u>A</u> ctive user sessions	Agent List	Show information about active user sessions	NetXMS agent should be available
<u>I</u> nfo-> <u>A</u> RP (Agent)	cache Agent List	Show ARP cache	NetXMS agent should be available
<u>I</u> nfo-> <u>T</u> opology table (CDP)	SNMP Table	Show topology table (CDP)	NetXMS should support SNMP
<u>I</u> nfo-> <u>T</u> opology table (LLDP)	SNMP Table	Show topology table (LLDP)	NetXMS should support SNMP
<u>I</u> nfo-> <u>T</u> opology table (Nortel)	SNMP Table	Show topology table (Nortel protocol)	NetXMS should support SNMP
<u>R</u> estart system	Action	Restart target node via NetXMS agent	NetXMS agent should be available
<u>S</u> hutdown system	Action	Shutdown target node via NetXMS agent	NetXMS agent should be available
<u>W</u> akeup node	Internal	Wakeup node using Wake-On-LAN magic packet	
<u>R</u> estart agent	Action	Restart NetXMS agent on target node	NetXMS agent should be available

NETWORK DISCOVERY

11.1 Introduction

NetXMS is capable of discovering your network automatically. The network discovery module can operate in two modes: passive and active.

In passive mode information about new hosts and devices are obtained from *ARP* tables and routing tables of already known devices. NetXMS starts with its own *ARP* cache and routing table.

In active discovery mode the NetXMS server will send an *ICMP* echo request to all IP addresses in the given range and consider each responding address for adding to database. If zoning is used the server sends an echo request only in zone 0. In other zones requests are sent by proxies. For each new device the NetXMS server tries to gather additional information using the *SNMP* and NetXMS agent and then adds it to database. By default the NetXMS server will add all discovered devices to database, but you can limit it by using discovery filters. Default *SNMP* credentials can be set in *Default SNMP credentials*.

The default intervals are 2 hours for active discovery and 15 minutes for passive discovery. These values can be changed in the Network Discovery configuration. The number of discovery poller threads changes dynamically and is defined by the server configuration parameters `ThreadPool.Discovery.BaseSize` and `ThreadPool.Discovery.MaxSize`. More information about server configuration parameters can be found at [here](#).

11.2 Configuring Network Discovery

To change network discovery settings, go to the main menu of the management client and choose *Configuration ▶ Network Discovery*. The configuration form will open:

11.2.1 General

In this section, you can choose the network discovery mode and choose if the source node of *SNMP Trap* or syslog source address should be used for discovery.

11.2.2 Schedule

For passive discovery the interval (in seconds) is selected. For active discovery you can choose either an interval (in seconds) or a cron format schedule. See [here](#) for more details.

11.2.3 Filter

In this section, you can define a filter for adding new nodes to NetXMS database. Available filtering options are:

No filtering

Any new device found will be added to the database. This is the default setting.

Custom script

You can choose a *NXSL* script from the *Script Library* to work as a discovery filter. This custom filtering script will get an object of class `NewNode` as its first parameter (special variable `$1`), and should return true to allow node inclusion into database.

Automatically generated script

This option can be used if you need only simple filtering. When selected, additional options control what nodes will be added to database:

Accept node if it has NetXMS agent	If checked, only nodes with NetXMS agent detected will pass the filter.
Accept node if it has SNMP agent	If checked, only nodes with SNMP agent detected will pass the filter.
Accept node if it is within given range or subnet	Only accept nodes within given address range or subnet. Address ranges can be configured in <i>Address Filters</i> section.

Please note that the first two options (NetXMS agent presence and SNMP agent presence) forms **OR** condition - if both are checked, any node with either SNMP agent or NetXMS agent will pass. Whereas the address range check and the first two options forms **AND** condition - so if a node does pass the agent presence check, but is not in an allowed IP address range, it will not be accepted. In other words, if all three options are checked, the condition for a new node to pass filter can be written as following:

if (node has NetXMS agent **or** node has SNMP agent) **and** node within given range **then** pass

11.2.4 Active Discovery Targets

In this section you can define address ranges for active discovery. The NetXMS server will periodically send ICMP echo requests to these addresses, and consider every responding device for addition to the database. This list has no effect if active discovery is off.

11.2.5 Address Filters

In this section you can define address ranges for the automatically generated discovery filter. This list has no effect if discovery is off or the filter is not set to *Automatically generated script*.

DATA COLLECTION

12.1 How data collection works

Every node can have many data collection items configured (see [Data Collection](#) for detailed description). NetXMS server has a set of threads dedicated to data collection, called *Data Collectors*, used to gather information from the nodes according to *DCI* configuration. You can control how many data collectors will run simultaneously, by changing server configuration parameter `ThreadPool.DataCollector.MaxSize`.

Node capabilities provide information about available sources for data collection in the *Overview->Capabilities* section. The last values of DCIs for the node can be found on the *Data Collection* tab. Additionally, specific DCIs can be displayed in the *Overview`->Last Values* section or as a graph on the *Performance* tab. More details about DCI display configuration options can be found in the [Other options](#) and [Performance View](#) chapters.

All configured DCIs are checked for polling requirement every second. If DCI needs to be polled, appropriate polling request is placed into internal data polling queue. First available data collector will pick up the request and gather information from the node according to DCI configuration. If a new value was received successfully, it's being stored in the database, and thresholds are checked. After threshold checking, data collector is ready for processing new request. If DCI is unsupported it will be polled only every tenth poll, this is not configurable. Processing of a newly received metric value is outlined on the figure below.

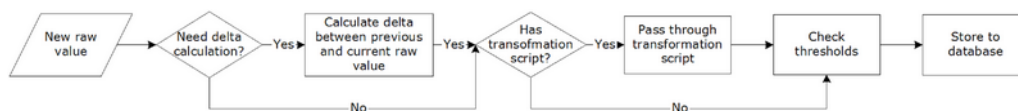


Fig. 1: Newly received metric processing

It is also possible to push data to server. If DCI source is set to *Push*, server just waits for new values instead of polling from a data source.

By default, DCI data is not collected for the duration while connection between server and agent is broken as poll request would not get to agent. There is special configuration that allows data collection and storage on agent till connection with server is restored and collected data is pushed to the server thereafter. This option is available for metrics, table metrics and proxy SNMP metrics as well as implemented for proxy SNMP table metrics and DCIs with custom schedule. In case of this setup, agent stores DCI configuration locally and does all metric collection and dispatch on its own. DCI configuration is synchronized on connect, DCI configuration change or SNMP proxy server change. Information about configuration options can be found here: [Agent caching mode](#).

12.2 DCI configuration

Data collection for a node can be configured using management client. To open data collection tab view, click on node object in *Infrastructure* or *Network* perspective, and click *Data Collection* tab. You will see the list of configured data collection items. From here, since DCI configuration and Last values are combined, one can see collected data and configure new or change existing metrics for monitoring. Right click on an item and all possible configuration options will be available.

Each DCI have multiple attributes which affects the way data is collected. Detailed information about each attribute is given below and can be accessed by selecting *Edit...*, *New parameter...* or *New table...*

12.2.1 General

Properties for System.CPU.Usage.User

General

Metric to collect

Origin: NetXMS Agent Source node override: None

Metric: System.CPU.Usage.User

Display name: CPU: usage (user)

Polling

Data Type: Float Units: % Use multipliers: Default

Collection schedule

☒ Server default interval (60 seconds)

☐ Custom interval

☐ Advanced schedule

History retention period

☒ Server default (30 days)

☐ Custom

☐ Do not save to the database

☐ Save only changed values

Restore Defaults Apply

Apply and Close Cancel

Fig. 2: DCI configuration general property page

Display name

Display name is a free form text string describing DCI. It is not used by the server and is intended for better information understanding by operators. If you use the *Select* button to choose a metric from the list, description field will be filled in automatically.

Metric

Name of the metric of interest, used for making a request to target node. For NetXMS Agent and Internal metrics it will be metric name, and for SNMP agent it will be an SNMP OID. You can use the *Select* button for easier selection of required metric name.

Available agent metric names are obtained during *Configuration poll*.

Origin

Origin of data (method of obtaining data). Possible single-value origins are:

Source	Description
Internal	Data generated inside NetXMS server process (server statistics, etc.)
NetXMS Agent	Data is collected from NetXMS agent, which should be installed on target node.
SNMP	Data is collected via SNMP transport.
Web service	Data is obtained from JSON, XML, or plain text retrieved via HTTP/HTTPS
Push	Values are pushed by external system (using <i>nxpush</i> , <i>nxapush</i> tools or API), from NXSL script or log file parser.
Windows Performance counters	Data is collected via NetXMS agent running on Windows machine. Windows Performance counters metric has format <code>Object (Instance)\Counter</code> , e.g. <code>\LogicalDisk(C:)\Avg. Disk Write Queue Length</code> .
SM-CLP	Data is collected via Server Management Command Line Protocol.
Script	Value is generated by NXSL script stored in <i>Script Library</i> . Script name and other options are set in <i>Metric</i> field: <ul style="list-style-type: none"> <code>my_script</code> - will call <code>main()</code> or <code>\$main()</code> function from <code>my_script</code> script library script <code>my_script(param1, param2)</code> - will call <code>main()</code> or <code>\$main()</code> function from <code>my_script</code> passing parameters <i>param1</i>, <i>param2</i> to it <code>my_script.my_function</code> - will call <code>my_function()</code> function from <code>my_script</code> <code>my_script.my_function(param1, param2)</code> - will call <code>my_function()</code> function from <code>my_script</code> passing parameters <i>param1</i>, <i>param2</i> to it
SSH	Data is obtained from output of <code>ssh</code> command executed through SSH connection.
MQTT	Data is obtained by subscribing to MQTT broker topics.
Network Device Driver	Some SNMP drivers (e.g. NET-SNMP, RITTAL) provide metrics for data collection. E.g. NET-SNMP provides information about storage this way.
Modbus	Data is collected via Modbus-TCP industrial protocol. See Modbus for more information.
Ethernet/TP	Data is collected via Ethernet/TP industrial protocol.

Push Agent origin is different from all others, because it represents DCIs whose values are pushed to server by external program (usually via *nxapush* or *nxpush* command line tool) instead of being polled by the server based on the schedule. Values can also be pushed from a NXSL script launched on the server.

Possible table metric origins are Internal, NetXMS agent, SNMP, Script. Please refer to description in above table.

Data Type

Data type for column. Can be one of the following: *Integer*, *Unsigned Integer*, *Integer 64-bit*, *Unsigned Integer 64-bit*, *Counter 32-bit*, *Counter 64-bit*, *Float* (floating point number), or *String*. Selected data type affects collected data processing - for example, you cannot use operations like `less than` or `greater than` on strings. If you select metric from the list using the *Select* button, correct data type will be set automatically.

Units

For user convenience collected DCI values can have the following predefined units assigned, but it is possible to enter any unit one requires. Most of the units are just displayed after the value, but some of them are special and affect how collected data is displayed:

Unit	Description
%	Percent - symbol used to indicate a percentage, a number or ratio as a fraction of 100. For more details please check Wikipedia
°C	Degree in Celsius, unit of temperature. For more details please check Wikipedia
°F	Degree in Fahrenheit, unit of temperature. For more details please check Wikipedia
A	Ampere, unit of electric current. For more details please check Wikipedia
B (IEC)	Bytes in IEC format. Please note that "(IEC)" part will be removed when value is displayed. For more details on difference between IEC and SI please check Wikipedia
b (IEC)	Bits in IEC format. Please note that "(IEC)" part will be removed when value is displayed.
B (Metric)	Bytes in SI format. Please note that "(Metric)" part will be removed when value is displayed.
b (Metric)	Bits in SI format. Please note that "(Metric)" part will be removed when value is displayed.
B/s	Bytes per second. For more details please check Wikipedia
b/s	Bits per second. For more details please check Wikipedia
dBm	Unit of power level expressed using a logarithmic decibel. For more details please check Wikipedia
Epoch time	Unix time, measures time by the number of non-leap seconds that have elapsed since 00:00:00 UTC on 1 January 1970. Converts collected into human readable timestamp. For more details please check Wikipedia
Hz	Hertz, the unit of frequency. For more details please check Wikipedia
J	Joule, unit of energy. For more details please check Wikipedia
lm	Lumen, a measure of the perceived power of visible light emitted by a source. For more details please check Wikipedia
lx	Lux, unit of illuminance or luminous flux per unit area. For more details please check Wikipedia
N	Newton, unit of force. For more details please check Wikipedia
Pa	Pascal, unit of pressure. For more details please check Wikipedia
rpm	Revolutions per minute. For more details please check Wikipedia
s	Second, unit of time. For more details please check Wikipedia
T	Tesla, unit of magnetic flux density. For more details please check Wikipedia
Uptime	Measure of system reliability. Converts number of seconds since uptime into human readable format. For more details please check Wikipedia
W	Watt, unit of power or radiant flux. For more details please check Wikipedia
V	Volt, electric potential between two points of a conducting wire. For more details please check Wikipedia
Ω	Ohm, unit of electrical resistance. For more details please check Wikipedia

Use multipliers

This boolean setting gives convenience of displaying some measurements in more readable form. For example, if enabled, 1230000 becomes 1.23 M. Please note - setting has no effect on units “%”, “°C”, “°F”, “dBm” and “rpm”. Everything with (IEC) will use binary multipliers both for calculation and to display value. This setting is taken into consideration only to display value; it is not converting value in the database. Selection here will be taken to format value when macro `%<{format-specifier}name>` with formatting is used. In Other options property page it is possible to set fixed multiplier degree. Again, it is used for display purposes only, however will be used when macro `%<{format-specifier}name>` is used.

Source node override

Source node of metrics collection. This can be used when other node provides information about current node. In this way, platform provides additional flexibility of where metrics collection is taking place.

Other example of usage is virtual nodes (nodes with IP 0.0.0.0). In this case, node state can be obtained from the DCI created on current node, but collected from the other one.

Data is collected from the current node if no value is set.

Collection schedule

Polling mode and interval describe schedule type and interval between consecutive polls, in seconds. However, collecting too many values for too long will lead to significant increase of your database size and possible performance degradation.

Following options can be selected:

- *Server default interval* - default value will be taken from *DataCollection.DefaultDCIPollingInterval* server configuration parameter.
- *Custom interval* - Allows to enter a custom value. This field supports macro resolution, so e.g. you can use `%{polling_interval:600}` macro that will take value of `polling_interval` custom attribute or 600, if such custom attribute is not present on the node.
- *Advanced scheduling* - schedules configured in *Custom Schedule* page will be used.

If you turn on *Advanced Schedule* flag, additional link to *Custom Schedule* will appear and, once configured, server will use custom schedule for collecting DCI values instead of fixed intervals. Advanced schedule consists of one or more records; each representing desired data collection time in cron-style format.

See *Cron format* for supported cron format options.

For DCI Collection schedule it's possible to specify optional sixth (first from left) cron field for resolution in seconds. It's not recommended to use seconds in custom schedules as your main data collection strategy though. Use seconds only if it is absolutely necessary.

History retention period

This attribute specifies how long the collected data should be kept in database, in days. Minimum retention time is 1 day and maximum has not limit. However, keeping too many collected values for too long may lead to significant increase of your database size and possible performance degradation.

Following options can be selected:

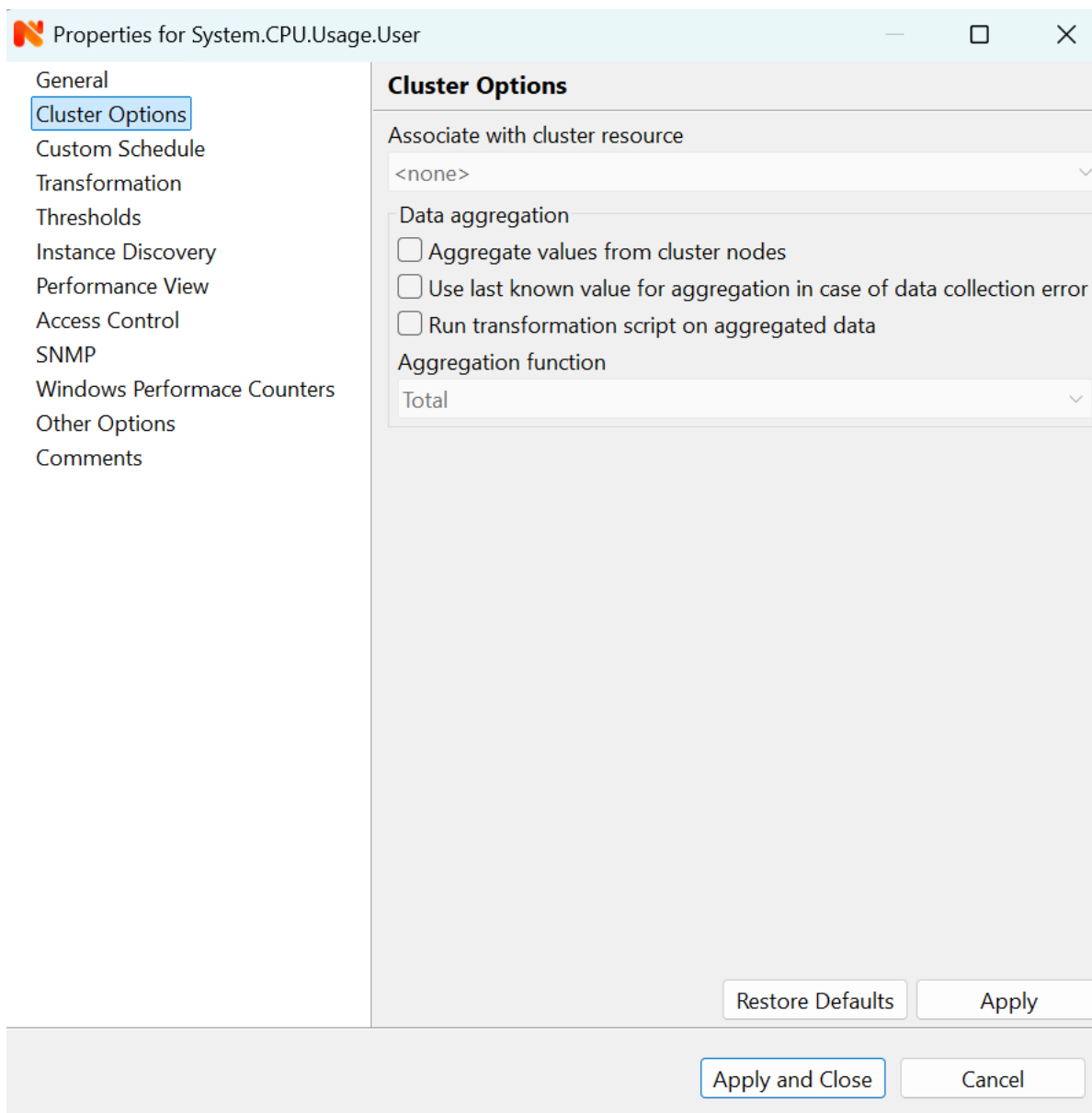
- *Server default* - default value will be taken from *DataCollection.DefaultDCIRetentionTime* server configuration parameter.
- *Custom* - Allows to enter a custom value. This field supports macro resolution, so for example you can use `%{storage_period:30}` macro that will take value of `storage_period` custom attribute or 30 if such custom attribute is not present on the node.
- *Do not save collected data to database* - will not save collected data to database, but will store last value in memory

Last option is used when it is required to show latest (every 1 second collected) data on Dashboard, however it would result in excessive data stored in database. So, 2 DCI configurations are created - one to store historical data collected once per minute and the second one, that is not stored in database, but is collected every second and displayed on dashboards in close to real time.

- *Save only changed values* - if enabled, value is saved to the database only if it differs from last saved value.

12.2.2 Cluster

This section is available only for DCI's collected on cluster.



The screenshot shows a window titled "Properties for System.CPU.Usage.User". On the left is a sidebar with the following options: General, Cluster Options (highlighted), Custom Schedule, Transformation, Thresholds, Instance Discovery, Performance View, Access Control, SNMP, Windows Performace Counters, Other Options, and Comments. The main area is titled "Cluster Options" and contains the following settings:

- Associate with cluster resource:** A dropdown menu showing "<none>".
- Data aggregation:** A section with three checkboxes:
 - ☐ Aggregate values from cluster nodes
 - ☐ Use last known value for aggregation in case of data collection error
 - ☐ Run transformation script on aggregated data
- Aggregation function:** A dropdown menu showing "Total".

At the bottom right of the main area are two buttons: "Restore Defaults" and "Apply". At the bottom of the dialog are two buttons: "Apply and Close" and "Cancel".

Fig. 3: DCI configuration cluster property page

Associate with cluster resource

In this field one can specify cluster resource associated with DCI. Data collection and processing will occur only if node, you configured DCI for, is current owner of this resource. This field is valid only for cluster member nodes.

Data aggregation

This section specifies how cluster data aggregation is done. *Aggregate values from cluster nodes* option means that DCI from cluster will be collected on each node separately and aggregated on cluster using one of the aggregation options.

Aggregation options:

- Total
- Average
- Min
- Max

12.2.3 Data Transformations

In simplest case, NetXMS server collects values of specified metrics and stores them in database. However, you can also specify various transformations for original value. For example, you may be interested in a delta value, not in a raw value of some metric. Or, you may want to have metric's value converted from bytes to kilobytes. All transformations will take place after receiving new value and before threshold processing.

Data type after transformation - drop down menu of required data type.

Data transformation consists of two steps. In the first step, delta calculation is performed. You can choose four types of delta calculation:

Function	Description
None	No delta calculation performed. This is the default setting for newly created DCI.
Simple	Resulting value will be calculated as a difference between current raw value and previous raw value. By raw value it is meant the metric's value originally received from host.
Average per second	Resulting value will be calculated as a difference between current raw value and previous raw value, divided by number of seconds passed between current and previous polls.
Average per minute	Resulting value will be calculated as a difference between current raw value and previous raw value, divided by number of minutes passed between current and previous polls.

In second step, custom transformation script is executed (if present). By default, newly created DCI does not have a transformation script. If transformation script is applied, the resulting value of the first step is passed to the transformation script as a parameter; and a result of script execution is the final DCI value. Transformation script gets original value as first argument (available via special variable `$1`), and also has two predefined global variables: `$node` (reference to current node object), and `$dci` (reference to current DCI object).

In case of table DCIs, `$1` special variable is an object of type Table.

For more information about NetXMS scripting language, please refer to [Scripting](#) chapter in this manual.

Transformation script can be tested in the same view, by clicking *Test...* and entering test input data.

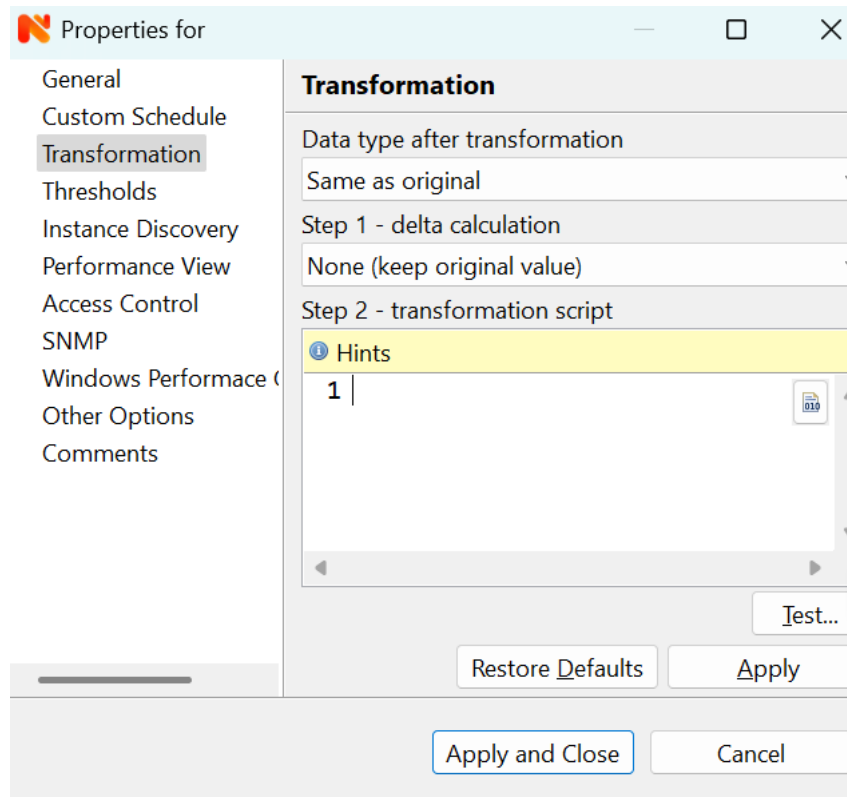


Fig. 4: DCI configuration transformation property page

12.2.4 Thresholds

For every DCI you can define one or more thresholds. For each threshold there is a pair of condition and event - if condition becomes true, associated event is generated. To configure thresholds, open data collection *Edit...* mode for node or template DCI. You can add, modify and delete thresholds using buttons below the threshold list. If you need to change the threshold order, select one threshold and use arrow buttons located on the right to move the selected threshold up or down.

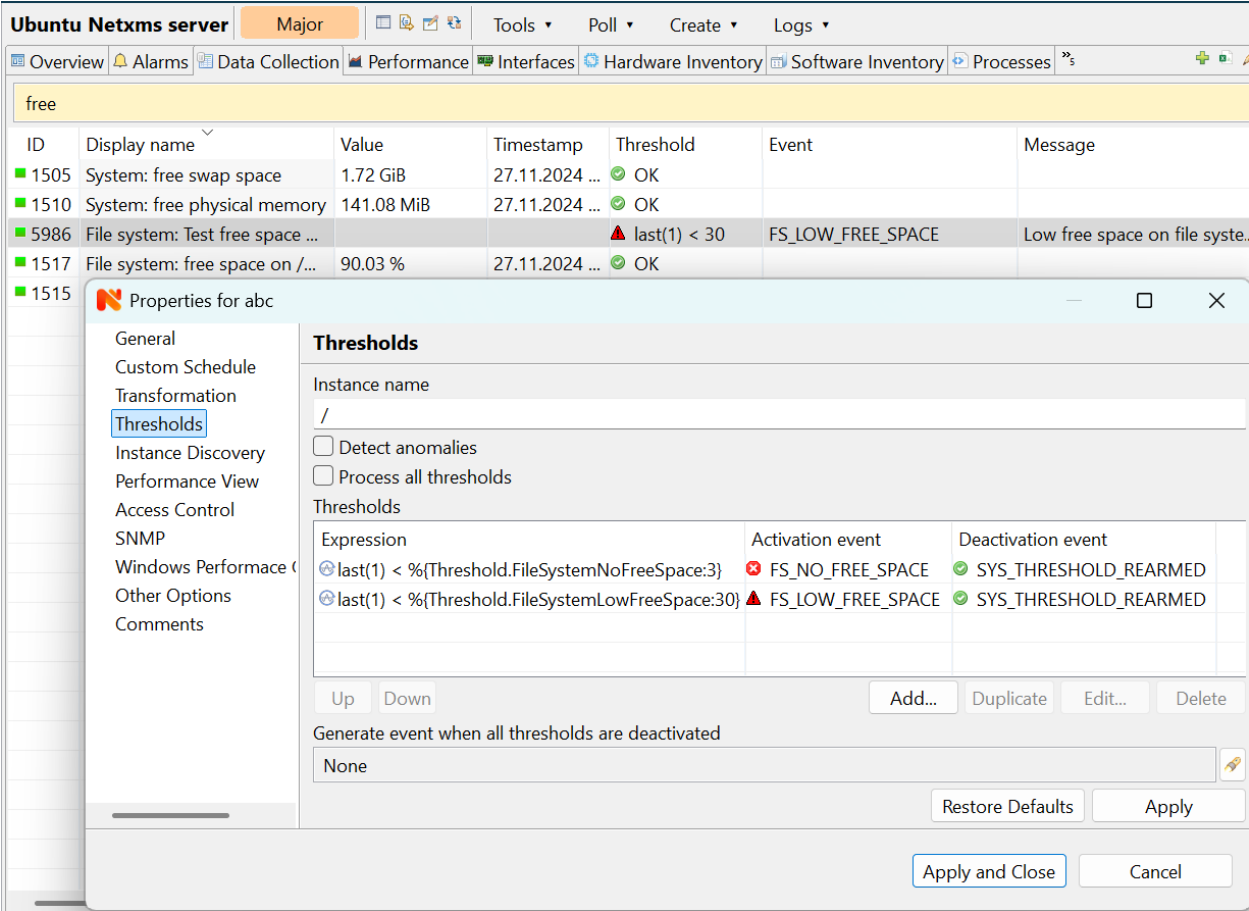


Fig. 5: DCI configuration threshold property page

Threshold Processing

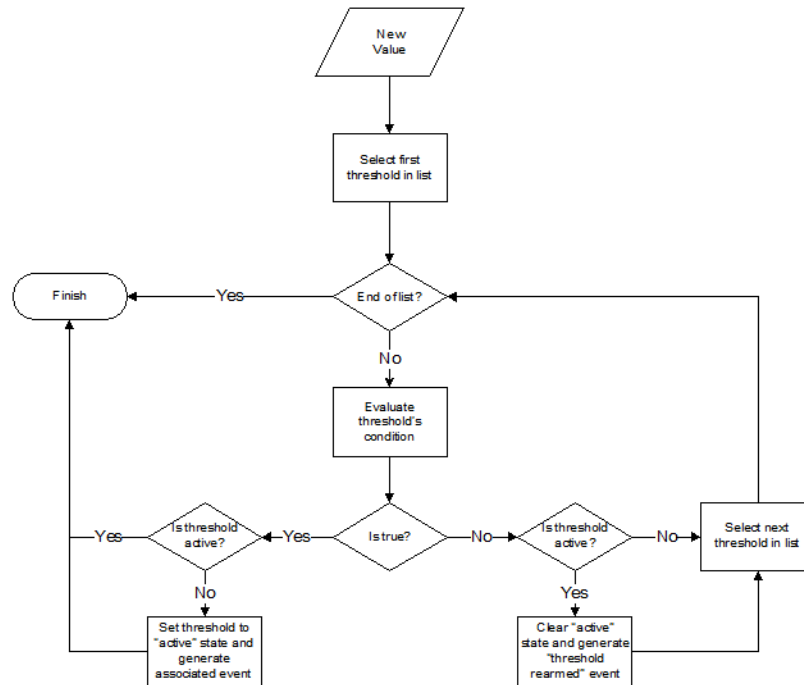


Fig. 6: Threshold processing algorithm

As you can see from above flowchart, threshold order is very important. Let's consider the following example: you have DCI representing CPU utilization on the node, and you wish two different events to be generated - one when CPU utilization exceeds 50%, and another one when it exceeds 90%. What happens when you place threshold > 50 first, and > 90 second? The following table shows values received from host and actions taken by monitoring system (assuming that all thresholds initially unarmed):

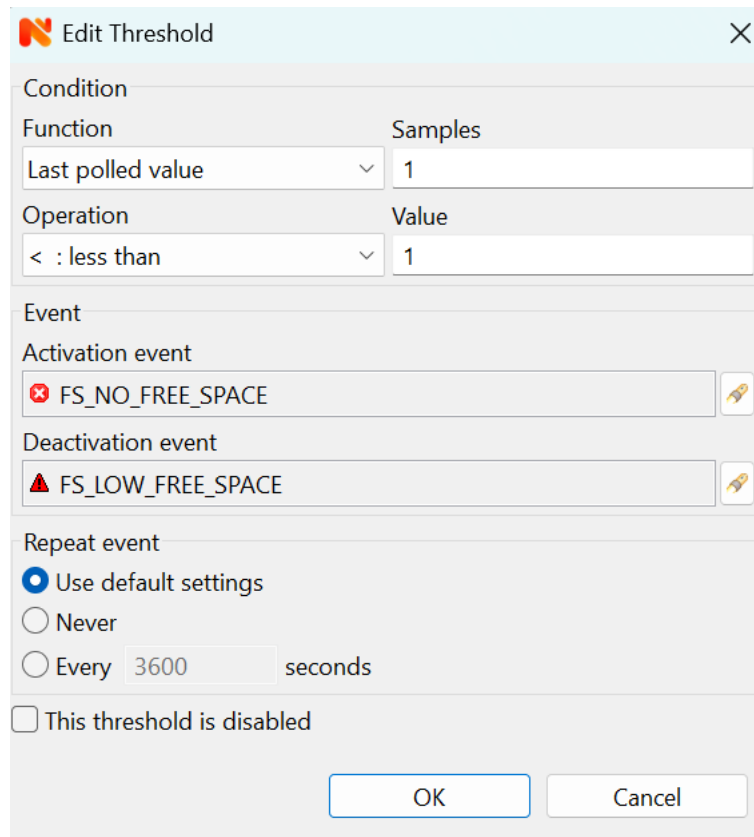
Value	Action
10	Nothing will happen.
55	When checking first threshold (> 50), the system will find that it's not active, but condition evaluates to true. So, the system will set threshold state to "active" and generate event associated with it.
70	When checking first threshold (> 50), the system will find that it's already active, and condition evaluates to true. So, the system will stop threshold checking and will not take any actions.
95	When checking first threshold (> 50), the system will find that it's already active, and condition evaluates to true. So, the system will stop threshold checking and will not take any actions.

Please note that second threshold actually is not working, because it is masked by the first threshold. To achieve desired results, you should place threshold > 90 first, and threshold > 50 second.

You can disable threshold ordering by checking *Always process all thresholds* checkbox. If enabled, system will always process all thresholds.

Threshold Configuration

When adding or modifying a threshold, you will see the following dialog:



The dialog box is titled "Edit Threshold" and contains the following sections:

- Condition**
 - Function**: A dropdown menu showing "Last polled value".
 - Samples**: A text input field containing the value "1".
- Operation**
 - Operation**: A dropdown menu showing "< : less than".
 - Value**: A text input field containing the value "1".
- Event**
 - Activation event**: A text input field containing "FS_NO_FREE_SPACE" with a red 'x' icon on the left and a pencil icon on the right.
 - Deactivation event**: A text input field containing "FS_LOW_FREE_SPACE" with a red triangle icon on the left and a pencil icon on the right.
- Repeat event**
 - ☒ Use default settings
 - ☐ Never
 - ☐ Every 3600 seconds
- ☐ This threshold is disabled

At the bottom right, there are two buttons: "OK" and "Cancel".

First, you have to select what value will be checked:

Last polled value	The last value will be used. If number of polls is set to more than 1, then condition will evaluate to true only if it's true for each individual value of last N polls.
Average value	Average value for last N polls will be used (you have to configure required number of polls).
Mean deviation	Mean absolute deviation for last N polls will be used (you have to configure required number of polls). Additional information on how mean absolute deviation is calculated can be found here .
Diff with previous value	Delta between the last and previous values will be used. If DCI data type is string and the last and previous values match, system will use 0, and if they don't - 1.
Data collection error	An indicator of data collection error. Instead of DCI's value, system will use 0 if data collection was successful, and 1 if there was a data collection error. You can use this type of thresholds to catch situations when DCI's value cannot be retrieved from agent.
Sum of values	Sum DCI values for the number of samples specified and will compare it with the value. Side note - in THRESHOLD_REACHED there are two parameters - one is last DCI value and the other is value calculated by the threshold, and if number of samples is >1, then these values can be different.
Script	This will enable script editor, so one can make a script that makes a decision. If it returns true it means to trigger the threshold, if false - rearm threshold. There are some variables available inside the script, \$dci, \$1 etc. Value input field (which is below Samples) can be read from there, which can be convenient, as one can still use this field to store some threshold value.
Absolute deviation	Similar to mean deviation - will take number of datapoints specified in Samples and calculate deviation from these.
Anomaly	If checkbox "Detect anomalies" is selected, server will use Isolation Forest algorithm to check if new value is an outlier within two set of data points - all values within 30 minutes of current time of the day for last 30 days, and all values within 30 minutes around current time of the day on the same day of the week for last 10 weeks. If new data point is classified as outlier in both data sets, DCI will be marked as having anomalous value. Using this setting may adversely affect your database performance. This is an experimental feature - use with caution.

Second, you have to select comparison function. Please note that not all functions can be used for all data types. Below is a compatibility table:

Type/Function	Integer	Unsigned Integer	Counter 32-bit	Integer 64-bit	Unsigned Integer 64-bit	Counter 64-bit	Float	String
Less	X	X	X	X	X	X	X	
Less or equal	X	X	X	X	X	X	X	
Equal	X	X	X	X	X	X	X	X
Greater or equal	X	X	X	X	X	X	X	
Greater	X	X	X	X	X	X	X	
Not equal	X	X	X	X	X	X	X	X
Like								X
Not like								X
Like (ignore case)								X
Not like (ignore case)								X

Third, you have to set a value to check against. If you use `like` or `not like` functions, value is a pattern string where you can use meta characters - asterisk (*), which means "any number of any characters", and/or question mark (?), which

means “any character”.

If you use numeric threshold value, the following multipliers are supported: K, M, G, T, Ki, Mi, Gi, Ti. So, e.g. instead of value “1000000000” you can put “1G” into the *Value* field.

Fourth, you have to select events to be generated when the condition becomes true or returns to false. By default, system uses `SYS_THRESHOLD_REACHED` and `SYS_THRESHOLD_REARMED` events, but in most cases you will change it to your custom events.

You can also configure threshold to resend activation event if threshold’s condition remain true for specific period of time. You have three options - default, which will use server-wide settings, never, which will disable resending of events, or specify interval in seconds between repeated events.

Thresholds and Events

You can choose any event to be generated when threshold becomes active or returns to inactive state. However, you should avoid using predefined system events (their names usually start with `SYS_` or `SNMP_`). For example, you may set event `SYS_NODE_CRITICAL` to be generated when CPU utilization exceeds 80%. System will generate this event, but it will also generate the same event when node status will change to *CRITICAL*. In your event processing configuration, you will be unable to determine actual reason for that event generation, and probably will get some unexpected results. If you need custom processing for specific threshold, you should create your own event first, and use this event in the threshold configuration. NetXMS has some preconfigured events that are intended to be used with thresholds. Such event names start with `DC_`.

System will pass the following parameters to events generated as a reaction to single-value DCI threshold violation:

Parameter number	Named parameter	Description
1	<code>dciParam</code>	Data collection item name
2	<code>dciParamDescription</code>	Data collection item description
3	<code>thresholdValue</code>	Threshold value
4	<code>currentValue</code>	Current value (e.g. average for several samples for averaging threshold) that is compared to threshold value
5	<code>dciParamId</code>	Data collection item ID
6	<code>instance</code>	Instance
7	<code>isRepeatedEvent</code>	Repeat flag
8	<code>dciParamValue</code>	Last collected DCI value
9	<code>operation</code>	Threshold’s operation code
10	<code>function</code>	Threshold’s function code
11	<code>pollCount</code>	Threshold’s required poll count
12	<code>thresholdDefinition</code>	Threshold’s textual definition

Event parameters can be accessed by number or by name via macros to form event message. For example, if you are creating a custom event that is intended to be generated when file system is low on free space, and wish to include file system name, actual free space, and threshold’s value into event’s message text, you can use message template like this:

```
File system %<instance> has only %<currentValue> bytes of free space (threshold:
%<thresholdValue> bytes)
```

For table threshold violation the following parameters are passed to generated events:

Parameter number	Named parameter	Description
1	dciName	Table DCI name
2	dciDescription	Table DCI description
3	dciId	Table DCI ID
4	row	Table row
5	instance	Instance

For events generated on threshold's return to inactive state (default event is `SYS_THRESHOLD_REARMED`), event parameter list is different:

Parameter number	Named parameter	Description
1	dciName	Data collection item name
2	dciDescription	Data collection item description
3	dciId	Data collection item ID
4	instance	Instance
5	thresholdValue	Threshold value
6	currentValue	Current value (e.g. average for several samples for averaging threshold) that is compared to threshold value
7	dciValue	Last collected DCI value
8	operation	Threshold's operation code
9	function	Threshold's function code
10	pollCount	Threshold's required poll count
11	thresholdDefinition	Threshold's textual definition

For table DCI threshold rearm the following parameters are passed to generated events:

Parameter number	Named parameter	Description
1	dciName	Table DCI name
2	dciDescription	Table DCI description
3	dciId	Table DCI ID
4	row	Table row
5	instance	Instance

12.2.5 Instance

Each DCI has an *Instance* attribute, which is a free-form text string, passed as a 6th parameter to events associated with thresholds. You can use this parameter to distinguish between similar events related to different instances of the same entity. For example, if you have an event generated when file system was low on free space, you can set the *Instance* attribute to file system mount point.

Sometimes you may need to monitor multiple instances of some entity, with exact names and number of instances not known or different from node to node. Typical example is file systems or network interfaces. To automate creation of DCIs for each instance, you can use instance discovery mechanism. First you have to create "master" DCI. Create DCI as

usual, but in places where normally you would put instance name, use the special macro {instance}. Then, go to *Instance Discovery* tab in DCI properties, and configure instance discovery method and optionally filter script.

Instance discovery creates 2 macros for substitution:

- {instance} - instance name
- {instance-name} - instance user-readable description

The screenshot shows the 'Properties for FileSystem.UsedPerc({instance})' dialog box. The 'Instance Discovery' tab is selected in the left sidebar. The main area contains the following configuration:

- Instance discovery method:** Agent List
- List name:** FileSystem.MountPoints
- Instance retention:**
 - Instance retention mode: Server default
 - Instance retention time (days): 0
- Instance discovery filter script:**
 - Hints:**

```

1 if (($1 == "/proc") or
2   ($1 like "/proc/*") or
3   ($1 == "/sys") or
4   ($1 like "/sys/*") or
5   ($1 == "/run") or
6   ($1 like "/run/*") or
7   ($1 == "/dev") or
8   ($1 like "/dev/*") or
9   ($1 == "/var/snap/firefox/common/host-hunspell") or
10  ($1 like "/var/snap/*"))
11   return false;
12
13 excludedFSTypes = %(
14   "ahafs",
15   "aufs",
16   "autofs",
17   "cgroup",
18   "configfs"

```

At the bottom of the dialog, there are buttons for 'Restore Defaults', 'Apply', 'Apply and Close', and 'Cancel'.

Fig. 7: DCI configuration instance discovery property page

Instance Discovery Methods

The following instance discovery methods are available:

Method	Input Data	Description
Agent List	List name	Read list from agent and use it's values as instance names.
Agent Table	Table name	Read table from agent and use it's instance column values as instance names. If there are several instance columns in that table, a concatenation of values will be used, separated by ~~~ (three tilda characters).
SNMP Walk - Values	Base OID	Do SNMP walk starting from given OID and use values of returned varbinds as instance names.
SNMP Walk - OIDs	Base OID	Do SNMP walk starting from given OID and use IDs of returned varbinds as instance names.
Script	Script name	Instance names are provided by a script from script library. The script should return an array (with elements representing instance names) or a map (keys represent instance names and values represent user-readable description)
Windows Performance Counters	Object name, e.g. LogicalDisk.	Instances of given object will be taken.
Web Service	Definition: path	Web service request field contains web service definition name with optional arguments and path to the root element of the document where enumeration will start. Each sub-element of given root element will be considered separate instance.
Internal Table	Table name	Read NetXMS server internal table and use it's instance column values as instance names. If there are several instance columns in that table, a concatenation of values will be used, separated by ~~~ (three tilda characters).

Instance Discovery Filter Script

You can optionally filter out unneeded instances, transform instance names and add user-readable description using filtering script written in NXSL. Script will be called for each instance and can return either a binary value or an array.

If binary value is returned, it has the following meaning: `TRUE` (to accept instance), `FALSE` (to reject instance).

If an array is returned, then instance is counted as accepted. Only first element of the array is mandatory, the rest elements are optional (but to include an element, all preceding elements should be included). Array structure:

Data type	Description
String	Instance name, that will be available as {instance} macro.
String	Instance user-readable description, that will be available as {instance-name} macro
NetObj	Object connected with this DCI

12.2.6 Performance view

This section provides configuration options for displaying DCI values as line charts on the *Performance* tab. Various options are available to visually represent the collected data; see [Data and Network visualization](#) for more details.

Note

Note: Not available for table metrics.

Properties for FileSystem.UsedPerc({instance})

Performance View

General
 Custom Schedule
 Transformation
 Thresholds
 Instance Discovery
Performance View
 Access Control
 SNMP
 Windows Performance
 Other Options
 Comments

Performance View

Warning: This DCI was added by template "Generic UNIX"
 All local changes can be overwritten at any moment

☒ Show in performance view

Title: File system Average

Group: OS_DiskAverage

Name in legend: 15 minutes

Time Period

Time interval: 1 Time units: Hours

Color: ☐ Automatic ☒ Custom (Green)

Type: Line

Order: 100

Options

☐ Show thresholds on graph
☐ Logarithmic scale
☐ Stacked
☐ Always show legend
☒ Extended legend
☒ Use multipliers
☐ Inverted values
☒ Translucent

Y Axis Settings

☒ Automatic range ☐ Set Y base to min value

☐ Manual range From: 1.0 To: 100.0

Axis label (leave empty to hide):

Restore Defaults Apply

Apply and Close Cancel

Fig. 8: DCI configuration instance discovery property page

Multiple DCIs can be grouped in one graph. To group them use the same group name in "Group" field.

12.2.7 Access Control

This page provides access control management option to each DCI. If no user set, then access rights are inherited from node. So any user that is able to read node is able to see last value of this DCI and user that is able to modify node is able to change and see DCI configuration. When list is not empty, then both access to node and access to DCI are checked on DCI configuration or value request.

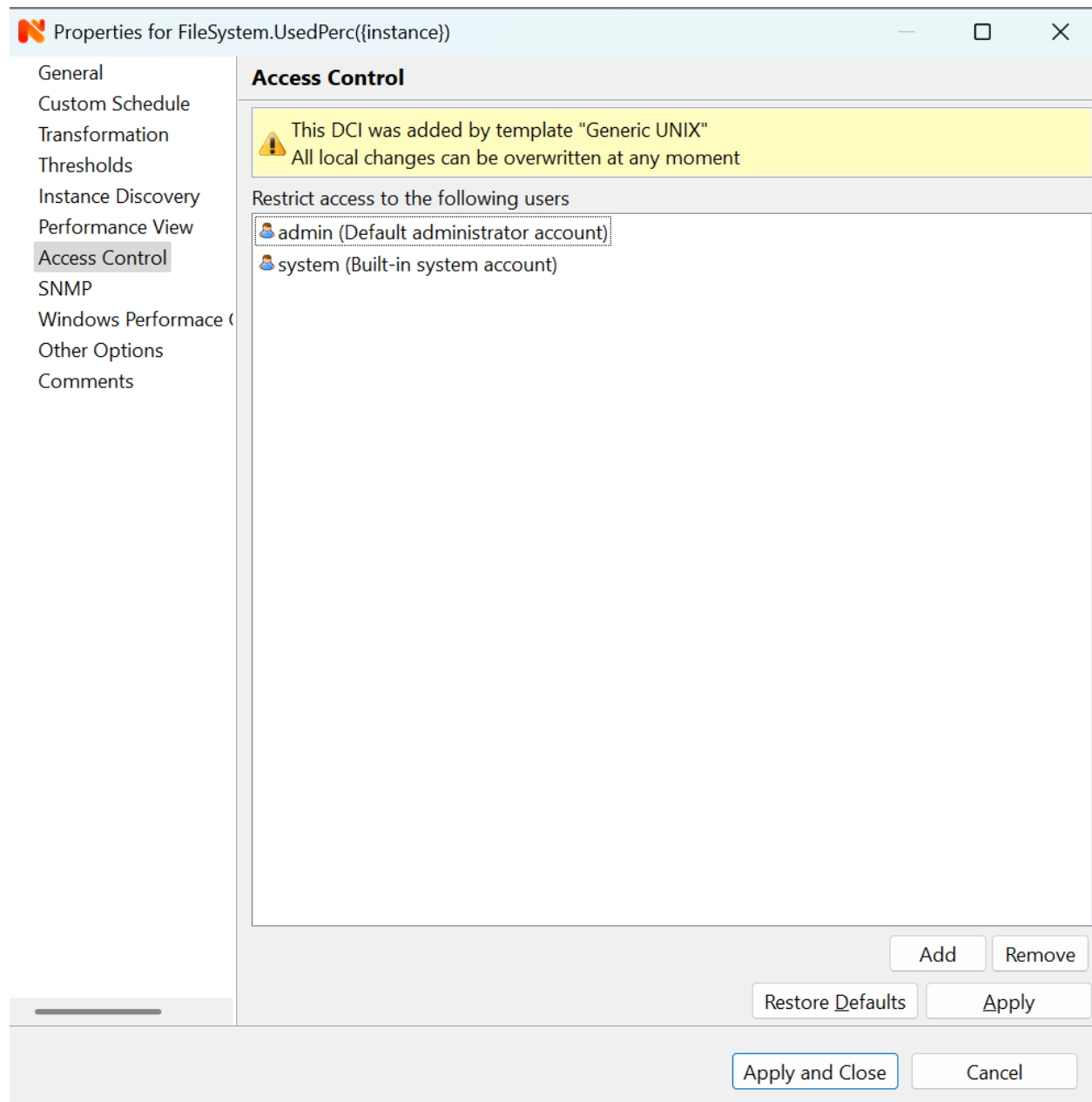
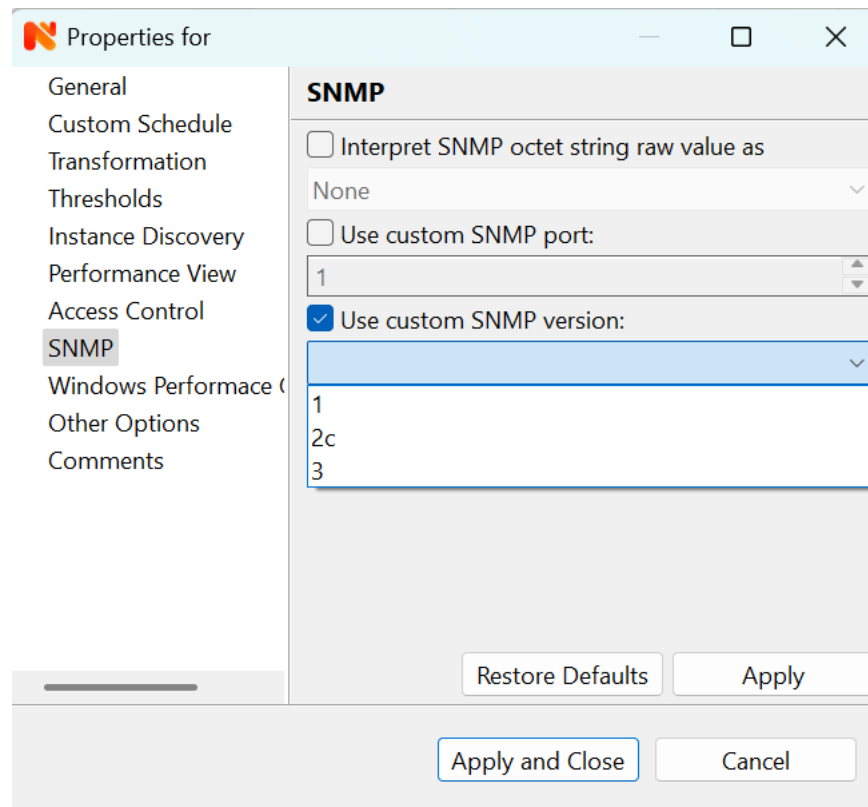


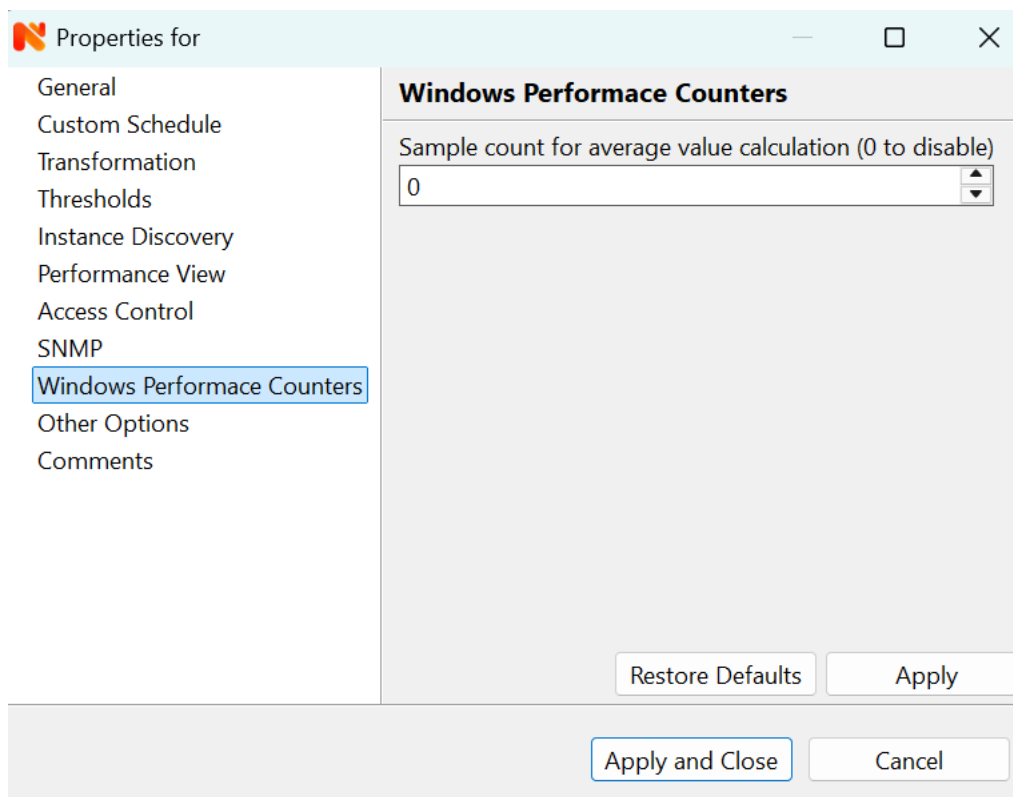
Fig. 9: DCI configuration access control property page

12.2.8 SNMP

SNMP page provides additional options for SNMP data collection or processing. Like: how to interpret collected SNMP octet string or to use custom port or version for data collection.



12.2.9 Windows Performance Counters



12.2.10 Other options

Other available options:

- Show last value in object tooltip - shows DCI last value on tooltip that is shown on network maps.
- Show last value in object overview - shows DCI last value on *Overview->Last Values* page.
- Use this DCI for node status calculation - Uses value returned by this DCI as a status, that participate in object status calculation. Such kind of DCI should return integer number from 0 till 4 representing object status.
- Related object - object that is related to collected DCI. Related object can be set by instance discovery filter script and accessed in NXSL from DCI object.

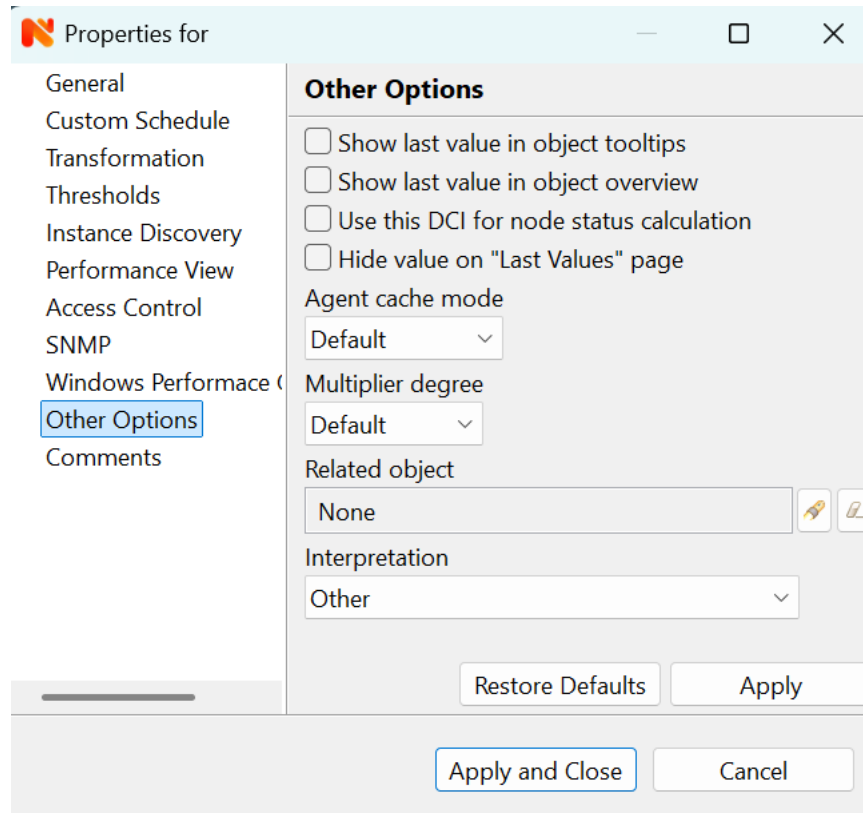


Fig. 10: DCI configuration other option property page

12.2.11 Comments

This configuration page can be used freely for text comments to add additional notes about DCI configuration or usage. These comments are added to alarms created from threshold violation events. For example, they can be used to inform operators about problem-solving approaches.

12.3 Push metrics

NetXMS gives you ability to push DCI values when you need it instead of polling them on specific time intervals. To be able to push data to the server, you should take the following steps:

1. Set your DCI's origin to Push Agent and configure other properties as usual, excluding polling interval which is meaningless in case of pushed data.

2. Create separate user account or pick an existing one and give “Push Data” access right on the DCI owning node to that user.
3. Use *nxapush* or *npxpush* utility or client API for pushing data.

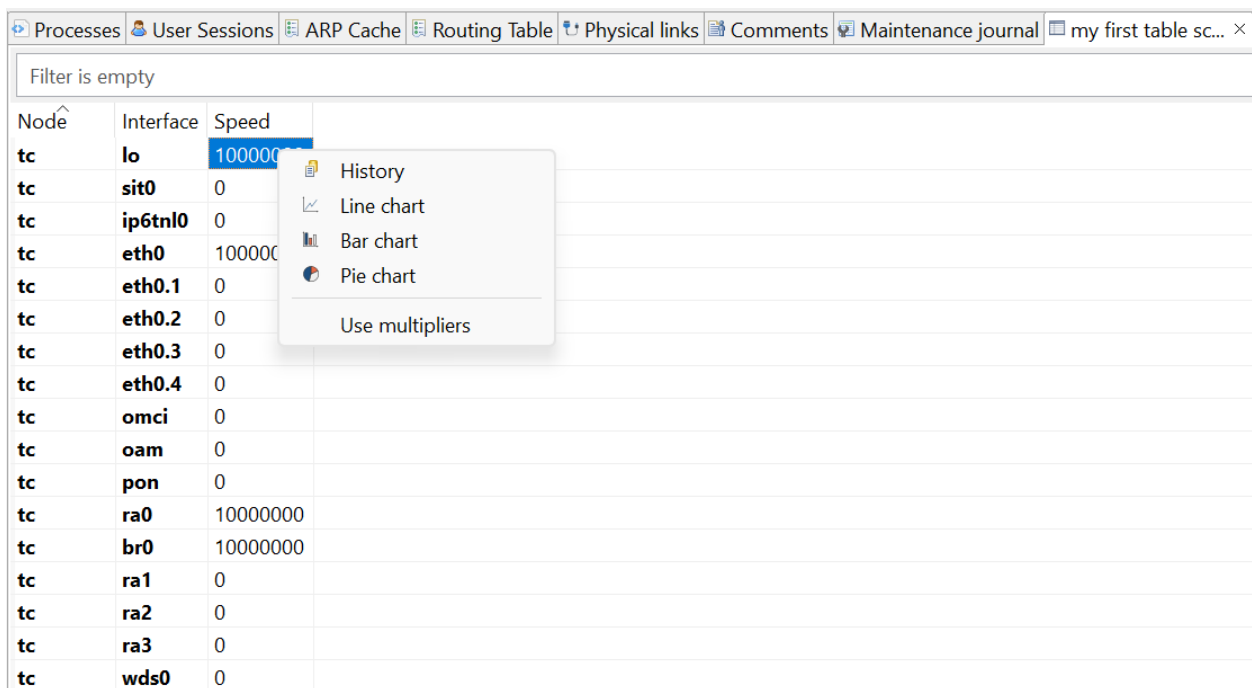
12.4 DCI types

12.4.1 Single-value DCIs

Single-value metrics, as the name suggests, collect only one data value.

12.4.2 Table DCIs

Table metrics can collect data in bulk, effectively encapsulating multiple values that can be collected simultaneously.



Node	Interface	Speed
tc	lo	1000000
tc	sit0	0
tc	ip6tnl0	0
tc	eth0	100000
tc	eth0.1	0
tc	eth0.2	0
tc	eth0.3	0
tc	eth0.4	0
tc	omci	0
tc	oam	0
tc	pon	0
tc	ra0	10000000
tc	br0	10000000
tc	ra1	0
tc	ra2	0
tc	ra3	0
tc	wds0	0

Fig. 11: Table example

They're primarily used when it is necessary to gather bulk data, like data sets that can be acquired together or for atomic collection. Atomic collection is when you need to take a data snapshot that consists of multiple items collected at the exact same time. By right-click on string or non string value one can access history, and line chart builds are possible for non string values.

There are distinct benefits to using table metrics. But they're not without their disadvantages. As tables are not single values, they require more storage, which can be one of the potential drawbacks.

Furthermore, the threshold configuration can be more complicated for table metrics because they have multiple rows and columns.

Unlike a single value where you can easily specify a threshold for when something is wrong, with a table, you have to specify which instance or item in a column has an issue.

12.4.3 List DCIs

Usually DCIs have scalar values. A list DCI is a special DCI which returns a list of values. List DCIs are mostly used by NetXMS internally (to get the list of network interfaces during the configuration poll, for example), but can also be utilized by user in some occasions. NetXMS Management Client does not support list DCIs directly, but their names are used as input parameters for Instance Discovery methods. List DCI values can be also obtained with **nxget** command line utility (e.g. for use in scripts).

12.5 Agent caching mode

Agent caching mode allows metric data to be obtained for the time being while connection between server and agent have been broken. This option is available for metrics, table metrics and proxy SNMP metrics as well as for proxy SNMP table metrics and DCIs with custom schedule. In absence of connection to the server, collected data is stored on agent and once connection is restored, data is sent to server. Detailed description can be found there: [How data collection works](#).

Agent side cache is configurable globally, on node and DCI levels. Configuration can be changed separately on each level. By default it's off.

All collected data goes through all transformations and thresholds only when it comes to server. In order to prevent generation of old events, one can set *DataCollection.OfflineDataRelevanceTime* configuration variable to time period in seconds within which received offline data still relevant for threshold validation. By default it is set to 1 day.

12.5.1 Configuration

Agent cache mode can be configured:

- globally - set configuration parameter *Agent.DefaultCacheMode* to *on* or *off* in *Configuration* perspective -> *Server configuration*.
- on node level - *Agent cache mode* can be changed to *on*, *off* or *default* (use global settings). Right click on a node in *Infrastructure* perspective and select *Properties* followed by *Polling* page.
- on DCI level - *Agent cache mode* can be changed to *on*, *off* or *default* (use node level settings) in DCI properties on *Other Options* page.

12.6 Data Collection tab

Data Collection tab provides information about all data collected on a node: DCI last value, last collection timestamp and threshold status.

It is possible to check last values or raw last values in textual format or as a chart by right clicking on DCI and selecting corresponding display format.

LAPTOP-K0E8LJNN Minor Tools Poll Create Logs

Overview Alarms Data Collection Performance Interfaces Hardware Inventory Software Inventory Services Processes User Sessions

Filter is empty

ID	Display name	Value	Timestamp	Threshold	Event	Message
1645	System: used physical memory	1.15 GiB	28.11.2024 10:46:34	OK		
1646	CPU: usage	23 %	28.11.2024 10:46:34	OK		
1649	System: free virtual memory	20 %	28.11.2024 10:46:34	OK		
1650	Agent thread pool TIMER: current load	0 %	28.11.2024 10:46:34	OK		
1651	Agent thread pool COMM: current load	12 %	28.11.2024 10:46:34	OK		
1652	Agent thread pool WEBSVC: current load	0 %	28.11.2024 10:46:34	OK		
1653	Agent thread pool PROCEXEC: current load	0 %	28.11.2024 10:46:34	OK		
1654	Agent thread pool DATACOLL: current load	0 %	28.11.2024 10:46:34	OK		
1655	Agent thread pool TIMER: current size	2	28.11.2024 10:46:34	OK		
1656	Agent thread pool COMM: current size	8	28.11.2024 10:46:34	OK		
1657	Agent thread pool WEBSVC: current size	4	28.11.2024 10:46:34	OK		
1658	Agent thread pool PROCEXEC: current size	1	28.11.2024 10:46:34	OK		
1659	Agent thread pool DATACOLL: current size	4	28.11.2024 10:46:34	OK		
1660	Agent thread pool TIMER: normalized load average...	0	28.11.2024 10:46:34	OK		
1661	Agent thread pool COMM: normalized load average...	0	28.11.2024 10:46:34	OK		
1662	Agent thread pool WEBSVC: normalized load average...	0	28.11.2024 10:46:34	OK		
1663	Agent thread pool PROCEXEC: normalized load average...	0	28.11.2024 10:46:34	OK		
1664	Agent thread pool DATACOLL: normalized load average...	0	28.11.2024 10:46:34	OK		
1665	Agent thread pool TIMER: usage	12 %	28.11.2024 10:46:34	OK		
1666	Agent thread pool COMM: usage	1 %	28.11.2024 10:46:34	OK		
1667	Agent thread pool WEBSVC: usage	6 %	28.11.2024 10:46:34	OK		
1668	Agent thread pool PROCEXEC: usage	6 %	28.11.2024 10:46:34	OK		

Click on *Edit mode* to obtain more detailed view.

LAPTOP-K0E8LJNN Minor Tools Poll Create Logs

Overview Alarms Data Collection Performance Interfaces Hardware Inventory Software Inventory Services Processes User Sessions

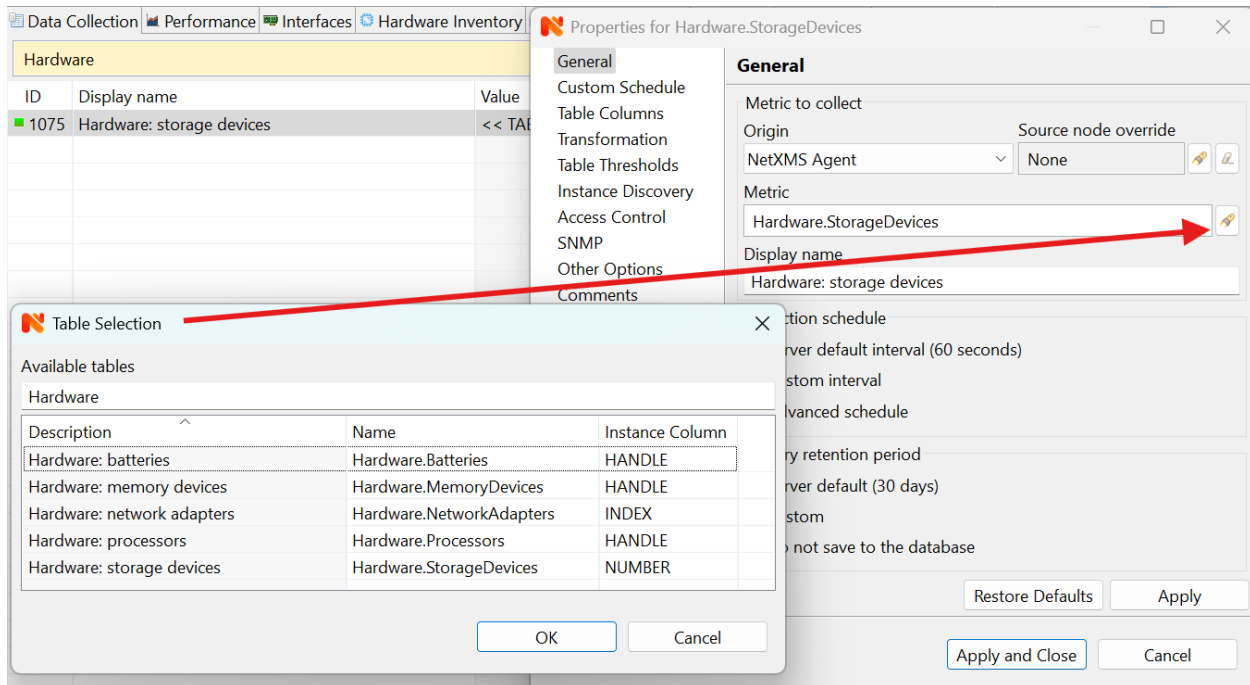
Filter is empty

ID	Display name	Origin	Metric	Units	Data Type	Polling I...	Retentio...	Status	Thresholds	Template
16...	Total number of processes	NetXMS Agent	System.ProcessCount		Unsigned Int...	default	default	Active		Templates/Operating System...
16...	System: used physical mem...	NetXMS Agent	System.Memory.Physical.U...	B (IEC)	Unsigned Int...	default	default	Active		Templates/Operating System...
16...	System: total threads	NetXMS Agent	System.ThreadCount		Unsigned Int...	default	default	Active		Templates/Operating System...
16...	System: free virtual memory	NetXMS Agent	System.Memory.Virtual.Fre...	%	Float	default	default	Active	last(1) < 10	Templates/Operating System...
16...	System: free physical mem...	NetXMS Agent	System.Memory.Physical.Fr...	B (IEC)	Unsigned Int...	default	default	Active		Templates/Operating System...
16...	Server: Total Number of Ob...	Internal	Server.ObjectCount.Total		Unsigned Int...	86400	365 days	Active		Templates/System/NetXMS S...
16...	Server: Syncer Min Run Time	Internal	Server.SyncerRunTime.Min	ms	Integer 64-bit	default	default	Active		Templates/System/NetXMS S...
16...	Server: Syncer Max Run Time	Internal	Server.SyncerRunTime.Max	ms	Integer 64-bit	default	default	Active		Templates/System/NetXMS S...
16...	Server: Syncer Last Run Time	Internal	Server.SyncerRunTime.Last	ms	Integer 64-bit	default	default	Active		Templates/System/NetXMS S...
16...	Server: Syncer Average Run...	Internal	Server.SyncerRunTime.Aver...	ms	Integer 64-bit	default	default	Active		Templates/System/NetXMS S...
16...	Server: Received Windows ...	Internal	Server.ReceivedWindowsEv...	events/...	Counter 64-bit	default	default	Active		Templates/System/NetXMS S...
16...	Server: Received Syslog Me...	Internal	ReceivedSyslogMessages	messag...	Counter 64-bit	default	default	Active		Templates/System/NetXMS S...
16...	Server: Received SNMP Trap...	Internal	ReceivedSNMPTraps	traps/m...	Counter 64-bit	default	default	Active		Templates/System/NetXMS S...
16...	Server: Number of Sensors	Internal	Server.ObjectCount.Sensors		Unsigned Int...	86400	365 days	Active		Templates/System/NetXMS S...
16...	Server: Number of Processe...	Internal	Server.TotalEventsProcessed	events/...	Unsigned Int...	default	default	Active		Templates/System/NetXMS S...
16...	Server: Number of Nodes	Internal	Server.ObjectCount.Nodes		Unsigned Int...	86400	365 days	Active		Templates/System/NetXMS S...
16...	Server: Number of Data Col...	Internal	Server.DataCollectionItems		Unsigned Int...	86400	365 days	Active		Templates/System/NetXMS S...
16...	Server: Number of Clusters	Internal	Server.ObjectCount.Clusters		Unsigned Int...	86400	365 days	Active		Templates/System/NetXMS S...
15...	Server: Memory Used by R...	Internal	Server.MemoryUsage.Raw...	B (IEC)	Unsigned Int...	default	default	Active		Templates/System/NetXMS S...
15...	Server: Memory Used by D...	Internal	Server.MemoryUsage.Data...	B (IEC)	Unsigned Int...	default	default	Active		Templates/System/NetXMS S...

12.6.1 DCI table creation example

Encapsulating earlier covered configuration options - in *Data Collection* tab view one can, for example, create DCI table with Agent cache mode enabled in the following way:

1. Create new table by right click in *Data Collection* tab view followed by selecting *New table*....
2. Select *Origin* on *General* page as NetXMS Agent (default option) and table metrics from *Table Selection* pop-up view when clicking on *Metric* selector.



Note

Pop up view from *Metric* selector may be different for other sources in *Origin*.

Currently supported DCI table sources are:

- Internal
- NetXMS Agent
- SNMP
- Script

Currently supported DCI table sources with agent cache enabled:

- NetXMS Agent
- SNMP

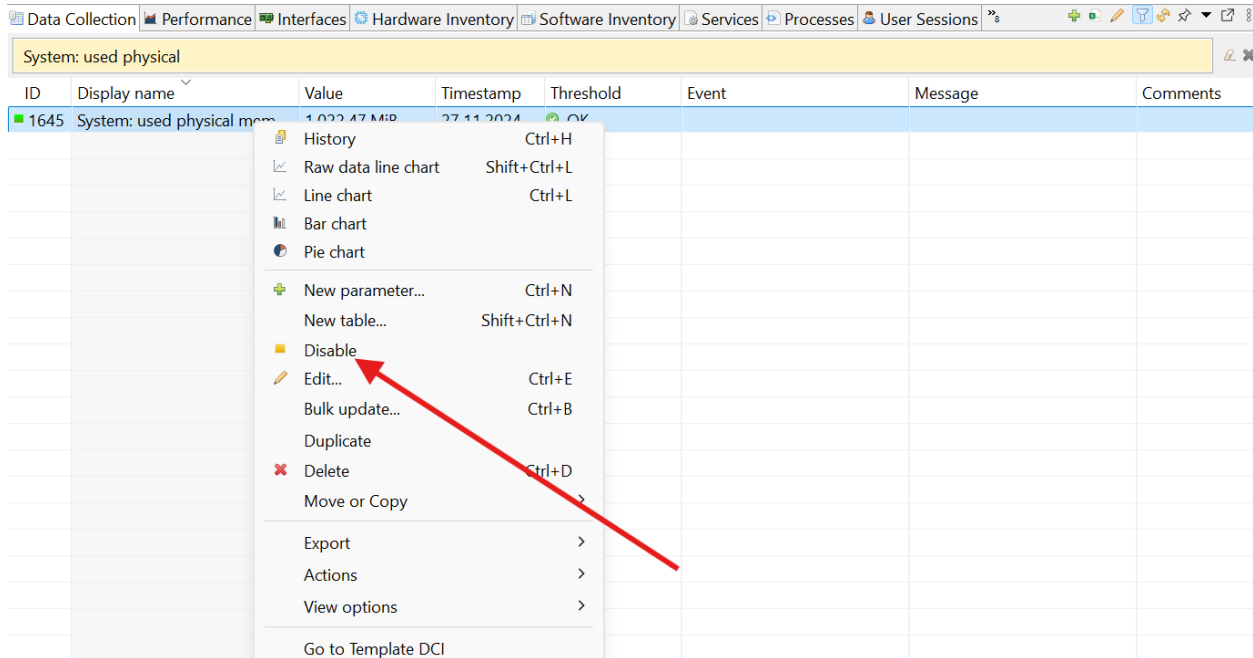
Currently supported DCI table sources with agent cache and proxy enabled:

- NetXMS Agent
- SNMP

3. Configure agent catching mode as per instructions [above](#).

12.6.2 Status

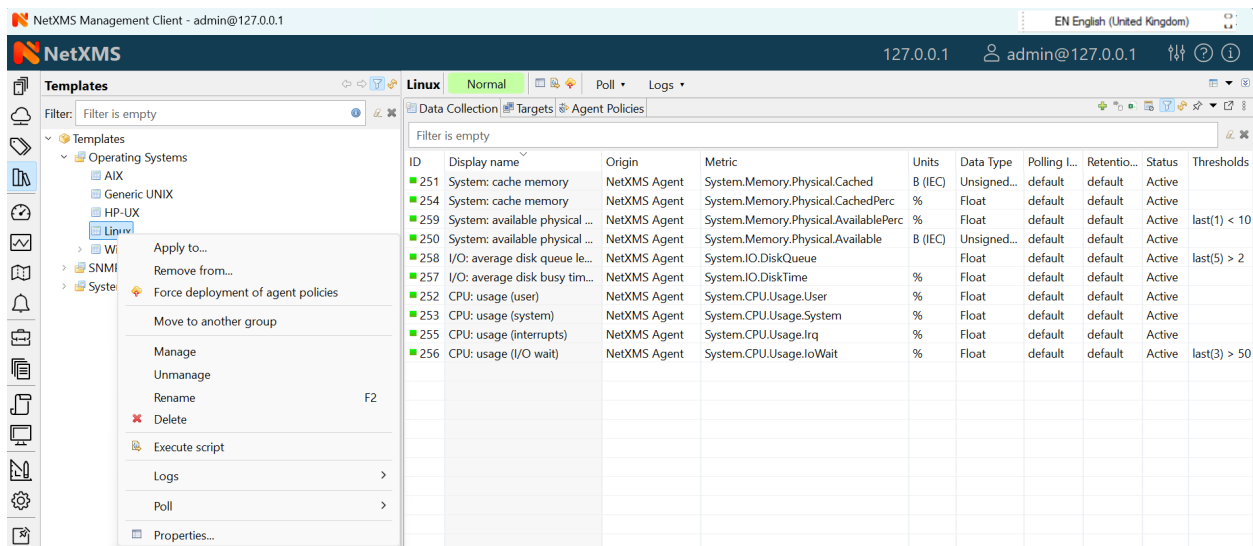
DCI status can be one of the following: *Active*, *Disabled*, *Not Supported*. Server will collect data only if the status is *Active*. If you wish to stop data collection without removing *DCI* configuration and collected data, the *Disabled* status can be set manually. If requested metric is not supported by target node, the *Not Supported* status is set by the server.



12.7 Templates

12.7.1 What is template

Often you have a situation when you need to collect same metrics from different nodes. Such configuration making may easily fall into repeating one action many times. Things may become even worse when you need to change something in already configured DCIs on all nodes - for example, increase threshold for CPU utilization. To avoid these problems, one can use data collection templates. Data collection template (or just template for short) is a special object, which can have DCIs configured and grouped for similar or logical purposes and applied to relevant node or node group (for example, Collector or Cluster in *Infrastructure* perspective). Templates can be accessed from *Template* perspective.



When you create template and configure DCIs for it, nothing happens - no data collection will occur. Then, you can apply this template to one or multiple nodes - and as soon as you do this, all DCIs configured in the template object will

appear in the target node objects, and server will start data collection for these DCIs. If you then change something in the template data collection settings - add new DCI, change DCI's configuration, or remove DCI - all changes will be reflected immediately in all nodes associated with the template. You can also choose to remove template from a node. In this case, you will have two options to deal with DCIs configured on the node through the template - remove all such DCIs or leave them, but remove relation to the template. If you delete template object itself, all DCIs created on nodes from this template will be deleted as well.

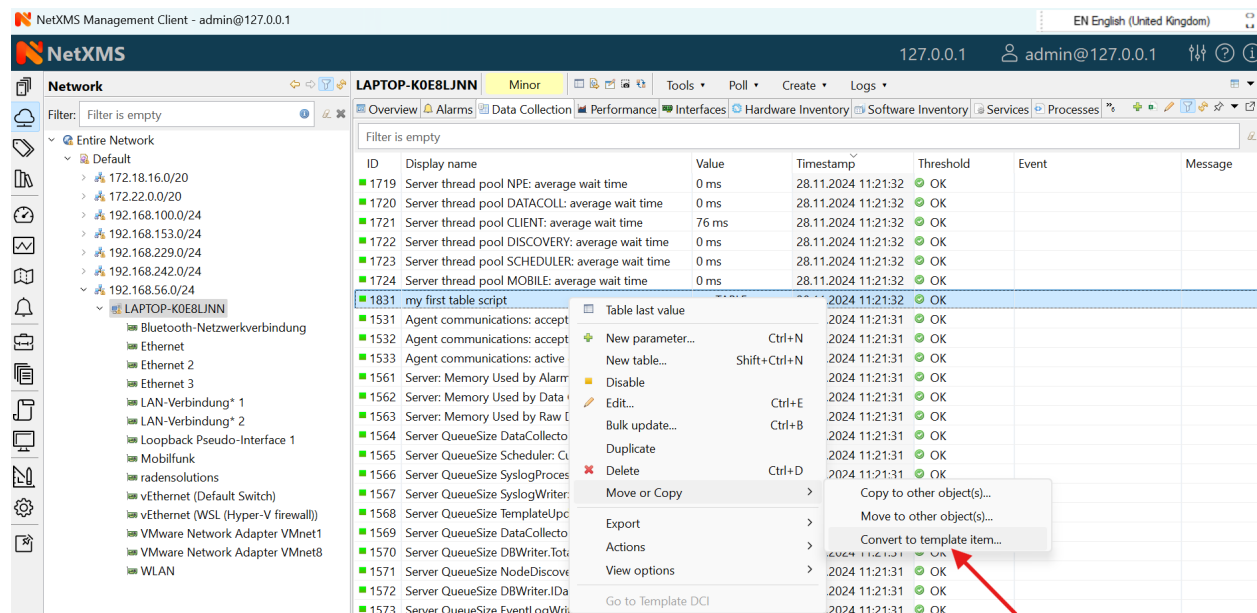
Please note that you can apply unlimited number of templates to a node - so you can create individual templates for each group of metrics (for example, generic performance metrics, MySQL metrics, network counters, etc.) and combine them, as per your business requirements.

12.7.2 Creating template

To create a template, right-click on *Template Root* or *Template group* object in *Template* perspective, and click *Create ▶ Template*. Enter a name for a new template and click *OK*.

12.7.3 Configuring templates

To configure DCIs in the template, click on *Template* object in the *Template* perspective, then right-click in *Data Collection* tab view and select *New parameter...* or *New table...* for further data collection configuration. You can configure DCIs in the same way as the node objects. Another way to apply configuration in *Template* - create DCI in *Infrastructure* or *Network* perspective and convert it to template item, as seen below.

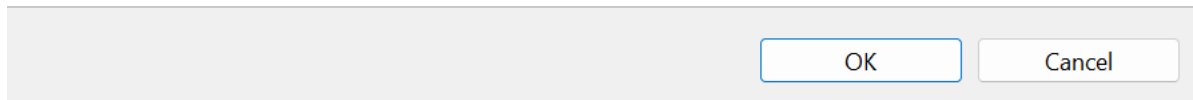
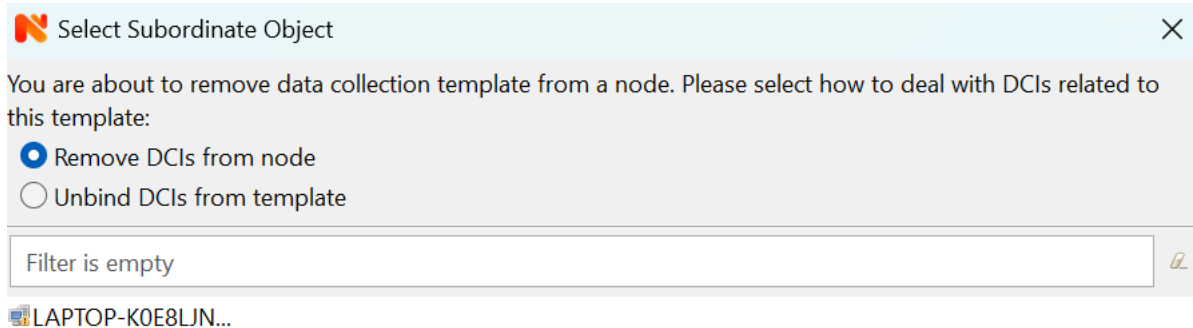


12.7.4 Applying template to node

To apply a template to one or more nodes, right-click on template object in *Template* perspective and select *Apply to...*. Pop-up menu will appear with objects in *Infrastructure* and *Network* perspectives available for selection. Select objects that you wish to apply template to, and click *OK* (you can select multiple nodes in the list by holding *Control* key). Please note that if data collection editor is open for any of the target nodes, either by you or another administrator, template applying will be delayed until data collection editor for that node will be closed. Another way to apply template to object - in *Infrastructure* or *Network* perspectives select one or more objects, right-click and select *Apply template...*

12.7.5 Removing template from node

To remove a link between template and node, right-click on *Template* object in *Template* perspective and select *Remove from....* Pop-up menu will appear with objects, which are having the template in question already applied. Select objects that you wish to remove template from, and click *OK*.



Another way to remove template from object - in *Infrastructure* or *Network* perspective select one or more objects, right-click and select *Remove template....* Pop-up window will appear with all applied templates to objects. Select templates to be removed and click *OK*.

If you select *Unbind DCIs from template*, all DCIs related to template will remain configured on a node, but association between the DCIs and template will be removed. Any further changes to the template will not be reflected in these DCIs. If you later reapply the template to the node, you will have two copies of each DCI - one standalone (remaining from unbind operation) and one related to template (from new apply operation). Selecting *Remove DCIs from node* will remove all DCIs associated with the template. After you click *OK*, node will be unbound from template.

12.7.6 Macros in template items

You can use various macros in name, description, and instance fields of template DCI. These macros will be expanded when template applies to node. Macro started with `%{` character combination and ends with `}` character. The following macros are currently available:

Macro	Expands to
node_id	Node unique id
node_name	Node name
node_primary_ip	Node primary IP address
script:name	String returned by script name. Script should be stored in script library (accessible via <i>Configuration ▶ Script Library</i>). Inside the script, you can access current node's properties via \$node variable.

For example, if you wish to insert node's IP address into DCI description, you can enter the following in the description field of template DCI:

```
My IP address is %{node_primary_ip}
```

When applying to node with primary IP address 10.0.0.1, on the node will be created DCI with the following description:

```
My IP address is 10.0.0.1
```

Please note that if you change something in the node, name for example, changes will not be reflected automatically in DCI texts generated from these macros. However, they will be updated if you reapply template to the node or on housekeeper run.

12.8 Working with collected data

Once you setup DCI, data starts collecting in the database. You can access this data and work with it in different ways. Data can be visualized in three ways: in graphical form, as a historical view(textual format) and as DCI summary table, this layout types can be combined in Dashboards. More detailed description about visualization and layout can be found there: *Data and Network visualisation*.

EVENT PROCESSING

13.1 Introduction

NetXMS is event based monitoring system. Events can come from different sources - polling processes (status, configuration, discovery), data collection, *SNMP* traps, from NXSL scripts and directly from external applications via client library. All events are forwarded to NetXMS Event Queue.

NetXMS Event Processor can process events from Event Queue in either sequential or parallel mode. In sequential mode events are processed one-by-one which guarantees that events will be processed in the same sequence as they arrive into the queue. For installation where a lot of events could be generated in a short period of time this mode can be a bottleneck.

Parallel processing mode allows to process events in several parallel threads, thus allowing to scale horizontally and to increase processing performance. Number of threads for parallel processing is set by *Events.Processor.PoolSize* server configuration parameter.

Event Processing Rules can read/write persistent storage and custom attributes, create/terminate alarms, can run scripts that are checking other node statuses and care should be taken to ensure that no race condition would occur when performing parallel processing.

Correct operation is ensured by properly setting *Events.Processor.QueueSelector* server configuration parameter. This parameter contains macros that are expanded when an event is created. Events that have same QueueSelector string will be processed sequentially by one and the same event processing thread, thus ensuring that there will be no race condition between these events.

13.2 Event Processing Policy

Actions taken by event processor for any specific event are determined by a set of rules called *Event Processing Policy* (EPP).

Every rule has two parts - matching part (called *Condition* in the rule configuration dialog), which determines if the rule is applicable to an event, and action part, which defines actions to be taken for matched events.

Each event passes through all rules in the policy, so if it matches more than one rule, actions specified in all matched rules will be executed. You can change this behavior by setting Stop Processing flag on a rule. If this flag is set for a rule and that rule is matched, subsequent rules (with higher rule number) will not be processed.

Event Processing Policy rules are managed using *Event Processing Policy Editor* available in *Configuration* → *Event Processing Policy*.

Only one user of NetXMS server can access *Event Processing Policy Editor* window at a time. Other users will receive `Component locked` error message when attempting to open this window.

Changes made in *Event Processing Policy Editor* are applied at the moment when `Save` button is clicked.

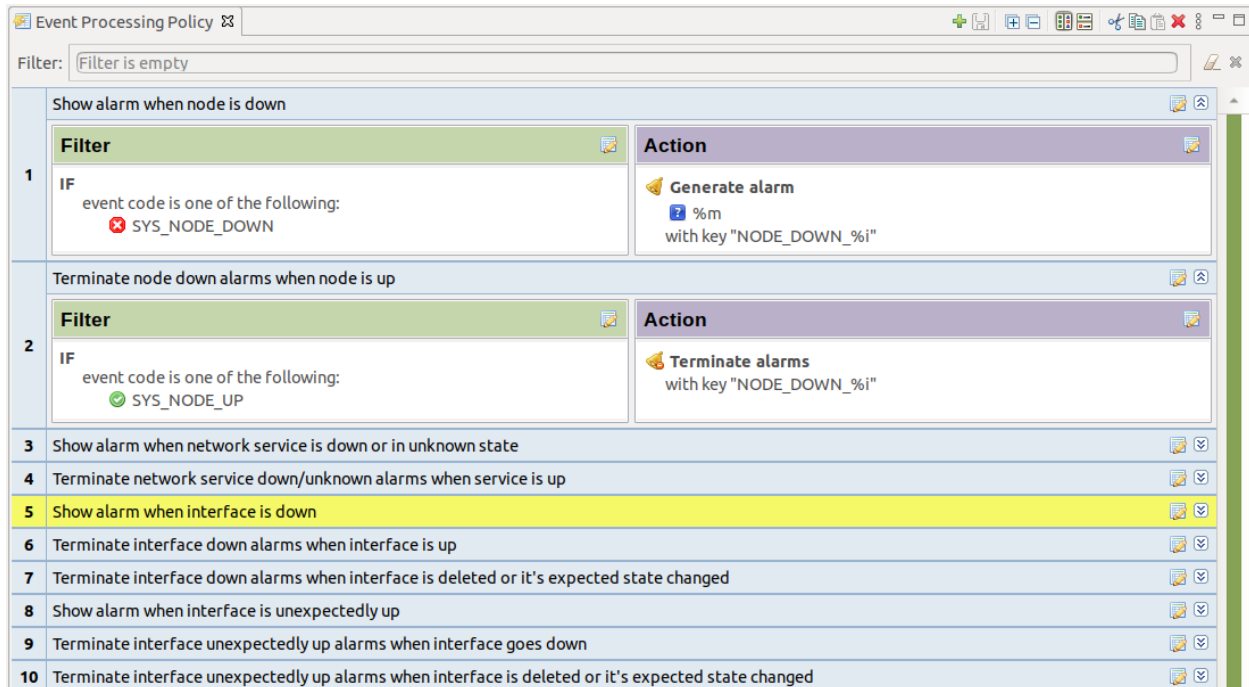


Fig. 1: Event Processing Policy Screen

To expand or collapse a rule, double click on its title or use `Expand/collapse` button on the right hand side of rule title.

Event Processing Policy Editor window toolbar buttons have the following meaning (from left to right): Add new rule, Save changes, Expand all, Collapse all, Horizontal layout, Vertical layout, Cut rule, Copy rule, Paste rule, Delete rule.

To create event policy rule, right click on entry before or after which new Event Processing Policy should appear and select *Insert before* or *Insert after*. Drag and drop can be used for rule reorganization.

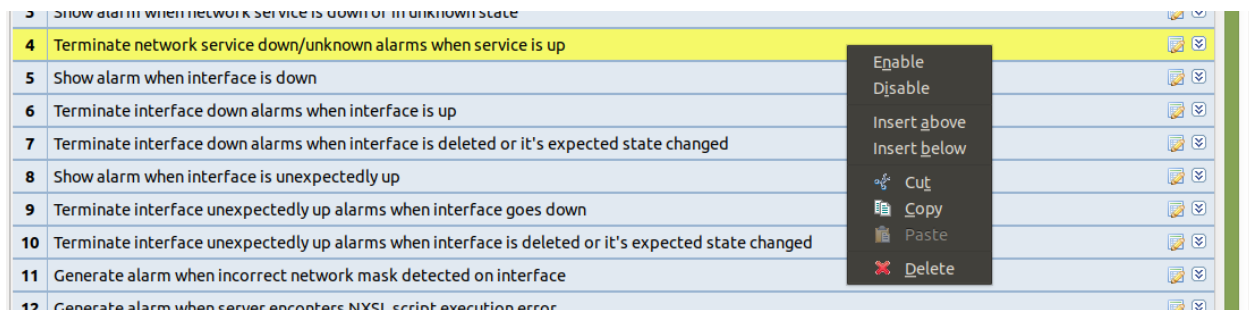


Fig. 2: Event Processing Policy item context menu

To edit Event Processing Policy's properties, click edit button in right corner of an entry, or double-click text in Filter or Action text.

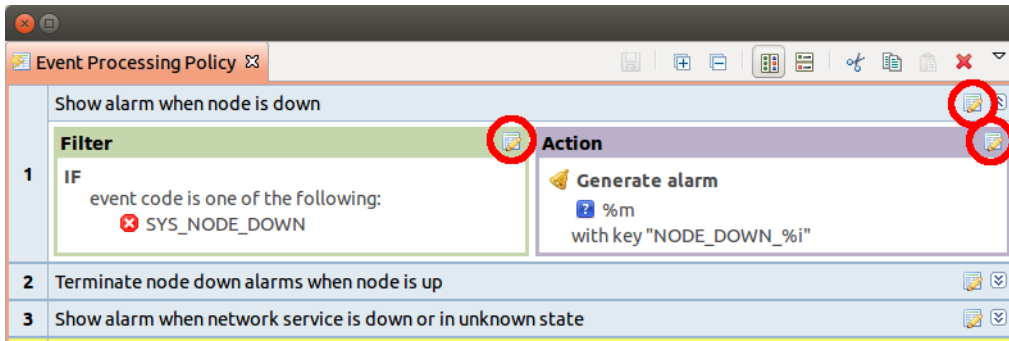


Fig. 3: Edit buttons

Properties of Event Processing Policy rule have the following sections:

Section	Description
Condition	Sub-sections of Condition section determine if the rule is applicable to a particular event. If checkbox <i>Rule is disabled</i> is set, this rule is ignored. Checkbox <i>Accept correlated events</i> defines, if events, which are correlated to another events should be processed (e.g. when node is in maintenance, all node events are correlated to the maintenance event).
Condition → Events	Event code. This field can be left empty, which matches any event, or contain list of applicable events. <i>Inverse rule</i> checkbox allows to react to all events except to those listed.
Condition → Source Objects	Source objects and exclusions lists allow to specify for which objects this rule is applicable. If source objects list is empty, rule would match any object. Multiple objects can be specified in the lists. If you specify subnet, container, collector, cluster, rack or chassis, any object within it will also be matched. If one and the same object is present both in source objects and exclusions, exclusions list has priority. E.g. you can specify a container in source objects and one specific node from that container in exclusions list - rule would match all nodes from that container except that one specified node. <i>Inverse rule</i> checkbox allows to invert the logic, so objects that would be matched by given combination of source objects and exclusions will not be matched and vice versa.
Condition → Time Filter	Allows to specify time frames when rule should be matched. Time frames allow to specify time range, days of week, days of month and months. Days of month are specified as comma-separated lists of days or ranges, e.g. <i>1,3,5,20-25</i> . Letter <i>L</i> denotes last day of month.
Condition → Severity Filter	Event's severity. This field contains selection of event severities to be matched.
Condition → Filtering Script	Optional matching script written in NXSL. If this field is empty (or only contains comments according to NXSL language specification), no additional checks are performed. Otherwise, the event will be considered as matched only if the script returns boolean <code>true</code> (or other value that is considered true in NXSL language, e.g. non-zero number or array). For more information about NetXMS scripting language please refer to the chapter Scripting in this manual. Note: Script execution is a blocking operation - event processor will wait for the script to complete. Make sure that script is written in a way that it would execute quickly.
Action	Sub-sections of Action section determine what actions are performed if an event meets all conditions of a rule. If checkbox <i>Stop event processing</i> is set, then subsequent rules (with higher rule number) will not be processed for a given event. However, actions of given rule will be performed.
Action → Alarm	Action in regard to alarms. Alarm can be created, resolved or terminated or no action to alarms is done. See Generating and Terminating Alarms from EPP for more information.
Action → Downtime Control	Allows to add records to <code>downtime_log</code> table in the DB which can later be used to generate downtime report using the reporting engine. Downtime tag allows to specify several types of downtime for one and the same object. When closing a downtime record, system will search for open record with same downtime tag. Downtime tag is 15 characters in length, macros are not supported in this field.
Action → Persistent Storage	NXLS Persistent Storage action like add/update or delete can be performed.
Action → Custom Attributes	Actions with Custom attributes like add/update or delete can be performed.
Action → Server Actions	List of predefined actions to be executed. Action execution could be delayed with ability to cancel a delayed action later on. Execution of action could be snoozed for a specified period of time. For action configuration refer to Actions chapter. Delayed execution and snoozing is controlled using timers which can be referred to using timer key. This allows cancelling a timer or checking, if its still running from NXSL script.
Action → Script	Script written in NXSL.

Note: Script execution is a blocking operation - event processor will wait for the script to complete. Make sure that script is written in a way that it would execute quickly. If you need to execute a long-running script, create *Execute NXSL script action* and execute it from EPP rule.

After all manipulations are done - save changes by pressing save icon.

13.2.1 Examples

This rule defines that for every major or critical event originated from a node named “IPSO” two e-mail actions will be executed.

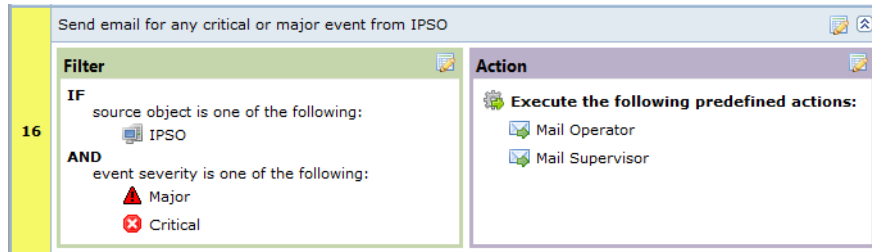


Fig. 4: Example 1

13.3 Events

13.4 Alarms

13.4.1 Alarms Overview

As a result of event processing some events can be shown up as alarms. Usually alarm represents something that needs attention of network administrators or network control center operators, for example low free disk space on a server.

All alarm events are logged to alarm log. The number of days the server keeps alarm history can be configured by “AlarmHistoryRetentionTime” server configuration parameter. Alarm log can be viewed in “Alarm Log View”(Alt+F8). This view gives option to query for required information from alarm log.

Alarm Log

Filter: AlarmLog

Condition

Repeat Count: ☐ AND condition ☐ OR condition

EQUAL

Ack by: ☐ AND condition ☐ OR condition

IS

[Add column](#)

Ordering

Column	Descending
Created	<input checked="" type="checkbox"/> Yes
Last Changed	<input checked="" type="checkbox"/> Yes

[Add](#) [Remove](#)

Alarm ID	State	Helpdesk State	Source	Zone	DCI	Severity	Original Severity	Event	Message
77649	Outstanding	Ignored	sw-poe.office.radensol	Default	0	Warning	Warning	SYS_MAC_ADD	MAC address
77166	Terminated	Ignored	wifi.office.radensol	Default	0	Major	Major	SYS_IF_UNEXPE	Interface "wla
72938	Terminated	Ignored	sw-poe.office.radensol	Test1	0	Critical	Critical	SYS_NODE_DO	Node down
71381	Terminated	Ignored	hp8570w	Default	0	Major	Major	SYS_AGENT_UN	Native agent
70472	Terminated	Ignored	hp8570w	Default	0	Major	Major	SYS_IF_UNEXPE	Interface "virt
70245	Terminated	Ignored	sw-core.office.radensol	Default	0	Major	Major	SYS_IF_UNEXPE	Interface "Gig
69991	Terminated	Ignored	wifi-2.office.radensol	Default	0	Major	Major	SYS_IF_UNEXPE	Interface "wla
69952	Terminated	Ignored	_gateway	Default	0	Critical	Critical	SYS_NODE_DO	Node down
69851	Terminated	Ignored	hp8570w	Default	0	Minor	Minor	SYS_IF_DOWN	Interface "vm
69630	Terminated	Ignored	esx1.office.radensol	Default	0	Critical	Critical	SYS_NODE_DO	Node down
69607	Terminated	Ignored	solaris.office.radensol	Default	0	Critical	Critical	SYS_NODE_DO	Node down
69605	Terminated	Ignored	mqtt.office.radensol	Default	0	Critical	Critical	SYS_NODE_DO	Node down
69329	Terminated	Ignored	hp8570w	Default	0	Minor	Minor	SYS_DCI_UNSU	Status of DCI
69330	Terminated	Ignored	hp8570w	Default	0	Minor	Minor	SYS_DCI_UNSU	Status of DCI
69331	Terminated	Ignored	hp8570w	Default	0	Minor	Minor	SYS_DCI_UNSU	Status of DCI
69328	Terminated	Ignored	hp8570w	Default	0	Minor	Minor	SYS_IF_DOWN	Interface "lo"
69263	Terminated	Ignored	ilo-esx2.office.radensol	Default	0	Critical	Critical	SYS_NODE_DO	Node down
69262	Terminated	Ignored	solaris.office.radensol	Default	0	Critical	Critical	SYS_NODE_DO	Node down
67441	Terminated	Ignored	solaris.office.radensol	Default	0	Minor	Minor	SYS_DCI_UNSU	Status of DCI
67445	Terminated	Ignored	solaris.office.radensol	Default	0	Minor	Minor	SYS_DCI_UNSU	Status of DCI
67446	Terminated	Ignored	solaris.office.radensol	Default	0	Minor	Minor	SYS_DCI_UNSU	Status of DCI
67447	Terminated	Ignored	solaris.office.radensol	Default	0	Minor	Minor	SYS_DCI_UNSU	Status of DCI
67425	Terminated	Ignored	fin.office.radensol	Default	0	Minor	Minor	SYS_DCI_UNSU	Status of DCI
67426	Terminated	Ignored	fin.office.radensol	Default	0	Minor	Minor	SYS_DCI_UNSU	Status of DCI
67427	Terminated	Ignored	fin.office.radensol	Default	0	Minor	Minor	SYS_DCI_UNSU	Status of DCI

Every alarm has the following attributes:

Attribute	Description
Creation time	Time when alarm was created.
Last change time	Time when alarm was last changed (for example, acknowledged).
State	Current state of the alarm, see table below
Message	Message text (usually derived from originating event's message text).
Severity	Alarm's severity - <i>Normal</i> , <i>Warning</i> , <i>Minor</i> , <i>Major</i> , or <i>Critical</i> .
Source	Source node (derived from originating event).
Key	Text string used to identify duplicate alarms and for automatic alarm termination.

Possible alarm states:

Outstanding	New alarm.
Acknowledged	When network administrator sees an alarm, he may acknowledge it to indicate that somebody already aware of that problem and working on it. A new event with the same alarm ID will reset the alarm state back to outstanding
Sticky Acknowledged for time	Alarm will remain acknowledged for given time interval even after new matching events, after time will pass alarm will be moved to outstanding state. This option can be used like snooze. When you know that there will be new matching events, but it will not change the situation. But after some time someone should check this problem. For example, if you have problem that cannot be solved until next week, so this alarm can be sticky acknowledged for 7 days. After 7 days this problem again will be in outstanding state. This type of acknowledge can be disabled by changing <i>EnableTimedAlarmAck</i> server configuration parameter.
Sticky Acknowledged	Alarm will remain acknowledged event after new matching events. This can be useful when you know that there will be new matching events, but it will not change the situation. For example, if you have network device which will send new SNMP trap every minute until problem solved, sticky acknowledge will help to eliminate unnecessary outstanding alarms.
Resolved	Network administrator sets this state when the problem is solved.
Terminated	Inactive alarm. When problem is solved, network administrator can terminate alarm. This will remove alarm from active alarms list and it will not be seen in Management Client, but alarm record will remain in database.

There are 2 types of alarm state flows: strict and not strict. This option can be configured in Preference page of Alarms or on server configuration page, parameter “StrictAlarmStatusFlow”. The difference between them is that in strict mode Terminate can be done only after Resolve state.

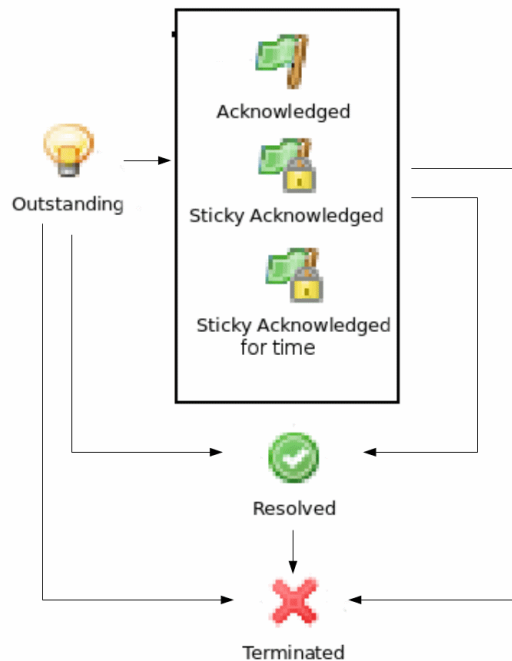


Fig. 5: Not strict(default)



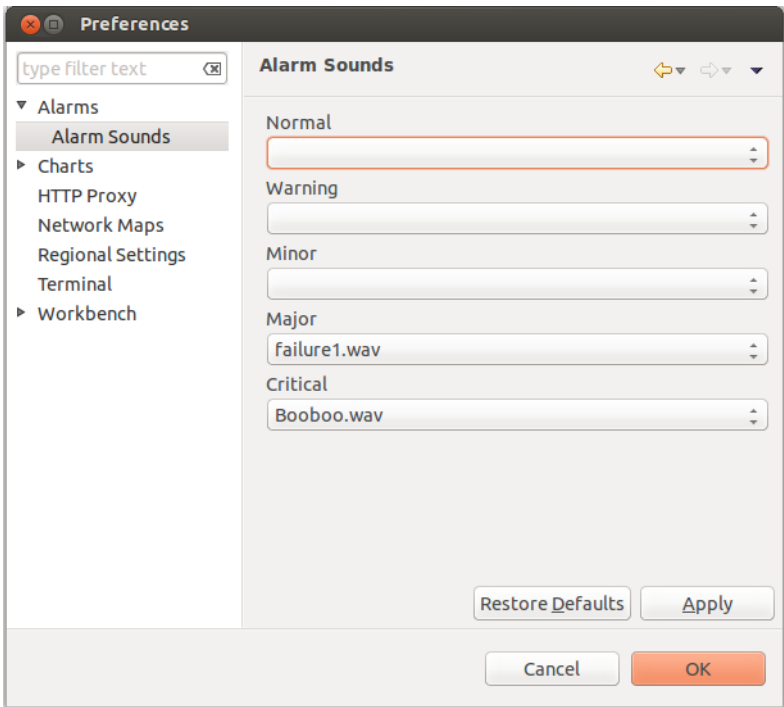
Fig. 6: Strict

13.4.2 Alarm Melodies

On each severity of alarm can be set melody to play. This melody will be played when new alarm in state outstanding will occur. Melody that should be played should exist on server in wav format. See instruction there: [Upload file on server](#). By default there are no sounds on alarms.

To set sound open preferences, there select *Alarms* ▶ *Alarm Sounds* tab. There in drop-down will be seen all possible options. If sound will not be chosen, alarm with this severity will come silently.

To configure sounds, open preferences and select *Alarms* ▶ *Alarm Sounds* tab. Drop-downs next to each severity level have a list of available sounds. If no sound is chosen, alarm for given severity will come silently.



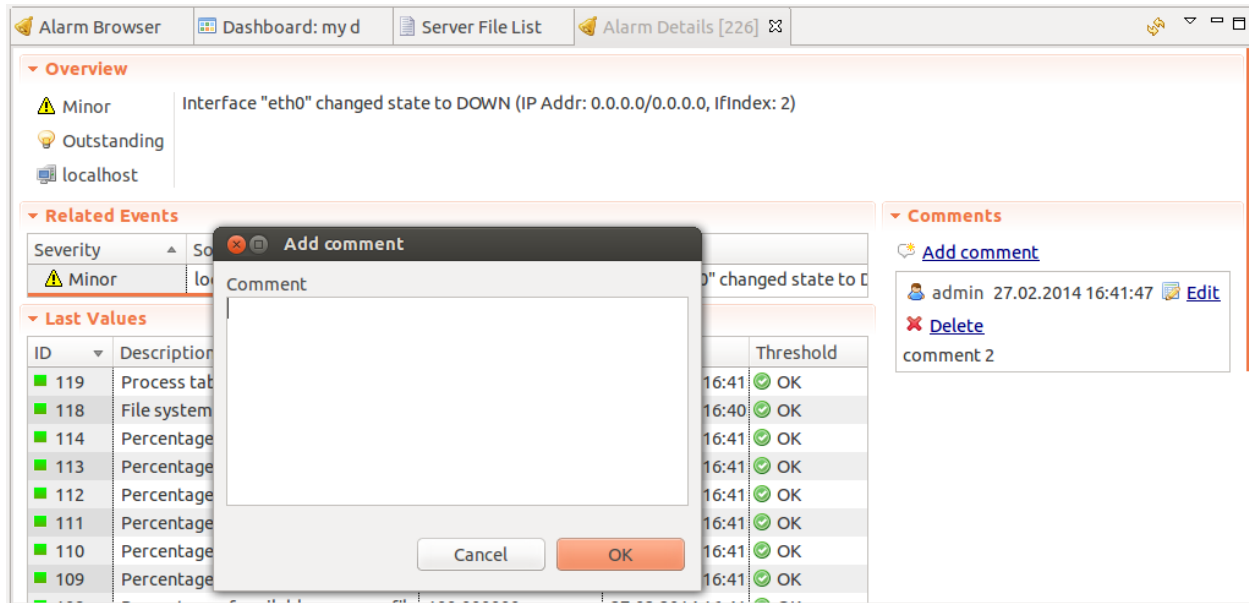
13.4.3 Alarm Browser

When an alarm is generated it will appear in the Alarm Browser where information about currently active alarms can be viewed.

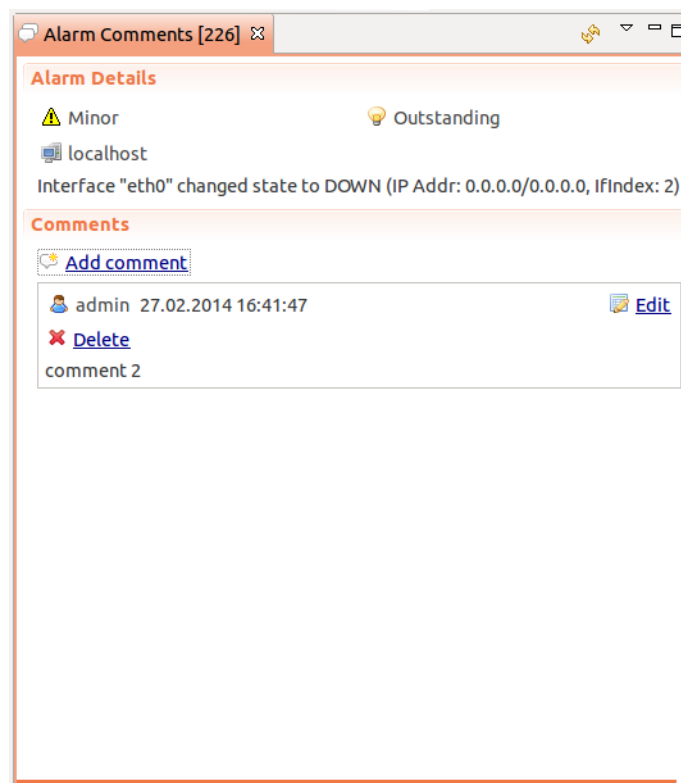
Alarm Browser							
Severity	State	Source	Message	Count	Comment	Helpdesk ID	Ack/Resolve By
Critical	Outstanding	unknown	Node down	1			
Major	Outstanding	sw-lab-1.office.radensolutions.com	Interface "Fa0/21" unexpectedly changed	6			
Critical	Outstanding	Eriks-ThinkPad	Node down	1			
Minor	Outstanding	ilo-sun-v240.office.radensolutions.co	Interface "unknown" changed state to DO	24			
Major	Outstanding	betelgeuse.office.radensolutions.com	Interface "tun0" unexpectedly changed st	24			
Minor	Outstanding	cisco-2600-branch2	Invalid network mask /24 on interface "Se	177			
Minor	Outstanding	Eriks-ThinkPad	Script (Template::Database writer::107) ex	193			
Minor	Outstanding	Eriks-ThinkPad	Script (Template::Windows::109) executio	193			
Minor	Outstanding	Eriks-ThinkPad	Script (Template::UNIX::110) execution er	193			
Minor	Outstanding	Eriks-ThinkPad	Script (Template::Server Performance::11	193			
Minor	Outstanding	Eriks-ThinkPad	Script (Template::HP-UX::112) execution	193			
Minor	Outstanding	Eriks-ThinkPad	Script (Template::Linux::509) execution er	193			
Minor	Outstanding	Eriks-ThinkPad	Script (Template::Generic UNIX::510) exe	193			
Minor	Outstanding	Eriks-ThinkPad	Script (Template::AIX::511) execution erre	193			
Minor	Outstanding	Eriks-ThinkPad	Script (Template::Thread pools::512) exec	193			
Minor	Outstanding	Eriks-ThinkPad	Script (Template::Windows::576) executio	193			
Major	Outstanding	aix.office.radensolutions.com	Problem with agent log: could not open	2059			

Alarm Comments

For each alarm can be created comments in “Alarm Details”



or “Alarm Comments” views.



Comment can be created, edited or deleted. All comments will be deleted after alarm termination.

Alarm Summary Emails

It is possible to schedule emails which contain a summary of all currently active alarms, similar to what can be seen in the Alarm Browser.

Summary emails are sent through SMTP notification channel with HTML formatting. It should be first configured in *Notification channels* configuration and then it's name should be set in "DefaultNotificationChannel.SMTP.Html" server configuration parameter.

To enable Alarm Summary Emails it is required to configure the following server parameters:

Name
DefaultNotificationChannel.SMTP.Html
EnableAlarmSummaryEmails
AlarmSummaryEmailSchedule
AlarmSummaryEmailRecipients

Further information on server configuration parameters can be found in *Server configuration parameters*.

13.4.4 Generating and Terminating Alarms from EPP

To generate alarms from events, you should edit *Alarm* field in appropriate rule of *Event Processing Policy*. Alarm configuration dialog will look like this:

Properties for Rule 1

type filter text

Condition

- Source Objects
- Events
- Severity Filter
- Filtering Script

Action

- Alarm**
- Situation
- Server Actions
- Comments

Alarm

☐ Do not change alarms

☒ Create new alarm

☐ Resolve alarms

☐ Terminate alarms

Message

%m

Alarm key

NODE_DOWN_%i

Alarm severity

From event

Alarm timeout

0

Timeout event

SYS_ALARM_TIMEOUT

Alarm category

<none>

Restore Defaults Apply

Cancel OK

You should select *Generate new alarm* radio button to enable alarm generation from current rule. In the *Message* field enter alarm's text, and in the alarm key enter value which will be used for repeated alarms detection and automatic alarm termination. In both fields you can use macros described in the *Macros for Event Processing* section.

You can also configure sending of additional event if alarm will stay in *Outstanding* state for given period of time. To enable this, enter desired number of seconds in *Seconds* field, and select event to be sent. Entering value of 0 for seconds will disable additional event sending.

Alarms generated by rules can be categorised to limit what alarms can be seen by what users. This can be done by applying a category in the *Alarm Category* field, which can be created and configured in the *Alarm Category Configurator*.

13.4.5 Alarm Category Configurator

Alarm categories can be created and configured in the *Alarm Category Configurator* which can be found in *Configuration*

► *Alarm Category Configurator* menu:

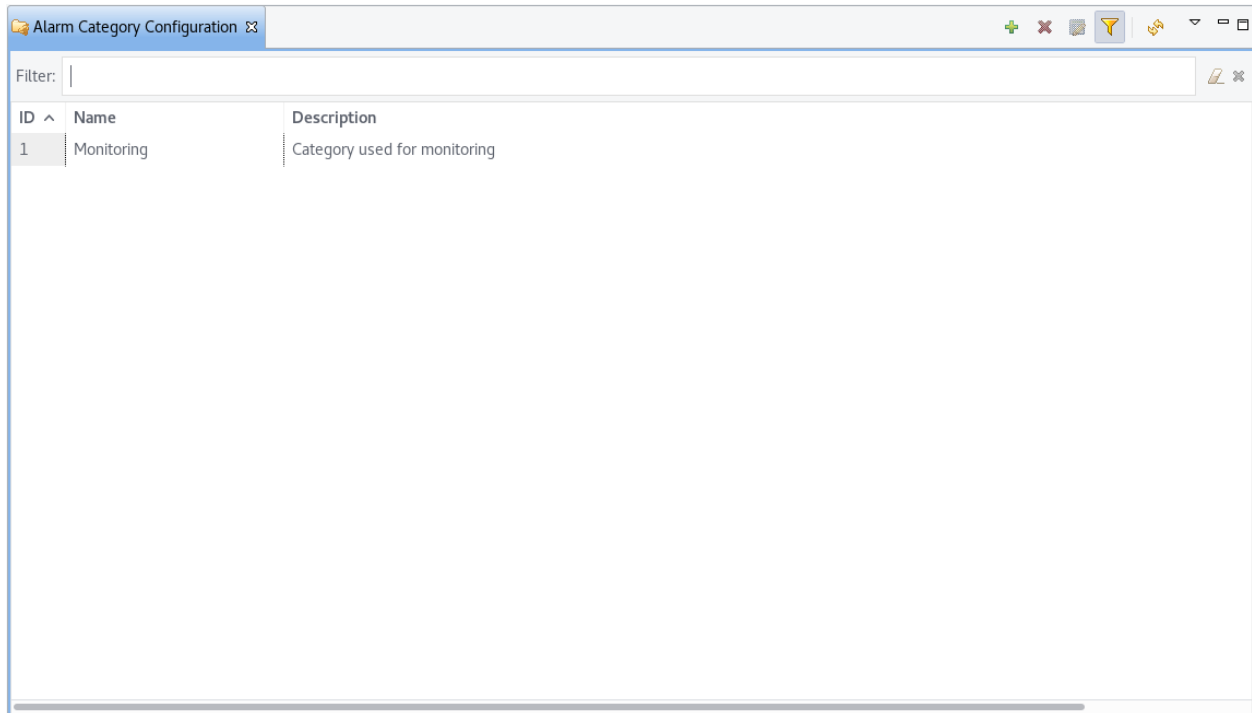


Fig. 7: Alarm Category Configurator

Alarm categories provide the possibility to configure access rights for viewing generated alarms on a per user or per group basis. When creating an alarm category, it is possible to set the *Category name*, *Description*.

The screenshot shows a 'Properties' dialog box with a dark header bar. On the left is a sidebar with a search bar labeled 'type filter text' and a plus icon. Below the search bar are two tabs: 'General' (highlighted in blue) and 'Access Control'. The main area of the dialog is titled 'General' and contains three input fields: 'Category ID' with the value '1', 'Category name' with the value 'Monitoring', and 'Description' with the value 'Category used for monitoring'. At the bottom right of the dialog are four buttons: 'Restore Defaults', 'Apply', 'Cancel', and 'OK'.

Fig. 8: Alarm Category properties

Alarm category access rights can be configured by adding users or groups to the access list of the category in the *Access Control* property page.

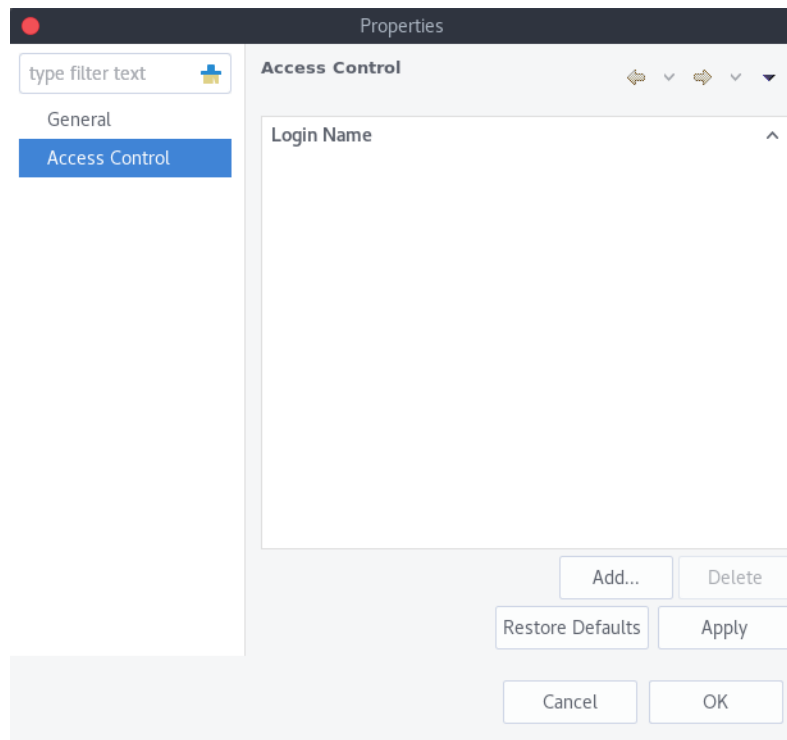


Fig. 9: Alarm Category Access Control

By default, all alarms can be viewed by all users due to the *View all alarms* system right being set as default to the *Everyone* user group. In order to limit the viewing of alarms, this system right should be removed and the access rights configured in the categories themselves. When the categories have been configured, they can be applied to the necessary *Event Processing Policy* rules.

If an alarm category has been applied to an *Event Processing Policy* rule, it will appear in the *Event Processing Policy Editor* when a rule is expanded under the *Action* section.

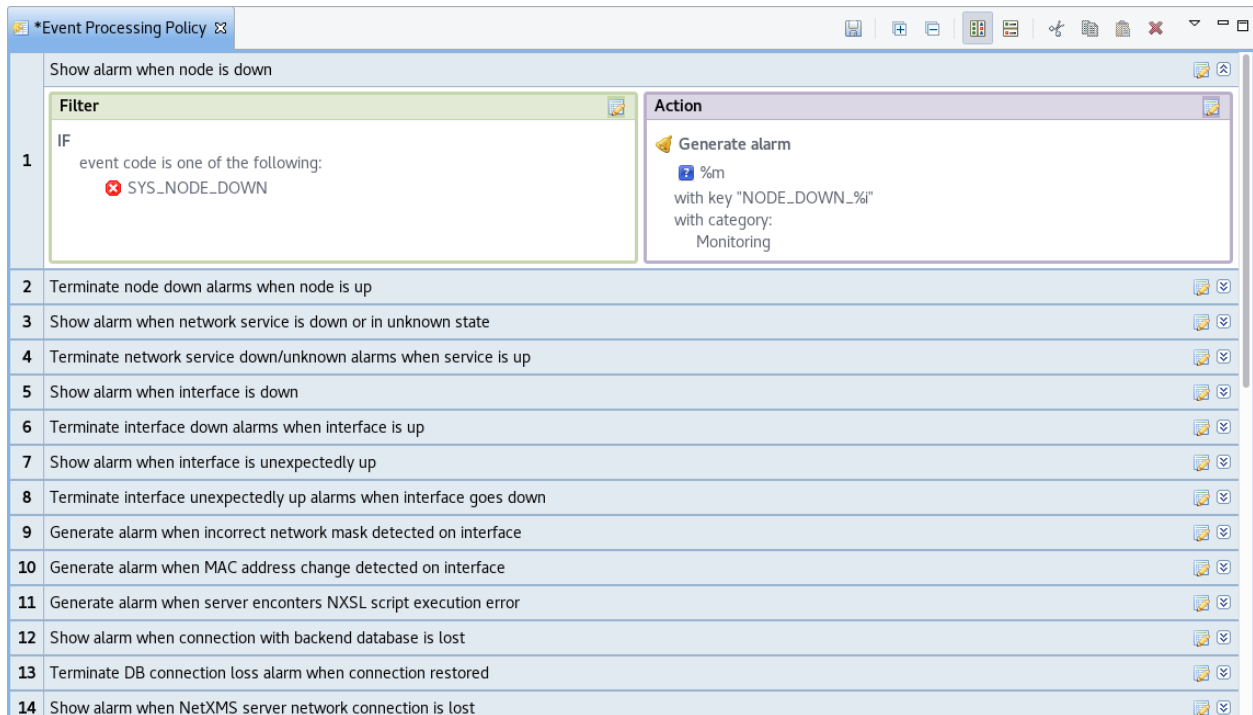


Fig. 10: Event Processing Policy expanded

13.4.6 Automatic Alarm Termination/Resolve

You can terminate or resolve all active alarms with given key as a reaction for the event. To do this, select *Terminate alarm* radio button or *Resolve alarm* radio button in alarm configuration dialog and enter value for alarm key. For that field you can use macros described in the *Macros for Event Processing* chapter.

13.4.7 Escalation

As it was described in *Generating and Terminating Alarms from EPP* chapter there is possibility to generate new event if alarm stay in *Outstanding* state for too long. Escalation is built on this option. When alarm was generated, but no action was done from operator in predefined time, new event can be generated and this time email or notification (SMS, instant message) can be sent to operator or to it's manager. This escalation process can have as many steps as it is required.

13.5 Actions

In addition to alarm generation server can perform various types of actions as a reaction to an event. Action types available in NetXMS are described in the following sections. Each action can be separately disabled in action configuration.

After the action is added, it can be edited to add delay time and timer key. This option can be used to prevent notification sending in case if problem solved quickly enough. Key is a free form string that supports *macros* and delay is the delay time in seconds before action is executed.

The following example shows the configuration for the situation when there is no need to notify anyone if node went down and back up in less then 5 minutes.

Show alarm when node is down	
Filter IF event code is one of the following: SYS_NODE_DOWN	Action Generate alarm with key "NODE_DOWN_%i" Execute the following predefined actions: Send email Delayed by 300 seconds with timer key set to "NODE_DOWN_NOTIFICATION_%i"
Terminate node down alarms when node is up	
Filter IF event code is one of the following: SYS_NODE_UP	Action Terminate alarms with key "NODE_DOWN_%i" Cancel the following timers: NODE_DOWN_NOTIFICATION_%i

If, in addition, we want to send notification when node goes up, but only if notification about node down was sent:

Show alarm when node is down	
Filter IF event code is one of the following: SYS_NODE_DOWN	Action Generate alarm with key "NODE_DOWN_%i" Execute the following predefined actions: Send email Delayed by 300 seconds with timer key set to "NODE_DOWN_NOTIFICATION_%i"
Terminate node down alarms when node is up	
Filter IF event code is one of the following: SYS_NODE_UP	Action Terminate alarms with key "NODE_DOWN_%i" Execute the following predefined actions: Send email Do not run if timer with key "NODE_DOWN_NOTIFICATION_%i" is active Cancel the following timers: NODE_DOWN_NOTIFICATION_%i

13.5.1 Escalation

One *EPP* rule can contain multiple actions with different delays. Delay timers are canceled by other rule in case of problem resolution.

The next example shows that if node went down, then

1. after 1 minute responsible person will be notified if the problem still persists
2. after 30 minutes the support manager will be notified if the problem still persists
3. after 1 hour the IT manager will be notified if the problem still persists

Test rule (Node down)	
Filter IF event code is one of the following: SYS_NODE_DOWN	Action Execute the following predefined actions: Notify node is down Delayed by 60 seconds with timer key set to "node %i down timer" Notify support manager Delayed by 1800 seconds with timer key set to "node %i 30min down" Notify IT manager Delayed by 3600 seconds with timer key set to "node %i 1h down"
Do not send notification	
Filter IF event code is one of the following: SYS_NODE_UP	Action Cancel the following timers: node %i 1h down node %i down timer node %i 30min down

13.5.2 Action types

Execute command on management server

Executes provided command on server node. Check that user under which `netxmsd` process run has permission to run this command.

Execute command on remote node

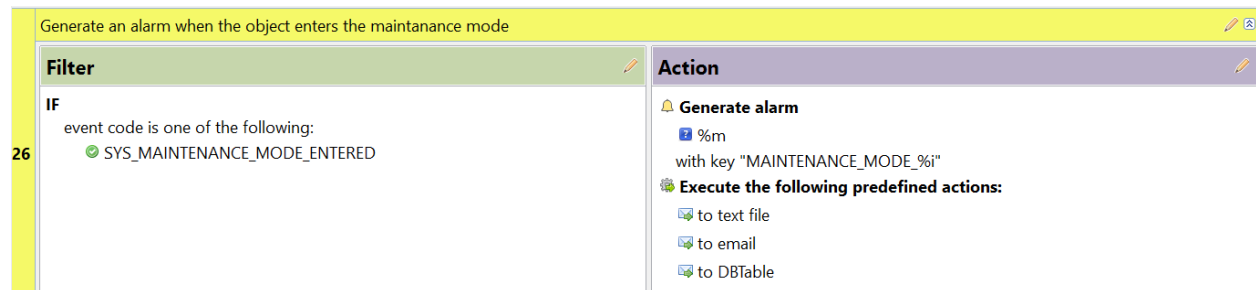
Executes provided command name defined in this nodes agent configuration file. To this command can be given parameters in format: `commandName param1 param2 param3...`. Check that user under which `nxagentd` process run has permission to run this command.

As the *Remote Host* can be used hostname or object name(int format: `@objectName`). Second option allows action execution on node behind proxy.

Send notification

Send notification, e.g. SMS, MicrosoftTeams, e-mail etc, to one or more recipients. This can be configured in Notification channels section described below and appropriate action created in Actions section and then available for use in EPP. Driver configuration parameters are detailed in Drivers section.

In message text can be used *Macros for Event Processing*.



Execute NXSL script

This action executes script from scrip library. In action configuration should be defined name of script. Information about scripting and library can be found [there](#).

Forward event

NetXMS does not support configuration synchronization between two NetXMS servers(Distributed Monitoring). But it is possible to forward events from one server to another. This option allow synchronize events between servers but there are some limitation.

Configuration

Source server configuration:

1. Create new action of type “forward event” - it will have destination server address property.
2. Create a rule in event processing policy with filter for events you want to forward and add forwarding action as action.

Destination server configuration:

1. Enable `EnableISCListener` and `ReceiveForwardedEvents` in server configuration.
2. Open port 4702.

3. Check that receiving server have all events as on a sending server

Limitation

Limitations of event forwarding:

1. Event template with same event code or event name must exist on recipient server
2. Node object with same IP address as event's source node's address must exist on recipient server
3. Does not work with zones

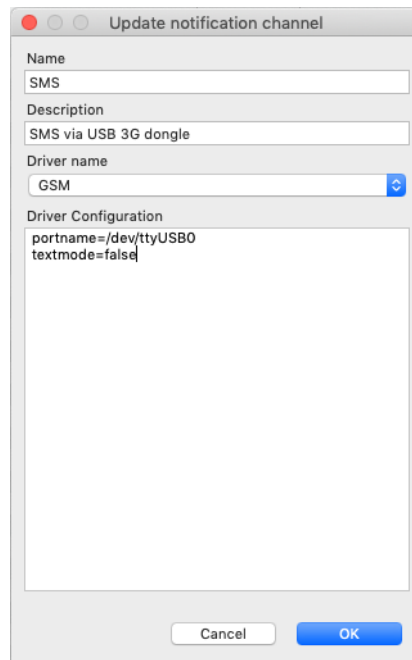
Events not met these conditions are discarded. It is possible to check if and why incoming events are discarded by turning on level 5 debug on receiving server.

There can be used one of two options if it is required to disable polling of sender server nodes on recipient server: disable all polling protocols or unmanage nodes. Chose depends on how you wish to see node's status. For unmanaged node, it always be "unmanaged", regardless of active alarms. If you disable polling, node's status will be "unknown" unless there will be active alarms for that node - in that case node's status will change to severity of most critical alarm.

13.5.3 Notification channels

NetXMS supports concept of notification channel drivers to provide SMS and instant message sending functionality. Role of notification channel driver is to provide level of abstraction on top of different notification sending mechanisms and uniform notification sending interface for server core. It is possible to set up and use several notification channels.

Configuration of notification channels is done in *Configuration ► Notification channels*.



Notification channel driver parameters are specified in *Driver configuration* input field. Each parameter is given on a separate line in format: *parameter_name=parameter_value*. Meaning of parameters is driver dependent and described separately for each driver. If a parameter is not given, its default value will be used.

Once notification channel is created it is seen in channel list with green or red square next to the name - it is channel status identifier. It should be green if driver initialization was successful or red in other cases. *Status* column displays last sent attempt status and *Error message* column provide more information about driver initialization or sending error.

Name	Description	Driver	Status	Error message
Generic com3	Generic	GSM	Unknown	
Slac2	Desc	Slack	Unknown	
Slack1	Desc	Slack	Unknown	
SMS_Modem	3G Module	GSM	Unknown	
Telegram		Telegram	Unknown	Unable to create instance of driver Telegram

Drivers

The following drivers are provided by default with NetXMS installation:

Driver	Description
AnySMS	SMS driver for any-sms.biz service (http://any-sms.biz). Configuration parameters: <ul style="list-style-type: none"> login (default: user) password (default: password) sender (default: NETXMS) gateway (default: 28)
DBTable	This driver saves notifications to a database. Configuration parameters: <ul style="list-style-type: none"> DBDriver (default: sqlite.ddd) DBName (default: netxms) DBLogin (default: netxms) DBPassword DBServer (default: localhost) DBSchema MaxMessageLength (default: 255) MaxNumberLength (default: 32) QueryTemplate
Dummy	Dummy driver for debugging purposes. Does not send any actual notifications and only logs them to server log file. This driver has no configuration parameters. It is necessary to set debug level to <i>debug=6</i> or higher to get records in the log file.
Google chat	Driver to send notifications to Google charts. You need to create incoming web hook first . Each web hook have it's own URL, you can either put it as recipient, or setup mapping in notification channel configuration. Mapping is done in the section "Rooms". Example: <pre>[Rooms] RoomName=URL AnotherRoomName=URL</pre>

continues on next page

Table 1 – continued from previous page

Driver	Description
GSM	<p>Driver for serial or USB attached GSM modems with support for standard GSM AT command set. Configuration parameters:</p> <ul style="list-style-type: none"> • BlockSize (default: 8) • DataBits (default: 8) • Parity (default: n) • Port (default: COM1: on Windows platforms, /dev/ttyS0 on other platforms) • Speed (default: 9600) • StopBits (default: 1) • TextMode (1 - text mode, 0 - PDU mode, default: 1) • UseQuotes (1 - use quotes, 0 - do not use quotes, default: 1) • WriteDelay (default: 100)
Kannel	<p>Driver for Kannel SMS gateway (http://www.kannel.org). Configuration parameters:</p> <ul style="list-style-type: none"> • login (default: user) • password (default: password) • host (default: 127.0.0.1) • port (default: 13001)
Mattermost	<p>Mattermost online chat service driver. Configuration parameters:</p> <ul style="list-style-type: none"> • AuthToken (required, example: f6ern7edy3ma9gtg9zdhaks9aw) • Color • Footer • ServerURL (required, example: your.mattermost.server.fqdn) • UseAttachments
MicrosoftTeams	<p>Notification channel driver for Microsoft Teams. Configuration parameters:</p> <ul style="list-style-type: none"> • ThemeColor - team color in RGB, default: FF6A00 (optional parameter) • UseMessageCards - flag if message cards should be used, default: no (optional parameter) <p>Optional configuration section “Channels” should contain list of channels in the following format: channelName=URL, where channelName is an arbitrary name later used as recipient in action configuration. More information about setting up the URL of incoming webhook available there</p> <pre>#config example ThemeColor=FF6A00 UseMessageCards = false [Channels] Channel=URL AnotherChannel=URL</pre> <p>MsTeams requires 2 fields in action configuration:</p> <ul style="list-style-type: none"> • Recipient name - channel name defined in <i>Channels</i> section or incoming webhook URL • Message - message to be sent

continues on next page

Table 1 – continued from previous page

Driver	Description
MQTT	Driver for sending messages to MQTT broker. Sending is done by NetXMS server process. When sending, MQTT topic is specified in recipient field, value in message body field. Configuration parameters: <ul style="list-style-type: none"> • hostname (default: 127.0.0.1) • port (default: 1883) • login • password
MyMobile	SMS driver for MyMobile API gateways. Configuration parameters: <ul style="list-style-type: none"> • username • password
Nexmo	SMS driver for Nexmo gateway. Configuration parameters: <ul style="list-style-type: none"> • apiKey (default: key) • apiSecret (default: secret) • from (default: NetXMS)
NXAgent	Similar to gsm.ncd, but sending is done via GSM modem, attached to NetXMS agent. Configuration parameters: <ul style="list-style-type: none"> • hostname (default localhost) • port (default: 4700) • timeout (seconds, default: 30) • secret • encryption - optional parameter. Encryption policy: <ul style="list-style-type: none"> 0 = Encryption disabled; 1 = Encrypt connection only if agent requires encryption; 2 = Encrypt connection if agent supports encryption; 3 = Force encrypted connection; • keyFile - optional parameter. Specify server's key file, if not specified will take default path.
Portech	Driver for Portech MV-372 and MV-374 GSM gateways (https://www.portech.com.tw/p3-product1_1.asp?Pid=14). Configuration parameters: <ul style="list-style-type: none"> • host (default: 10.0.0.1) • secondaryHost • login (default: admin) • password (default: admin) • mode (PDU or TEXT, default: PDU)
Shell	Driver executes shell commands on the server. Configuration parameter: <ul style="list-style-type: none"> • Command In the command \${recipient}, \${subject} and \${text} macros will be correspondingly replaced with values of recipient, subject and text.
Slack	Driver for slack.com service. Configuration parameters: <ul style="list-style-type: none"> • url • username

continues on next page

Table 1 – continued from previous page

Driver	Description
SMSEagle	<p>Driver for SMSEagle Hardware SMS Gateway. Configuration parameters:</p> <ul style="list-style-type: none"> • host (default: 127.0.0.1) • port (default: 80) • login (default: user) • password (default: password) • https (1 - use https, 0 - do not use https)
SMTP	<p>Driver to send notifications using SMTP protocol. Encryption and authentication are supported. Driver is using libcurl library to send emails. Mail encoding is always utf8.</p> <ul style="list-style-type: none"> • FromAddr (default: <code>netxms@localhost</code>) • FromName (default: NetXMS Server) • IsHTML (no - do not use HTML, yes - use HTML; default: no) • Login (default: none) • Password (default: none). Passwords encrypted by <code>nxencpasswd</code> are supported. If password provided by your email service is 44- or 88-character base64 string, it will be interpreted as a password encrypted by <code>nxencpasswd</code>, in this case encrypt password provided by your email service with <code>nxencpasswd</code>. • Port (default: 465 if TLSMode=TLS, 25 otherwise)) • Server (default: localhost) • TLSMode (NONE - No TLS (default), TLS - Enforced TLS, STARTTLS - Opportunistic TLS)

continues on next page

Table 1 – continued from previous page

Driver	Description
SNMPTrap	<p>Driver to send notifications as SNMP traps. Driver configuration parameters:</p> <ul style="list-style-type: none"> • Community (default: public) • Port (default: 162) • ProtocolVersion (possible values: 1, 2c, 3; default: 2c) <p>Driver configuration parameters applicable to SNMP v3 only:</p> <ul style="list-style-type: none"> • AuthMethod (possible values: none, sha1, sha224, sha256, sha384, sha512; default: none) • AuthPassword • PrivMethod (possible values: none, aes, des; default: none) • PrivPassword • UseInformRequest (default: false) • UserName (default: netxms) <p>Raden Solutions has IANA assigned Private Enterprise Number (57163). MIB files defining the OIDs (RADENSOLUTIONS-SMI.txt and NETXMS-MIB.txt) are included with NetXMS server. It's also possible to use custom OIDs by setting the following driver configuration parameters:</p> <ul style="list-style-type: none"> • AdditionalDataFieldID (default: .1.3.6.1.4.1.57163.1.1.6.0) • AlarmKeyFieldID (default: .1.3.6.1.4.1.57163.1.1.5.0) • MessageFieldID (default: .1.3.6.1.4.1.57163.1.1.3.0) • SeverityFieldID (default: .1.3.6.1.4.1.57163.1.1.2.0) • SourceFieldID (default: .1.3.6.1.4.1.57163.1.1.1.0) • TimestampFieldID (default: .1.3.6.1.4.1.57163.1.1.4.0) • TrapID (default: .1.3.6.1.4.1.57163.1.0.1) <p>Recipient's address should contain host name or IP address the trap is sent to. Message and subject are sent as separate fields (MessageFieldID and AdditionalDataFieldID) in the trap message. In addition to that, if subject contains semicolon-separated key=value pairs or JSON and the key is from below list, additional fields with these values will be added to trap message. List of supported keys:</p> <ul style="list-style-type: none"> • key - alarm key • source - source object name • severity - event severity (integer in range 0..4) • timestamp - original even timestamp as UNIX time <p>E.g. subject could be <code>key=%K;source=%n;severity=%s;timestamp=%T</code>. Subject field could be generated using NXSL script that is called using <code>%[script_name]</code> macro. This is convenient for generating JSON.</p> <p>JSON data can have more fields in addition to the above mentioned, this allows to send more information in the trap.</p>

continues on next page

Table 1 – continued from previous page

Driver	Description
Telegram	<p>Notification channel driver for Telegram messenger. Configuration parameters:</p> <ul style="list-style-type: none"> • AuthToken • DisableIPv4 - <code>true</code> to disable IPv4 usage • DisableIPv6 - <code>true</code> to disable IPv6 usage • ParseMode - Text formatting style: <code>Markdown</code>, <code>HTML</code> or <code>MarkdownV2</code>. See Telegram API documentation on formatting syntax: https://core.telegram.org/bots/api#formatting-options • Proxy - proxy url or ip or full configuration if format <code>[scheme]://[login:password]@IP:[PORT]</code> • ProxyPort - proxy port • ProxyType - proxy type: <code>http</code>, <code>https</code>, <code>socks4</code>, <code>socks4a</code>, <code>socks5</code> or <code>socks5h</code> • ProxyUser - proxy user name • ProxyPassword - proxy user password <p>Only AuthToken field is mandatory field all others are optional.</p> <p>It is necessary to create a telegram bot that NetXMS server will use to send messages. In order to create a new bot it's necessary to talk to BotFather and get bot authentication token (AUTH_TOKEN). Set authentication token in notification channel configuration, e.g.: <code>AuthToken=1234567890:jdiAiwdisUsWjvKpDenAlDjuqpx</code></p> <p>The bot can:</p> <ul style="list-style-type: none"> • Have a private chat with another Telegram user • Participate a group • Be channel admin <p>Telegram's bot can't initiate conversations with users in a private chat or a group. A user must either add bot to a group or send a private message to the bot first.</p> <p>Chat, group or channel is identified by ID or name (without @ prefix). For private chats only users who configured a Username can be identified by name (without @ prefix). NetXMS stores the correspondence between ID and name when the bot receives a message in chat or group (NetXMS server should be running at that moment). If group, channel name or username is changed, it's necessary to send any message to the bot so new correspondence could be stored.</p> <p>Telegram notification channel requires 2 fields in action configuration:</p> <ul style="list-style-type: none"> • Recipient name - It could be name (of a group, channel or username, without @ prefix) or ID of group, channel or chat. • Message - text that should be sent <p>If you want to use ID to identify a recipient, you can get it by opening Telegram API URL in your browser, e.g. https://api.telegram.org/bot1234567890:jdiAiwdisUsWjvKpDenAlDjuqpx/getUpdates After sending a message to the bot or adding it to a group you should see chat id there. You might need to temporarily deconfigure Telegram notification channel, otherwise if NetXMS server is running, it will read data from Telegram API first.</p>
Text2Reach	<p>Driver for Text2Reach.com service (http://www.text2reach.com). Configuration parameters:</p> <ul style="list-style-type: none"> • apikey (default: apikey) • from (default: from) • unicode (1 or 0, default: 1) • blacklist (1 or 0, default: 0)
TextFile	<p>Notification driver that writes messages to text file. Configuration parameter:</p> <ul style="list-style-type: none"> • OutputFile - path to file.

continues on next page

Table 1 – continued from previous page

Driver	Description
Twilio	Driver for Twilio.com service (http://www.twilio.com). Configuration parameters: <ul style="list-style-type: none"> • CallerId - caller ID • SID - account SID (for authentication) • Token - account security token (for authentication) • Voice - voice to be used for Text To Speech (man, woman, alicia, or any of the Amazon Polly voices. See here for more information https://www.twilio.com/docs/voice/twiml/say#voice) • UseTTS - true/false, enable or disable Text To Speech (default is false)
WebSMS	Driver for websms.ru service (https://websms.ru). Configuration parameters: <ul style="list-style-type: none"> • login (default: user) • password (default: password) • m_fromPhone
XMPP	Driver for XMPP/Jabber messages. Configuration parameters: <ul style="list-style-type: none"> • Server (default: localhost) • Port (default: user) • Login - may or may not contain XMPP domainpart. If no domainpart is specified server name will be added to login. (default: netxms@localhost) • m_fromPhone (default: 5222)

13.6 NXLS Persistent Storage

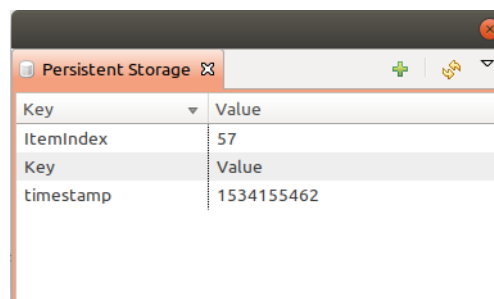
13.6.1 NXSL

There are 2 functions:

- ReadPersistentStorage("key") - read value by key
- WritePersistentStorage("key", "value") - insert or update value by key. If value will be empty - variable will be deleted.

13.6.2 View

Persistent Storage view (*Configuration* ► *Persistent Storage*) provide information about current state of Persistent Storage variables.



13.7 Macros for Event Processing

On various stages of event processing you may need to use macros to include information like event source, severity, or parameter in your event texts, alarms, or actions. You may use the following macros to accomplish this:

Macro	Description
%a	IP address of event source object.
%A	Alarm's text. This macro is populated when creating, resolving or terminating alarm in EPP rule. Macro is available in that EPP rule for persistent storage and server action and in subsequent EPP rules. Changed in version 3.8.314. Prior to 3.8.314 this macro was available only withing given EPP rule.
%c	Event's code.
%C	Comment of event source object. Added in version 4.4.3.
%D	Comment of Data Collection Item (only for threshold violation events) Added in version 4.4.3.
%E	List of comma-separated user tags associated with the event.
%g	Globally unique identifier (GUID) of event source object.
%i	Unique ID of event source object in hexadecimal form. Always prefixed with 0x and contains exactly 8 digits (for example 0x000029AC).
%I	Unique ID of event source object in decimal form.
%K	Alarm's key (can be used only in actions to put text of alarm from the same event processing policy rule).
%L	Alias of event source object. Added in version 4.4.3.
%m	Event's message text (meaningless in event template).
%M	Custom message text. Can be set in filtering script by setting CUS-TOM_MESSAGE variable.
%n	Name of event source object. Name of interface when interface name is generated using macros.
%N	Event's name.
%s	Event's severity code as number. Possible values are: <ul style="list-style-type: none"> • 0 - <i>Normal</i> • 1 - <i>Warning</i> • 2 - <i>Minor</i> • 3 - <i>Major</i> • 4 - <i>Critical</i>
%S	Event's severity code as text.
%t	Event's timestamp is a form day-month-year hour:minute:second.
%T	Event's timestamp as a number of seconds since epoch (as returned by <code>time()</code> function).
%v	NetXMS server's version.
%z	Zone UIN of event source object.
%Z	Zone name of event source object.
%[name]	Value returned by script. You should specify name of the script from script library. It's possible to specify script entry point separating it by /, e.g. to call a function named <code>calculate</code> : %[name/calculate]. Script parameters can be specified in brackets, e.g.: %[name(123, "A textual parameter")]

continues on next page

Table 2 – continued from previous page

Macro	Description
<code>%{name}</code>	Value of custom attribute. Expansion is attempted in the following order: <ol style="list-style-type: none"> 1. If information about a DCI is available during expansion (when processing threshold violation event or if macro is used in a field in DCI properties), custom attribute <code>name::instance</code> is taken, where <code>instance</code> is instance of a DCI. 2. If above custom attribute is not found, <code>name</code> custom attribute is taken. If custom attribute exists, but has empty value, this empty value is taken (if this macro is used in a place where its value is converted to numeric value - e.g. as threshold value for a numeric DCI - then empty value will be converted to 0).
<code>%{name:default_value}</code>	Value of custom attribute. Expansion is attempted in the following order: <ol style="list-style-type: none"> 1. If information about a DCI is available during expansion (when processing threshold violation event or if macro is used in a field in DCI properties), custom attribute <code>name::instance</code> is taken, where <code>instance</code> is instance of a DCI. 2. If above custom attribute is not found, <code>name</code> custom attribute is taken. 3. If above custom attribute is not found, <code>default_value</code> is taken. If custom attribute exists, but has empty value, this empty value is taken (if this macro is used in a place where its value is converted to numeric value - e.g. as threshold value for a numeric DCI - then empty value will be converted to 0).
<code>%<name></code>	Event's parameter with given name.
<code>%<{format-specifier}name></code>	Formatted event's parameter with given name. This is applicable to DCI value and threshold value parameters. <code>format-specifier</code> is comma-separated list supporting the following options: <ul style="list-style-type: none"> • <code>units</code> - add measurement units from DCI's properties. For <i>Epoch</i> time and <i>Uptime</i> this will also convert the value. • <code>u</code> - same as <code>units</code> • <code>multipliers</code> - display values with multipliers (e.g. 1230000 becomes 1.23 M) • <code>m</code> - same as <code>multipliers</code>
<code>%1 - %99</code>	Event's parameter number 1 .. 99.
<code>%%</code>	Insert % character.

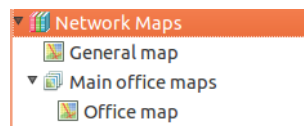
If you need to insert special characters (like carriage return) you can use the following notations:

Char	Description
<code>\t</code>	Tab Character (0x09)
<code>\n</code>	New line, CR/LF character pair
<code>\\</code>	Backslash character

DATA AND NETWORK VISUALISATION

14.1 Network maps

Network map objects can be found in “Object browser” under “Network Maps”. There can be created and deleted maps and map groups. Maps can be organized in groups.



14.1.1 Creating Maps

There are 3 types of map that can be created:

- Custom - will be created empty map.
- Layer 2 Topology - will create map(if possible) with layer 2 topology of selected object. Will be automatically updated on topology change.
- IP Topology - will create map with known IP Topology of selected object. (More about network topology can be found there [Network topology](#)) Will be automatically updated on topology change.
- Internal communication topology - map created based on internal communication between server and node (will show SNMP, ICMP,).

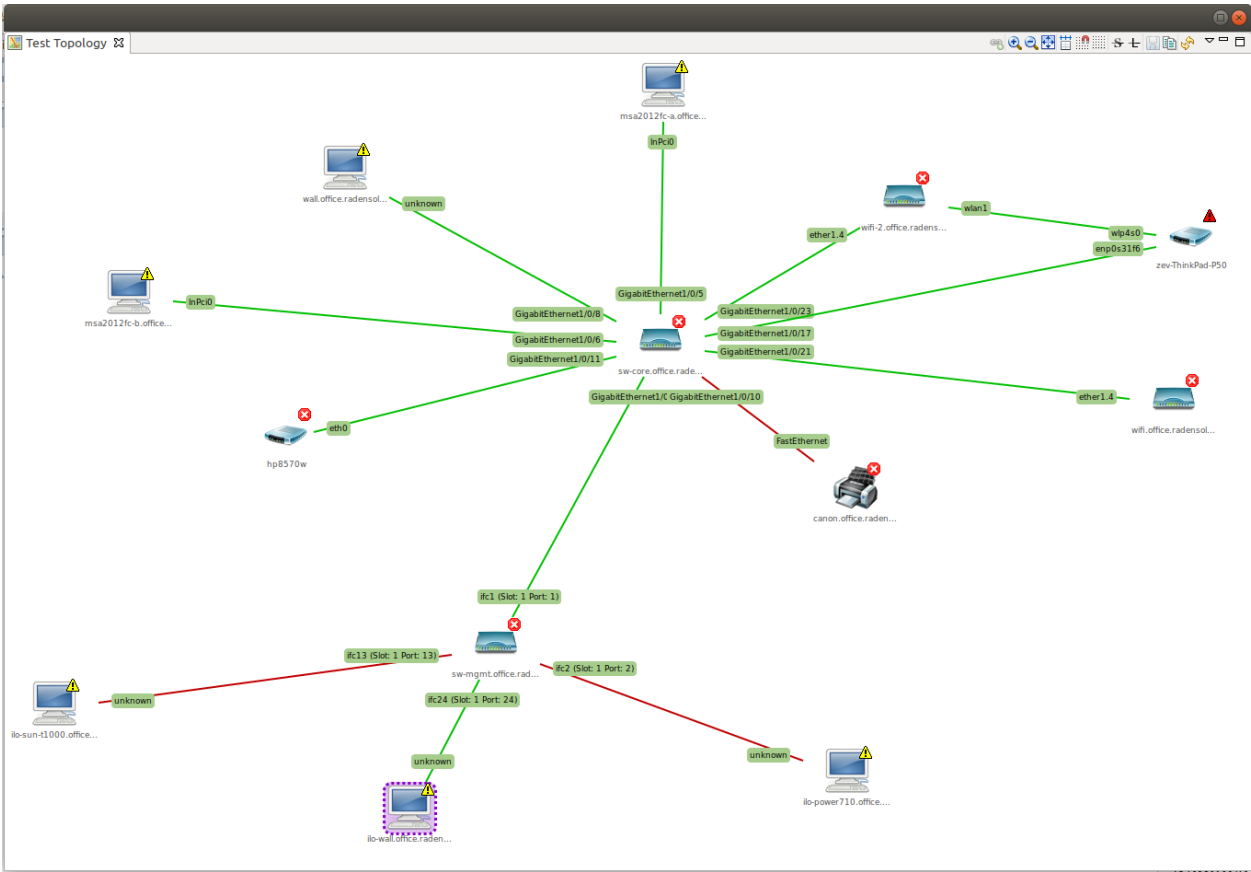


Fig. 1: Network map layer 2

Type of created map affects only on initial map setup.

14.1.2 Edit Maps

Add object...Ctrl+A

Add decoration

Show status background

☒ Show status icon

Show status frame

Layout

Routing

Zoom

Display objects as

Align to grid

Snap to grid

Show grid

Refresh

Map properties

Input Methods

14.1.3 Adding Objects

Network map can be populated in 2 different ways: automatically and manually. Automatically are populated Layer 2, IP Topology and Internal communication topology. Object filter (in properties of the map) can be created for automatically populated maps to filter out unrequired nodes.

Objects to map can be added in tow ways:

1. Just drag and drop object to map from object browser.
2. “Add object...” from menu.

To remove object from map:

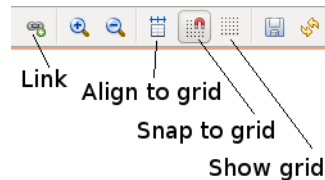
- Select object, right click and select “Remove from map” option.

14.1.4 Adding Links between Objects

Objects can be linked with a line.

To link objects:

- Select two of objects with help of CTRL button and press “Link selected objects” button.



To remove the link:

- Select line, right click and select “Remove from map” option.

Link properties:

Select link line, right click and select “Properties”.

The following properties can be configured:

- Link name
- Connector names (shown on the link line near each connected object)
- **Line color**
 - Default - grey
 - Based on object status - object(s) should be selected
 - Custom color
- **Routing algorithm**
 - Map Default - algorithm selected in map properties will be used
 - Direct - straight line without bend points
 - Manhattan - line with automatic bend points
 - Bend points - bend point can be added manually with double click on the line
- Label position - defines position of label containing link name and DCI values on the link. 50 means middle of the link.

- Data Source (allows to configure DCI values and text near them that will be displayed on a link).
- For each Data Source can be configured: Data collection item, name, format string, in case of table DCI also column and instance. If format string is not provided, default formatting including multipliers and measurement units is used.

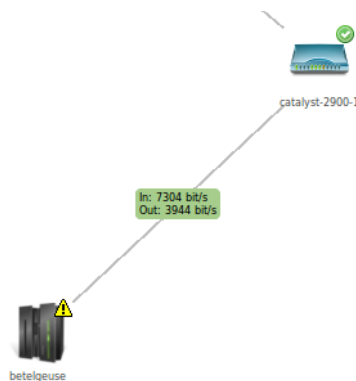
Java format string syntax is used, e.g. Text : `%.4f`, syntax description is available here: <http://docs.oracle.com/javase/7/docs/api/java/util/Formatter.html#syntax>.

Additional format specifier can be provided in curly brackets after `%` sign to display multipliers and measurement units, e.g. `%{units,multipliers}f`.

Format specifier is comma-separated list supporting the following options:

- `units` - add measurement units from DCI's properties. For *Epoch* time and *Uptime* this will also convert the value.
- `u` - same as `units`
- `multipliers` - display values with multipliers (e.g. 1230000 becomes 1.23 M)
- `m` - same as `multipliers`

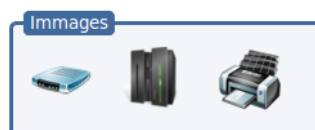
Example of DCI data displayed on a link:



14.1.5 Decorations

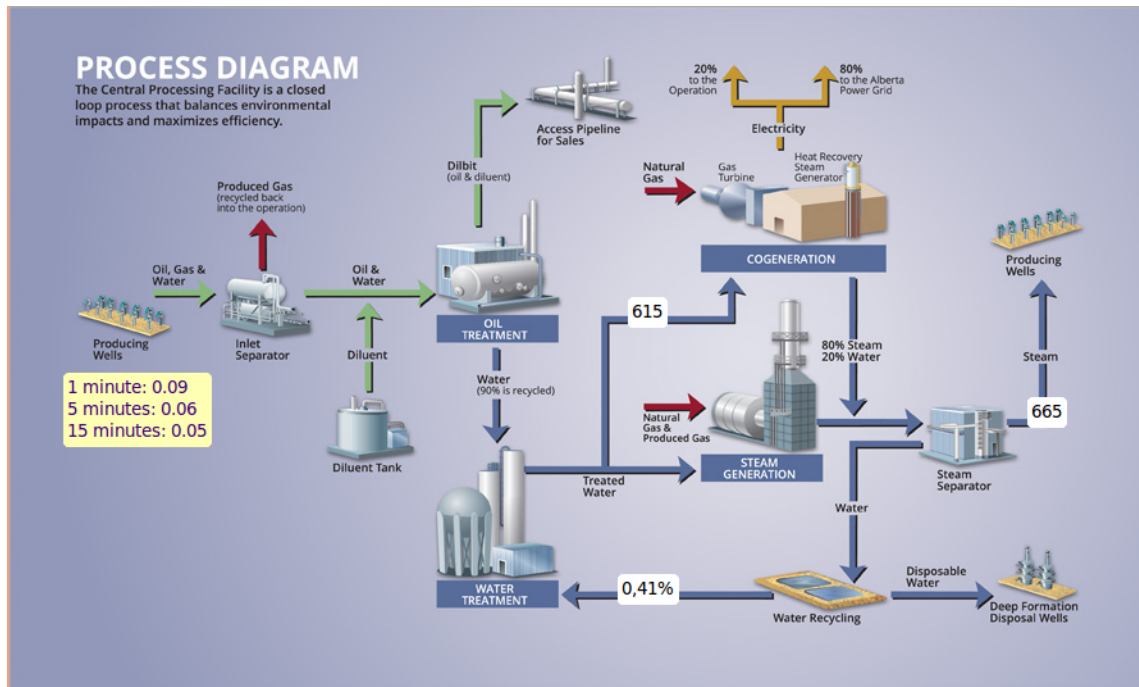
Decorations like picture and group box can be added to maps. To add picture it should be previously be uploaded to “Image Library”.

When creating group box you should specify it's size, color and name.



14.1.6 DCI Container

DCI Container is part of decorations. It can be used to display separate dci values on a map.



Container properties:

- Background color
- Text color
- If border should be shown and it's color
- **Data Source** - there can be configured DCI values and text near them that will be displayed
 - For each Data Source can be configured: Data collection item, name, format string (e.g. "Text: %.4f" or "Text: %*s"), in case of table DCI also column and instance

More examples:

Load avg 0.73

Status 4
CPU util: 8,320795

Status 4

14.1.7 DCI Image

DCI Image is part of decorations. It can be used to display DCI status change in pictures.

DCI image properties

- Data source - DCI which data will be taken to process picture display rules
- Column - required only for table DCI
- Instance - required only for table DCI

- Default image - image that will be displayed if no rule is applicable on current value
- **Rules**
 - For each rule can be configured: operation, value, comment and image that will be displayed if this rule is applicable

Hints:

To use image it should be first uploaded to image library.

Rules are processed from up to down, so if you want to describe in rules something like:

- DCI > 3 => image1
- DCI > 2 => image2
- DCI > 4 => image3

They should go in this sequence:

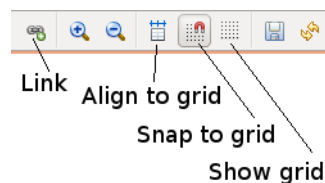
- DCI > 4 => image3
- DCI > 3 => image1
- DCI > 2 => image2

14.1.8 Object Layout and display options

All object layout properties and display options are applicable only on objects, not on decorations.

Grid

- Align to grid - will move all objects to grids
- Snap to grid - all objects will be moved in grids and it will not be possible to place them not inside grid.
- Show grid - will show grid according to which objects are located.



Layout

Objects can be placed manually on a map or can be chosen one of automatic layouts:

- Spring
- Radial
- Horizontal tree
- Vertical tree
- Sparse vertical tree

If there is chosen automatic layout, then after each refresh object best matching place will be recalculated. So if new object is add - it is just required to refresh map to have correctly placed objects.

If there is chosen manual layout, then after each object movement map should be saved, to save the new place of object.

Display object as

- Show status background - will display background behind object image according to it's state.
- Show status icon - will display icon of object state near each object
- Show status frame - will display frame around object icon according to it's state
- Floor plan - will display nodes as adjustable rectangles. This can be used to display hardware placement on room plan.

Routing

Default routing type for whole map:

- Direct - objects are connected by links drawn to shortest route
- Manhattan - objects are connected by grid-based links

Zoom

Map can be zoomed in and out with help of top menu buttons and to predefined percentage selected from menu.

Object display options

Objects can be displayed in 3 ways:

- Icons
- Small labels
- Large labels

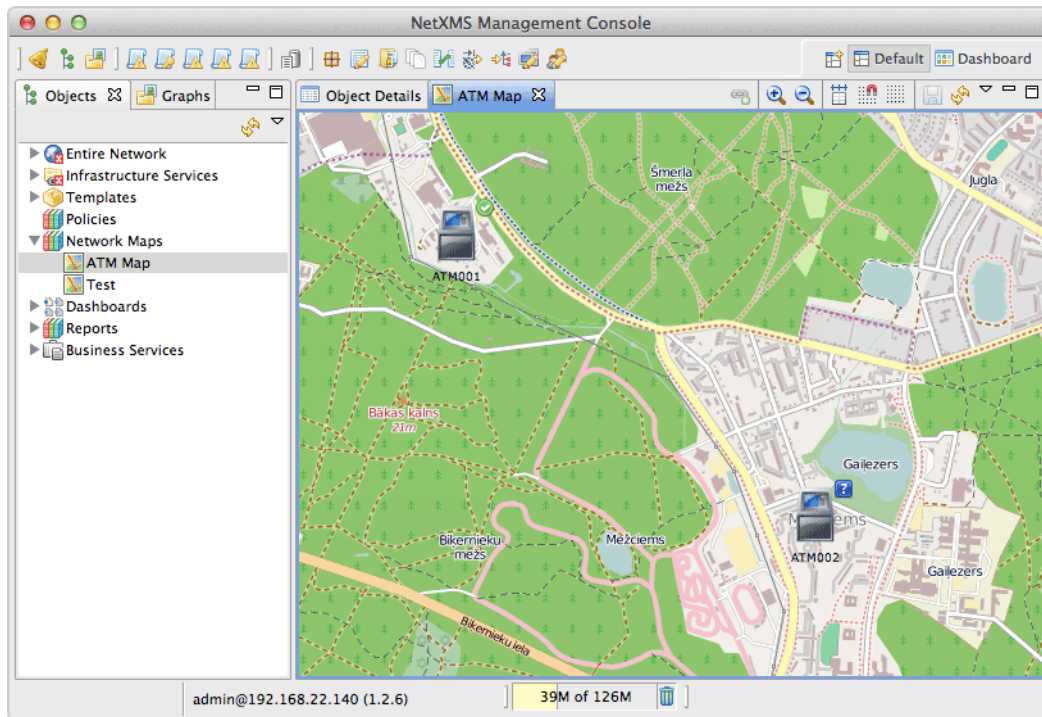
14.1.9 Map Background

It can be set background for map:

- Colour
- Image - image should be uploaded to "Image Library" before.
- Geographic Map - place on map is chose with help of zoom and coordinates

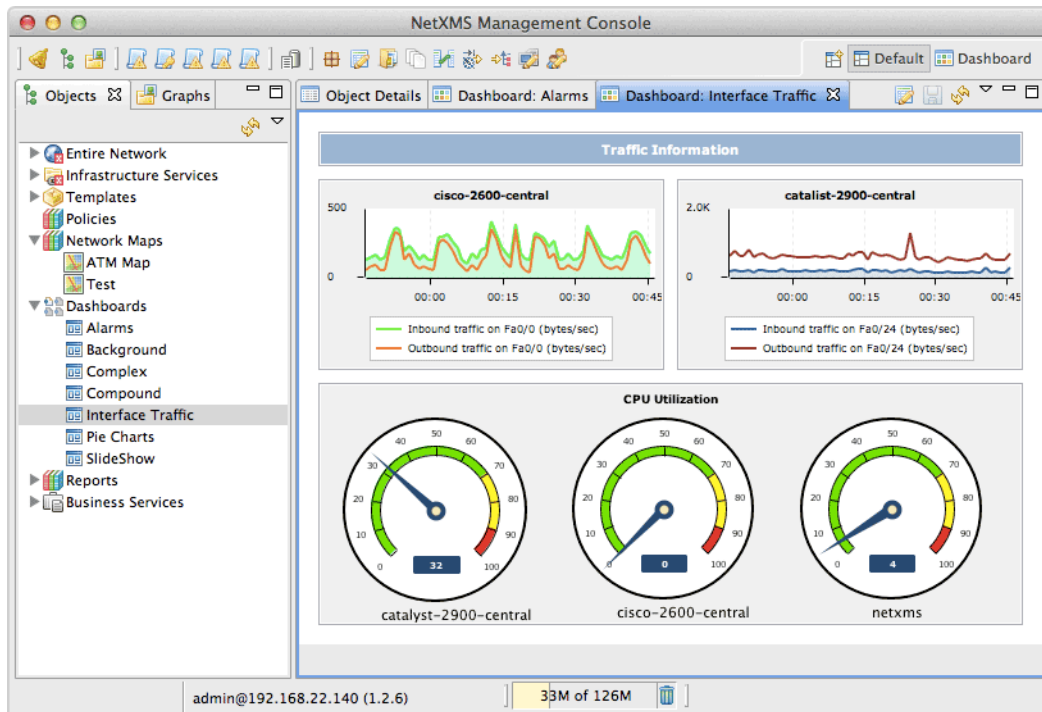
This can be used to show object physical please on map or on building plan.

Examples:



14.2 Dashboards

Dashboards are defined by administrator and allow to combine any available visualization components with data from multiple sources in order to create high-level views to see network (or parts of it) health at a glance. For example, below is a dashboard showing traffic information from core router, as well as CPU usage from vital nodes:



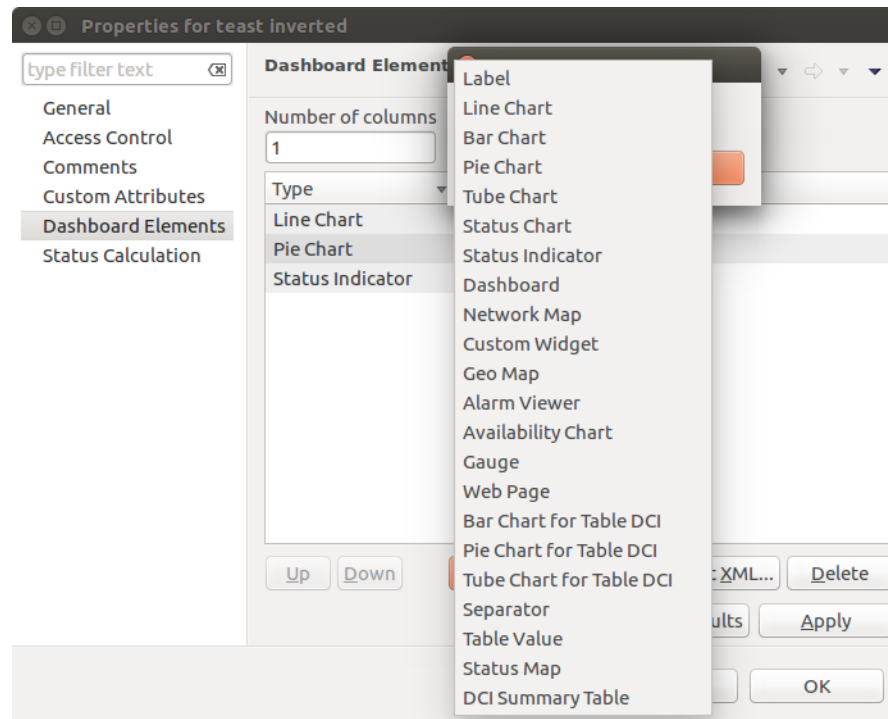
There are two ways to access dashboards:

Open dashboard from Object Browser

- Open dashboard from *Object Browser*
- Switch to *Dashboard* perspective and select dashboard with left-click

14.2.1 Configuration

Dashboards is a special type of objects created in *Dashboards* tree. To create a new dashboard, right click on *Dashboards* root object or any other existing dashboard and select *Create dashboard*. To configure dashboard content, open object's properties and go to *Dashboard Elements:guilabel:* page. Here you can define number of columns and manage list of elements. Press *Add:guilabel:* to add new element. You will be prompted with element type selection dialog:



When a new element is added, you can edit it by double-clicking on it's record in the elements list, or by pressing the *Edit* button. Each element have *Layout* property page which controls the element's layout inside the dashboard, and one or more element type specific pages to control element's appearance and displayed information. The following element types are available:

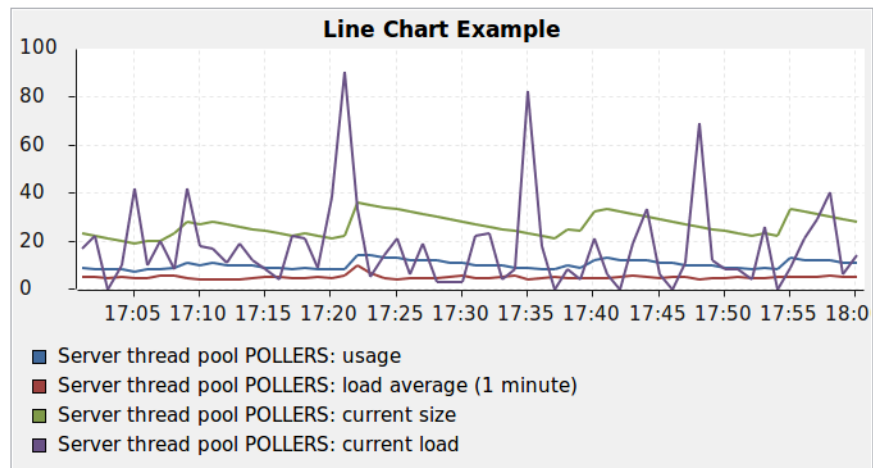
Label

Text label with configurable text and colors.



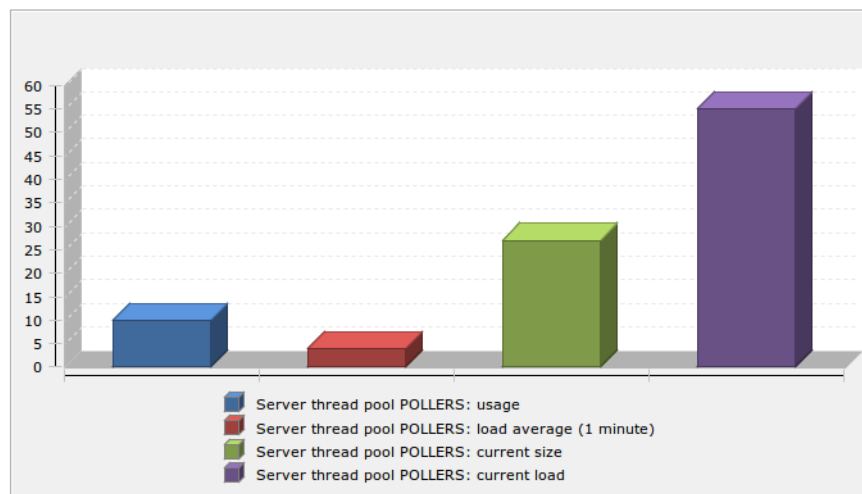
Line Chart

Line chart.



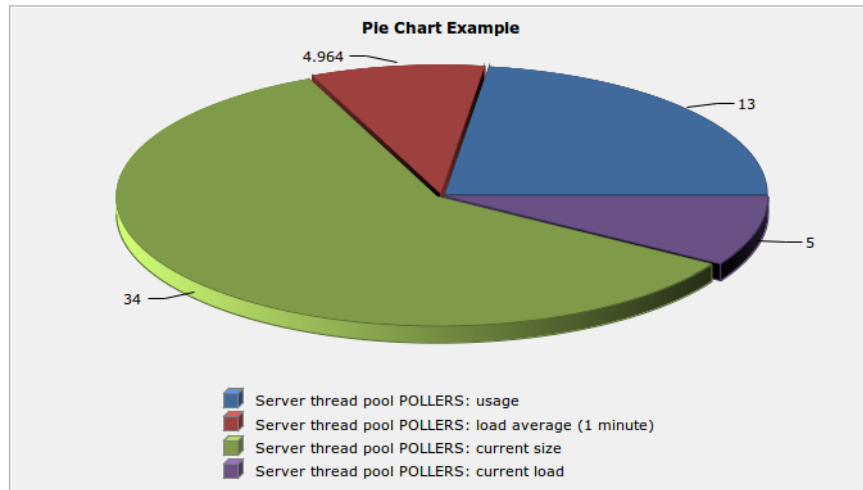
Bar Chart

Bar chart.



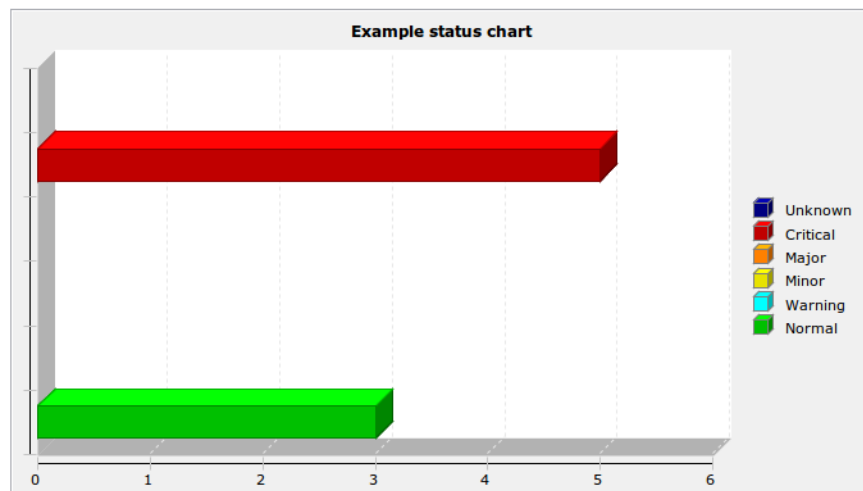
Pie Chart

Pie chart.



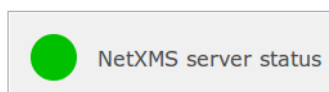
Status Chart

Bar chart which shows current status distribution for nodes under given root.



Status Indicator

Shows current status of selected object.



Dashboard

Another dashboard object (or multiple objects) rendered as element of this dashboard.

Network Map

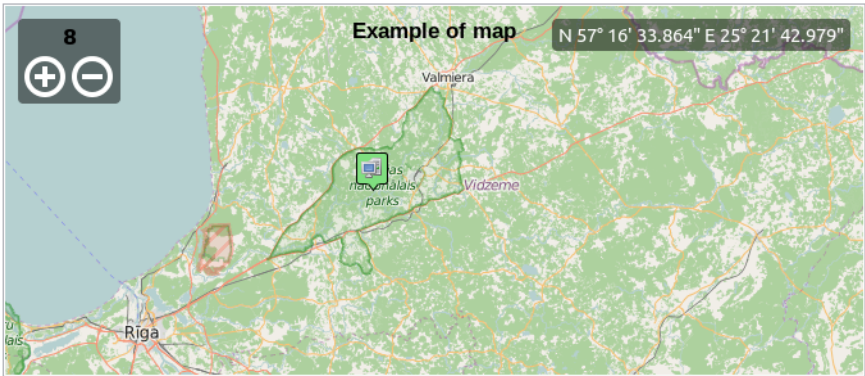
Network map object rendered as dashboard element.

Custom Widget

Custom widget provided by third party management client plugin. This options allows to add widget from third party loaded plugin.

Get Map

Geographic map centered at given location.



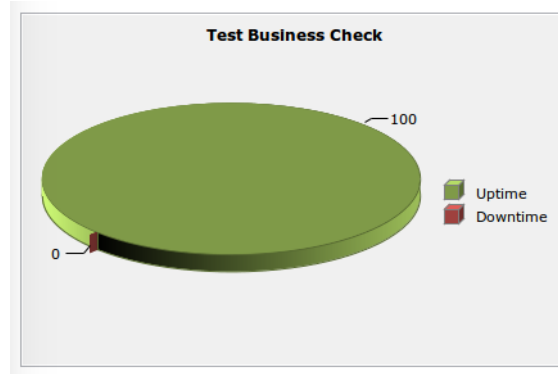
Alarm Viewer

List of alarms for given object subtree.

Severity	State	Source	Message	Count	Comment	Helpdesk ID	Ack/Resolve	Created	Last Change
Major	Outstanding	zev-ThinkPad-P50	Native agent is not responding	1				20.06.2016 19:59:01	20.06.2016 19:59:01
Major	Outstanding	zev-ThinkPad-P50	Interface "virbr0" unexpectedly changed state	1				20.06.2016 19:59:01	20.06.2016 19:59:01
Minor	Outstanding	zev-ThinkPad-P50	Status of DCI 665 (Internal: Server.ThreadPool)	1				20.06.2016 19:58:27	20.06.2016 19:58:27
Minor	Outstanding	zev-ThinkPad-P50	Status of DCI 668 (Internal: Server.ThreadPool)	1				20.06.2016 19:58:27	20.06.2016 19:58:27
Minor	Outstanding	zev-ThinkPad-P50	Status of DCI 669 (Internal: Server.ThreadPool)	1				20.06.2016 19:58:27	20.06.2016 19:58:27
Minor	Outstanding	zev-ThinkPad-P50	Status of DCI 670 (Internal: Server.ThreadPool)	1				20.06.2016 19:58:27	20.06.2016 19:58:27
Minor	Outstanding	zev-ThinkPad-P50	Status of DCI 676 (Internal: Server.ThreadPool)	1				20.06.2016 19:58:27	20.06.2016 19:58:27
Minor	Outstanding	zev-ThinkPad-P50	Status of DCI 673 (Internal: Server.ThreadPool)	1				20.06.2016 19:58:27	20.06.2016 19:58:27
Minor	Outstanding	zev-ThinkPad-P50	Status of DCI 671 (Internal: Server.ThreadPool)	1				20.06.2016 19:58:27	20.06.2016 19:58:27
Minor	Outstanding	zev-ThinkPad-P50	Status of DCI 672 (Internal: Server.ThreadPool)	1				20.06.2016 19:58:27	20.06.2016 19:58:27
Minor	Outstanding	zev-ThinkPad-P50	Status of DCI 675 (Internal: Server.ThreadPool)	1				20.06.2016 19:58:27	20.06.2016 19:58:27

Availability Chart

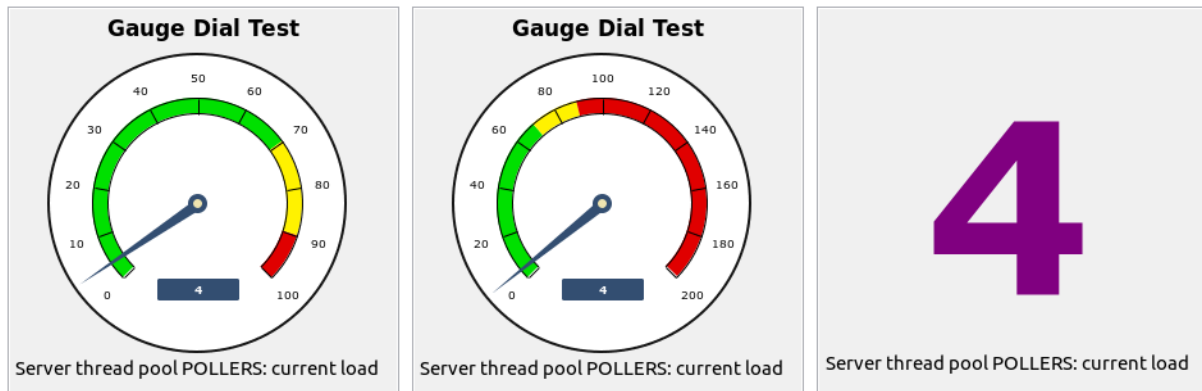
Pie chart showing availability percentage for given business service



Gauge

Gauge have 3 types of widgets

- Dial is radial gauge with configurable maximum, minimum values. Scale can have fixed color or can be separated to 3 color configurable zones.
- Bar is linear gauge with configurable maximum, minimum values. Scale can have fixed color or can be separated to 3 color configurable zones. (Not yet implemented)
- Text is text gauge, that can be colored using fixed color, changed depending on 3 configurable color zones or colored using threshold color (severity).

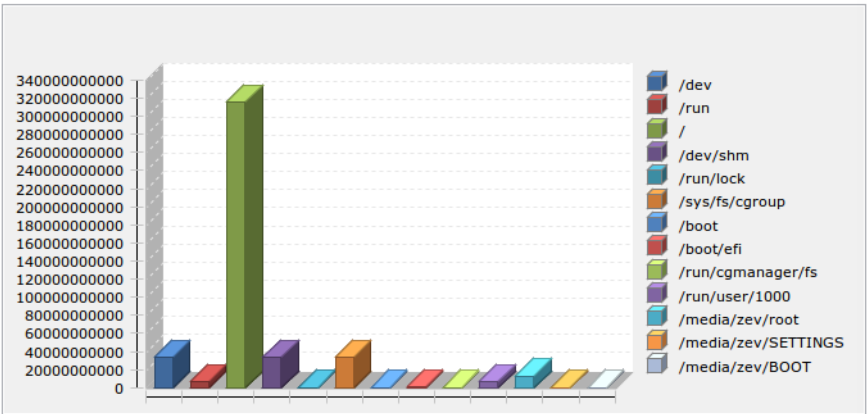


Web Page

Web page at given URL rendered within dashboard.

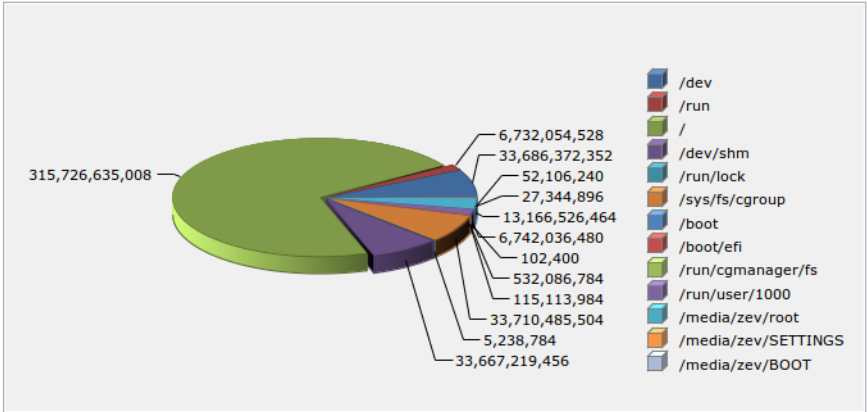
Bar Chart for Table DCI

Bar chart built from data collected via single table DCI.



Pie Chart for Table DCI

Pie chart built from data collected via single table DCI.



Separator

Separator, can be shown as line, box, or simply empty space.

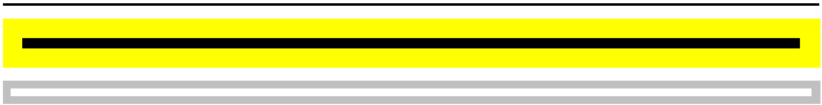


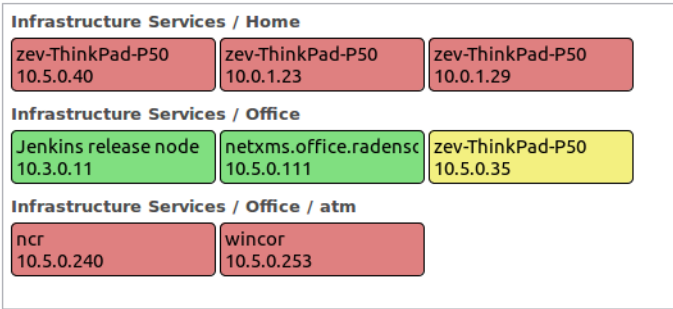
Table Value

This widget displays table with last values of Table DCI.

Status Map

Status map has three views: Flat view, Group view and Radial view.

Flat view and Group view display nodes as rectangles, using color to indicate their status. In Flat view nodes are displayed without grouping, whether in Group view nodes are grouped by containers.



Radial view displays containers and nodes as hierarchical colored radial layout.

DCI Summary Table

DCI Summary Table widget provides summary DCI information about objects under container.

Node	Status	Agent's versi	Agent log sta	Get database statu
Jenkins release node	0			
netxms.office.radensolutions.com	0	2.0.4		
zev-ThinkPad-P50	2	2.1-M1	0	0



Syslog Monitor

Syslog monitor widget. Has additional option to set root object to filter objects what will be shown in monitor. One object or a container that contains required objects can be set as root object.

Timestamp	Source	Severity	Facility	Host Name	Tag	Message
11.03.2021 19:37:20	zev-ThinkPad-P50	Informational	System	zev-ThinkPad-P50	dbus	dbus-daemon[2669]: [session uid=1000 pid
11.03.2021 19:37:20	zev-ThinkPad-P50	Informational	System	zev-ThinkPad-P50	dbus	dbus-daemon[2669]: [session uid=1000 pid
11.03.2021 19:37:10	zev-ThinkPad-P50	Informational	System	zev-ThinkPad-P50	gnome	gnome-shell[3082]: Window manager warn
11.03.2021 19:37:05	zev-ThinkPad-P50	Warning	System	zev-ThinkPad-P50	systemd	systemd-resolved[1343]: message repeate
11.03.2021 19:37:04	zev-ThinkPad-P50	Warning	System	zev-ThinkPad-P50	systemd	systemd-resolved[1343]: Server returned e
11.03.2021 19:36:30	zev-ThinkPad-P50	Informational	System	zev-ThinkPad-P50	gnome	gnome-shell[3082]: Window manager warn
11.03.2021 19:36:10	zev-ThinkPad-P50	Informational	System	zev-ThinkPad-P50	gnome	gnome-shell[3082]: Window manager warn
11.03.2021 19:36:10	zev-ThinkPad-P50	Informational	System	zev-ThinkPad-P50	gnome	gnome-shell[3082]: Window manager warn
11.03.2021 19:36:10	zev-ThinkPad-P50	Informational	System	zev-ThinkPad-P50	gnome	gnome-shell[3082]: Window manager warn
11.03.2021 19:36:10	zev-ThinkPad-P50	Informational	System	zev-ThinkPad-P50	gnome	gnome-shell[3082]: Window manager warn
11.03.2021 19:36:10	zev-ThinkPad-P50	Informational	System	zev-ThinkPad-P50	gnome	gnome-shell[3082]: Window manager warn
11.03.2021 19:36:10	zev-ThinkPad-P50	Informational	System	zev-ThinkPad-P50	gnome	gnome-shell[3082]: Window manager warn
11.03.2021 19:36:10	zev-ThinkPad-P50	Informational	System	zev-ThinkPad-P50	gnome	gnome-shell[3082]: Window manager warn
11.03.2021 19:36:04	zev-ThinkPad-P50	Warning	System	zev-ThinkPad-P50	systemd	systemd-resolved[1343]: message repeate
11.03.2021 19:36:04	zev-ThinkPad-P50	Warning	System	zev-ThinkPad-P50	systemd	systemd-resolved[1343]: Server returned e
11.03.2021 19:36:03	zev-ThinkPad-P50	Informational	System	zev-ThinkPad-P50	gnome	gnome-shell[3082]: Window manager warn
11.03.2021 19:36:03	zev-ThinkPad-P50	Informational	System	zev-ThinkPad-P50	gnome	gnome-shell[3082]: Window manager warn
11.03.2021 19:36:03	zev-ThinkPad-P50	Informational	System	zev-ThinkPad-P50	gnome	gnome-shell[3082]: Window manager warn
11.03.2021 19:36:03	zev-ThinkPad-P50	Informational	System	zev-ThinkPad-P50	gnome	gnome-shell[3082]: Window manager warn
11.03.2021 19:36:03	zev-ThinkPad-P50	Informational	System	zev-ThinkPad-P50	gnome	gnome-shell[3082]: Window manager warn
11.03.2021 19:36:03	zev-ThinkPad-P50	Informational	System	zev-ThinkPad-P50	gnome	gnome-shell[3082]: Window manager warn
11.03.2021 19:36:03	zev-ThinkPad-P50	Informational	System	zev-ThinkPad-P50	gnome	gnome-shell[3082]: Window manager warn
11.03.2021 19:36:03	zev-ThinkPad-P50	Informational	System	zev-ThinkPad-P50	gnome	gnome-shell[3082]: Window manager warn
11.03.2021 19:36:03	zev-ThinkPad-P50	Informational	System	zev-ThinkPad-P50	gnome	gnome-shell[3082]: Window manager warn
11.03.2021 19:35:56	zev-ThinkPad-P50	Informational	System	zev-ThinkPad-P50	gnome	gnome-shell[3082]: Window manager warn
11.03.2021 19:35:52	zev-ThinkPad-P50	Warning	Auth	zev-ThinkPad-P50	gnome	gnome-keyring-daemon[2675]: asked to re

SNMP Trap Monitor

SNMP Trap monitor widget. Has additional option to set root object to filter objects what will be shown in monitor. One object or a container that contains required objects can be set as root object.

Filter: <input type="text" value="Filter is empty"/>  				
Timestamp	Source IP	Source node	OID	Varbinds
12.03.2021 11:35:28	10.0.1.38	zev-ThinkPad-P50	.1.3.6.1.2.1.43.18.2.0.1	
12.03.2021 11:35:27	10.0.1.38	zev-ThinkPad-P50	.1.3.6.1.6.3.1.1.5.4.0.33	.1.3.6.1.6.3.1.1.5.4 == 'eth0'

Event monitor

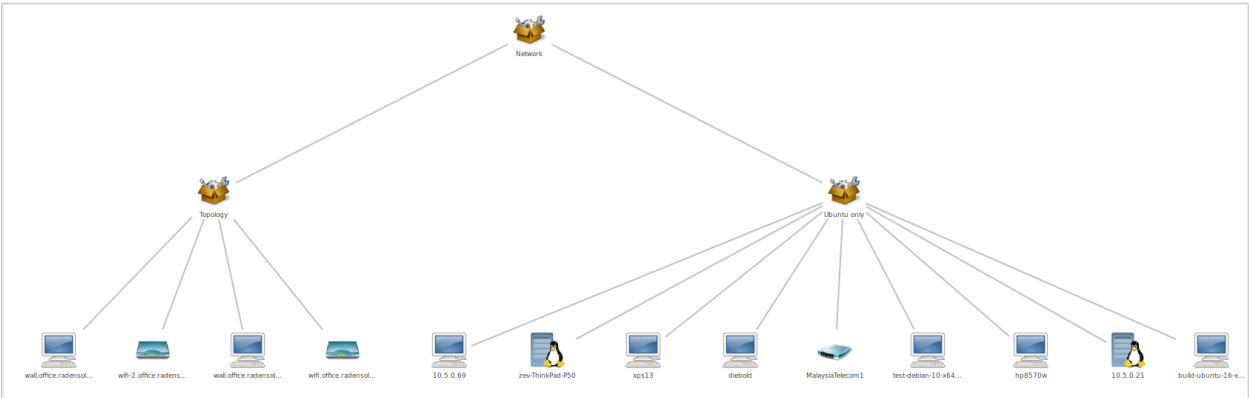
Event monitor widget. Has additional option to set root object to filter objects what will be shown in monitor. One object or a container that contains required objects can be set as root object.

Filter:

Timestamp	Source	Severity	Event	Message
12.03.2021 11:45:14	wifi-2.office.radensolutions.con	Warning	SYS_IF_UNKNOWN	Interface "bridge-vlan100" changed state to UNKNOWN (IP Addr: UNSPEC/0, IIndex: 8)
12.03.2021 11:45:14	wifi-2.office.radensolutions.con	Warning	SYS_IF_UNKNOWN	Interface "ether1.100" changed state to UNKNOWN (IP Addr: UNSPEC/0, IIndex: 7)
12.03.2021 11:45:14	wifi-2.office.radensolutions.con	Minor	SYS_IF_DOWN	Interface "ether1.4" changed state to DOWN (IP Addr: 10.5.4.16/24, IIndex: 6)
12.03.2021 11:45:06	wifi-2.office.radensolutions.con	Warning	SYS_IF_UNKNOWN	Interface "ether2" changed state to UNKNOWN (IP Addr: UNSPEC/0, IIndex: 4)
12.03.2021 11:45:06	wifi-2.office.radensolutions.con	Warning	SYS_IF_UNKNOWN	Interface "ether1" changed state to UNKNOWN (IP Addr: UNSPEC/0, IIndex: 3)
12.03.2021 11:45:06	wifi-2.office.radensolutions.con	Warning	SYS_IF_UNKNOWN	Interface "wlan2" changed state to UNKNOWN (IP Addr: UNSPEC/0, IIndex: 2)
12.03.2021 11:45:06	wifi-2.office.radensolutions.con	Warning	SYS_IF_UNKNOWN	Interface "wlan1" changed state to UNKNOWN (IP Addr: UNSPEC/0, IIndex: 1)
12.03.2021 11:45:06	wifi-2.office.radensolutions.con	Major	SYS_SNMP_UNREACHABLE	SNMP agent is not responding
12.03.2021 11:45:14	wifi-2.office.radensolutions.con	Critical	SYS_NODE_DOWN	Node down

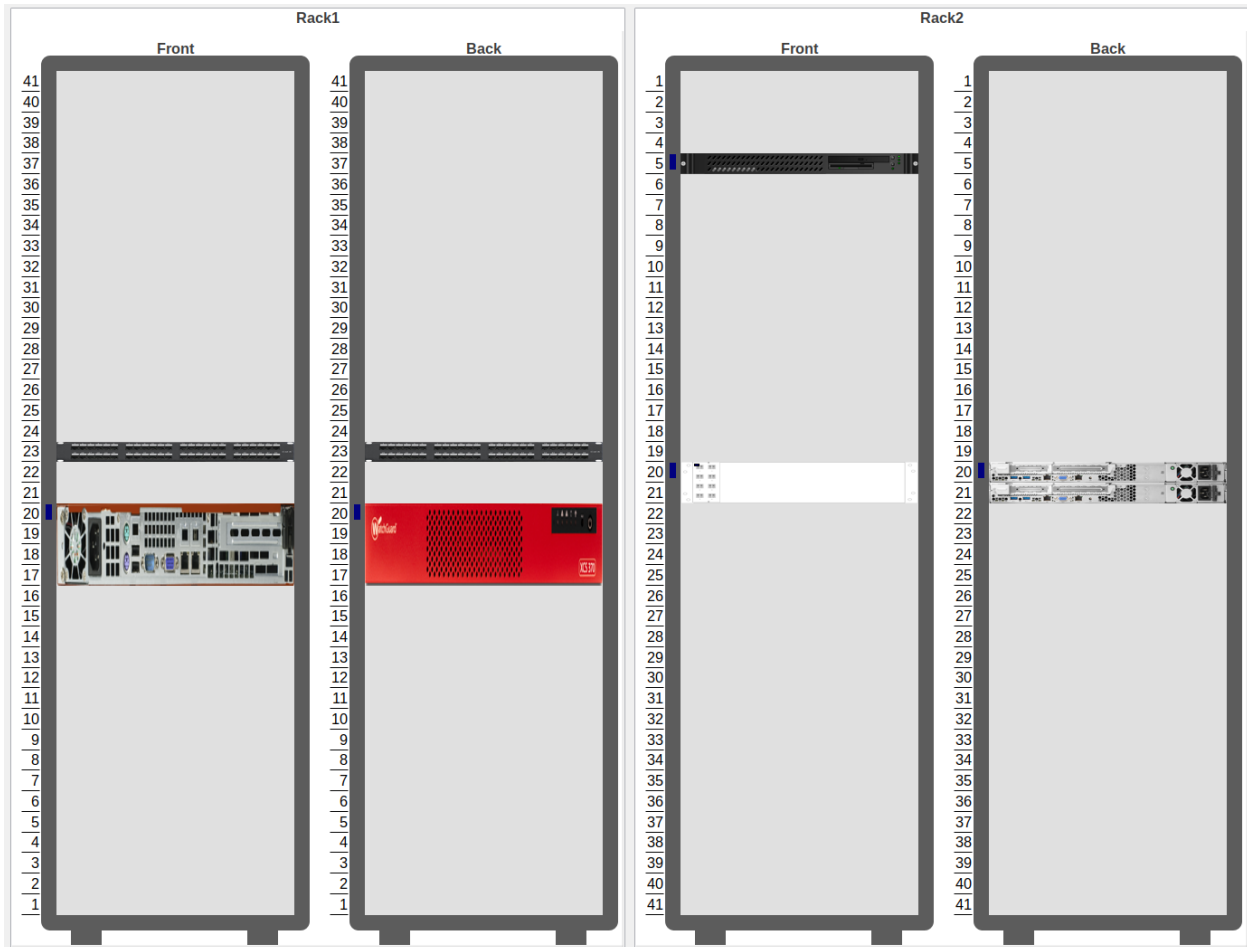
Service component map

Map displays hierarchy of objects in *Infrastructure Service* starting from selected root object.



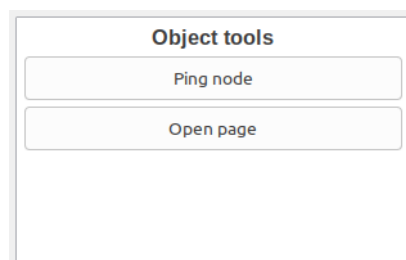
Rack diagram

Shows rack front, back or both views with object placement in it.



Object tools

Shows buttons with pre configured object tools, that are executed on click.



Object query

Shows columns with filtered objects' information.

Object query has 2 main configurations. *Query* that filterers objects and provide option to create additional information about object in columns and *Object Properties* that lists information that should be shown in table.

Query

It is a script that is executed on each object and should return true if object should be displayed in the table and false if it should not. It has special syntax that provides option to calculate additional values for columns in *Object Properties*

section. This syntax is optional and usual NXSL script can be used instead. Usual NXSL script should return true or map (where key is column name and value is value for this column) if node should be shown and false if not, additional self calculated columns can be defined as global variables.

Syntax:

```
with
  varName = { code or expression },
  varName = { code or expression }
/* Might be as many blocks as required.
 * varName is a name of the variable where result of a code will be assigned.
 * It can be used later in the code in expression or to be displayed in table
 * using the same name in the Object Properties part.
 */
expression
/* Short circuit evaluated expression. This expression is executed first and if it
↳contains not yet calculated
 * varName then variable is calculated and used in expression. Expression that should
↳result as true or false
 * as a sign if this object should be displayed in table or not. No semicolon at the
↳end.
 */
```

This page provides option to configure columns that should be used for ordering, refresh interval and record limit. To order column write a coma separated list of attribute named or varNames with - sign to order in descending order and with + sign to order in ascending order.

Object Properties

This property page is used to organize required columns and column order in table. Each column configuration consists of name of object's attribute or varName defined in Query page, display name used as a name for a column and data type of the column.

Example

This example will show how to filter nodes that only have alarms on them, are not in maintenance mode and show count of critical alarms on the node, order by critical alarm count the list and then by node name. Example shows two different options how to write the same script so only one of them should be used.

Configuration:

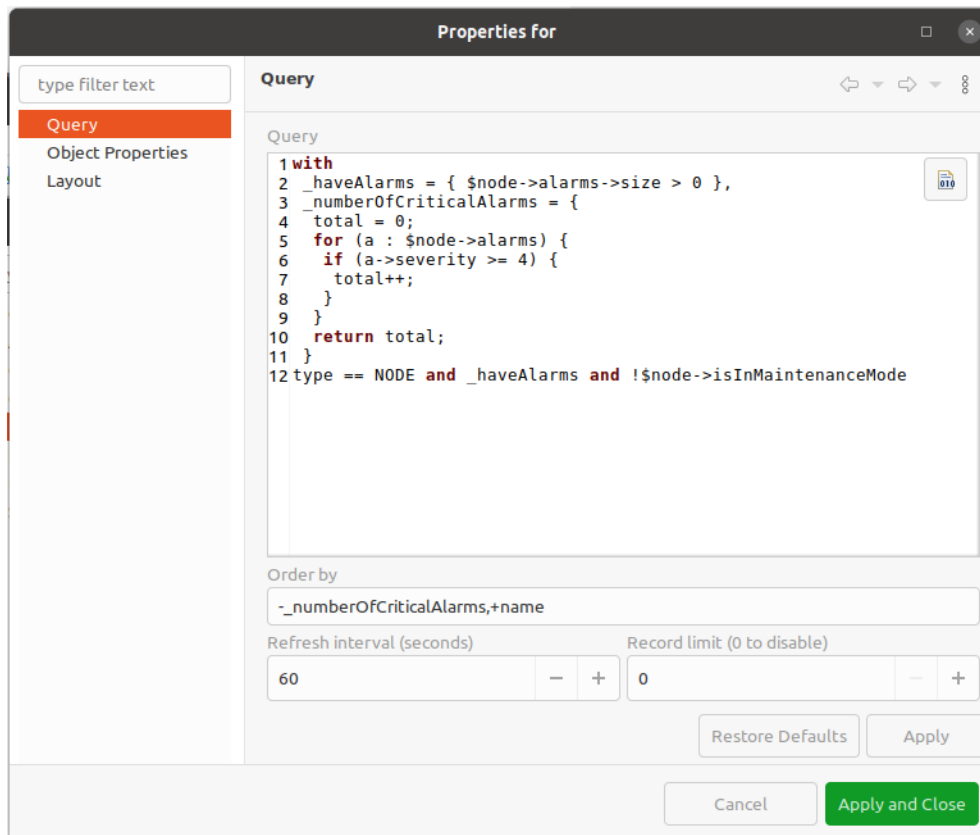


Fig. 2: Option 1. Query script with “with” syntax

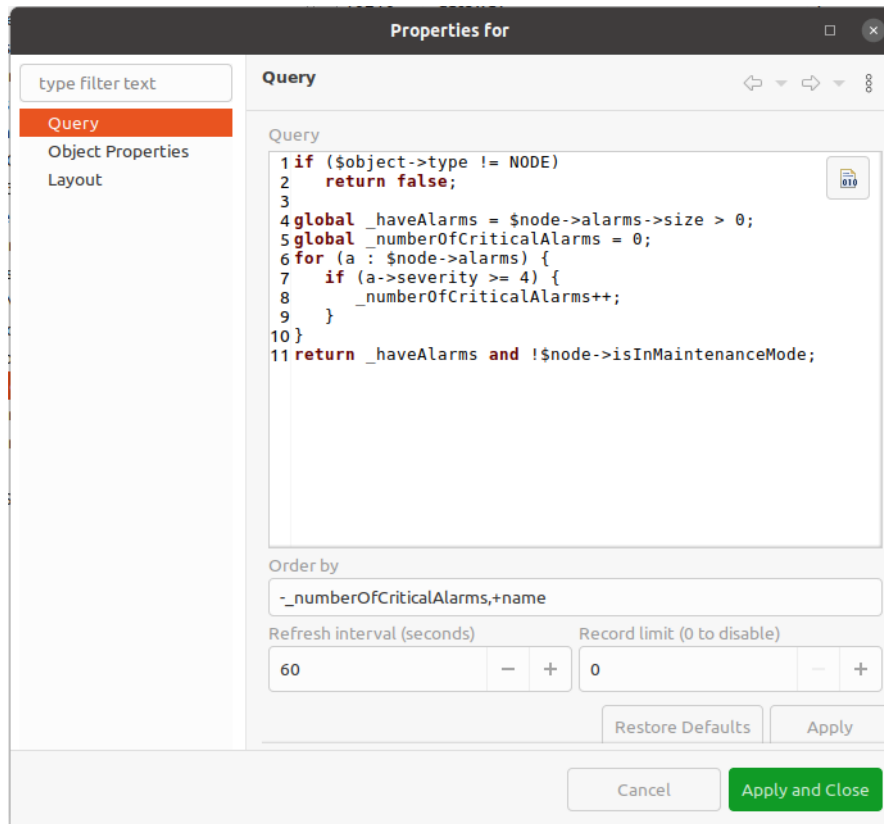


Fig. 3: Option 2. Query script with usual NXSL script and global variables

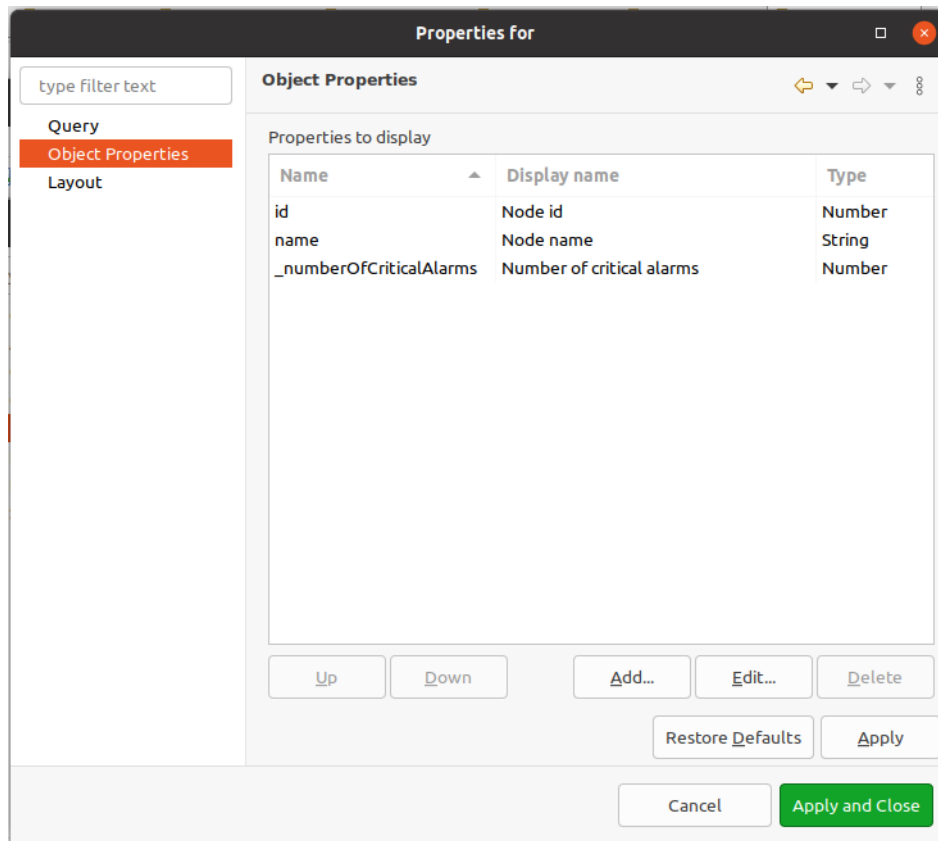


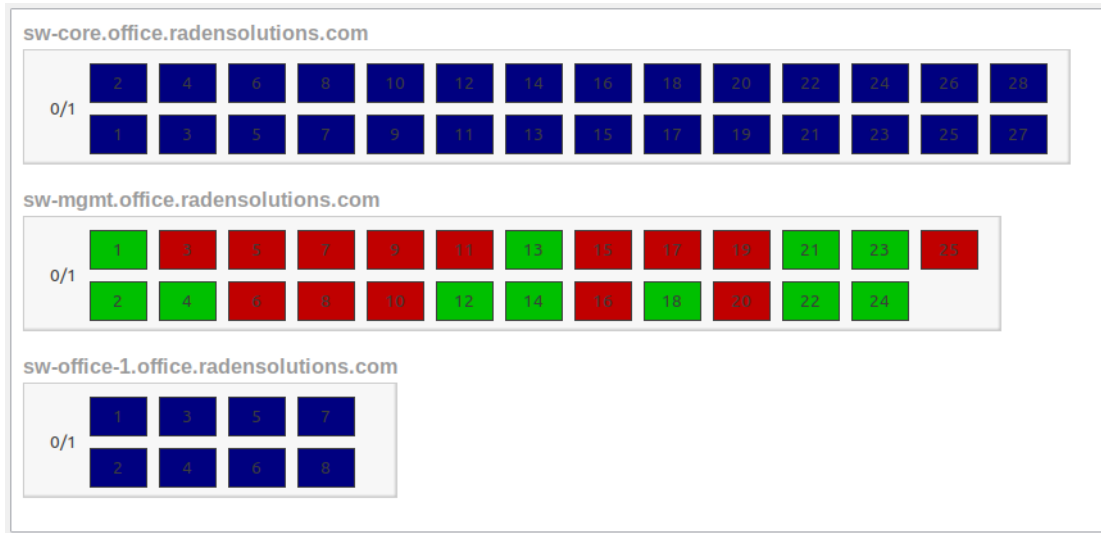
Fig. 4: Configuration of *Properties to display* will be the same for both scripts

Result:

Node id	Node name	Number of critical alarms
10230	DESKTOP-UU54OHE	2
10180	zev-ThinkPad-P50	2
10210	_gateway	1
7430	_gateway	1
4912	Alexs-MacBook-Pro.local	1
7196	build-debian-6-x64.office.radensolutions.com	1
7178	build-debian-6-x86.office.radensolutions.com	1
7202	build-freebsd-12-x64.office.radensolutions.com	1
7198	build-opensuse-leap15-x64.office.radensolutions.com	1
7530	hp8570w	1
5009	ilo-power710.office.radensolutions.com	1
6741	MSEDGEWIN10	1
5512	NPI096EF9	1
4967	sun-v240.office.radensolutions.com	1
6646	sw-office-1.office.radensolutions.com	1
6711	syslog.office.radensolutions.com	1
10196	user-PC	1
7400	XPS13	1
9187	XPS13	1
9110	10.0.1.22	0
9133	10.5.5.60	0
7176	build-rpi.office.radensolutions.com	0
7534	build-ubuntu-16-x64.office.radensolutions.com	0
4916	canon.office.radensolutions.com	0
4863	dc.office.radensolutions.com	0
6679	diebold	0
4969	docker1.office.radensolutions.com	0
5047	endurox.office.radensolutions.com	0
4873	fin.office.radensolutions.com	0
4965	ilo-sun-v240.office.radensolutions.com	0

Port view

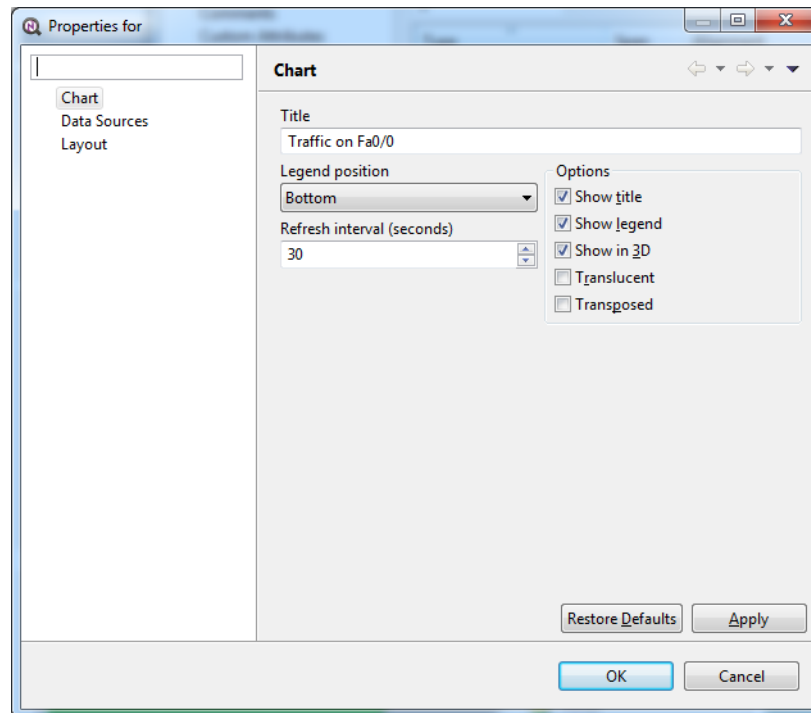
Shows ports schematic with each port status. One object or a container that contains required objects can be set as root object.



14.2.2 Element Property Pages

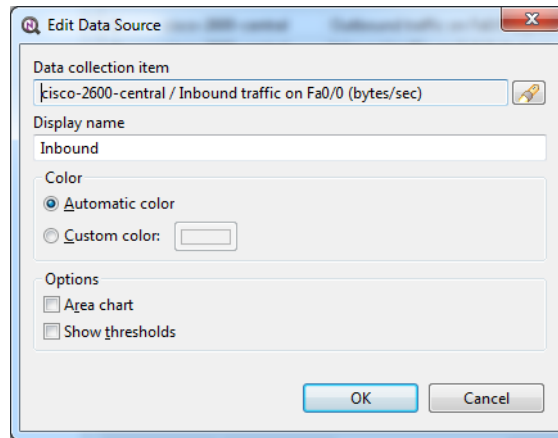
Chart

Chart page is available for all chart type elements: Bar Chart, Bar Chart for Table DCI, Dial Chart, Line Chart, Pie Chart, Pie Chart for Table DCI. It defines basic properties of a chart.



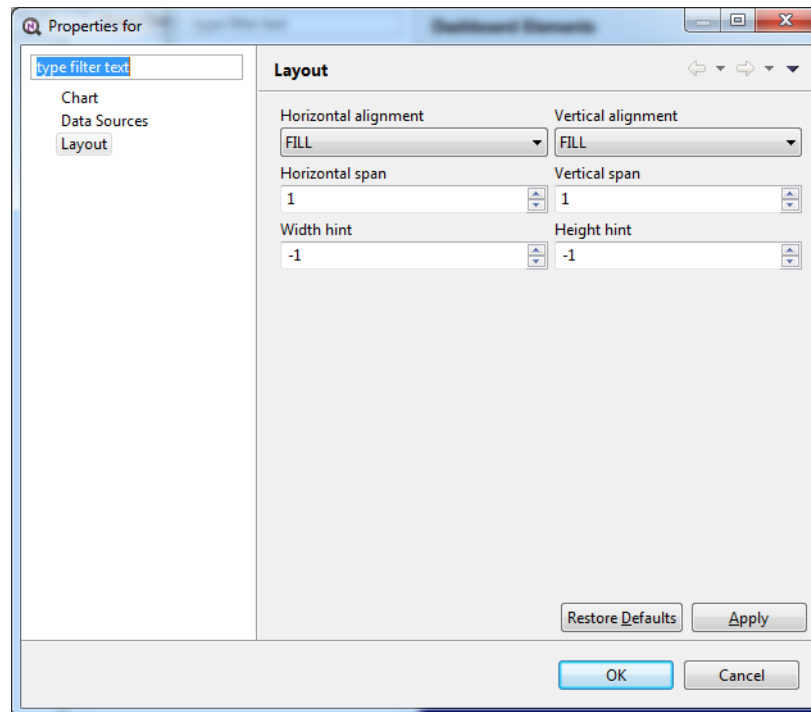
Data Sources

Data sources page is available for all DCI based elements: Bar Chart, Dial Chart, Line Chart and Pie Chart. Here you can define what DCIs should be used as data sources for the chart. Up to 16 DCIs can be added to a single chart. You can configure multiple properties for each data source. To edit data source, either double click on appropriate item in the list, or press *Edit* button. Data source configuration dialog looks like following:



Property	Description
Data collection item	DCI object to be used.
Display name	Name for this data source to be used in chart's legend. If left empty, DCI description will be used.
Colour	Allows you to define specific color for this data source or let system to pick one automatically.
Area chart	This option is valid only for line charts and toggles data source display as filled area instead of line.
Show thresholds	This option is valid only for line charts and toggles display of configured thresholds.

Layout



Property	Description
Horizontal alignment	Horizontal alignment for this element. Possible values are <i>FILL</i> , <i>CENTER</i> , <i>LEFT</i> , and <i>RIGHT</i> .
Vertical alignment	Vertical alignment for this element. Possible values are <i>FILL</i> , <i>CENTER</i> , <i>TOP</i> , and <i>BOTTOM</i> .
Horizontal span	Specify how many grid cells this element will occupy horizontally.
Vertical span	Specify how many grid cells this element will occupy vertically.
Width hint	Hint for element's width in pixels. Default value of <i>-1</i> means that layout manager will decide width for element.
Height hint	Hint for element's height in pixels. Default value of <i>-1</i> means that layout manager will decide width for element.

See detailed information about layout in section [Understanding Element Layout](#).

Web Page

:guilabel`Web Page` property page is available for web page type elements. Here you can define URL to be displayed and optional title. If title is not empty, it will be displayed above page content.

14.2.3 Understanding Element Layout

Dashboard uses grid concept to layout it's elements. Available space is divided into rows and columns, and each element occupies one or more cells. The number of columns is configured in dashboard object properties, and number of rows is calculated automatically based on number of columns, elements, and cells occupied by each element. Elements are laid out in columns from left to right, and a new row is created when there are no space left for next element on current row. Each element has horizontal and vertical alignment properties. Default for both is *FILL*. Possible alignment values are following:

Value	Description
FILL	Make element to fill whole cell. Also causes to grab excess free space available inside dashboard. If more than one element is trying to grab the same space, then the excess space is shared evenly among the grabbing elements.
CENTER	Center element within cell.
LEFT/TOP	Align element to left/top of the cell.
RIGHT/BOTTOM	Align element to right/bottom of the cell.

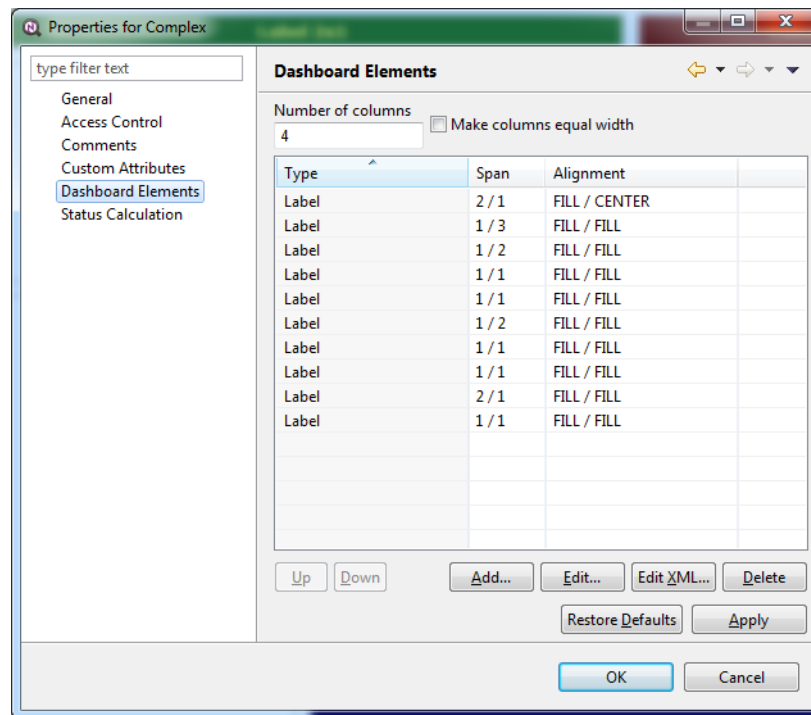
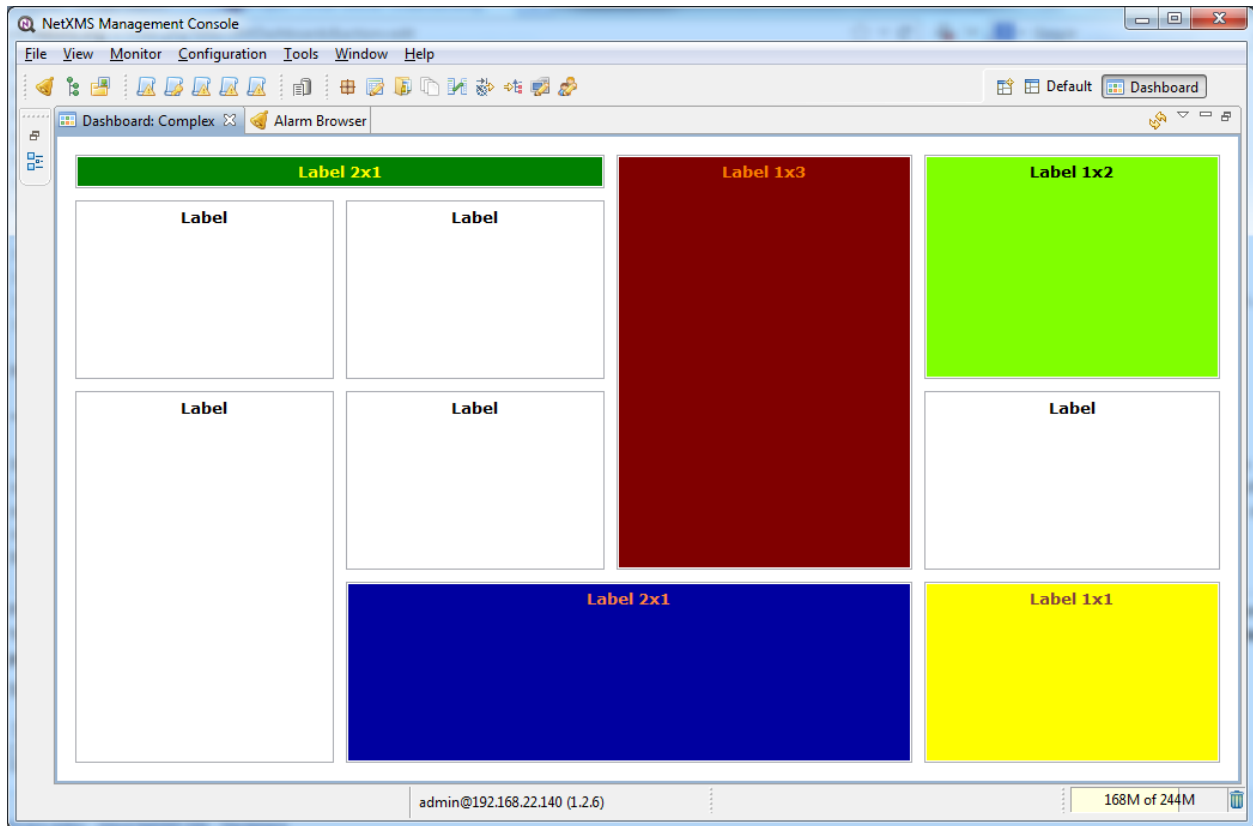


Fig. 5: Complex layout configuration

This configuration will be rendered into this layout:



14.2.4 Dashboard Rotation

To create configuration when management client displays multiple dashboards one by one in a loop, follow these steps:

- Create all dashboards you want to show
- Create additional dashboard object, with single element of type *Dashboard* inside
- Add all dashboards you want to show to dashboard list of that element and set desired time between changing dashboards.

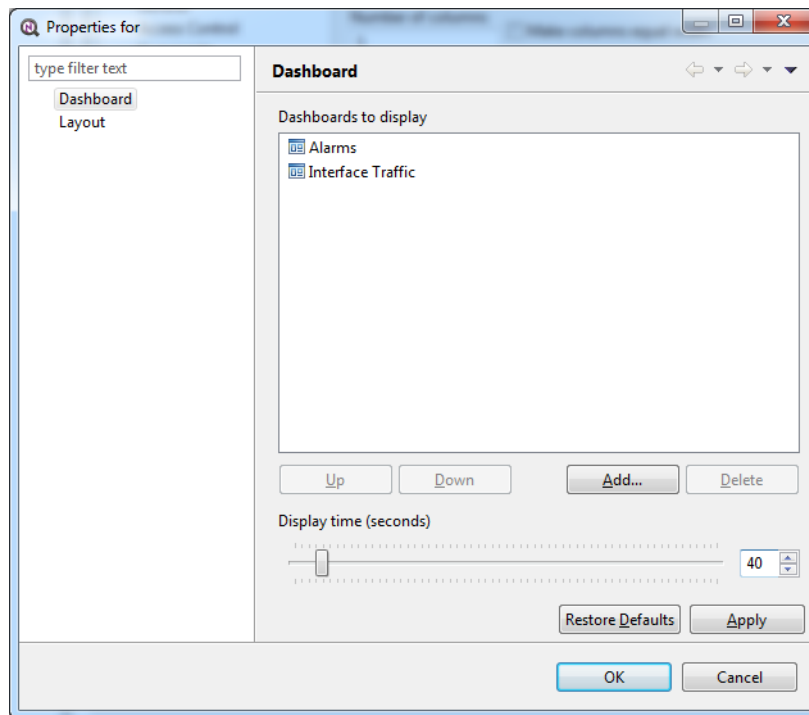


Fig. 6: Sample configuration of two dashboards displayed in a loop for 40 seconds each.

14.2.5 Tutorials

Dashboard creation tutorial available on [Youtube](#)

14.3 Graphs

You can view collected data in a graphical form, as a line chart. To view values of some DCI as a chart, first open either *Data Collection Editor* or *Last Values* view for a host. You can do it from the *Object Browser* or map by selection host, right-clicking on it, and selecting *Data collection* or *Last DCI values*. Then, select one or more DCIs (you can put up to 16 DCIs on one graph), right-click on them and choose *Graph* from the pop-up menu. You will see graphical representation of DCI values for the last hour.

When the graph is open, you can do various tasks:

14.3.1 Select different time interval

By default, you will see data for the last hour. You can select different time interval in two ways:

1. Select new time interval from presets, by right-clicking on the graph, and then selecting *Presets* and appropriate time interval from the pop-up menu.
2. Set time interval in graph properties dialog. To access graph properties, right-click on the graph, and then select *Properties* from the pop-up menu. Alternatively, you can use main application menu: *Graph* ▶ *Properties*. In the properties dialog, you will have two options: select exact time interval (like 12/10/2005 from 10:00 to 14:00) or select time interval based on current time (like last two hours).

14.3.2 Turn on automatic refresh

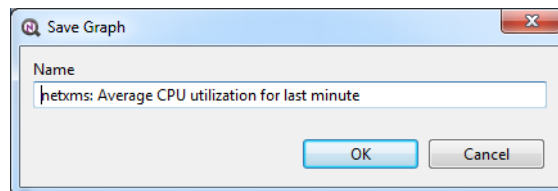
You can turn on automatic graph refresh at a given interval in graph properties dialog. To access graph properties, right-click on it, and select *Properties* from the pop-up menu. Alternatively, you can use main application menu: *Graph ▶ Properties*. In the properties dialog, select the *Refresh automatically* checkbox and enter a desired refresh interval in seconds in edit box below. When automatic refresh is on, you will see *Autoupdate* message in the status bar of graph window.

14.3.3 Change colors

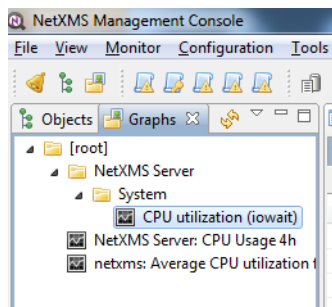
You can change colors used to paint lines and graph elements in the graph properties dialog. To access graph properties, right-click on it, and select *Properties* from the pop-up menu. Alternatively, you can use main application menu: *Graph ▶ Properties*. In the properties dialog, click on colored box for appropriate element to choose different color.

14.3.4 Save current settings as predefined graph

You can save current graph settings as predefined graph to allow quick and easy access in the future to information presented on graph. Preconfigured graphs can be used either by you or by other NetXMS users, depending on settings. To save current graph configuration as predefined graph, select *Save as predefined* from graph view menu. The following dialog will appear:



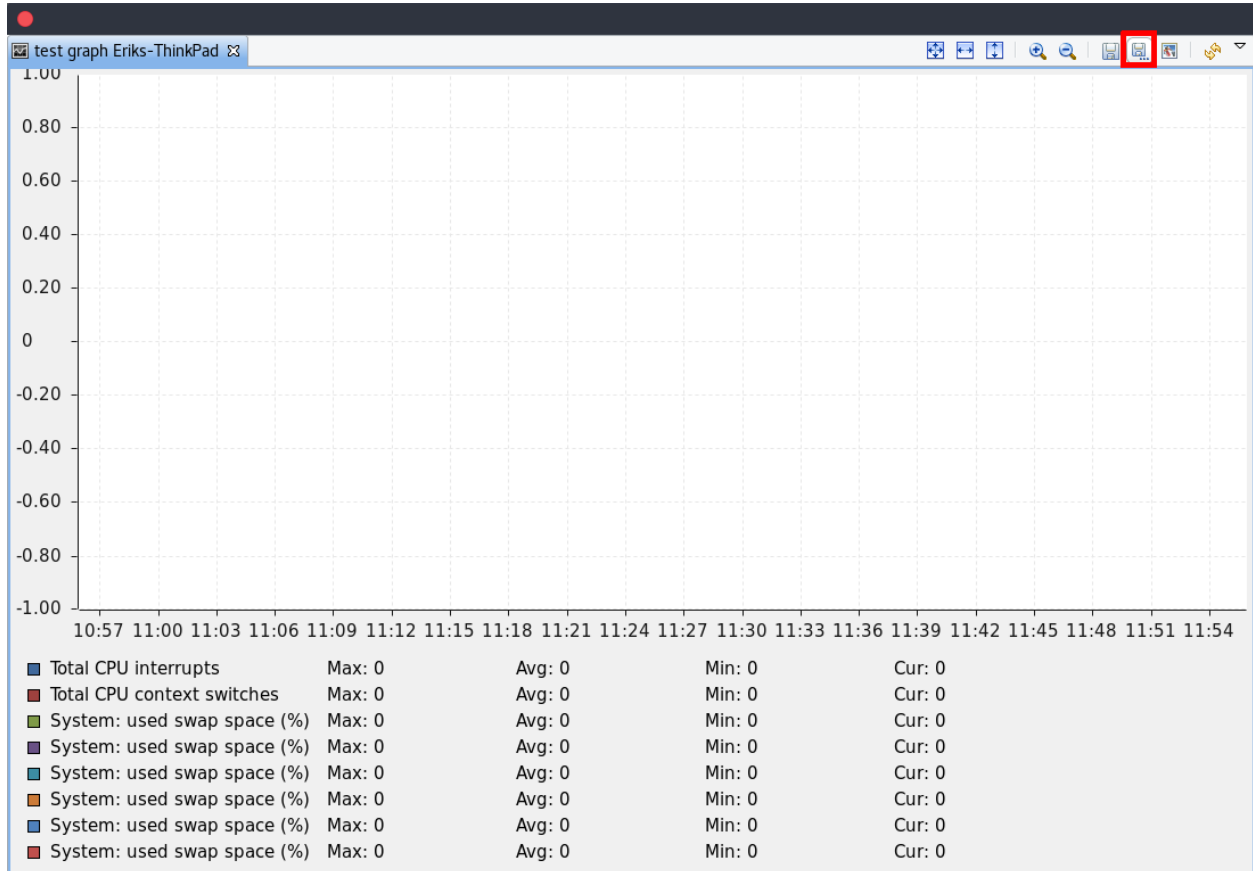
In *Graph name* field, enter desired name for your predefined graph. It will appear in predefined graph tree exactly as written here. You can use `->` character pair to create subtree. For example, if you name your graph `NetXMS Server->System->CPU utilization (iowait)` it will appear in the tree as following:



You can edit predefined graph by right-clicking on it in predefined graph tree, and selecting *Properties* from context menu. On *Predefined Graph* property page you can add users and groups who will have access to this graph. Note that user creating the graph will always have full access to it, even if he is not in access list.

If you need to delete predefined graph, you can do it by right-clicking on it in predefined graph tree, and selecting *Delete* from context menu.

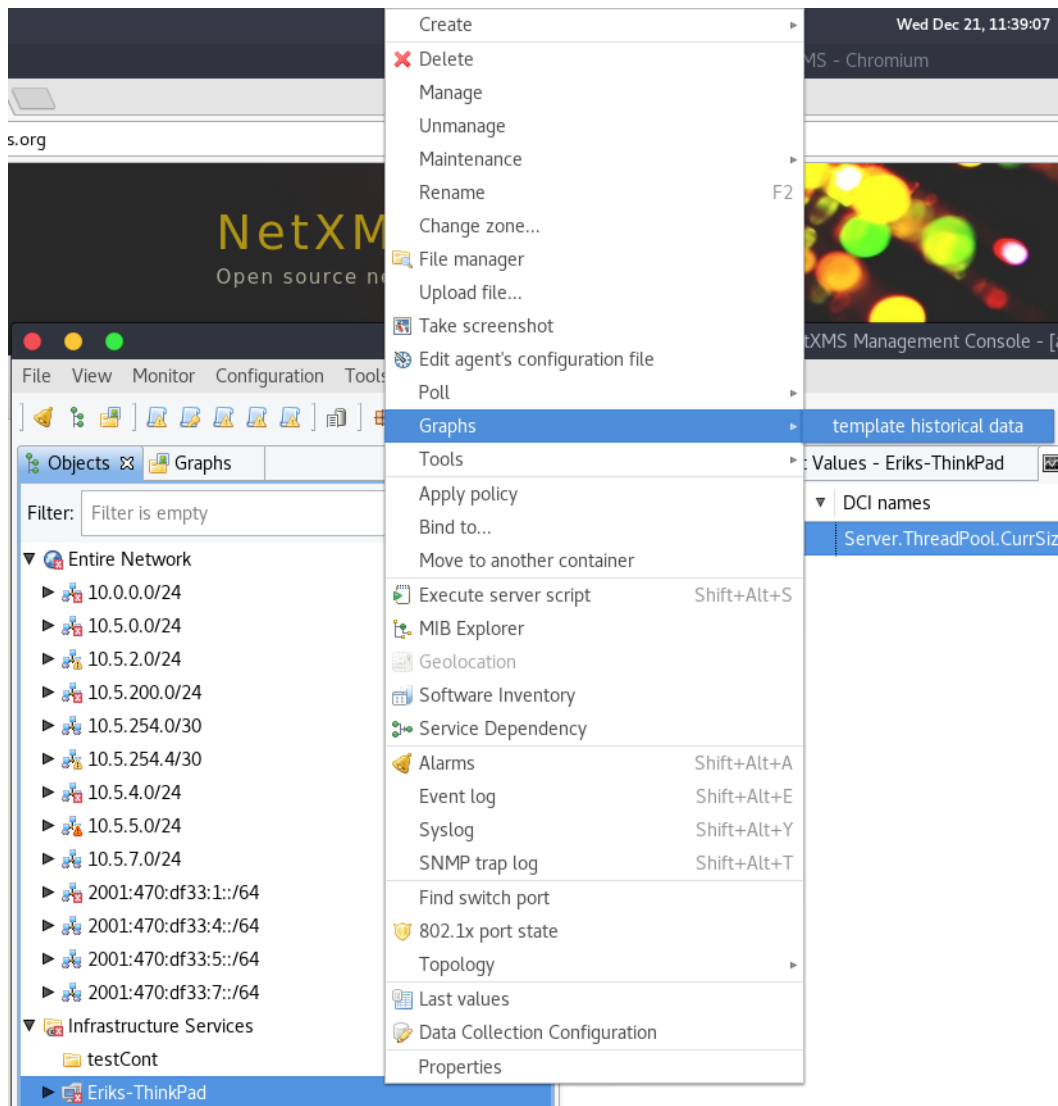
14.3.5 Save current settings as template graph



Current graph settings can be saved as a template graph for an easy template graph creation. The difference between predefined graphs and template graphs are that template graphs are not configured to view specific DCI's on a node, instead they are configured to view DCI names that can be found on many nodes (e.g. `FileSystem.FreePerc()`). This allows for the creation of certain graph templates to monitor, for example, disk usage that can be reused on any node to which the appropriate DCI's are applied on via *DCI configuration*.

See detailed information on template graphs in the section *Template Graph Configuration*.

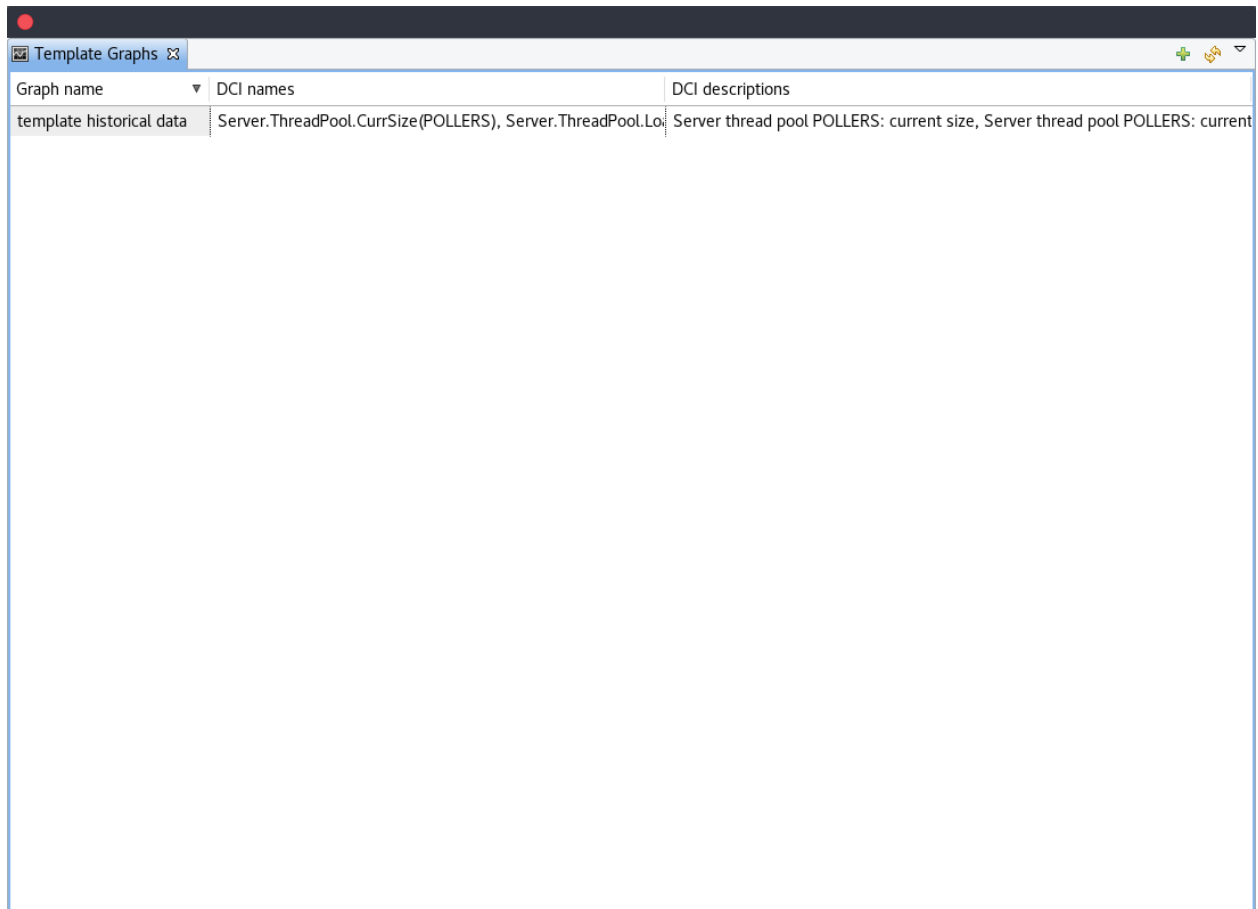
In the Graph name field of the pop-up save dialog, enter the desired name for the template graph by which you can later identify your it in the *Template Graph Configuration* which can be found in *Configuration*•*Template Graph Configuration*.



Template graphs can be accessed in the *Object Browser* as seen on the screenshot above. When a template graph is created, it will appear in the sub-menus of the nodes found in *Object Browser*, the rest of the settings can be accessed by editing a template graph in the *Template Graph Configuration*.

14.3.6 Template Graph Configuration

Template graphs are used to ease the monitoring of a pre-set list of DCI's on multiple nodes by adding a list of DCI names to the template source. This allows for the possibility to create templates to monitor specific data on any node to which the appropriate DCI's are applied on.



The *Template Graph Configuration* is used to create and edit template graphs. Properties for already created template graphs can be brought up by double clicking the template graph you wish to edit and new ones can be added by pressing the green cross on the top right or by right clicking and selecting *Create new template graph*.

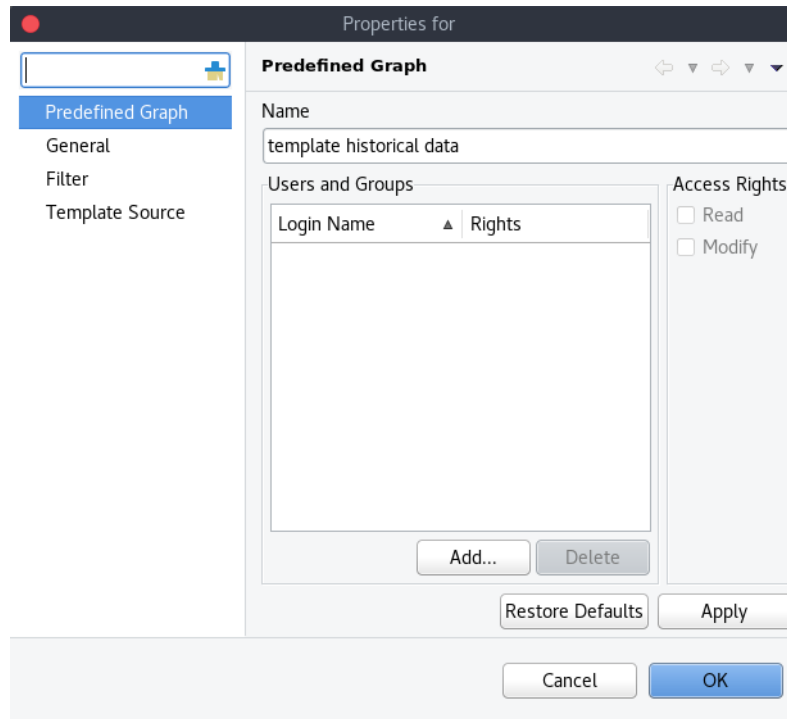


Fig. 7: Name and access rights of a graph

The above property page provides the possibility to configure the name of the template graph and the access rights. The user who has created the template graph will have full access to it even though the username will not show up in the access right list.

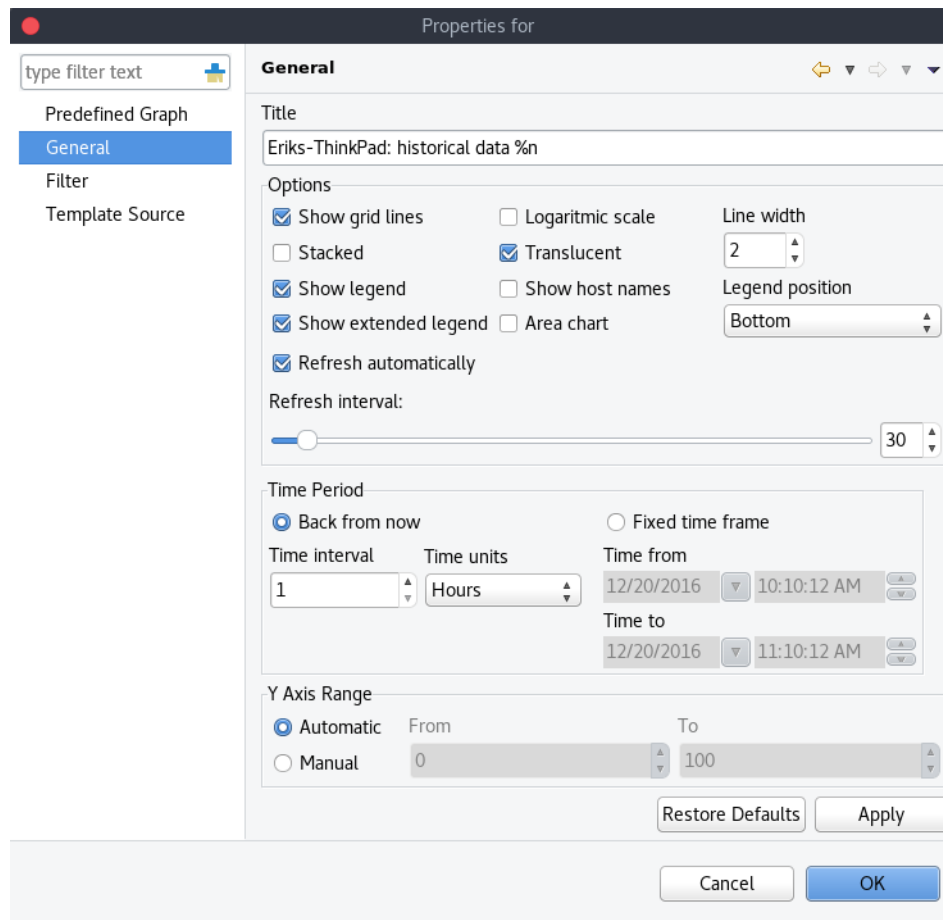


Fig. 8: General graph properties.

Title:

- The title that the graph will have when opened.
- The title can contain special characters described in [Macro Substitution](#).

Options:

Option	Description
Show grid lines	Enable or disable grid lines for the graph.
Stacked	Stacks the graphs of each value on top of one another to be able to see the total value easier (e.g. useful when monitoring cpu usage).
Show legend	Enable or disable the legend of the graph.
Show extended legend	Enable or disable the extended legend of the graph (Max, Avg, Min, Curr).
Refresh automatically	Enable or disable auto-refresh.
Logarithmic scale	Use the logarithmic scale for the graph.
Translucent	Enable or disable the translucency of the graph.
Show host names	Show host name of the node from which the value is taken.
Area chart	Highlights the area underneath the graph.
Line width	Adjust the width of the lines.
Legend position	Set the position of the legend.
Refresh interval	Set the refresh interval.

Time Period:

Provides the possibility to configure the time period of the graph. It is possible to set a dynamic time frame (Back from now) and a static time frame (Fixed time frame).

Y Axis Range:

Adjust the range of the Y axis on the graph.

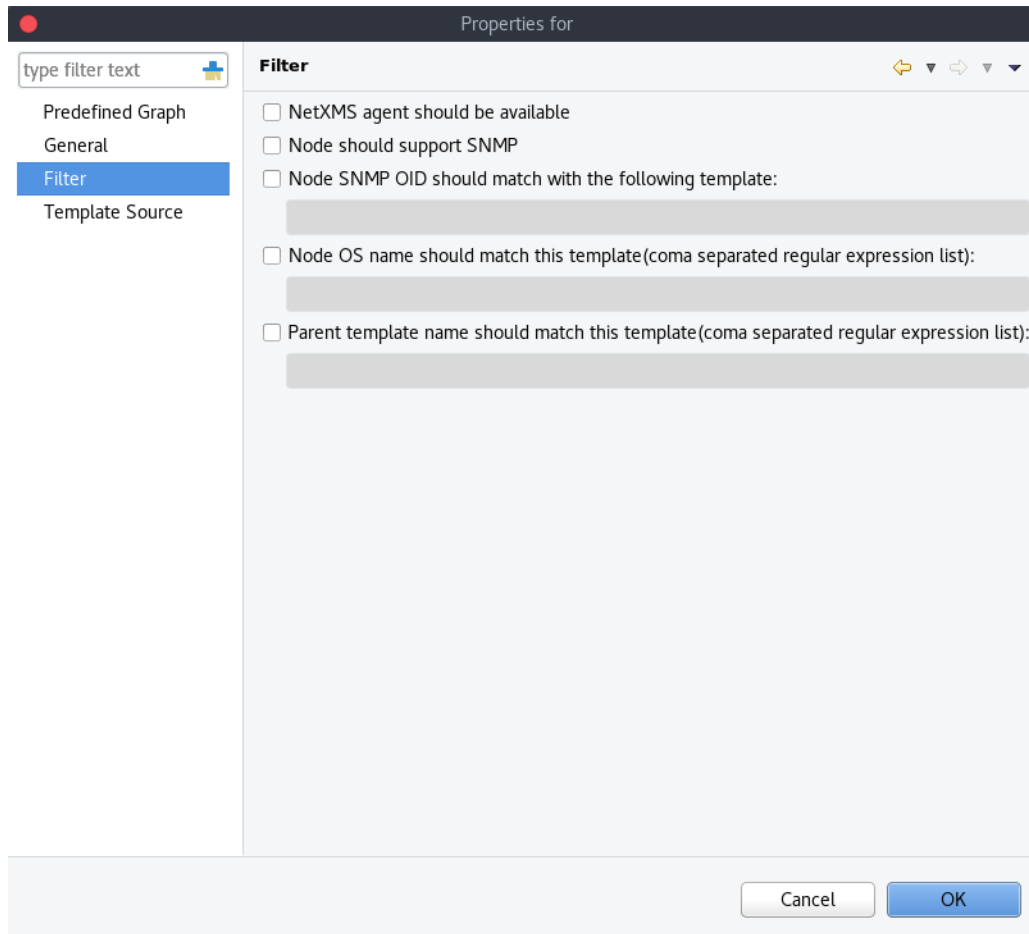


Fig. 9: Template graph filter properties.

It may be necessary to set certain filters for a template graph. This can be useful if the graph contains DCI names that are only available on NetXMS agent or are SNMP dependant.

More information on filters can be found in [Filter](#).

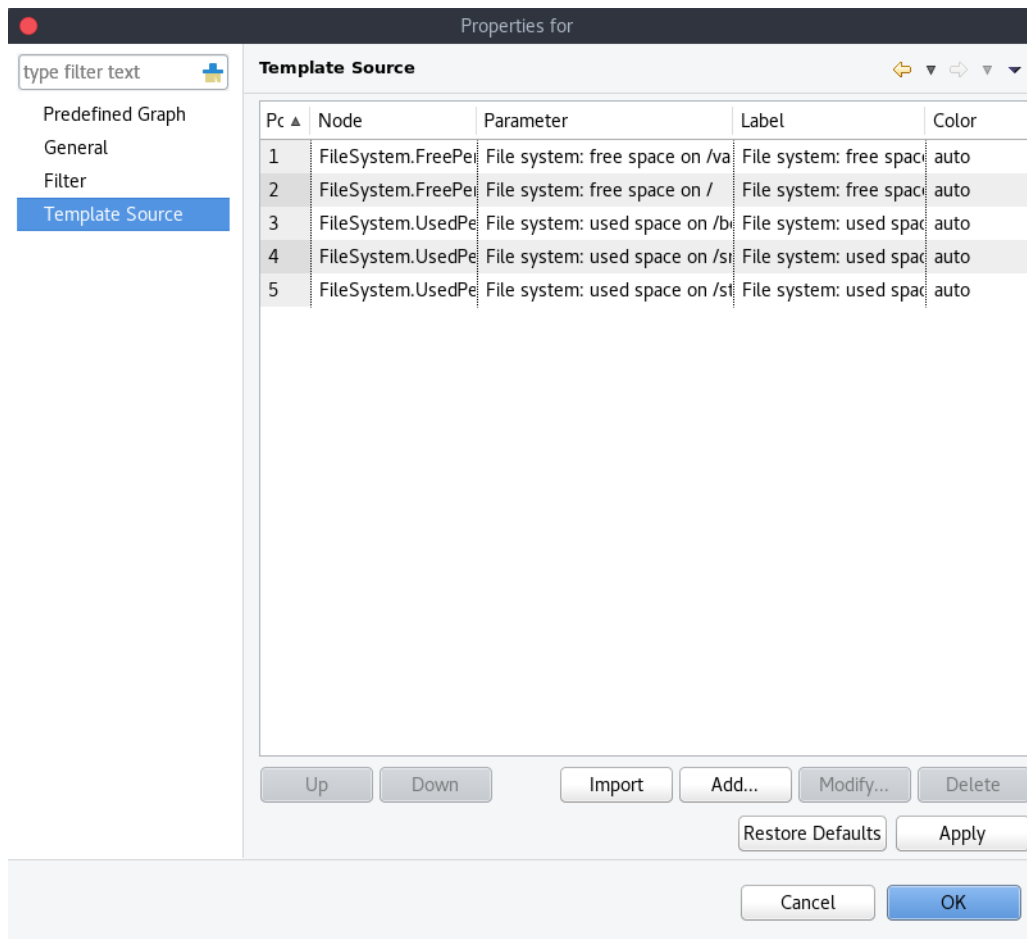


Fig. 10: Template graph sources

There are two options to add sources to the template graph. Sources can be added manually by configuring the Data Source parameters yourself or by importing data source information from DCI's that have already been applied to other nodes.

Edit Data Source

Display name
Server thread pool AGENT: current size

Display format
%s

DCI Name
Server.ThreadPool.CurrSize(AGENT)

DCI Description
Server thread pool AGENT: current size

Display type
Default

Color
☒ Automatic color
☐ Custom color:

Options
☐ Show thresholds
☐ Invert values
☐ Multiple match

Cancel OK

When adding or editing a source, it is possible to use Java regex in the DCI Name and DCI Description fields. This can be handy when used with the Multiple match option which will use all DCI's that match the particular regex. The order in which the DCI list is searched is first by DCI Name and then by DCI Description.

14.4 History

You can view collected data in a textual form, as a table with two columns - *timestamp* and *value*. To view values of some DCI as a table, first open either *Data Collection Editor* or *Last Values* view for a host. You can do it from the *Object Browser* or map by selection host, right-clicking on it, and selecting *Data collection* or *Last DCI values*. Then, select one or more DCIs (each DCI data will be shown in separate view), right-click on them and choose *Show history* from the pop-up menu. You will see the last 1000 values of the DCI.

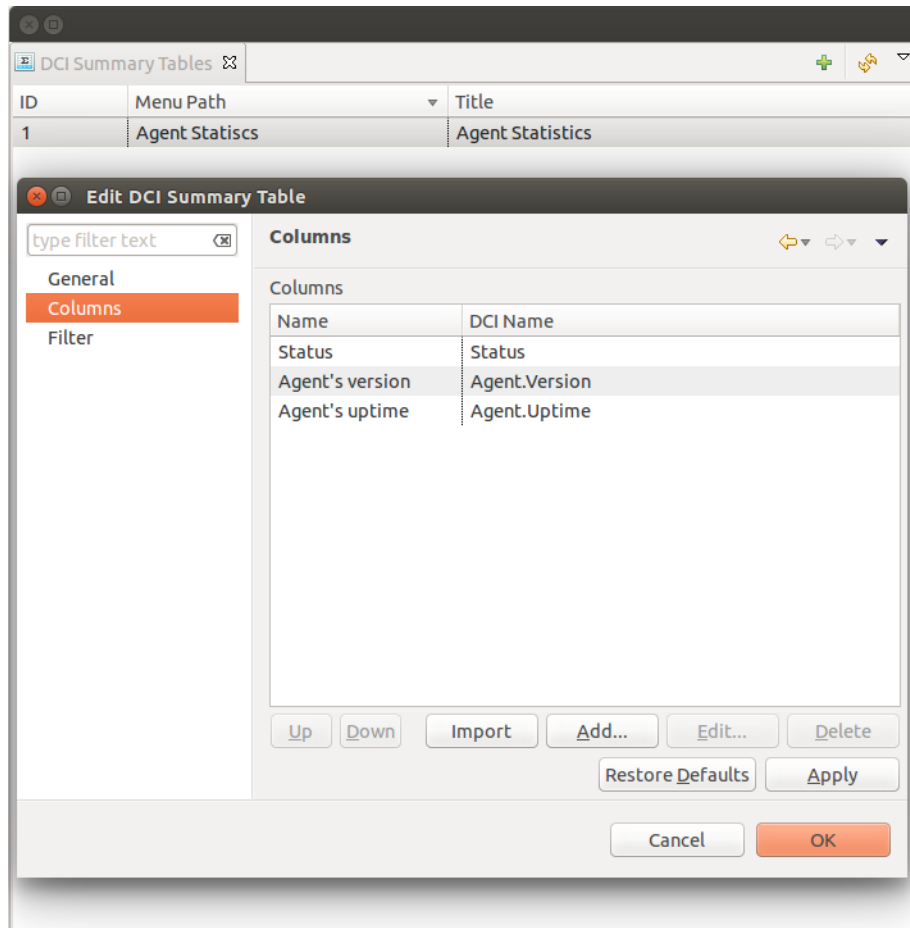
14.5 Summary table

It is possible to see DCI data as a table where each line is one node and each column is a DCI. It can be configured for each summary table which DCIs should be present on it.

Node	Status	Agent's version	Agent's uptime
aix.radensolutions.com	0	1.2.15	1230123
netxms.radensolutions.com	0	1.2.15	192440
static-5-0-20.radensolutions.com	2		
zev-VirtualBox	0	1.2.15	21124

14.5.1 Configuration

DCI summary table can be configured in Configuration -> Summary Table.



General:

- Menu path - path where this summary table can be found. You can use -> character pair to create subtree like "Linux->System information".
- Title - title of the summary table.

Columns:

- This is the list of DCIs that will be shown on the summary table. Name is the name of column and DCI Name is DCI parameter name.
 - Multivalued column is intended to present string DCIs that contain several values divided by specified separator. Each value is presented on a separate line in the column.
 - If **Use regular expression for parameter name matching** is enabled, a regular expression is specified in **DCI name** field. If several DCIs will be matched on a node, only one will be displayed.
- Import button allows to select a DCI from existing object.

Filter:

- Filter script is executed for each node to determine, if that node should be included in a summary table. Filter script is defined with help of *NXSL* scripting language.

14.5.2 Usage

After DCI summary table is configured it can be accessed in container object (Subnet, container...) context menu under "Summary tables".

GRAFANA INTEGRATION

NetXMS Grafana integration provides the possibility to display important data using the Grafana platform and the *NetXMS WebAPI*.

15.1 Integration with Grafana

The NetXMS Grafana datasource provides an alternative way of monitoring to that of the NetXMS Web and Desktop consoles or the Android app, by using the Grafana platform and the NetXMS WebAPI.

15.1.1 Requirements

The following prerequisites need to be installed first:

A running instance of the NetXMS Server. A running instance of the NetXMS WebAPI. A running instance Grafana (more information in <https://grafana.com/get>).

15.1.2 Installation

See <https://grafana.com/grafana/plugins/radensolutions-netxms-datasource/?tab=installation>

For installation from source:

1. Clone the NetXMS Grafana datasource GitHub repository from <https://github.com/netxms/grafana>.
2. Copy the files from the repository to `GRAFANA_HOME/data/plugins/datasources/netxms`
3. Restart your Grafana server.
4. Login to your Grafana web interface and add the NetXMS datasource in the Data Sources section.

15.1.3 Features

The datasource currently supports the following functionality:

- Visualization of configured data collection items for objects in graphs and tables.
- Listing of active alarms on a general or a per object basis

15.2 Configuration

Edit data source

Name	NetXMS data source	Default	<input checked="" type="checkbox"/>
Type	NetXMS		

NetXMS settings

API base URL	https://office.radensolutions.com/		
Login	grafana	Password	*****

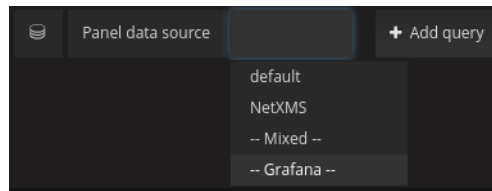
Save & Test Delete Cancel

The data source can be configured in the data source management section in the Grafana web ui. The required settings are the base URL of the NetXMS WebAPI, the username and the password of an account that exists on your NetXMS server. It is suggested to create a dedicated account to be used with Grafana.

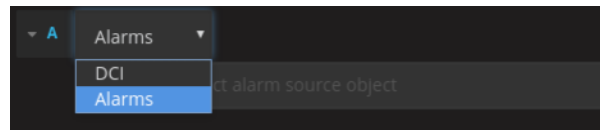
15.3 Alarm Browser

Alarm browser									
Severity	State	Source	Message	Count	Helpdesk ID	Ack/Resolved By	Created	Last Change	
MINOR	Outstanding	WINCOR	Low cash in 01	16.00			30.11.2016 19:33:09	07.04.2017 13:21:09	
MINOR	Outstanding	sw-4510g	Interface "GigabitEthernet1/0/4" changed state to DOWN (IP Addr: UNSPEC'd, ifindex: 4)	5.00			28.10.2016 09:41:03	14.02.2017 13:12:52	
MINOR	Outstanding	WINCOR	Threshold activated on table "Cash units information" row 2 (02)	1.00			13.02.2017 15:27:30	13.02.2017 15:27:30	
MINOR	Outstanding	sw-4510g	Interface "GigabitEthernet1/0/12" changed state to DOWN (IP Addr: UNSPEC'd, ifindex: 12)	5.00			28.10.2016 09:41:03	14.02.2017 13:12:52	
MINOR	Outstanding	sw-4510g	Interface "GigabitEthernet1/0/18" changed state to DOWN (IP Addr: UNSPEC'd, ifindex: 18)	5.00			28.10.2016 09:41:03	14.02.2017 13:12:52	
MINOR	Outstanding	sw-4510g	Interface "GigabitEthernet1/0/10" changed state to DOWN (IP Addr: UNSPEC'd, ifindex: 10)	5.00			28.10.2016 09:41:03	14.02.2017 13:12:52	
MINOR	Outstanding	sw-4510g	Interface "GigabitEthernet1/0/22" changed state to DOWN (IP Addr: UNSPEC'd, ifindex: 22)	5.00			28.10.2016 09:41:03	14.02.2017 13:12:52	
MINOR	Outstanding	sw-lab-1-office.radensolutions.com	Interface "Fa0/2" changed state to DOWN (IP Addr: UNSPEC'd, ifindex: 3)	3.00			19.01.2017 09:53:26	14.02.2017 13:12:46	
MINOR	Outstanding	demo-netxms	Script (ATM-SetCurrentCashForLogicalUnit) execution error: Error 14 in line 1: Function or operation argument is not an object	527.00			25.10.2016 18:44:20	25.10.2016 18:13:31	
MINOR	Outstanding	demo-netxms	Script (Template-Cash Common:236) execution error: Error 14 in line 1: Function or operation argument is not an object	726.00			13.12.2016 12:08:17	12.01.2017 14:58:47	
MINOR	Outstanding	WINCOR	Threshold activated on table "Cash units information" row 2 (02)	1.00			14.02.2017 15:15:54	14.02.2017 15:15:54	
MINOR	Outstanding	sw-4510g	Interface "GigabitEthernet1/0/13" changed state to DOWN (IP Addr: UNSPEC'd, ifindex: 13)	7.00			19.08.2016 16:13:18	14.02.2017 13:12:52	
MINOR	Outstanding	sw-4510g	Interface "GigabitEthernet1/0/14" changed state to DOWN (IP Addr: UNSPEC'd, ifindex: 14)	7.00			19.08.2016 16:13:18	14.02.2017 13:12:52	
MINOR	Outstanding	sw-4510g	Interface "GigabitEthernet1/0/11" changed state to DOWN (IP Addr: UNSPEC'd, ifindex: 11)	7.00			19.08.2016 16:13:18	14.02.2017 13:12:52	
MINOR	Outstanding	sw-4510g	Interface "GigabitEthernet1/0/17" changed state to DOWN (IP Addr: UNSPEC'd, ifindex: 17)	7.00			19.08.2016 16:13:18	14.02.2017 13:12:52	
MINOR	Outstanding	sw-4510g	Interface "GigabitEthernet1/0/19" changed state to DOWN (IP Addr: UNSPEC'd, ifindex: 19)	7.00			19.08.2016 16:13:18	14.02.2017 13:12:52	
MINOR	Outstanding	sw-4510g	Interface "GigabitEthernet1/0/15" changed state to DOWN (IP Addr: UNSPEC'd, ifindex: 15)	7.00			19.08.2016 16:13:18	14.02.2017 13:12:52	
MINOR	Outstanding	sw-4510g	Interface "GigabitEthernet1/0/16" changed state to DOWN (IP Addr: UNSPEC'd, ifindex: 16)	7.00			19.08.2016 16:13:18	14.02.2017 13:12:52	
MINOR	Outstanding	MGDebold529	Interface "tslap (ZAGADAF-AD3A-4D5A-8187-73A5267F6438)" changed state to DOWN (IP Addr: UNSPEC'd, ifindex: 18)	8.00			21.12.2016 10:56:50	21.12.2016 12:55:26	
MINOR	Outstanding	demo-netxms	Script (Template-NetXMS Agent: 6389) execution error: Error 14 in line 1: Function or operation argument is not an object	726.00			13.12.2016 12:08:17	12.01.2017 14:58:47	
MINOR	Outstanding	MGDebold529	Interface "tslap (8BA8D069-058A-4464-802C-69D5CA6C3A)" changed state to DOWN (IP Addr: UNSPEC'd, ifindex: 23)	7.00			21.12.2016 11:16:11	21.12.2016 12:55:26	
MINOR	Outstanding	HTC One	Device not reporting for more than 600 seconds (1493810509 seconds since last report)	2.61 K			11.01.2017 18:03:53	03.05.2017 14:23:29	
MINOR	Outstanding	MGDebold529	Low cash in 04	1.00			21.12.2016 11:54:16	21.12.2016 11:54:16	
MINOR	Outstanding	GNP NCB P77	Low cash in 01	1.00			03.01.2017 17:35:51	03.01.2017 17:35:51	
MINOR	Outstanding	MGDebold529	Low cash in 01	1.00			21.12.2016 11:54:16	21.12.2016 11:54:16	
MINOR	Outstanding	MGDebold529	Low cash in 02	1.00			21.12.2016 11:54:16	21.12.2016 11:54:16	
MINOR	Outstanding	demo-netxms	Script (Template-Server Performance:8440) execution error: Error 14 in line 1: Function or operation argument is not an object	726.00			13.12.2016 12:08:17	12.01.2017 14:58:47	
MINOR	Outstanding	MGDebold	Disk queue length is too high (2,612543)	2.00			01.03.2017 04:58:35	01.03.2017 08:58:36	

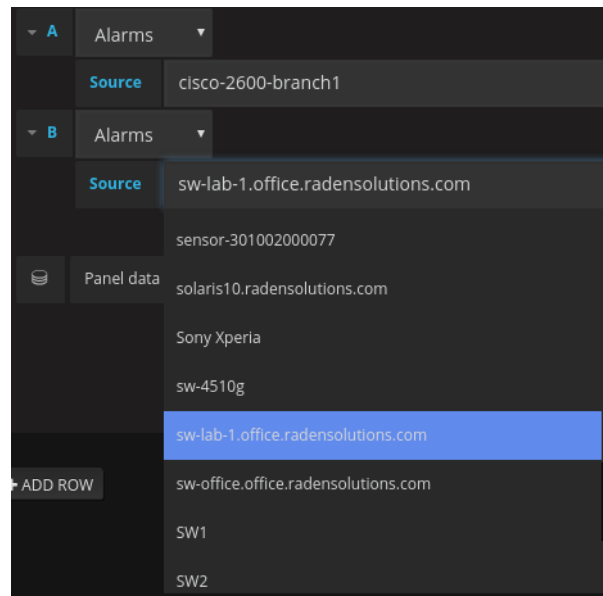
The data source provides the possibility to view currently active *Alarms* on all nodes or on a per node basis. To do this, you need to add a new Table Panel to your Grafana dashboard and then edit the Metrics section of the panel settings. If the NetXMS data source is set as the default data source, it should have been added to the panel automatically. If not, select the name of the installed NetXMS data source from the *Panel data source* list and press *Add query* to add the data source.



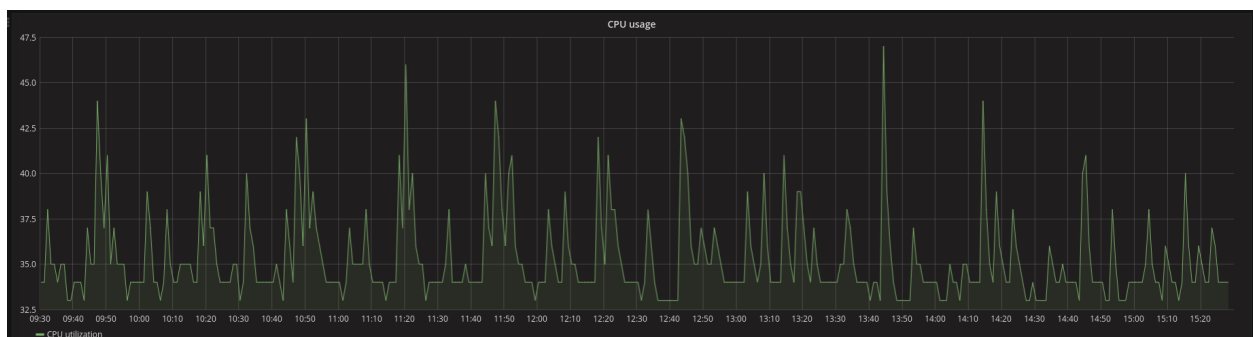
Once the data source is added to the panel, it is necessary to set the necessary type of data for the data source to provide, in this case - *Alarms*.



After the data type has been set, you should see the active alarms appear on the table panel. If you wish to view alarms from specific nodes only, you can add multiple data sources to your table panel and for each specify the node you wish to see the active alarms of.



15.4 Data Collection Items



The data source provides the possibility to visualize metrics collected from data collection items configured on nodes. This can be achieved by adding a Graph Panel to your Grafana dashboard, adding the NetXMS data source to it and selecting the *DCI* data type in the Metrics section of the graph panel settings. Once this is done, it is possible to select the *Target* node from the list of targets which will then provide a list of the configured DCIs for the particular node in the *DCI* section. By default, the legend of the data provided by the DCI will be the DCI description as configured on the server. It is also possible to set a legend of your choice by entering it in the *Legend* section.

▼ B	DCI ▼	
Target	sw-lab-1.office.radensolutions.com	
DCI	CPU utilization	
Legend	CPU Usage	

It is possible to view multiple DCIs on the same graph by adding multiple data sources to it.

OPERATING SYSTEM MONITORING

Most OS-related metrics (file system, CPU, network) are provided by “platform subagent”, which is loaded automatically by the agent on the startup.

List of available subagents:

- linux
- aix
- hpux
- winnt (all Windows flavors)
- sunos (Solaris)
- darwin (MacOS)
- freebsd
- netbsd
- openbsd

In this section we cover only most common metrics. *Detailed list* available below.

16.1 Example

In examples will be shown only DCI configuration with threshold. Generated event processing options can be found in *Event processing* chapter.

16.1.1 Process monitoring

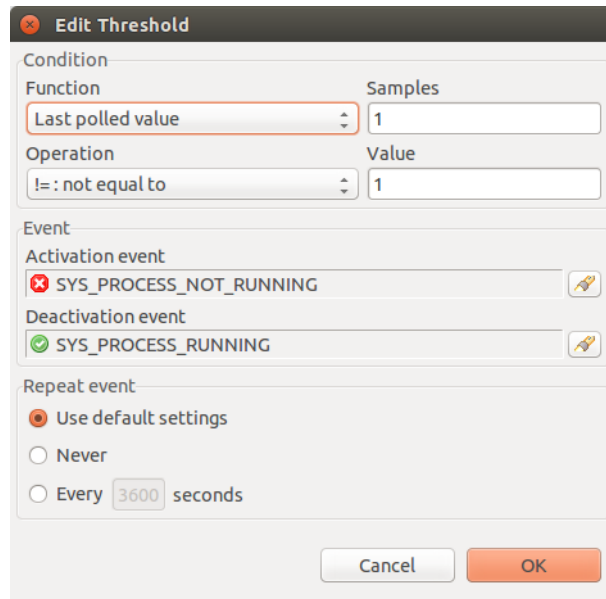
In this example monitoring of running “mysqld” process will be configured and one threshold will be added: when process count is less then 1 (process is not running).

Create DCI for Process.Count(*) metric to monitor “mysqld” process count.

Create threshold. It will be triggered when process count is not equal to 1(process is not running). As prerequisite it was created 2 events.

100003	SYS_PROCESS_NOT_RUNNING	Critical	L	Process %6 is not running.	Generated when threshold value reached for specific data collection item. Parameters: 1) Parameter name 2) Item description 3) Threshold value 4) Actual value 5) Data collection item ID 6) Instance 7) Repeat flag
100004	SYS_PROCESS_RUNNING	Normal	L	Process %6 is running.	Generated when threshold value reached for specific data collection item. Parameters: 1) Parameter name 2) Item description 3) Threshold value 4) Actual value 5) Data collection item ID 6) Instance 7) Repeat flag

Fig. 1: Events



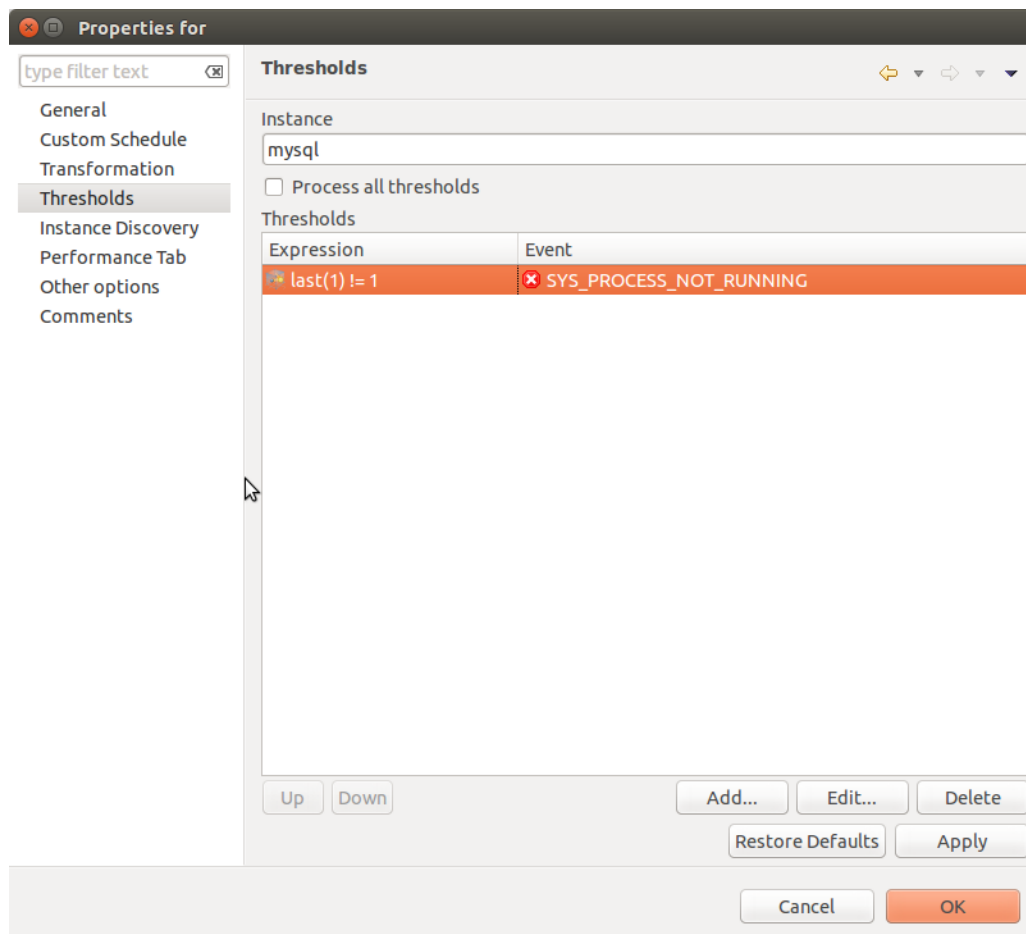
The **Edit Threshold** dialog box is used to configure a threshold condition. It contains the following sections:

- Condition**
 - Function:** Last polled value (dropdown)
 - Samples:** 1 (text input)
- Operation**
 - Operation:** != : not equal to (dropdown)
 - Value:** 1 (text input)
- Event**
 - Activation event:** ☒ SYS_PROCESS_NOT_RUNNING (with edit icon)
 - Deactivation event:** ☒ SYS_PROCESS_RUNNING (with edit icon)
- Repeat event**
 - ☒ Use default settings
 - ☐ Never
 - ☐ Every 3600 seconds

Buttons: Cancel, OK

Fig. 2: Threshold 1

As in message of error is used Instance parameter, it should be set in *Threshold* window.



The **Properties for - Thresholds** window displays the configuration for a specific instance. It includes a sidebar with navigation options and a main area for threshold management.

Sidebar:

- General
- Custom Schedule
- Transformation
- Thresholds** (selected)
- Instance Discovery
- Performance Tab
- Other options
- Comments

Main Area:

- Instance:** mysql
- ☐ Process all thresholds
- Thresholds Table:**

Expression	Event
last(1) != 1	<input checked="" type="checkbox"/> SYS_PROCESS_NOT_RUNNING

Buttons: Up, Down, Add..., Edit..., Delete, Restore Defaults, Apply, Cancel, OK

16.1.2 Disk free space monitoring

In this example monitoring of free space in percents for / disk will be configured and two thresholds will be added: when disk space less then 15% and less then 7%.

Create DCI for FileSystem.FreePerc(*) metric to monitor space on /.

The screenshot shows the 'Properties for' dialog box with the 'General' tab selected. The configuration is as follows:

- Description:** Percentage of free space on file system /
- Data:**
 - Parameter:** FileSystem.FreePerc(/)
 - Origin:** NetXMS Agent
 - Data Type:** Floating Point Number
 - ☐ Interpret SNMP octet string raw value as
 - ☐ Use custom SNMP port:
 - Sample count for average value calculation (0 to disable):** 0
 - Proxy node:** <none>
- Polling:**
 - Polling mode:** Fixed intervals
 - Polling interval (seconds):** 60
- Status:**
 - ☒ Active
 - ☐ Disabled
 - ☐ Not supported
- Storage:**
 - Retention time (days):** 30
 - ☐ Do not save collected data to database

Buttons at the bottom: Restore Defaults, Apply, Cancel, and OK.

Create 2 thresholds. One will be triggered when free space is less than 15% and other one when free space is less than 7%. Before threshold creation was created 3 events:

100000	SYS_DISK_LOW	Warning	L	Disk %6 has less then %3 disk space available. Current value is %4.	Generated when threshold value reached for specific data collection item. Parameters: 1) Parameter name 2) Item description 3) Threshold value 4) Actual value 5) Data collection item ID 6) Instance 7) Repeat flag
100001	SYS_DISK_NORMAL	Normal	L	Disk space for %6 back to normal.	Generated when threshold value reached for specific data collection item. Parameters: 1) Parameter name 2) Item description 3) Threshold value 4) Actual value 5) Data collection item ID 6) Instance 7) Repeat flag
100002	SYS_DISK_FULL	Critical	L	Disk %6 has less then %3 disk space available. Current value is %4.	Generated when threshold value reached for specific data collection item. Parameters: 1) Parameter name 2) Item description 3) Threshold value 4) Actual value 5) Data collection item ID 6) Instance 7) Repeat flag

Fig. 3: Events

Edit Threshold

Condition

Function

Last polled value

Samples

1

Operation

< : less then

Value

15

Event

Activation event

SYS_DISK_LOW

Deactivation event

SYS_DISK_NORMAL

Repeat event

☒ Use default settings

☐ Never

☐ Every 3600 seconds

Cancel

OK

Fig. 4: Threshold 1

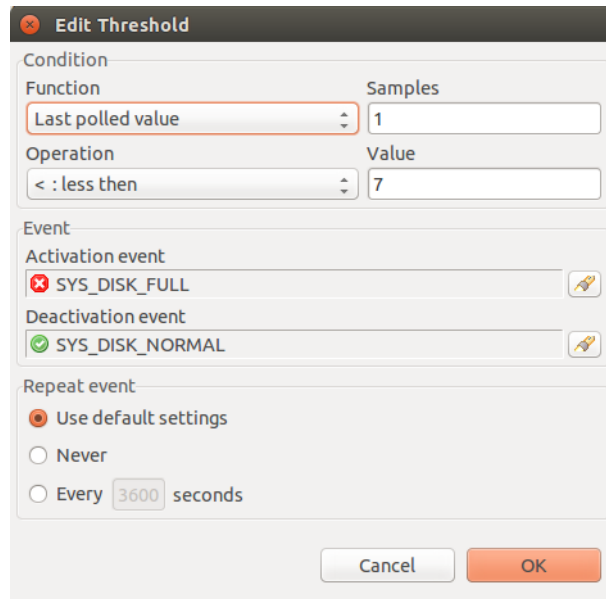


Fig. 5: Threshold 2

As in message of error is used Instance parameter, it should be set in *Threshold* window.

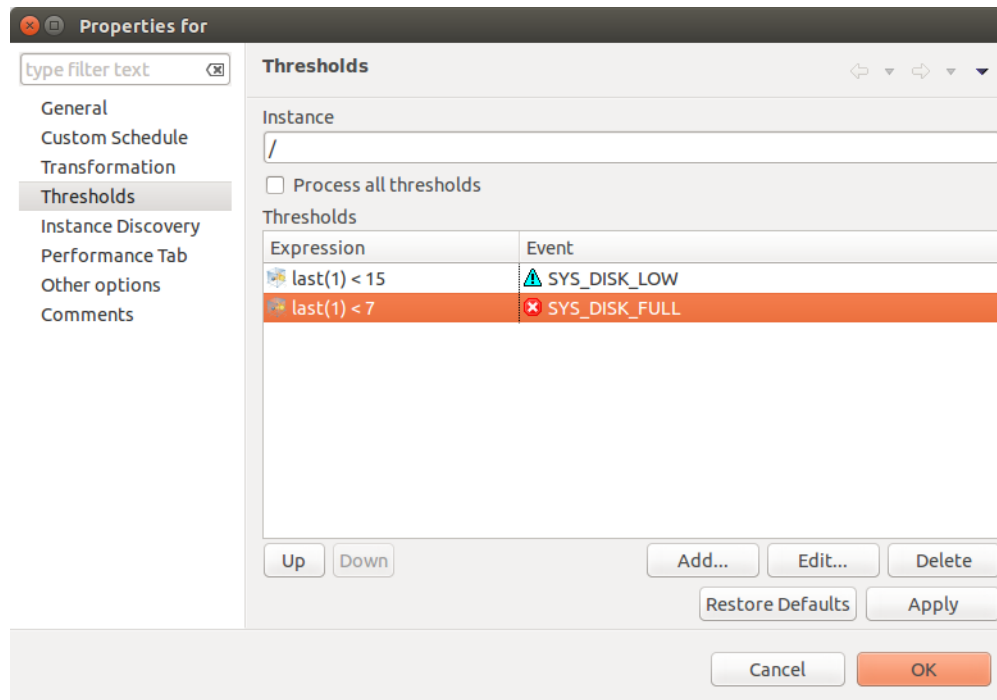


Fig. 6: Both

16.1.3 CPU usage

This example will show how to configure monitoring of CPU usage and create event when CPU usage is more than 90% for more than 5 minutes.

Create DCI for System.CPU.LoadAvg metric.

The screenshot shows the 'Properties for' dialog box with the 'General' tab selected. The configuration is as follows:

- Description:** Average CPU load for last minute
- Data:**
 - Parameter:** System.CPU.LoadAvg
 - Origin:** NetXMS Agent
 - Data Type:** Floating Point Number
 - ☐ Interpret SNMP octet string raw value as
 - ☐ Use custom SNMP port:
 - Sample count for average value calculation (0 to disable):** 0
 - Proxy node:** <none>
- Polling:**
 - Polling mode:** Fixed intervals
 - Polling interval (seconds):** 60
- Status:**
 - ☒ Active
 - ☐ Disabled
 - ☐ Not supported
- Storage:**
 - Retention time (days):** 30
 - ☐ Do not save collected data to database

Buttons at the bottom: Restore Defaults, Apply, Cancel, and OK.

Create threshold that will create event in case if last 5 values are more than 90 (last 5 minutes CPU usage is more than 90%).

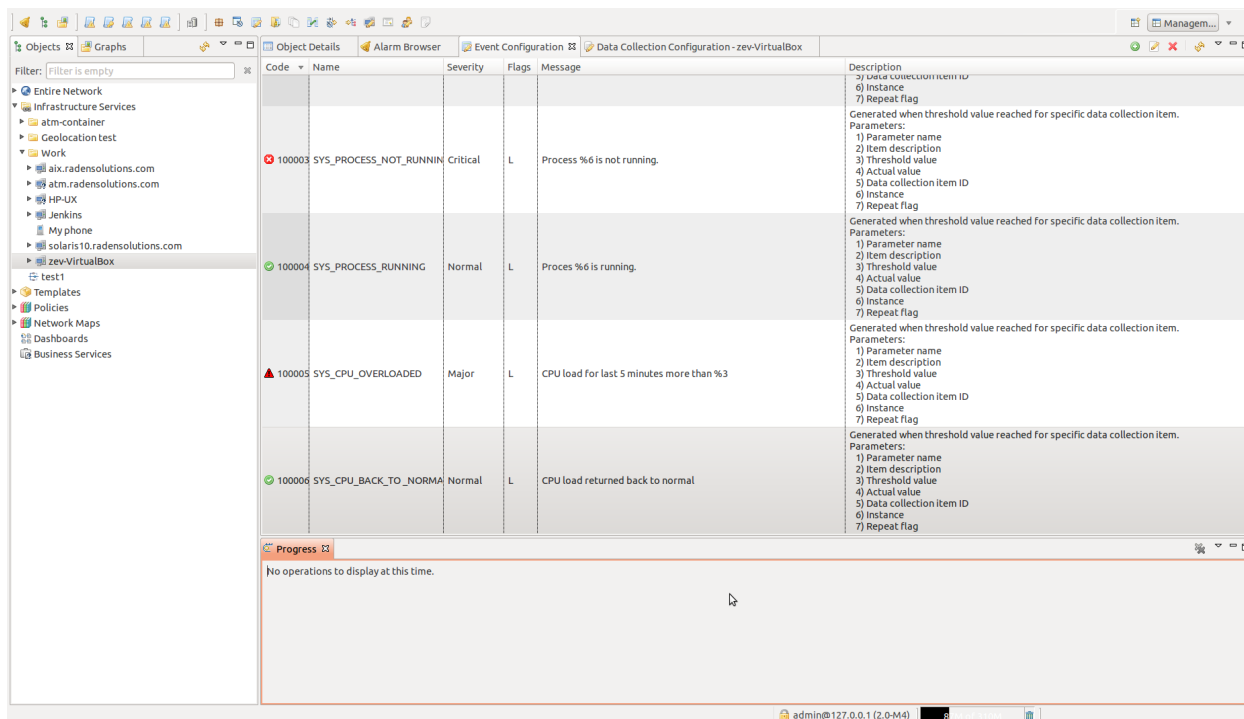


Fig. 7: Events

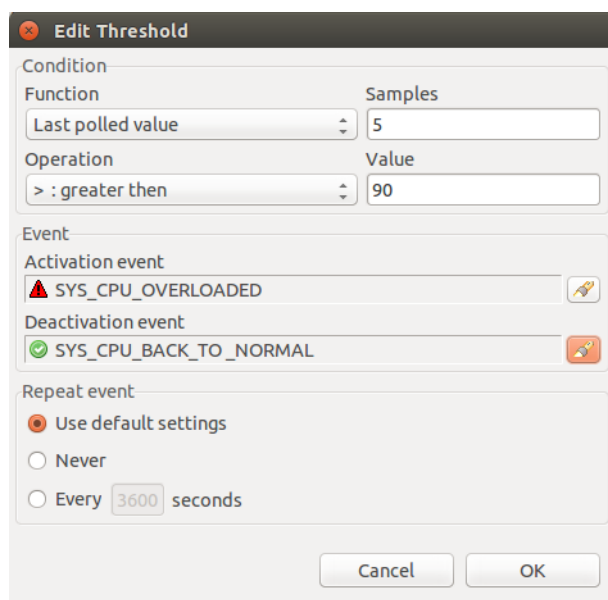


Fig. 8: Threshold

16.1.4 WMI

Windows Management Instrumentation subagent provides interface to Windows Driver Model and thus enables information and notification gathering and further manipulation for monitoring purpose.

Configuration example:

```
MasterServers = netxms.demo
SubAgent=wmi.nsm
```

Provides access to WMI data via WMI class queries. In below example, DCI *New table ...* is created with NetXMS Agent as Origin and WMI query as Metric

Properties for WMI.Query(root\CIMV2,\"SELECT * FROM Win32_Pr...

General

Metric to collect

Origin: NetXMS Agent Source node override: None

Metric: WMI.Query(root\CIMV2,\"SELECT * FROM Win32_Process\")

Display name: Generic WMI query

Collection schedule

☒ Server default interval (60 seconds)

☐ Custom interval

☐ Advanced schedule

History retention period

☒ Server default (30 days)

☐ Custom

☐ Do not save to the database

Restore Defaults Apply

Apply and Close Cancel

Following parameters are available for this subagent:

Parameter	Description
ACPI.ThermalZone.CurrentTemp	Current temperature in ACPI thermal zone.
ACPI.ThermalZone.CurrentTemp(*)	Current temperature in ACPI thermal zone {instance}. Argument is thermal zone name, one of those returned by list ACPI.ThermalZones (actually InstanceName from WMI class MSAcpi_ThermalZoneTemperature).
Hardware.NetworkAdapter.Availability(*)	Availability. Argument is physical network adapter index, one of those returned by list Hardware.NetworkAdapters or column "INDEX" in table Hardware.NetworkAdapters.

continues on next page

Table 1 – continued from previous page

Parameter	Description
Hardware.NetworkAdapter.Description(*)	Description. Argument is physical network adapter index, one of those returned by list Hardware.NetworkAdapters or column “INDEX” in table Hardware.NetworkAdapters.
Hardware.NetworkAdapter.InterfaceIndex(*)	InterfaceIndex. Argument is physical network adapter index, one of those returned by list Hardware.NetworkAdapters or column “INDEX” in table Hardware.NetworkAdapters.
Hardware.NetworkAdapter.MACAddress(*)	MACAddress. Argument is physical network adapter index, one of those returned by list Hardware.NetworkAdapters or column “INDEX” in table Hardware.NetworkAdapters.
Hardware.NetworkAdapter.Manufacturer(*)	Manufacturer. Argument is physical network adapter index, one of those returned by list Hardware.NetworkAdapters or column “INDEX” in table Hardware.NetworkAdapters.
Hardware.NetworkAdapter.Product(*)	ProductName. Argument is physical network adapter index, one of those returned by list Hardware.NetworkAdapters or column “INDEX” in table Hardware.NetworkAdapters.
Hardware.NetworkAdapter.Speed(*)	Speed. Argument is physical network adapter index, one of those returned by list Hardware.NetworkAdapters or column “INDEX” in table Hardware.NetworkAdapters.
Hardware.NetworkAdapter.Type(*)	AdapterType. Argument is physical network adapter index, one of those returned by list Hardware.NetworkAdapters or column “INDEX” in table Hardware.NetworkAdapters.
System.AntiSpywareProduct.Active	Anti-spyware product active.
System.AntiSpywareProduct.DisplayName	Anti-spyware product display name.
System.AntiSpywareProduct.UpToDate	Anti-spyware product up to date.
System.AntiVirusProduct.Active	Anti-virus product active.
System.AntiVirusProduct.DisplayName	Anti-virus product display name.
System.AntiVirusProduct.UpToDate	Anti-virus product up to date.
System.FirewallProduct.Active	Firewall active.
System.FirewallProduct.DisplayName	Firewall product display name.
System.FirewallProduct.UpToDate	Firewall product up to date.
WMI.Query(*)	Generic WMI query. Arguments are namespace, query, property. For example: WMI.Query(rootcimv2, SELECT * FROM Win32_Process WHERE ProcessId=252, Caption)

Following lists are available for this subagent:

- ACPI.ThermalZones
- Hardware.NetworkAdapters
- WMI.Classes(*), argument is WMI namespace (for example rootcimv2). List of available namespaces can also be retrieved using agent list WMI.NameSpaces (output will not contain “root”)
- WMI.NameSpaces
- WMI.Query(*), arguments are namespace, query, property (for example: WMI.Query(rootcimv2, SELECT * FROM Win32_Process, Caption) - will return all process names)

Below list of supported tables for this subagent:

- Hardware.NetworkAdapters
- WMI.Query(*), arguments are namespace and query and it will return query output with column for each attribute (for example: WMI.Query(rootcimv2, SELECT * FROM Win32_Process) - all processes in the system)

Some of the most commonly used WMI classes are listed below:

Static

- Computer System - Win32_ComputerSystem
- Operating System - Win32_OperatingSystem
- Processor Info - Win32_Processor
- HDD - Win32_DiskDrive
- Disk Partitions - Win32_DiskPartition
- Logical Disks - Win32_LogicalDisk
- Logical Disk to Partition - Win32_LogicalDiskToPartition
- Memory - Win32_PhysicalMemory, Win32_PhysicalMemoryArray
- Network - Win32_NetworkAdapter , Win32_NetworkAdapterConfiguration

Performance Counters

- Processor Utilization - Win32_PerfRawData_PerfOS_Processor
- Memory Utilization - Win32_PerfRawData_PerfOS_Memory
- Network Utilization - Win32_PerfRawData_Tcpip_NetworkInterface

The result is a table with appropriate WMI data.

Filter is empty	
Caption	CommandLine
System Idle Process	
System	
Secure System	
Registry	
smss.exe	
csrss.exe	
wininit.exe	
csrss.exe	
services.exe	
lsass.exe	
svchost.exe	C:\WINDOWS\system32\svchost.exe -k DcomLaunch -p
fontdrvhost.exe	"fontdrvhost.exe"
WUDFHost.exe	"C:\Windows\System32\WUDFHost.exe" -HostGUID:{193a1820-d9ac-4997-8c55-be817523f6aa} -IoEventPortName:\UMDFCommunica
svchost.exe	C:\WINDOWS\system32\svchost.exe -k RPCSS -p
svchost.exe	C:\WINDOWS\system32\svchost.exe -k DcomLaunch -p -s LSM
winlogon.exe	winlogon.exe
WUDFHost.exe	"C:\Windows\System32\WUDFHost.exe" -HostGUID:{193a1820-d9ac-4997-8c55-be817523f6aa} -IoEventPortName:\UMDFCommunica
fontdrvhost.exe	"fontdrvhost.exe"
svchost.exe	C:\WINDOWS\System32\svchost.exe -k netsvcs -p -s BDESVC
svchost.exe	C:\WINDOWS\system32\svchost.exe -k LocalSystemNetworkRestricted -p -s HvHost
svchost.exe	C:\WINDOWS\system32\svchost.exe -k LocalServiceNetworkRestricted -p -s lmhosts
svchost.exe	C:\WINDOWS\system32\svchost.exe -k LocalServiceNetworkRestricted -s BTAGService
svchost.exe	C:\WINDOWS\system32\svchost.exe -k osprivity -p -s camsvc
svchost.exe	C:\WINDOWS\system32\svchost.exe -k LocalService -n -s hthserv

FILE SYSTEM MONITORING

NetXMS has two options to monitor files: one is to use build in agent file monitoring functionality, that is described in next chapter and another is to create *DCI* that will collect file information and create your own thresholds for collected data. Second approach is describe in *DCI Metrics for file system monitoring* chapter.

17.1 File Monitoring

NetXMS provides a feature to monitor hash value of a file, last modification time and permissions changes. One file is added to monitoring any changes to those file parameters will be detected and reported to the server via events. Those events are `SYS_AGENT_FILE_ADDED`, `SYS_AGENT_FILE_CHANGED` and `SYS_AGENT_FILE_DELETED` for files creations, alterations and deletions correspondingly.

Specify the path to a file for monitoring by adding `[FileMonitor]` section to *Agent configuration files*. If the path to a directory is specified, then all files in that directory and it's subdirectories will be monitored.

Configuration parameters:

1. `Path` - The path to monitored file. This parameter should be specified once for each file/directory.
2. `Interval` - Check interval in seconds. This parameter should not be specified multiple times. This parameter is optional and will be set to 6 hours by default.

```
# Example
[FileMonitor]
Interval=10800
Path=/home/user/file_name
Path=/home/user/directory
```

17.2 DCI Metrics for file system monitoring

17.2.1 'FileSystem.*' Metrics

Metrics with prefix 'FileSystem' are used to monitor file system. They provide information about free and user space, inode information, etc.

Full list of available metrics can be found in *FileSystem.** section.

17.2.2 'File.*' Metrics

Metrics with prefix 'File' are used to monitor files. They provide information about file size, count, modification time, etc.

Full list of available metrics can be found in *File.** section.

17.2.3 Examples

In examples will be shown only DCI configuration with threshold. Generated event processing options can be found in *Event processing* chapter.

Example 1

In this example will be shown how to check that specific folder exceed specified size.

Create DCI for File.Size(*) metric to monitor folder size. Required parameters: /path,*,1.

The screenshot shows the 'Properties for' dialog box in NetXMS. The 'General' tab is selected in the left sidebar. The main panel contains the following fields and options:

- Description:** A text field containing 'Size of folder /path'.
- Data:**
 - Parameter:** A text field containing 'File.Size(/path,*,1)' and a 'Select...' button.
 - Origin:** A dropdown menu set to 'NetXMS Agent'.
 - Data Type:** A dropdown menu set to 'Unsigned Integer 64 bit'.
 - ☐ Interpret SNMP octet string raw value as
 - ☐ Use custom SNMP port:
 - Sample count for average value calculation (0 to disable):** A dropdown menu set to '0'.
 - Proxy node:** A text field set to '<none>' with edit and delete icons.
- Polling:**
 - Polling mode:** A dropdown menu set to 'Fixed intervals'.
 - Polling interval (seconds):** A text field set to '60'.
 - Status:** Three radio buttons: 'Active' (selected), 'Disabled', and 'Not supported'.
- Storage:**
 - Retention time (days):** A text field set to '30'.
 - ☐ Do not save collected data to database

At the bottom of the dialog are buttons for 'Restore Defaults', 'Apply', 'Cancel', and 'OK'.

In threshold it should be checked that last value is less than 2 GB. That mean that returned value should be less than 2 000 000 000 bytes.

Edit Threshold

Condition

Function: Last polled value Samples: 1

Operation: > : greater then Value: 2000000000

Event

Activation event: SYS_THRESHOLD_REACHED

Deactivation event: SYS_THRESHOLD_REARMED

Repeat event

☒ Use default settings

☐ Never

☐ Every 3600 seconds

Cancel OK

Fig. 1: Threshold

Example 2

In this example will be configured monitoring that in exact folder exist files that was modified less then half an hour ago.

Create DCI for File.Count(*) metric to monitor file count in folder /path, that match any pattern, folder should be checked recursively, file match any size, files are created less than 30 minutes ago. This conditions will be given to metric as this parameters: path,*,1,0,-1800.

type filter text

General

Custom Schedule

Transformation

Thresholds

Instance Discovery

Performance Tab

Other options

Comments

Properties for

General

Description

Number of files that were created less than 30 min before now in /path catalog

Data

Parameter

File.Count(/path,*,1,0,-1800)

Select...

Origin

NetXMS Agent

Data Type

Unsigned Integer

☐ Interpret SNMP octet string raw value as
☐ Use custom SNMP port:

None

1

Sample count for average value calculation (0 to disable)

0

Proxy node

<none>

Polling

Polling mode

Fixed intervals

Polling interval (seconds)

60

Status

☒ Active
☐ Disabled
☐ Not supported

Storage

Retention time (days)

30

☐ Do not save collected data to database

Restore Defaults

Apply

Cancel

OK

In threshold it should be checked that at least one file meeting conditions exists. That mean that file count should be more than 1. Prerequisite is to create 2 events.

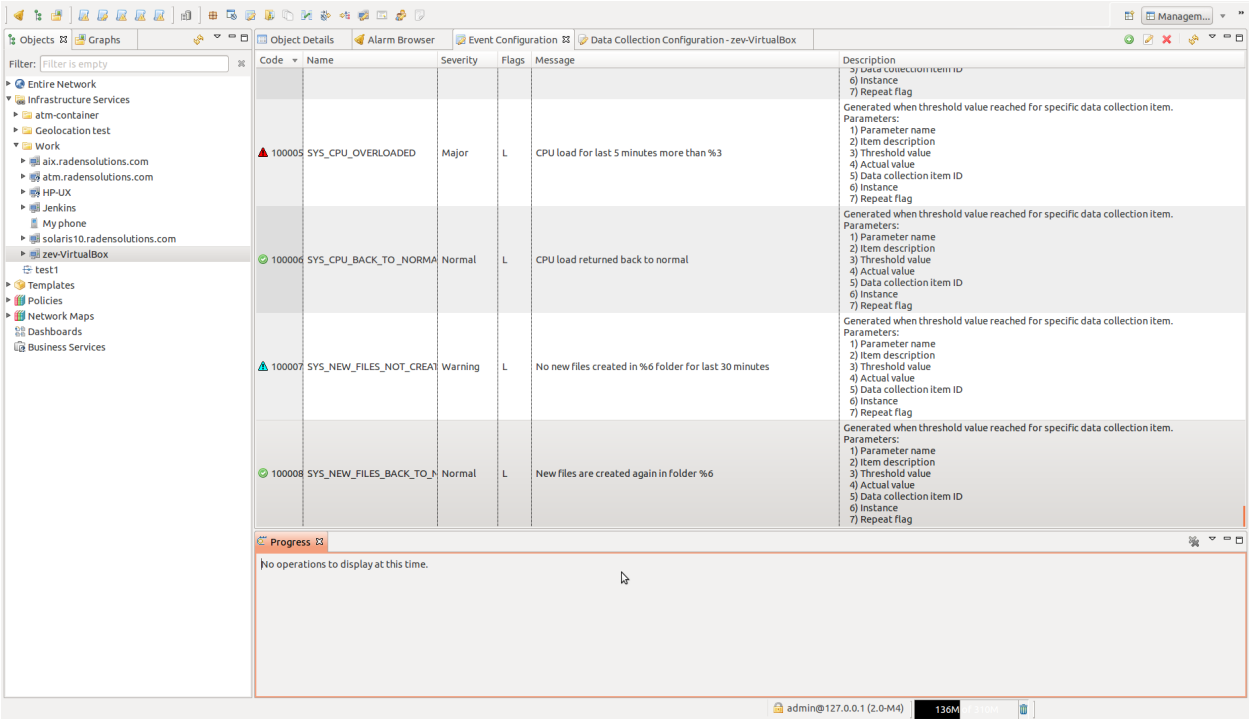


Fig. 2: Events

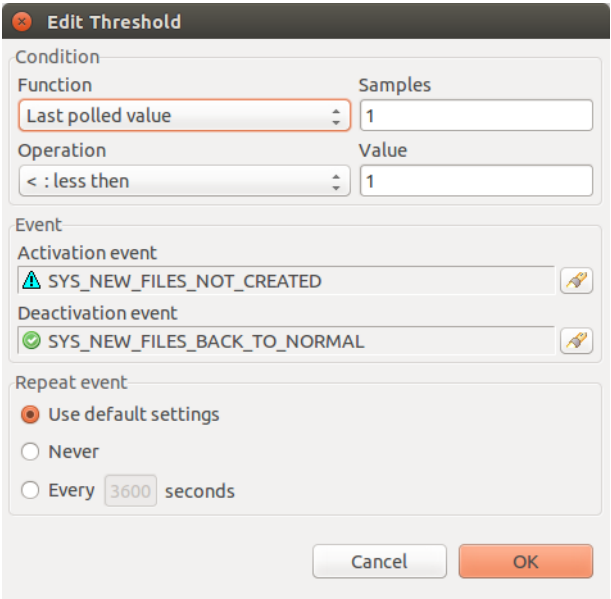
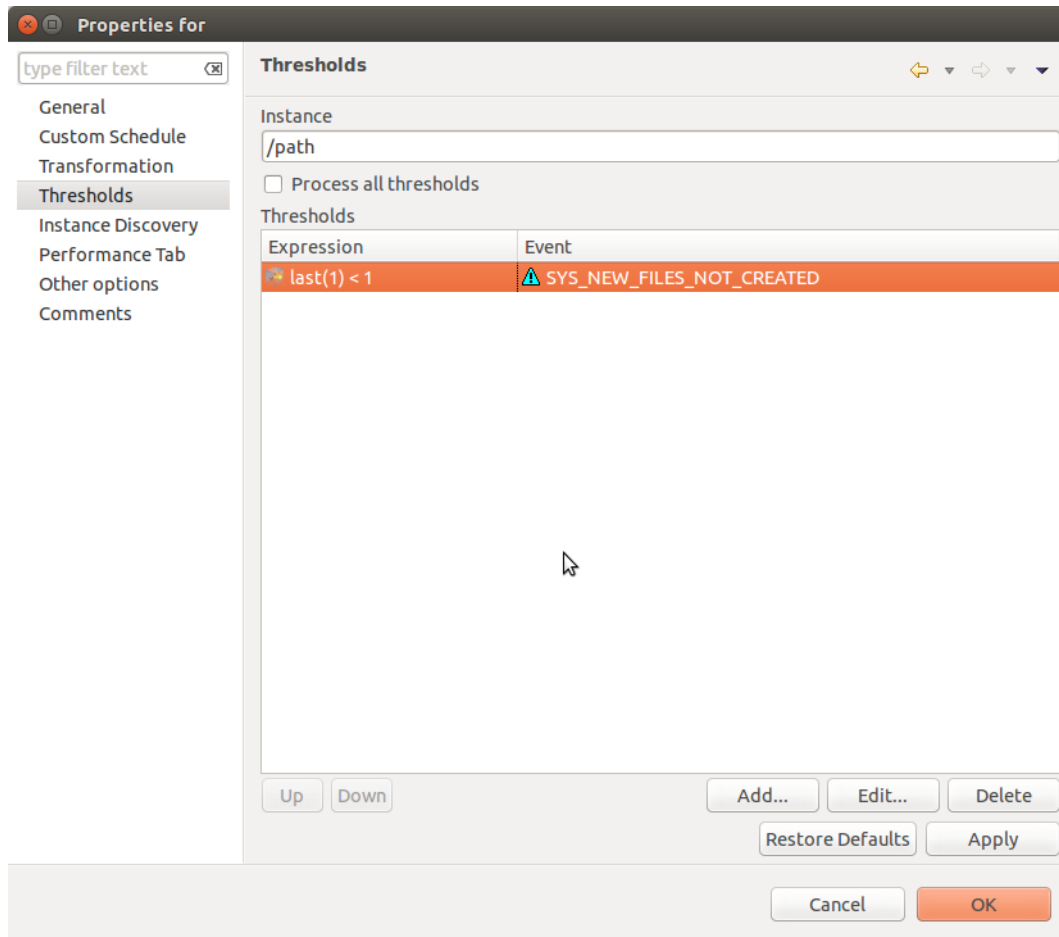


Fig. 3: Threshold

As in message of error is used Instance parameter, it should be set in *Threshold* window.



LOG MONITORING

With NetXMS you can monitor changes in text log files, Windows Event Log, and built-in syslog server. All log monitoring done by agents, except for built-in syslog server. In general, most common log processing goes as following:

1. When new line added to log file, it is passed to appropriate log parser
2. If line matched one of the patterns, an event associated with this pattern is sent to NetXMS server.
3. Server receives event and passes it to event processing policy as usual, with event source set to node from which event was received.

For text log files, agent keeps status information about monitored files in memory only. This means that if the agent was stopped for a period of time, lines that were added to log file during that time will not be parsed.

For Windows Event Log, agent can keep status information in Windows registry. This function should be explicitly enabled by setting `ProcessOfflineEvents = true` in `LogWatch` section. On agent start records that were added while the agent was stopped will be parsed.

Log parser also provides some additional statistic information through [Metrics](#). More information can be found in [Log parser metrics](#) chapter.

18.1 Agent Configuration for Log Monitoring

To be able to monitor logs with NetXMS agent, you should load `LOGWATCH` subagent. There are two options to define parser configuration:

1. Create log parser rule XML files on the monitored system and define them in `LOGWATCH` part of agent configuration.
2. Create log parser agent policy on a template and apply that template to all required nodes. This provides graphical editor that allows to specify monitored files, conditions and events. Graphical editor automatically generates log parser rule XML file that is being uploaded to agents. More information about [Agent Policies](#)

Example of agent configuration file:

```
SubAgent = logwatch.nsm

# Below is log parsers definitions
[LOGWATCH]
Parser = C:\log_monitoring_definitions\parser1.xml
Parser = C:\log_monitoring_definitions\parser2.xml
```

18.2 Syslog Monitoring

NetXMS has built-in syslog server, which can be used to receive logs from network devices and servers. It is also possible to parse incoming syslog messages in a way similar to Windows Event Log monitoring. To parse syslog messages, LOGWATCH subagent is not required - parsing is done by the server itself. You only need to define monitoring rules in *Configuration ▶ Syslog Parser*

18.3 Parser Definition File

Parser definition file is an XML document with the following structure:

```
<parser>
  <file>file name</file>
  <!-- more <file> tags can follow -->
  <macros>
    <macro name="name">macro body</macro>
    <!-- more <macro> tags can follow -->
  </macros>
  <rules>
    <rule>
      <match>regexp</match>
      <id>event id</id>
      <level>severity level</level>
      <source>event source</source>
      <event>event</event>
      <context>context</context>
    </rule>
    <!-- more <rule> tags can follow -->
  </rules>
</parser>
```

Note

Entire <macros> section can be omitted. Empty <rule> tag will match any line (like <rule> <match>.*</match> </rule>).

18.4 Global Parser Options

In the <parser> tag you can specify the following options:

Option	Description	Default value
processAll	If this option set to 1, parser will always pass log record through all rules. If this option set to 0, processing will stop after first match.	0
name	Parser name that is used in statistic information <i>Metrics</i> . See <i>Log parser metrics</i> for more information.	empty

18.5 <file> Tag

In the <file> tag you should specify full path of log file to apply this parser to. To specify Windows Event Log, prepend it's name with asterisk (*), for example *System. Multiple <file> tags can be used - in this case same rules will be applied to all files.

In the <file> tag it's possible to use wildcards. Wildcards can be used in file name, not in directory names in the path. Two wildcard characters are supported: * - represents zero, one or multiple characters. ? - represents any single character.

In file and folder names the following macros can be used:

- Environment variables as \${ENV_VAR_NAME}
- `strftime(3C)` macros (e.g. C:\Windows\system32\dhcp\DhcpSrvLog-%a)
- Text inside ` braces will be executed as a command and first line of output will be taken

Option	Description	Default value
encoding	It is possible to specify the encoding of the log file by adding the <code>encoding</code> attribute. File encodings that can be defined: <ul style="list-style-type: none"> • ACP • UTF-8 • UCS-2 • UCS-2LE • UCS-2BE • UCS-4 • UCS-4LE • UCS-4BE When using UCS-2 or UCS-4 values, the endianness of the system will be detected automatically.	By default, the parser will attempt to detect the encoding by scanning the file's BOM.
preallocated	Should be set when log file is preallocated (filled with zeros) before logs get written into it.	0
snapshot	Create VSS snapshot and uses snapshot file for parsing. Can be used when log is opened by other application as exclusive open. Windows only. Can highly increase CPU usage.	0
keepOpen	Defines if the file is kept open or reopened on each parsing iteration.	1
ignore-ModificationTime	Ignores modification time of log file	0
rescan	When file modification is detected, parse the file from it's beginning. The file is also parsed on agent startup and when log parsing policy is reapplied.	0
followSym-links	Follow symlinks.	0
removeEscapeSequences	Remove ANSI escape sequences when reading file.	0

18.6 Macros

In the `<macros>` section you can define macros for use in matching rules. For example, it can be useful to define macro for a timestamp preceding each log record and use it in matching rules instead of actual regular expression. You can define as many macros as you wish, each within its own `<macro>` tag. Each macro should have unique name, defined in `name` attribute, and can be used in matching rules in form `@{name}`.

Example: you need to parse log file where each line starts with timestamp in format `dd/mm/yy HH:MM:SS`. You can define the following macro:

```
<macros>
  <macro name="timestamp">dd/mm/yy HH:MM:SS</macro>
</macros>
<rules>
  <rule>
    <match>@{timestamp}.*([A-Za-z]+) failed.*</match>
    <event>12345</event>
  </rule>
  <rule>
    <match>@{timestamp}.*error.*</match>
    <event>45678</event>
  </rule>
</rules>
```

Please note that `<macros>` section always should be located before `<rules>` section in parser definition file.

18.7 Matching rules

In the `<rules>` section you define matching rules for log records.

18.7.1 <rule> Tag

Each rule is placed inside its own `<rule>` tag. Each rule can have additional options:

Option	Description	Default value
break	If this option set to 1 and current line match to regular expression in the rule, parser will stop processing of current line, even if global parser option <code>processAll</code> was set to 1. If this option set to 0 (which is default), processing will stop according to <code>processAll</code> option settings.	0
context	Name of the context this rule belongs to. If this option is set, rule will be processed only if given context was already activated with <code><context></code> tag in one of the rules processed earlier (it can be either same line or one of the previous lines).	<i>empty</i>
name	Name of rule	<i>empty</i>

Inside the `<rule>` section there are the following additional tags: `<match>`, `<description>`, `<event>`, and `<context>`. Only `<match>` section is mandatory - it specifies regular expression against which log record should be matched. All other tags are optional and define parser behavior if a record matches the regular expression.

18.7.2 <match> Tag

Tag <match> contains a PCRE compliant regular expression that is used to match log records. Parts enclosed in parenthesis are extracted from log record and passed as arguments of generated event. You can use macros defined in [Macros](#) section. Also, it is possible to define inverted match rules (rules when log record considered matching if it does not match regular expression). Inverted match can be set by setting attribute `invert` to 1. Other possible option that can be configured is number of times that expression should be matched to generate event.

Some examples:

```
<match>^Error: (.*)</match>
```

This regular expression will match any line starting with word `Error:`, and everything after this word will be extracted from the log record for use with an event.

```
<match repeatCount="3" repeatInterval="120" reset="false">[0-9]{3}</match>
```

This regular expression will match any line containing at least 3 consecutive digits. And event will be generated only if this regular expression will be matched 3 or more times in 2 minutes(120 seconds). Matched count won't be reset once mark is reached, so if expression is matched more than 3 times in 2 minutes, event will be generated more than one time.

```
<match invert="1">abc</match>
```

This regular expression will match any line not containing character sequence `abc`.

Possible attributes for tag <match>:

Option	Description	Default value
<code>invert</code>	If this option set to <code>true</code> , it will be matched any line that does not contain matching expression.	<code>false</code>
<code>repeatCount</code>	The number of times expression should be matched within specified time interval to generate event. Actual count is passed to generated event as parameter. Setting this option to 0 disables this functionality, event will be generated immediately on expression match.	0
<code>repeatInterval</code>	The time interval during which the expression should be matched specified number of times.	1
<code>reset</code>	If this option set to <code>true</code> , the count will be reset on expression match. In order to generate next event, <code>repeatCount</code> number of matches should be accumulated again within <code>repeatInterval</code> time.	<code>true</code>

18.7.3 <id> Tag

Tag <id> can be used to filter records from Windows Event Log by event ID. You can specify either single event ID or ID range (by using two numbers separated with minus sign). For example:

```
<id>7</id>
```

will match records with event ID equal 7, and

```
<id>10-20</id>
```

will match records with ID in range from 10 to 20 (inclusive). This tag has no effect for text log files, and can be used as a synonym for `<facility>` tag for syslog monitoring.

18.7.4 `<source>` Tag

Tag `<source>` can be used to filter records from Windows Event Log by event source. You can specify exact event source name or pattern with `*` and `?` meta characters.

Some examples:

```
<source>Tcpip</source>
```

will match records with event source `Tcpip` (case-insensitive), and

```
<source>X*</source>
```

will match records with event source started from letter `x`. This tag has no effect for text log files, and can be used as a synonym for `<tag>` tag for syslog monitoring.

18.7.5 `<level>` Tag

Tag `<level>` can be used to filter records from Windows Event log by event severity level (also called *event type* in older Windows versions). Each severity level has its own numeric value, and to filter by multiple severity levels you should specify sum of appropriate values (bitmask). Severity level numerical values are the following:

Severity level	Decimal value
Error	1
Warning	2
Information	4
Audit Success	8
Audit Failure	16
Critical (only on Windows 7/Windows Server 2008 and higher)	256

Some examples:

```
<level>1</level>
```

will match all records with severity level *Error*, and

```
<level>6</level>
```

will match all records with severity level *Warning* or *Information*. This tag has no effect for text log files, and can be used as a synonym for `<severity>` tag for syslog monitoring.

18.7.6 `<facility>` Tag

Tag `<facility>` can be used to filter syslog records (received by NetXMS built-in syslog server) by facility code. The following facility codes can be used:

Code	Facility
0	kernel messages
1	user-level messages
2	mail system
3	system daemons
4	security/authorization messages
5	messages generated internally by syslogd
6	line printer subsystem
7	network news subsystem
8	UUCP subsystem
9	clock daemon
10	security/authorization messages
11	FTP daemon
12	NTP subsystem
13	log audit
14	log alert
15	clock daemon
16	local use 0 (local0)
17	local use 1 (local1)
18	local use 2 (local2)
19	local use 3 (local3)
20	local use 4 (local4)
21	local use 5 (local5)
22	local use 6 (local6)
23	local use 7 (local7)

You can specify either single facility code or facility code range (by using two numbers separated by minus sign). For example:

```
<facility>7</facility>
```

will match records with facility code equal 7, and

```
<facility>10-20</facility>
```

will match records with facility code in range from 10 to 20 (inclusive). This tag has no effect for text log files, and can be used as a synonym for `<id>` tag for Windows Event Log monitoring.

18.7.7 <tag> Tag

Tag `<tag>` can be used to filter syslog records (received by NetXMS built-in syslog server) by content of `tag` field. You can specify exact value or pattern with `*` and `?` meta characters.

Some examples:

```
<tag>httpd</tag>
```

will match records with tag “httpd” (case-insensitive), and

```
<tag>X*</tag>
```

will match records with tag started from letter x. This tag has no effect for text log files, and can be used as a synonym for `<source>` tag for Windows Event Log monitoring.

18.7.8 <severity> Tag

Tag `<severity>` can be used to filter syslog records (received by NetXMS built-in syslog server) by severity level. Each severity level has its own code, and to filter by multiple severity levels you should specify sum of appropriate codes. Severity level codes are following:

Code	Severity
1	Emergency
2	Alert
4	Critical
8	Error
16	Warning
32	Notice
64	Informational
128	Debug

Some examples:

```
<severity>1</severity>
```

will match all records with severity level *Emergency*, and

```
<severity>6</severity>
```

will match all records with severity level *Alert* or *Critical*. This tag has no effect for text log files, and can be used as a synonym for `<level>` tag for Windows Event Log monitoring.

18.7.9 <description> Tag

Tag `<description>` contains textual description of the rule.

18.7.10 <event> Tag

Tag `<event>` defines event to be generated if current log record match to regular expression defined in `<match>` tag. Inside `<event>` tag you should specify event name or event code to be generated. All matched capture groups will be given to the event as an event parameters.

Event tag has `tag` attribute. If the attribute is set, then it will be added to the selected event tag list.

18.7.11 <context> Tag

Tag `<context>` defines activation or deactivation of contexts. This option can be used for multi line match. First line sets context and next generates event in case if context was set. Examples can be found further in *Examples of Parser Definition File* section.

It has the following format:

```
<context action="action" reset="reset mode">context name</context>
```

Possible actions are:

Action	Description
clear	Deactivate (clear “active” flag of) given context.
set	Activate (set “active” flag of) given context.
reset	Defines how context will be deactivated

Possible values for reset mode are:

Reset mode	Description
auto	Deactivate context automatically after first match in context (match rule with <code>context</code> attribute set to given context).
manual	Context can be deactivated only by explicit <code><context action="clear"></code> statement.

Both `action` and `reset` attributes can be omitted; default value for `action` is `set`, and default value for `reset` is `auto`.

18.7.12 <exclusionSchedules> Tag

Tag `<exclusionSchedules>` defines time when file should not be parsed. Each cron expression should be defined in `<schedule>`. This should be used to define time when file should not be opened. Once time does not match cron file will be reopened and all added lines will be parsed. See [Cron format](#) for supported cron format options.

Example:

```
<parser>
  <file>/var/log/messages</file>
  <rules>
    <rule>
      <match>error</match>
      <event>USR_APP_ERROR</event>
    </rule>
  </rules>
  <exclusionSchedules>
    <schedule>0-2 0 * * *</schedule>
  </exclusionSchedules>
</parser>
```

18.8 Examples of Parser Definition File

Generate event with name `USR_APP_ERROR` if line in the log file `/var/log/messages` contains word `error`:

```
<parser>
  <file>/var/log/messages</file>
  <rules>
    <rule>
      <match>error</match>
      <event>USR_APP_ERROR</event>
    </rule>
  </rules>
</parser>
```

Generate event with name `SYS_PROCESS_START_FAILED` if line in the log file `C:\demo.log` contains word `process:` and is immediately following line containing text `process startup failed`; everything after word `process:` will be sent as event's parameter:

```
<parser>
  <file>C:\demo.log</file>
  <rules>
    <rule>
      <match>process startup failed</match>
      <context action="set" reset="auto">STARTUP_FAILED</context>
    </rule>
    <rule context="STARTUP_FAILED">
      <match>process: (.*)</match>
      <event>SYS_PROCESS_START_FAILED</event>
    </rule>
  </rules>
</parser>
```

18.9 Passing parameters to events

The log parser adds parameters to events. For non-Windows platforms the following parameters are provided:

Number	Description
1 to n	Capture groups
n+1	Event tag (if set in log parser policy configuration, otherwise this field is omitted)
n+2	Repeat count - how many times this rule was matched previously.

For Windows the following parameters are provided:

Number	Description
1 to n	Capture groups
n+1	Event tag (if set in log parser policy configuration, otherwise this field is omitted)
n+2	Windows publisher name
n+3	Windows event id
n+4	Windows severity
n+5	Windows record Id
n+6	Repeat count - how many times this rule was matched previously.
n+7 to k	Windows event strings

Consider the following line is received via syslog, or added to a monitored file:

```
24.04.2015 12:22:15 1 5 system,error,critical login failure for user
testUser from 11.2.33.41 via ssh
```

We can extract username and login method from the syslog message, and pass it as parameters to an event with the following rule:

```
<match>system,error,critical login failure for user (.*) from .* via
(.*)</match> <event>10000</event>
```

Username will be sent to the event as `%1`, IP address will not be sent, and login method will be sent as `%2`.

18.10 Log parser metrics

Log parser provides some additional statistic information through *Metrics*. Metrics take name of particular parser as an argument. If name is not set, then file name is used.

Statistic information is reset on agent startup and when log parser policy is reapplied.

Available metrics:

Metric Name	Description
Log-Watch.Pa	Parser <i>name</i> status
Log-Watch.Pa	Number of records matched by parser <i>name</i>
Log-Watch.Pa	Number of records processed by parser <i>name</i>

Available lists:

List Name	Description
Log-Watch.Pa	List of parser names. If no name is defined then parser file name will be used.

WINDOWS EVENT LOG SYNCHRONIZATION

NetXMS can collect and centrally store Windows event logs. Collection is performed by NetXMS agents. It's possible to filter by log type, Source and Event IDs at agent side to reduce network traffic consumption.

Windows events received by NetXMS server are stored in the database and can later be viewed in *View ▸ Windows event log*. Upon reception event logs can be parsed according to rules and NetXMS events can be generated.

19.1 Agent Configuration for Event Log Synchronization

Agent configuration to enable Windows Event Log Synchronization can be done in two ways:

1. In agent's configuration file
2. Using Agent Configuration policy. For more information see *Agent Policies*.

Windows Event Log Synchronization subagent should be enabled in agent configuration:

```
SubAgent=wineventsync.nsm
```

Logs that should be monitored (Application, Security, etc) are specified in `WinEventSync` section:

```
[WinEventSync]
EventLog=Application
EventLog=Security
EventLog=System
```

With above configuration all records in the specified logs will be synchronized. It is possible to configure per-log settings to filter only part of records. Per-log configuration is specified in sections named according to log name, e.g. `WinEventSync/System`.

Filtering is done in two stages. First is pre-filter, which allows to independently filter events by Event ID, Source and Severity level. Second stage - Filter (added in version 5.2) allows to define chain of rules to filter by combinations of Event ID, Source and Severity level.

19.1.1 Pre-filter

Event ID

Filtering by Event IDs is done using options `IncludeEvent` and `ExcludeEvent`. You can configure a range like 100-200. Comma separated lists are not supported, you can however add multiple `Include/ExcludeEvent` lines.

By default, if no `IncludeEvent` or `ExcludeEvent` are given, all IDs in that log will be synced. Explicit Includes override Excludes. So if you configure an `IncludeEvent=201` and an `ExcludeEvent=200-300`, you will receive all Events except 200 and 202-300.

To exclude all Event IDs, use `ExcludeEvent=0-65535`, then you can use `IncludeEvent` to select only the IDs you need.

```
[WinEventSync/Security]
IncludeEvent=4624-4625
IncludeEvent=4800-4803
ExcludeEvent=0-65535
```

Source

Filtering by Source is done using options `IncludeSource` and `ExcludeSource`. By default, if no `IncludeSource` are `ExcludeSource` are given, all sources in that log will be synchronized. You can use `ExcludeSource=*` to exclude every source and specify `IncludeSource` to override the exclude for specific sources.

```
[WinEventSync/System]
IncludeSource=Microsoft-Windows-WindowsUpdateClient
ExcludeSource=*
```

Severity level

Filtering by severity level (also called *event type* in older Windows versions) is done using option `SeverityFilter`. Each severity level has it's own numeric value, and to filter by multiple severity levels you should specify sum of appropriate values (bitmask). Or alternatively you can specify severity level names separated by commas. Below are level names and their values:

Severity level name	Hexadecimal value	Decimal value
Error	0x001	1
Warning	0x002	2
Information / Info	0x004	4
AuditSuccess	0x008	8
AuditFailure	0x010	16
Critical	0x100	256

Below examples will have same result of filtering only Warning and Error records:

```
[WinEventSync/System]
SeverityFilter = 0x012
```

```
[WinEventSync/System]
SeverityFilter = 18
```

```
[WinEventSync/System]
SeverityFilter = Warning,Error
```

19.1.2 Filter

Added in version 5.2.

This stage allows to specify chain of rules to filter by combinations of Event ID, Source and Severity level. Rules are specified using `Filter` option.

```
Filter = Action:Source:Id:Severity
```

Name	Re-quired	Description
Action	Yes	Either <code>accept</code> or <code>reject</code>
Source	No	Name of event source. Two wildcard characters are supported: * - represents zero, one or multiple characters. ? - represents any single character.
Id	No	Event ID. Ranges are supported (e.g. 4800–4803). * means any ID.
Severity	No	Severity level. Bitmask or comma-separated severity level names are supported in same way as in pre-filter. * means any severity level.

If event matches specific rule, then it is accepted or rejected, depending on action set for this rule. Unmatched events proceed to subsequent rules. If event is not matched by any rule, it is accepted - it is recommended to have `Filter=reject` as the last rule to avoid that.

Agent log messages related to windows event log synchronization are written with tag `winsyncevent`. For debugging you can add `DebugTags=winsyncevent:6` to agent configuration - this will set debug level 6 for that tag.

19.2 Server Configuration for Event Log Synchronization

Upon being received on server Windows events are parsed according to rules defined in *Configuration ▶ Windows event parser*. Rules can be edited in two ways - using graphical editor or XML editor. When switching from one editor to another all entered information is automatically converted.

If *Process all* checkbox is not set, rules are processed until first match. If it's set, all rules are always processed.

In the *Macros* section you can define macros for use in matching rules. For example, it can be useful to define macro for IP address and use it in matching rules instead of actual regular expression. You can define as many macros as you wish. Each macro should have unique name, and can be used in matching rules in form `@{name}`.

A rule can have multiple conditions - regular expression match, severity level, Event ID, Source, log type.

Matching regular expression contains a PCRE compliant regular expression that is used to match Windows event log records. Parts enclosed in parenthesis are extracted from Windows event log record and passed as arguments of generated NetXMS event. You can use macros defined in *Macros* section. If *Invert* checkbox is set, Windows event log record will be considered matching if it does not match regular expression.

Level can be used to filter records from Windows Event log by event severity level (also called *event type* in older Windows versions). Each severity level has its own numeric value, and to filter by multiple severity levels you should specify sum of appropriate values (bitmask). Severity level numerical values are the following:

Severity level	Decimal value
Error	1
Warning	2
Information	4
Audit Success	8
Audit Failure	16
Critical (only on Windows 7/Windows Server 2008 and higher)	256

Id can be used to filter records from Windows Event Log by event ID. You can specify either single event ID (e.g. 7) or ID range by using two numbers separated with minus sign (e.g. 10–20 will match records with ID in range from 10 to 20 inclusive).

Source can be used to filter records from Windows Event Log by event source. You can specify exact event source name or pattern with * and ? meta characters. E.g. `Tcpip` will match records with event source `Tcpip` (case-insensitive), and `X*` will match records with event source started from letter `X`.

Log name allows to filter records by Windows Event Log name. You can specify exact name or pattern with * and ? meta characters.

Description contains textual description of the rule. It is printed in parser trace in the log file.

When a rule is matched the following actions can be performed:

- Generate NetXMS event. Event generation is optional - it could be useful to have rules that work as exclusion - match specific conditions and do not perform any actions.
- Break. In this case the following rules will not be processed even if *Process all* is set.
- Do not save to database. If this is set, matched Windows Event Log record will not be saved to the database.

19.3 Passing parameters to events

The log parser can send parameters to events. All capture groups will be sent to the event as parameters.

Number	Description
1...n	Capture groups

SSH MONITORING

20.1 SSH configuration

NetXMS can execute commands via an SSH connection and save the output as DCI values.

SSH connections are always established via an agent. For this to work, the `ssh.nsm` subagent should be enabled in the agent config file.

The subagent uses the built-in `libssh`. It reads the configuration in standard `ssh` format from `~/ssh/config`. It is also possible to specify a custom location for the configuration file by adding `ConfigFile=` to the `[SSH]` section of the agent configuration file.

If zoning is not used, the agent running on the NetXMS server is used for SSH connections. If zoning is used, zone proxies are used. If a zone has no proxies configured, the agent on the NetXMS server is used as a last resort.

The username and password are specified in *Node properties -> Communications -> SSH*. The same properties page can be used to specify an `ssh` port for node, the proxy for `ssh` polling and an `ssh` key if required. If a proxy node is specified on this property page, the connection will be performed via that node only.

The screenshot shows the 'Properties for jworker.office.radensolutions.com' window. On the left is a sidebar with a search bar and a tree view. The tree view has categories like General, Communications, Agent, Ethernet/IP, ICMP, SNMP, SSH (highlighted in orange), Syslog, Web Services, Polling, Access Control, Comments, Custom Attributes, Dashboards, External Resources, Location, Map Appearance, Rack or Chassis, and Responsible Users. The main panel is titled 'SSH' and contains the following fields: 'Login' with the value 'jenkins', 'Password' with masked characters and an eye icon, 'Port' with the value '22', 'Key from configuration' with a dropdown menu showing 'Local', and 'Proxy' with the value '<default>' and edit/delete icons. At the bottom right are buttons for 'Restore Defaults', 'Apply', 'Cancel', and 'Apply and Close'.

In DCI properties the `SSH` origin should be chosen. The parameter is the actual `ssh` command that is executed.

Only the first line of the output is stored as a DCI value. For numeric data type output is parsed from its beginning until the first non-numeric character.

The screenshot shows the 'Properties for' dialog box with the 'General' tab selected. The left sidebar contains a list of categories: General, Custom Schedule, Transformation, Thresholds, Instance Discovery, Performance Tab, Access Control, Other options, and Comments. The main area is divided into several sections:

- Description:** A text field containing 'System information'.
- Data:**
 - Parameter:** A text field containing 'uname -a' and a 'Select...' button.
 - Origin:** A dropdown menu set to 'SSH'.
 - Data Type:** A dropdown menu set to 'String'.
 - Three checkboxes: 'Interpret SNMP octet string raw value as' (unchecked), 'Use custom SNMP port:' (unchecked), and 'Use custom SNMP version:' (unchecked).
 - Below the checkboxes are three input fields: the first contains 'None', the second contains '1', and the third is empty.
 - Sample count for average value calculation (0 to disable):** An input field containing '0'.
- Source node:** A dropdown menu set to '<none>'.
- Agent cache mode:** A dropdown menu set to 'Default'.
- Polling:**
 - Polling mode:** A dropdown menu set to 'Fixed intervals (default)'.
 - Polling interval (seconds):** An empty input field.
- Status:** Three radio buttons: 'Active' (selected), 'Disabled', and 'Not supported'.
- Storage:**
 - Retention mode:** A dropdown menu set to 'Use default retention time'.
 - Retention time (days):** An empty input field.

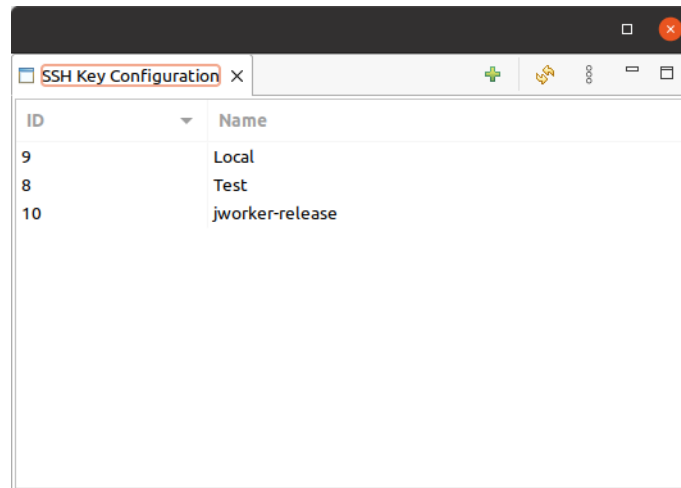
At the bottom right, there are four buttons: 'Restore Defaults', 'Apply', 'Cancel', and 'Apply and Close'.

There is also the `SSH.Command(*)` metric of origin NetXMS Agent that works in a similar way, but where target and credentials are specified as arguments. It is also necessary to manually specify the Source node, otherwise the agent of the monitored node will be used for establishing the ssh connection.

Metric Name	Description
<code>SSH.Command(target,login,password,command,[pattern],[ssh_key_id])</code>	<code>%{node_primary_ip}</code> macro can be used to specify the nodes primary IP address as <i>target</i> .

20.2 SSH key configuration

An SSH key can be added in *Configuration -> SSH key configuration* and then used in the object configuration for the SSH connection.



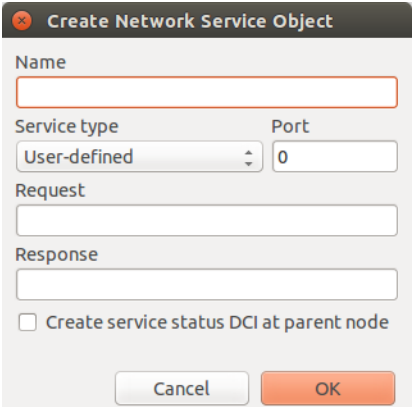
NETWORK SERVICE MONITORING

There are two options to add service monitoring: the first one is to add it through node menu option *Create* → *Create Network Service...* as an object with the status that will be propagated on a node, and the second one is to add its monitoring as DCI.

In both cases monitoring is done by the help of NetXMS agent. In agent's configuration file *NetSVC* subagent should be enabled.

21.1 Network Service Object

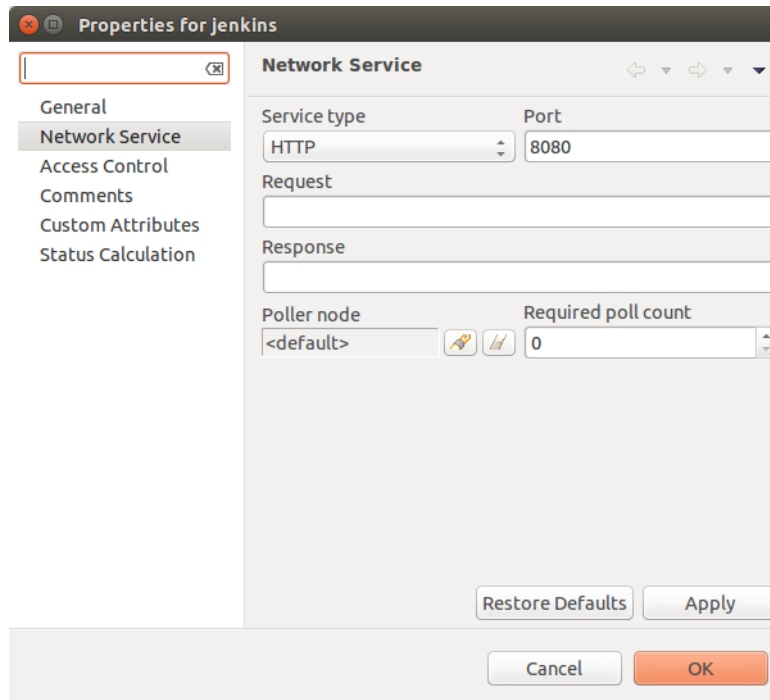
Object representing network service running on a node (like http or ssh), which is accessible online (via TCP IP). Network Service objects are always created manually. Currently, the system works with the following protocols - SSH, POP3, SMTP, FTP, HTTP, HTTPS, Telnet and Custom protocol type. For Custom protocol, user should define TCP port number and the system will be checking if it's possible to establish connection to that port. For the predefined standard services the system will also check whether an appropriate response is returned. In case of SMTP, the system will send a test mail, in case of POP3 - try to log in with a certain user, in case of HTTP - check whether the contents of a desired web page correspond to a certain given template. As soon as the Network Service object is created, it will be automatically included into the status poll. Each time when the status poll for the particular node is carried out, all Network Service objects are polled for a reply. If an object's reply corresponds to a certain condition, its status is set as NORMAL. If an object is not responding, its status will be changed to CRITICAL. It is possible to create a *DCI* that will collect status of Network Service object.



The image shows a dialog box titled "Create Network Service Object". It contains the following fields and controls:

- Name:** A text input field.
- Service type:** A dropdown menu currently showing "User-defined".
- Port:** A text input field containing the value "0".
- Request:** A text input field.
- Response:** A text input field.
- Checkboxes:** A checkbox labeled "Create service status DCI at parent node" which is currently unchecked.
- Buttons:** "Cancel" and "OK" buttons at the bottom right.

In default configuration request is done with the help of NetXMS agent (by its NetSVC subagent) on the server node. If it should be done through different node it should be changed in its properties after service creation by selecting Poller node. There is also possibility to set number of polls that is required to be sure that state have changed.



21.2 Network service monitoring using DCI

Second option is to use *DCI* to monitor service. Service monitoring metrics are provided NetXMS agent (by it's NetSVC *subagent*). DCIs should either be created on the node where agent is running, or they can be created on another node and the node with agent can be specified in *Source node override* in DCI's properties.

More about URL options can be found there: <https://curl.se/docs/url-syntax.html>

This subagent will add the following metrics to list of metrics available on agent:

Metric Name	Description
HTTP.Checksum.MD5(URL, [named parameters])	Calculate hash for the provided URL. Port number can be specified in the URL. <i>http</i> and <i>https</i> schemes are supported in the URL. Calculates hash only if web server returns 200 status code. Starting from second parameter this metric accepts named parameters in <i>name = value</i> form. When parameter(s) are used, they should be used without []. The following parameters are supported (all parameters are optional):
HTTP.Checksum.SHA1(URL, [named parameters])	
HTTP.Checksum.SHA256(URL, [named parameters])	
	<ul style="list-style-type: none"> <i>follow-location</i> - <i>true</i> - follow redirects which web server sends as part of an HTTP header in a 3xx response; <i>false</i> (default) - do not follow redirects <i>timeout</i> - timeout in milliseconds <i>verify-host</i> - <i>true</i> (default) - verify that host name from URL matches one from certificate (CURLOPT_SSL_VERIFYHOST = 2); <i>false</i> - do not verify that host name from URL match one from certificate (CURLOPT_SSL_VERIFYHOST = 0) <i>verify-peer</i> - <i>true</i> (default) - verify peer certificate; <i>false</i> - do not verify peer certificate.

continues on next page

Table 1 – continued from previous page

Metric Name	Description
NetworkService.Status(<i>URL</i> , [<i>named parameters</i>])	<p>Check status of network service and return numeric value denoting the result. Port number can be specified in the URL. URL supports the following schemes: <i>http</i>, <i>https</i>, <i>ssh</i>, <i>telnet</i>, <i>tcp</i>, <i>smtp</i> and <i>smtps</i>. For <i>ssh</i> protocol connection is established. For <i>telnet</i> it's checked that host sends some characters after connection is established. For <i>tcp</i> only ability to establish connection to specified port is checked. For <i>smtp</i> and <i>smtps</i> test email is being sent.</p> <p>Starting from second parameter this metric accepts named parameters in <i>name = value</i> form. When parameter(s) are used, they should be used without [].</p> <p>Optional parameter supported for all schemes:</p> <ul style="list-style-type: none"> • <i>timeout</i> - timeout in milliseconds <p>Parameters supported for <i>http</i> and <i>https</i> schemes (all parameters are optional):</p> <ul style="list-style-type: none"> • <i>follow-location</i> - <i>true</i> - follow redirects which web server sends as part of an HTTP header in a 3xx response; <i>false</i> (default) - do not follow redirects • <i>include-headers</i> - if set to <i>true</i> (default), <i>pattern</i> is matched within headers and body of the web page. If set to <i>false</i>, <i>pattern</i> is matched in web page body only. • <i>pattern</i> - regular expression to match. • <i>response-code</i> - web server response code to match. <p>Parameters supported for <i>smtp</i> and <i>smtps</i> schemes:</p> <ul style="list-style-type: none"> • <i>to</i> - test email will be sent to this address. Obligatory parameter • <i>from</i> - test email will be sent from this address. Optional parameter, default value depends on configuration of NetSVC subagent. <p>Parameters supported for all schemas except <i>ssh</i>, <i>telnet</i>, <i>tcp</i>:</p> <ul style="list-style-type: none"> • <i>verify-host</i> - <i>true</i> (default) - verify that host name from URL matches one from certificate (CURLOPT_SSL_VERIFYHOST = 2); <i>false</i> - do not verify that host name from URL match one from certificate (CURLOPT_SSL_VERIFYHOST = 0) • <i>verify-peer</i> - <i>true</i> (default) - verify peer certificate; <i>false</i> - do not verify peer certificate. • <i>tls-mode</i> - TLS mode that should be used. One of: <i>none</i>, <i>try</i>, <i>always</i> • <i>login</i> - login • <i>password</i> - password (can be encrypted by <i>nxencpasswd</i> tool) <p>Metric returns one of the following values:</p> <ul style="list-style-type: none"> • 0 - Success, connection to target was established and expected response was received. • 2 - Can not connect to target (connection refused or connection timeout) • 3 - Invalid / unexpected response from target (e.g. pattern or response-code not matched) • 4 - Agent internal error • 5 - Protocol handshake error (e.g. wrong data or no data expected by protocol received, SSL certificate problem)

continues on next page

Table 1 – continued from previous page

Metric Name	Description
NetworkService.ResponseTime(<i>URL</i> , [<i>named parameters</i>])	Measures response time, returns value in milliseconds. For <i>http</i> and <i>https</i> schemas time to fully load the web page is measured. Metric support same parameters as NetworkService.Status.
NetworkService.TLSStatus(<i>host</i> , <i>port</i> , [<i>named parameters</i>])	Check remote TLS service and return numeric value denoting the result. Starting from third parameter this metric accepts named parameters in <i>name = value</i> form. When parameter(s) are used, they should be used without []. The following optional parameter is supported: <ul style="list-style-type: none"> <i>timeout</i> - timeout in milliseconds Metric returns one of the following values: <ul style="list-style-type: none"> 0 - Success, connection to target was established and expected response was received. 2 - Can not connect to target (connection refused or connection timeout) 3 - Invalid / unexpected response from target 4 - Agent internal error 5 - Protocol handshake error
NetworkService.TLSResponseTime(<i>host</i> , <i>port</i> , [<i>named parameters</i>])	Measures time to perform TLS handshake, returns value in milliseconds. Metric support same parameters as NetworkService.TLSStatus.
TLS.Certificate.ExpirationDate(<i>host</i> , <i>port</i>)	Returns expiration date (YYYY-MM-DD) of X.509 certificate of remote TLS service
TLS.Certificate.ExpirationTime(<i>host</i> , <i>port</i>)	Returns expiration time (Unix time) of X.509 certificate of remote TLS service
TLS.Certificate.ExpiresIn(<i>host</i> , <i>port</i>)	Returns number of days until expiration of X.509 certificate of remote TLS service
TLS.Certificate.Issuer(<i>host</i> , <i>port</i>)	Returns issuer of X.509 certificate of remote TLS service
TLS.Certificate.Subject(<i>host</i> , <i>port</i>)	Returns subject of X.509 certificate of remote TLS service
TLS.Certificate.TemplateID(<i>host</i> , <i>port</i>)	Returns template ID of X.509 certificate of remote TLS service

21.2.1 Examples

```
NetworkService.Status(http://www.netxms.org)
```

This metric will return 0 (success). In this case we are just checking that web server provides response, without checking for pattern or status code (which is 301 in this case, as we receive redirect to <https://www.netxms.org/>)

```
NetworkService.Status(http://www.netxms.org, response-code=200)
```

Returns 3 (unexpected response) as response code (301) does not match the value we are checking for.

```
NetworkService.Status(http://www.netxms.org, follow-location=true, response-code=200)
```

Returns 0 (success) as it follows redirects and ultimately gets web page with response code 200.

```
NetworkService.Status(https://netxms.org, pattern="^HTTP/(1\.[01]|2) 200 .*")
```

Here we are checking for specific pattern both in headers and web page (*include-headers* parameter is not specified and it's default value is *true*).

```
NetworkService.Status(http://www.netxms.org, include-headers=false, pattern=".*Moved Permanently.*")
```

Checking for specific pattern only in web page itself, but not in headers.

```
NetworkService.Status(https://a.web.site.with.self.signed.certificate)
```

Returns 5 (Protocol handshake error) because libcurl can not verify the self-signed certificate.

```
NetworkService.Status(https://a.web.site.with.self.signed.certificate, verify-peer=false)
```

Returns 0 (Success) as we disabled peer certificate verification.

```
NetworkService.Status(tcp://netxms.org:80)
```

Returns 0 (Success) as we were able to establish TCP connection to port 80

```
NetworkService.Status(tcp://netxms.org:88, timeout=2000)
```

Returns 2 (Timeout) as it was not possible to establish TCP connection to port 1. Waits for 2 seconds according to *timeout* that we have specified.

```
NetworkService.ResponseTime(https://www.google.com)
```

Returns time in milliseconds it took to fully retrieve the web page from the server.

```
NetworkService.TLSStatus(netxms.org, 443)
```

Returns 0 (success). This only performs TLS handshake, without retrieving any web page from the server.

```
NetworkService.TLSResponseTime(www.google.com, 443)
```

Returns the time it takes to perform TLS handshake with the server.

21.3 NetSVC configuration

This subagent performs network services checks by employing libcurl. More information about syntax can be found here: <http://curl.haxx.se/docs/manpage.html>.

Note

If agent is build from sources, then libcurl-dev should be installed to build netsvc subagent.

To operate, NetSVC subagent should be loaded. All configuration parameters related to NetSVC subagent should be placed into **[netsvc]** section of agent's configuration file. The following configuration parameters are supported:

Parameter	Description	Default value
CA	Path to a file holding one or more certificates to verify the peer with (CURLOPT_CAINFO)	
DomainName	Used in SMTP check. Default <i>from</i> email address is composed as <i>noreply@DomainName</i> .	netxms.org
NegativeResponseTimeOnError	For metrics that measure response time, return negative time value instead of data collection error.	false
VerifyPeer	Verify peer certificate	true
Timeout	Timeout in milliseconds.	

Agent's configuration file example:

```
SubAgent = netsvc
[netsvc]
Timeout = 3000
```


DATA COLLECTION FROM WEB SERVICES

NetXMS has a built-in data collection mechanism using web services, allowing to extract data for DCIs from JSON, XML, or plain text responses to HTTP requests. Data collection from web services is done via the NetXMS agent. If zoning is not used (or for the Default zone), the agent running on the NetXMS server is used. If zoning is used, zone proxies are used (and if a zone has no proxies configured, the agent on NetXMS server is used as last resort).

22.1 Configuring Web Service Data collection

22.1.1 Agent configuration

Starting from version 3.8 of the NetXMS agent, data collection from web services is disabled by default. To enable it, add `EnableWebServiceProxy=yes` to the agent configuration file and restart the agent.

22.1.2 Web service definitions

Common configuration related to multiple metrics and nodes is set up in the web service definition editor accessible via the *Configuration -> Web Service Definitions* menu.

The screenshot shows a macOS-style window titled "Edit Web Service Definition". On the left is a sidebar with two tabs: "General" (selected) and "Headers". The main area is divided into several sections: "Name" with a text field containing "Web Service 1"; "URL" with an empty text field; "Authentication" with a dropdown menu set to "BASIC"; "Login" with a text field containing "username"; "Password" with a text field containing "password"; "Options" with two sub-sections: "Cache retention time" with a spinner set to "0" and "Request timeout" with a spinner set to "0"; and "Description" with a large empty text area. At the bottom right are two buttons: "Cancel" and "Apply and Close".

The following parameters can be configured:

- Web service name
- Web service URL
- Additional HTTP headers
- Authentication data (authentication type, login, password)
- Cache retention time (in seconds)
- Request timeout (in seconds)

The web service URL and additional HTTP headers fields can contain macros that are expanded when the actual request is made. So you can, for example, set the URL as `%{url}` and keep the actual URL in a custom attribute of the node with the name `url`.

22.1.3 DCI Configuration

DCI configuration provides the DCI origin “web service”. Metric name for this origin contains the web service definition name with optional arguments and the path to the document element that has to be retrieved, or a PCRE compliant regex with one capture group for text responses.

For example:

- `WebService1:/system/cpu/usage`
- `WebService2(eth0):/stat/bytesIn`
- `WebService3(10,20,30):^(\d*)`

Service arguments can be inserted into the request URL or headers using macros `%1`, `%2`, and so on. For XML and JSON responses, the path to the document element should start with `/`. An XML response, according to the standard, should only have one upper level tag. For text responses, the first capture group of the regular expression is returned.

22.1.4 Instance discovery

For web service discovery the “Web Service” instance discovery method can be used. It accepts a web service name with optional arguments and the path to the root element of the document where enumeration will start. Each sub-element of a given root element will be considered as a separate instance.

For example:

- `WebService1:/system/cpu` will enumerate all elements under “/system/cpu”
- `WebService2(eth0):/stat` will enumerate all elements under “/stat”

22.2 Data collection process

The data collection process from the server point of view is:

1. The server finds the web service definition by the given name, passes any parameters to it, and gets back the URL and headers with all macros expanded.
2. The server determines the agent to be used for the request based on zone settings, node settings, agent availability, etc.
3. The server sends the request to the selected agent. A request consists of an URL, headers, and document path.
4. The server waits for a response from the agent and processes the retrieved data similar to any other DCI type. For instance, the discovery server provides a new instance discovery method - “web service” which accepts a web service name with optional arguments and path to the root element of the document where enumeration will start. Each sub-element of the given root element will be considered a separate instance.

Actual requests and response parsing is implemented on the agent level. This provides the necessary flexibility for accessing services not directly reachable from the management server as well as offloads response parsing from the server to agents.

The data collection process from the agent point of view is:

1. The agent receives a web service request (URL, authentication data, headers) and list of elements to retrieve from the server.
2. The agent checks the document cache if the requested URL was already retrieved and data is within configured cache retention time. If so, values of the requested elements from cached data are returned to the server.
3. The agent performs an HTTP request using the provided service data. If the request is successfully retrieved, the document is parsed into tree form and values of the requested elements are returned to the server. No additional configuration should be required on the agent side.

22.3 Examples

This example shows how to use the same web service JSON output for instances and then to collect data.

We assume that the configuration is already done and we have a web service with the “WebService1” name, that returns a JSON data structure as:

```
[
  {
    "name": "Object1",
    "status": "Online",
    "position": "Front"
  },
  {
    "name": "Object2",
    "position": "Back"
  },
  {
    "name": "Object3",
    "status": "Offline",
    "position": "Front"
  }
]
```

From this JSON document we want to get a separate DCI for each object. We will collect status if exist and will set status to Offline if the object does not contain status parameter.

The DCI will have the following configuration:

- Instance discovery method: Web Service
- Web service request: WebService1:[.[]].name]

This will create an array with names. Each name will be taken as an instance:

```
["Object1", "Object2", "Object3"]
```

- Origin: Web service
- Metric: (.[[] | select(.name == "{instance}").status) // “failed”

This configuration will get the status for the object with name like {instance} (will be replaced by its real name on instance discovery) and it will return “failed” if this object does not contain the status field.

MODBUS

Added in version 4.4.

NetXMS can collect data via the Modbus-TCP protocol. Data collection is performed by the NetXMS server or by NetXMS agents operating in proxy mode.

To enable agent operation as a Modbus proxy, add `EnableModbusProxy=yes` to the agent configuration file and restart the agent.

The metric for Modbus data collection items has a special format denoting the type of Modbus unit id, register type, register address and the way how obtained data should be interpreted:

`[[unit-id:] register-type:] register-address [| conversion]`

Metric component	Description
unit-id	Modbus unit ID. Optional, if used, should be specified without []. To use it, <code>register-type</code> should also be provided.
register-type	Type of Modbus register. Optional, if not specified, <code>hold</code> will be used. Should be specified without [] if used. Supports following values: <ul style="list-style-type: none">• <code>coil</code> - Coil• <code>discrete</code> - Discrete Input• <code>hold</code> - Holding Register• <code>input</code> - Input Register
register-address	Address of Modbus register. Can be provided as decimal number or hexadecimal number prefixed by <code>0x</code> .

continues on next page

Table 1 – continued from previous page

Metric component	Description
conversion	<p>Conversion of Modbus data. Optional, if not specified, <code>uint16</code> will be used. Should be specified without [] if used. Affects the number of Modbus registers being read and how read data is interpreted:</p> <ul style="list-style-type: none"> • <code>int16</code> - 16 bit signed integer • <code>uint16</code> - 16 bit unsigned integer • <code>int32</code> - 32 bit signed integer (will read 2 registers) • <code>uint32</code> - 32 bit unsigned integer (will read 2 registers) • <code>int64</code> - 64 bit signed integer (will read 4 registers) • <code>uint64</code> - 64 bit unsigned integer (will read 4 registers) • <code>float</code> - same as <code>float-abcd</code> • <code>float-abcd</code> - 4 byte floating point number, ABCD byte order • <code>float-cdab</code> - 4 byte floating point number, CDAB byte order • <code>float-badc</code> - 4 byte floating point number, BADC byte order • <code>float-dcba</code> - 4 byte floating point number, DCBA byte order • <code>double</code> - same as <code>double-be</code> • <code>double-be</code> - 8 byte floating point number, big endian byte order • <code>double-le</code> - 8 byte floating point number, little endian byte order • <code>string-N</code> - string of N characters (will read (N + 1) / 2 registers) • <code>string-N-CP</code> - string of N characters encoded using codepage CP (will read (N + 1) / 2 registers)

23.1 Modbus metric examples

`0x2A`

Read holding register at address 2A hexadecimal (42 decimal), interpret as `uint16`.

`input:8`

Read input register at address 8 decimal, interpret as `uint16`.

`10|int16`

Read holding register at address 10 decimal, interpret as `int16`.

`input:55|float`

Read two input registers starting from 55 decimal, interpret as float with ABCD byte order.

DATABASE MONITORING

There are several *subagents* for database monitoring: DB2, Informix, Oracle, MySQL, MongoDB, PostgreSQL. Below we will describe how to configure and use these subagents. Besides it's also possible to monitor other types of databases supported by NetXMS server([link to supported database list](#)) using database query subagent as these databases support receiving performance parameters using queries. This subagent details are described in *Application Database Monitoring* chapter.

24.1 Oracle

NetXMS subagent for Oracle DBMS monitoring (further referred to as Oracle subagent) monitors one or more instances of Oracle databases and reports various database-related metrics.

All metrics available from Oracle subagent are collected or calculated once per minute thus it's recommended to set DCI poll interval for these items to 60 seconds or more. All metrics are obtained or derived from the data available in Oracle's data dictionary tables and views through regular select queries. Oracle subagent does not monitor any of the metrics related to lower level database layers, such as database processes. Monitoring of such metrics can be achieved through the standard NetXMS functionality.

24.1.1 Pre-requisites

An Oracle user with the role **select_catalog_role** assigned.

Required rights can be assigned to user with the following query:

```
grant select_catalog_role to user;
```

Where *user* is the user configured in Oracle subagent for database access.

24.1.2 Configuration file

Oracle subagent can be configured using XML configuration file (usually created as separate file in configuration include directory), or in simplified INI format, in main agent configuration file (nxagentd.conf).

Database definition supports the following parameters:

Parameter	Description	Default value
Id	Database identifier. It will be used to address this database in parameters.	
TnsName	Database TNS name or connection string.	
ConnectionTTL	Time in seconds. When this time gets elapsed, connection to the DB is closed and reopened again.	3600
Username	User name for connecting to database.	
Password	Database user password. When using INI format, remember to enclose password in double quotes ("password") if it contains # character. This parameter automatically detects and accepts password encrypted with <i>nx-encpasswd</i> tool.	
EncryptedPassword	Database user password encrypted with <i>nxencpasswd</i> tool. DEPRECATED. Use Password instead.	

XML configuration allows to specify multiple databases in the **oracle** section. Each database description must be surrounded by database tags with the **id** attribute. It can be any unique integer and instructs the Oracle subagent about the order in which database sections will be processed.

Sample Oracle subagent configuration file in XML format:

```
<config>
  <agent>
    <subagent>oracle.nsm</subagent>
  </agent>
  <oracle>
    <databases>
      <database id="1">
        <id>DB1</id>
        <tnsname>TEST</tnsname>
        <username>NXMONITOR</username>
        <password>NXMONITOR</password>
      </database>
      <database id="2">
        <id>DB2</id>
        <tnsname>PROD</tnsname>
        <username>NETXMS</username>
        <password>PASSWORD</password>
      </database>
    </databases>
  </oracle>
</config>
```

You can specify only one database when using INI configuration format. If you need to monitor multiple databases from same agent, you should use configuration file in XML format.

Sample Oracle subagent configuration file in INI format:

```
[ORACLE]
ID = DB1
Name = TEST
Username = dbuser
Password = "mypass123"
```


24.1.3 Metrics

When loaded, Oracle subagent adds the following metrics to agent (all metrics require database ID as first argument):

Metric	Description
Oracle.CriticalStats.AutoArchivingOff(<i>dbid</i>)	Archive logs enabled but auto archiving off (YES/NO)
Oracle.CriticalStats.DatafilesNeedMediaRecovery(<i>dbid</i>)	Number of datafiles that need media recovery
Oracle.CriticalStats.DFOffCount(<i>dbid</i>)	Number of offline datafiles
Oracle.CriticalStats.FailedJobs(<i>dbid</i>)	Number of failed jobs
Oracle.CriticalStats.FullSegmentsCount(<i>dbid</i>)	Number of segments that cannot extend
Oracle.CriticalStats.RBSegsNotOnlineCount(<i>dbid</i>)	Number of rollback segments not online
Oracle.CriticalStats.TSOffCount(<i>dbid</i>)	Number of offline tablespaces
Oracle.Cursors.Count(<i>dbid</i>)	Current number of opened cursors system-wide
Oracle.DataFile.AvgIoTime(<i>dbid</i> , <i>datafile</i>)	Average time spent on single I/O operation for <i>datafile</i> in milliseconds
Oracle.DataFile.Blocks(<i>dbid</i> , <i>datafile</i>)	<i>datafile</i> size in blocks
Oracle.DataFile.BlockSize(<i>dbid</i> , <i>datafile</i>)	<i>datafile</i> block size
Oracle.DataFile.Bytes(<i>dbid</i> , <i>datafile</i>)	<i>datafile</i> size in bytes
Oracle.DataFile.FullName(<i>dbid</i> , <i>datafile</i>)	<i>datafile</i> full name
Oracle.DataFile.MaxIoReadTime(<i>dbid</i> , <i>datafile</i>)	Maximum time spent on a single read for <i>datafile</i> in milliseconds
Oracle.DataFile.MaxIoWriteTime(<i>dbid</i> , <i>datafile</i>)	Maximum time spent on a single write for <i>datafile</i> in milliseconds
Oracle.DataFile.MinIoTime(<i>dbid</i> , <i>datafile</i>)	Minimum time spent on a single I/O operation for <i>datafile</i> in milliseconds
Oracle.DataFile.PhysicalReads(<i>dbid</i> , <i>datafile</i>)	Total number of physical reads from <i>datafile</i>
Oracle.DataFile.PhysicalWrites(<i>dbid</i> , <i>datafile</i>)	Total number of physical writes to <i>datafile</i>
Oracle.DataFile.ReadTime(<i>dbid</i> , <i>datafile</i>)	Total read time for <i>datafile</i> in milliseconds
Oracle.DataFile.Status(<i>dbid</i> , <i>datafile</i>)	<i>datafile</i> status
Oracle.DataFile.Tablespace(<i>dbid</i> , <i>datafile</i>)	<i>datafile</i> tablespace
Oracle.DataFile.WriteTime(<i>dbid</i> , <i>datafile</i>)	Total write time for <i>datafile</i> in milliseconds
Oracle.DBInfo.CreateDate(<i>dbid</i>)	Database creation date
Oracle.DBInfo.IsReachable(<i>dbid</i>)	Database is reachable (YES/NO)
Oracle.DBInfo.LogMode(<i>dbid</i>)	Database log mode
Oracle.DBInfo.Name(<i>dbid</i>)	Database name
Oracle.DBInfo.OpenMode(<i>dbid</i>)	Database open mode
Oracle.DBInfo.Version(<i>dbid</i>)	Database version
Oracle.Dual.ExcessRows(<i>dbid</i>)	Excessive rows in DUAL table
Oracle.Instance.ArchiverStatus(<i>dbid</i>)	Archiver status
Oracle.Instance.Status(<i>dbid</i>)	Database instance status
Oracle.Instance.ShutdownPending(<i>dbid</i>)	Is shutdown pending (YES/NO)
Oracle.Instance.Version(<i>dbid</i>)	DBMS Version
Oracle.Objects.InvalidCount(<i>dbid</i>)	Number of invalid objects in DB
Oracle.Performance.CacheHitRatio(<i>dbid</i>)	Data buffer cache hit ratio
Oracle.Performance.DictCacheHitRatio(<i>dbid</i>)	Dictionary cache hit ratio
Oracle.Performance.DispatcherWorkload(<i>dbid</i>)	Dispatcher workload (percentage)
Oracle.Performance.FreeSharedPool(<i>dbid</i>)	Free space in shared pool (bytes)
Oracle.Performance.Locks(<i>dbid</i>)	Number of locks
Oracle.Performance.LogicalReads(<i>dbid</i>)	Number of logical reads
Oracle.Performance.LibCacheHitRatio(<i>dbid</i>)	Library cache hit ratio
Oracle.Performance.MemorySortRatio(<i>dbid</i>)	PGA memory sort ratio
Oracle.Performance.PhysicalReads(<i>dbid</i>)	Number of physical reads
Oracle.Performance.PhysicalWrites(<i>dbid</i>)	Number of physical writes
Oracle.Performance.RollbackWaitRatio(<i>dbid</i>)	Ratio of waits for requests to rollback segments
Oracle.Sessions.Count(<i>dbid</i>)	Number of sessions opened
Oracle.Sessions.CountByProgram(<i>dbid</i> , <i>program</i>)	Number of sessions opened by specific program

continues on next page

Table 1 – continued from previous page

Metric	Description
Oracle.Sessions.CountBySchema(<i>dbid</i> , <i>schema</i>)	Number of sessions opened with specific schema
Oracle.Sessions.CountByUser(<i>dbid</i> , <i>user</i>)	Number of sessions opened with specific Oracle user
Oracle.TableSpace.BlockSize(<i>dbid</i> , <i>tablespace</i>)	<i>tablespace</i> block size
Oracle.TableSpace.DataFiles(<i>dbid</i> , <i>tablespace</i>)	Number of datafiles in <i>tablespace</i>
Oracle.TableSpace.FreeBytes(<i>dbid</i> , <i>tablespace</i>)	Free bytes in <i>tablespace</i>
Oracle.TableSpace.FreePct(<i>dbid</i> , <i>tablespace</i>)	Free space percentage in <i>tablespace</i>
Oracle.TableSpace.Logging(<i>dbid</i> , <i>tablespace</i>)	<i>tablespace</i> logging mode
Oracle.TableSpace.Status(<i>dbid</i> , <i>tablespace</i>)	<i>tablespace</i> status
Oracle.TableSpace.TotalBytes(<i>dbid</i> , <i>tablespace</i>)	Total size in bytes of <i>tablespace</i>
Oracle.TableSpace.Type(<i>dbid</i> , <i>tablespace</i>)	<i>tablespace</i> type
Oracle.TableSpace.UsedBytes(<i>dbid</i> , <i>tablespace</i>)	Used bytes in <i>tablespace</i>
Oracle.TableSpace.UsedPct(<i>dbid</i> , <i>tablespace</i>)	Used space percentage in <i>tablespace</i>

24.1.4 Lists

When loaded, Oracle subagent adds the following lists to agent:

List	Description
Oracle.DataFiles(<i>dbid</i>)	All known datafiles in database identified by <i>dbid</i> .
Oracle.DataTags(<i>dbid</i>)	All data tags for database identified by <i>dbid</i> . Used only for internal diagnostics.
Oracle.TableSpaces(<i>dbid</i>)	All known tablespaces in database identified by <i>dbid</i> .

24.1.5 Tables

When loaded, Oracle subagent adds the following tables to agent:

Table	Description
Oracle.DataFiles(<i>dbid</i>)	Datafiles in database identified by <i>dbid</i> .
Oracle.Sessions(<i>dbid</i>)	Open sessions in database identified by <i>dbid</i> .
Oracle.TableSpaces(<i>dbid</i>)	Tablespaces in database identified by <i>dbid</i> .

24.2 DB2

NetXMS subagent for DB2 monitoring is designed to provide a way to extract various metrics known as Data Collection Items (DCI) from an instance or several instances of DB2 database.

24.2.1 Configuration

DB2 subagent configuration is specified in agent configuration file (nxagentd.conf). Configuration can be done in two ways, the first one would be a simple INI file and the second one would be an XML configuration file. Please note that to use the XML configuration, you first need to declare the XML file in the DB2 section of the INI configuration file. The details are below.

Database definition supports the following parameters:

Parameter	Format	Description	Default value
DBName	string	The name of the database to connect to	
DBAlias	string	The alias of the database to connect to	
UserName	string	The name of the user for the database to connect to	
Password	string	The password for the database to connect to. When using INI format, remember to enclose password in double quotes ("password") if it contains # character. This parameter automatically detects and accepts password encrypted with <i>nxencpasswd</i> tool.	
Encrypted-Password	string	Database user password encrypted with <i>nxencpasswd</i> tool. DEPRECATED. Use Password instead.	
QueryInterval	seconds	The interval to perform queries with	60
ReconnectInterval	seconds	The interval to try to reconnect to the database if the connection was lost or could not be established	30

Sample DB2 subagent configuration file in INI format:

```
SubAgent      = db2.nsm

[DB2]
DBName        = dbname
DBAlias       = dbalias
UserName      = dbuser
Password      = "mypass123"
QueryInterval = 60
ReconnectInterval = 30
```

XML configuration allows the monitoring of several database instances.

To be able to use the XML configuration file, you first need to specify the file to use in the DB2 section of the INI file. The syntax is as follows:

```
SubAgent      = db2.nsm

[DB2]
ConfigFile    = /myhome/configs/db2.xml
```

Parameter	Format	Description	Default value
ConfigFile	string	The path to the XML configuration file	

The XML configuration file itself should look like this:

```
<config>
  <db2sub>
    <db2 id="1">
      <dbname>dbname</dbname>
      <dbalias>dbalias</dbalias>
      <username>dbuser</username>
      <password>mypass123</password>
      <queryinterval>60</queryinterval>
```

(continues on next page)

(continued from previous page)

```

        <reconnectinterval>30</reconnectinterval>
    </db2>
    <db2 id="2">
        <dbname>dbname1</dbname>
        <dbalias>dbalias1</dbalias>
        <username>dbuser1</username>
        <password>mypass456</password>
        <queryinterval>60</queryinterval>
        <reconnectinterval>30</reconnectinterval>
    </db2>
</db2sub>
</config>

```

As you can see, the parameters are the same as the ones from the INI configuration. Each database declaration must be placed under the `db2sub` tag and enclosed in the `db2` tag. The `db2` tag must have a numerical id which has to be a positive integer greater than 0.

Provided metrics

To get a DCI from the subagent, you need to specify the id from the `db2` entry in the XML configuration file (in case of INI configuration, the id will be **1**). To specify the id, you need to add it enclosed in brackets to the name of the metric that is being requested (e.g., `db2.metric.to.request(**1**)`). In the example, the metric `db2.metric.to.request` from the database with the id **1** will be returned.

Parameter	Arguments	Return type	Description
DB2.Instance.Version(*)	Database id	DCI_DT_STR	DBMS version
DB2.Table.Available(*)	Database id	DCI_DT_INT	The number of available tables
DB2.Table.Unavailable(*)	Database id	DCI_DT_INT	The number of unavailable tables
DB2.Table.Data.LogicalSize(*)	Database id	DCI_DT_INT	Data object logical size in kilobytes
DB2.Table.Data.PhysicalSize(*)	Database id	DCI_DT_INT	Data object physical size in kilobytes
DB2.Table.Index.LogicalSize(*)	Database id	DCI_DT_INT	Index object logical size in kilobytes
DB2.Table.Index.PhysicalSize(*)	Database id	DCI_DT_INT	Index object physical size in kilobytes
DB2.Table.Long.LogicalSize(*)	Database id	DCI_DT_INT	Long object logical size in kilobytes
DB2.Table.Long.PhysicalSize(*)	Database id	DCI_DT_INT	Long object physical size in kilobytes
DB2.Table.Lob.LogicalSize(*)	Database id	DCI_DT_INT	LOB object logical size in kilobytes
DB2.Table.Lob.PhysicalSize(*)	Database id	DCI_DT_INT	LOB object physical size in kilobytes
DB2.Table.Xml.LogicalSize(*)	Database id	DCI_DT_INT	XML object logical size in kilobytes
DB2.Table.Xml.PhysicalSize(*)	Database id	DCI_DT_INT	XML object physical size in kilobytes
DB2.Table.Index.Type1(*)	Database id	DCI_DT_INT	The number of tables using type-1 indexes
DB2.Table.Index.Type2(*)	Database id	DCI_DT_INT	The number of tables using type-2 indexes
DB2.Table.Reorg.Pending(*)	Database id	DCI_DT_INT	The number of tables pending reorganization
DB2.Table.Reorg.Aborted(*)	Database id	DCI_DT_INT	The number of tables in aborted reorganization state
DB2.Table.Reorg.Executing(*)	Database id	DCI_DT_INT	The number of tables in executing reorganization state
DB2.Table.Reorg.Null(*)	Database id	DCI_DT_INT	The number of tables in null reorganization state
DB2.Table.Reorg.Paused(*)	Database id	DCI_DT_INT	The number of tables in paused reorganization state
DB2.Table.Reorg.Alters(*)	Database id	DCI_DT_INT	The number of reorg recommend alter operations
DB2.Table.Load.InProgress(*)	Database id	DCI_DT_INT	The number of tables with load in progress status
DB2.Table.Load.Pending(*)	Database id	DCI_DT_INT	The number of tables with load pending status
DB2.Table.Load.Null(*)	Database id	DCI_DT_INT	The number of tables with load status neither in progress nor pending
DB2.Table.ReadOnly(*)	Database id	DCI_DT_INT	The number of tables in Read Access Only state

continues on next page

Table 2 – continued from previous page

Parameter	Arguments	Return type	Description
DB2.Table.NoLoadRestart(*)	Database id	DCI_DT_INT	The number of tables in a state that won't allow a load restart
DB2.Table.Index.Rebuild(*)	Database id	DCI_DT_INT	The number of tables with indexes that require rebuild
DB2.Table.Rid.Large(*)	Database id	DCI_DT_INT	The number of tables that use large row IDs
DB2.Table.Rid.Usual(*)	Database id	DCI_DT_INT	The number of tables that don't use large row IDs
DB2.Table.Rid.Pending(*)	Database id	DCI_DT_INT	The number of tables that use large row IDs but not all indexes have been rebuilt yet
DB2.Table.Slot.Large(*)	Database id	DCI_DT_INT	The number of tables that use large slots
DB2.Table.Slot.Usual(*)	Database id	DCI_DT_INT	The number of tables that don't use large slots
DB2.Table.Slot.Pending(*)	Database id	DCI_DT_INT	The number of tables that use large slots but there has not yet been an offline table reorganization or table truncation operation
DB2.Table.DictSize(*)	Database id	DCI_DT_INT	Size of the dictionary in bytes
DB2.Table.Scans(*)	Database id	DCI_DT_INT	The number of scans on all tables
DB2.Table.Row.Read(*)	Database id	DCI_DT_INT	The number of reads on all tables
DB2.Table.Row.Inserted(*)	Database id	DCI_DT_INT	The number of insertions attempted on all tables
DB2.Table.Row.Updated(*)	Database id	DCI_DT_INT	The number of updates attempted on all tables
DB2.Table.Row.Deleted(*)	Database id	DCI_DT_INT	The number of deletes attempted on all tables
DB2.Table.Overflow.Accesses	Database id	DCI_DT_INT	The number of r/w operations on overflowed rows of all tables
DB2.Table.Overflow.Creates(*)	Database id	DCI_DT_INT	The number of overflowed rows created on all tables
DB2.Table.Reorg.Page(*)	Database id	DCI_DT_INT	The number of page reorganizations executed for all tables
DB2.Table.Data.LogicalPages	Database id	DCI_DT_INT	The number of logical pages used on disk by data
DB2.Table.Lob.LogicalPages	Database id	DCI_DT_INT	The number of logical pages used on disk by LOBs
DB2.Table.Long.LogicalPages	Database id	DCI_DT_INT	The number of logical pages used on disk by long data
DB2.Table.Index.LogicalPages	Database id	DCI_DT_INT	The number of logical pages used on disk by indexes
DB2.Table.Xda.LogicalPages	Database id	DCI_DT_INT	The number of logical pages used on disk by XDA (XML storage object)
DB2.Table.Row.NoChange(*)	Database id	DCI_DT_INT	The number of row updates that yielded no changes
DB2.Table.Lock.WaitTime(*)	Database id	DCI_DT_INT	The total elapsed time spent waiting for locks (ms)
DB2.Table.Lock.WaitTimeGlob	Database id	DCI_DT_INT	The total elapsed time spent on global lock waits (ms)
DB2.Table.Lock.Waits(*)	Database id	DCI_DT_INT	The total amount of locks occurred
DB2.Table.Lock.WaitsGlob(*)	Database id	DCI_DT_INT	The total amount of global locks occurred
DB2.Table.Lock.EscalsGlob(*)	Database id	DCI_DT_INT	The number of lock escalations on a global lock
DB2.Table.Data.Sharing.Share	Database id	DCI_DT_INT	The number of fully shared tables
DB2.Table.Data.Sharing.Beco	Database id	DCI_DT_INT	The number of tables being in the process of becoming shared
DB2.Table.Data.Sharing.NotS	Database id	DCI_DT_INT	The number of tables not being shared
DB2.Table.Data.Sharing.Beco	Database id	DCI_DT_INT	The number of tables being in the process of becoming not shared
DB2.Table.Data.Sharing.Rem	Database id	DCI_DT_INT	The number of exits from the NOT_SHARED data sharing state
DB2.Table.Data.Sharing.Rem	Database id	DCI_DT_INT	The time spent on waiting for a table to become shared
DB2.Table.DirectWrites(*)	Database id	DCI_DT_INT	The number of write operations that don't use the buffer pool
DB2.Table.DirectWriteReqs(*)	Database id	DCI_DT_INT	The number of request to perform a direct write operation
DB2.Table.DirectRead(*)	Database id	DCI_DT_INT	The number of read operations that don't use the buffer pool

continues on next page

Table 2 – continued from previous page

Parameter	Arguments	Return type	Description
DB2.Table.DirectReadReqs(*)	Database id	DCI_DT_INT	The number of request to perform a direct read operation
DB2.Table.Data.LogicalReads	Database id	DCI_DT_INT	The number of data pages that are logically read from the buffer pool
DB2.Table.Data.PhysicalRead	Database id	DCI_DT_INT	The number of data pages that are physically read
DB2.Table.Data.Gbp.LogicalF	Database id	DCI_DT_INT	The number of times that a group buffer pool (GBP) page is requested from the GBP
DB2.Table.Data.Gbp.Physical	Database id	DCI_DT_INT	The number of times that a group buffer pool (GBP) page is read into the local buffer pool (LBP)
DB2.Table.Data.Gbp.InvalidP	Database id	DCI_DT_INT	The number of times that a group buffer pool (GBP) page is requested from the GBP when the version stored in the local buffer pool (LBP) is invalid
DB2.Table.Data.Lbp.PagesFou	Database id	DCI_DT_INT	The number of times that a data page is present in the local buffer pool (LBP)
DB2.Table.Data.Lbp.IndepPag	Database id	DCI_DT_INT	The number of group buffer pool (GBP) independent pages found in a local buffer pool (LBP)
DB2.Table.Xda.LogicalReads	Database id	DCI_DT_INT	The number of data pages for XML storage objects (XDA) that are logically read from the buffer pool
DB2.Table.Xda.PhysicalReads	Database id	DCI_DT_INT	The number of data pages for XML storage objects (XDA) that are physically read
DB2.Table.Xda.Gbp.LogicalR	Database id	DCI_DT_INT	The number of times that a data page for an XML storage object (XDA) is requested from the group buffer pool (GBP)
DB2.Table.Xda.Gbp.PhysicalF	Database id	DCI_DT_INT	The number of times that a group buffer pool (GBP) dependent data page for an XML storage object (XDA) is read into the local buffer pool (LBP)
DB2.Table.Xda.Gbp.InvalidPa	Database id	DCI_DT_INT	The number of times that a page for an XML storage objects (XDA) is requested from the group buffer pool (GBP) because the version in the local buffer pool (LBP) is invalid
DB2.Table.Xda.Lbp.PagesFou	Database id	DCI_DT_INT	The number of times that an XML storage objects (XDA) page is present in the local buffer pool (LBP)
DB2.Table.Xda.Gbp.IndepPag	Database id	DCI_DT_INT	The number of group buffer pool (GBP) independent XML storage object (XDA) pages found in the local buffer pool (LBP)
DB2.Table.DictNum(*)	Database id	DCI_DT_INT	The number of page-level compression dictionaries created or recreated
DB2.Table.StatsRowsModified	Database id	DCI_DT_INT	The number of rows modified since the last RUN-STATS
DB2.Table.ColObjectLogicalF	Database id	DCI_DT_INT	The number of logical pages used on disk by column-organized data
DB2.Table.Organization.Rows	Database id	DCI_DT_INT	The number of tables with row-organized data
DB2.Table.Organization.Cols	Database id	DCI_DT_INT	The number of tables with column-organized data
DB2.Table.Col.LogicalReads	Database id	DCI_DT_INT	The number of column-organized pages that are logically read from the buffer pool
DB2.Table.Col.PhysicalReads	Database id	DCI_DT_INT	The number of column-organized pages that are physically read
DB2.Table.Col.Gbp.LogicalR	Database id	DCI_DT_INT	The number of times that a group buffer pool (GBP) dependent column-organized page is requested from the GBP

continues on next page

Table 2 – continued from previous page

Parameter	Arguments	Return type	Description
DB2.Table.Col.Gbp.PhysicalR	Database id	DCI_DT_INT	The number of times that a group buffer pool (GBP) dependent column-organized page is read into the local buffer pool (LBP) from disk
DB2.Table.Col.Gbp.InvalidPa	Database id	DCI_DT_INT	The number of times that a column-organized page is requested from the group buffer pool (GBP) when the page in the local buffer pool (LBP) is invalid
DB2.Table.Col.Lbp.PagesFou	Database id	DCI_DT_INT	The number of times that a column-organized page is present in the local buffer pool (LBP)
DB2.Table.Col.Gbp.IndepPag	Database id	DCI_DT_INT	The number of group buffer pool (GBP) independent column-organized pages found in the local buffer pool (LBP)
DB2.Table.ColsReferenced(*)	Database id	DCI_DT_INT	The number of columns referenced during the execution of a section for an SQL statement
DB2.Table.SectionExecutions	Database id	DCI_DT_INT	The number of section executions that referenced columns in tables using a scan

24.3 MongoDB

NetXMS subagent for MongoDB monitoring. Monitors one or more instances of MongoDB databases and reports various database-related metrics.

All metrics available from MongoDB subagent gathered or calculated once per minute thus it's recommended to set DCI poll interval for these items to 60 seconds or more. It is supposed that only databases with same version are monitored by one agent.

24.3.1 Building mongodb subagent

Use `--with-mongodb=/path/to/mongoc driver` parameter to include MongoDB subagent in build. Was tested with `mongo-c-driver-1.1.0`.

24.3.2 Agent Start

While start of subagent at least one database should be up and running. Otherwise subagent will not start. On start subagent requests `serverStatus` to get list of possible DCI. This list may vary from version to version of MongoDB.

24.3.3 Configuration file

24.3.4 Metrics

There are 2 types of metrics: `serverStatus` metrics, that are generated from response on a subagent start and predefined for database status.

Description of `serverStatus` metrics can be found there: [serverStatus](#). In this type of DCI should be given id of server from where the metric should be taken.

Description of database status metrics can be found there: [dbStats](#).

Metric	Description
Mon-goDB.collectionsNum(<i>id,databaseName</i>)	Contains a count of the number of collections in that database.
Mon-goDB.objectsNum(<i>id,databaseName</i>)	Contains a count of the number of objects (i.e. documents) in the database across all collections.
Mon-goDB.avgObjSize(<i>id,databaseName</i>)	The average size of each document in bytes.
Mon-goDB.dataSize(<i>id,databaseName</i>)	The total size in bytes of the data held in this database including the padding factor.
Mon-goDB.storageSize(<i>id,databaseName</i>)	The total amount of space in bytes allocated to collections in this database for document storage.
Mon-goDB.numExtents(<i>id,databaseName</i>)	Contains a count of the number of extents in the database across all collections.
Mon-goDB.indexesNum(<i>id,databaseName</i>)	Contains a count of the total number of indexes across all collections in the database.
Mon-goDB.indexSize(<i>id,databaseName</i>)	The total size in bytes of all indexes created on this database.
Mon-goDB.fileSize(<i>id,databaseName</i>)	The total size in bytes of the data files that hold the database.
Mon-goDB.nsSizeMB(<i>id,databaseName</i>)	The total size of the namespace files (i.e. that end with .ns) for this database.

24.3.5 List

Metric	Description
MongoDB.ListDatabases(<i>id</i>)	Returns list of databases existing on this server

24.4 Informix

NetXMS subagent for Informix (further referred to as Informix subagent) monitors one or more Informix databases and reports database-related metrics.

All metrics available from Informix subagent are collected or calculated once per minute, thus its recommended to set DCI poll interval for these items to 60 seconds or more. All metrics are obtained or derived from the data available in Informix system catalogs. Informix subagent does not monitor any of the metrics related to lower level database layers, such as database processes. Monitoring of such metrics can be achieved through the standard NetXMS functionality.

24.4.1 Pre-requisites

A database user must have access rights to Informix system catalog tables.

24.4.2 Configuration

You can specify multiple databases in the [informix] section of agent configuration file. Each database description must be surrounded by database tags with the id attribute. Id can be any unique integer, it instructs the Informix subagent about the order in which database sections will be processed.

Each database definition supports the following parameters:

Parameter	Description
Id	Database identifier. It will be used to address this database in parameters.
DBName	Database name. This is a name of Informix DSN.
DBServer	Name of the Informix server.
DBLogin	User name for connecting to database.
DBPassword	The password for the database to connect to. When using INI format, remember to enclose password in double quotes ("password") if it contains # character. This parameter automatically detects and accepts password encrypted with <i>nxencpasswd</i> tool.

Configuration example in INI format:

```
Subagent=informix.nsm

[informix]
ID=db1
DBName = instance1
DBLogin = user
DBPassword = "password"
```

Configuration example in XML format:

```
<config>
  <agent>
    <subagent>informix.nsm</subagent>
  </agent>
  <informix>
    <databases>
      <database id="1">
        <id>DB1</id>
        <DBName>TEST</DBName>
        <DBLogin>NXMONITOR</DBLogin>
        <DBPassword>NXMONITOR</DBPassword>
      </database>
      <database id="2">
        <id>DB2</id>
        <DBName>PROD</DBName>
        <DBLogin>NETXMS</DBLogin>
        <DBPassword>PASSWORD</DBPassword>
      </database>
    </databases>
  </informix>
</config>
```

Provided metrics

To get a metric from the subagent, you need to specify the id from the `informix` entry in configuration file. To specify the id, you need to add it enclosed in brackets to the name of the metric that is being requested (e.g., `informix.metric.to.request(**1**)`). In the example, the metric `informix.metric.to.request` from the database with the id **1** will be returned.

Metric	Arguments	Return type	Description
Informix.Session.Count(*)	Database id	DCI_DT_INT	Number of sessions opened
In-formix.Database.Owner(*)	Database id	DCI_DT_STR	The database creation date
In-formix.Database.Logged(*)	Database id	DCI_DT_INT	Returns 1 if the database is logged, 0 - otherwise
In-formix.Dbspace.Pages.PageSize	Database id	DCI_DT_INT	A size of a dbspace page in bytes
In-formix.Dbspace.Pages.PageSize	Database id	DCI_DT_INT	A number of pages used in the dbspace
In-formix.Dbspace.Pages.Free(*)	Database id	DCI_DT_INT	A number of free pages in the dbspace
In-formix.Dbspace.Pages.FreePe	Database id	DCI_DT_INT	Percentage of free space in the dbspace

24.5 MySQL

NetXMS subagent for MySQL monitoring. Monitors one or more instances of MySQL databases and reports various database-related metrics.

MySQL subagent requires MySQL driver to be available in the system.

24.5.1 Configuration

Configuration of MySQL subagent is done in agent configuration file (nxagentd.conf). One or multiple MySQL server instances can be specified. In case of single database definition simply set all required parameters under `[mysql]` section. In multi database configuration define each database under `mysql/databases/<name>` section with unique `<name>` for each database. If no id provided `<name>` of the section will be used as a database id.

Each database definition supports the following parameters:

Parameter	Description	Default value
Id	Database identifier. It will be used to address this database in parameters.	localdb - for single DB definition; last part of section name - for multi database definition
Database	Database name. This is a name of MySQL DSN.	information_schema
Server	Name or IP of the MySQL server.	127.0.0.1
ConnectionTTL	Time in seconds. When this time gets elapsed, connection to the DB is closed and reopened again.	3600
Login	User name for connecting to database.	netxms
Password	Database user password. When using INI format, remember to enclose password in double quotes ("password") if it contains # character. This parameter automatically detects and accepts password encrypted with <i>nxencpasswd</i> tool.	

Single database configuration example:

```
Subagent=mysql.nsm
```

(continues on next page)

(continued from previous page)

```
[mysql]
Id=db1
Database = instance1
Login = user
Password = password
```

Multi database configuration example:

```
Subagent=mysql.nsm

[mysql/databases/somedatabase]
Database = instance1
Login = user
Password = password
Server = netxms.demo

[mysql/databases/local]
Database = information_schema
Login = user
Password = encPassword
Server = 127.0.0.1
```

24.5.2 Provided metrics

Metric	Description
MySQL.Connections.Aborted(<i>id</i>)	aborted connections
MySQL.Connections.BytesReceived(<i>i</i>	bytes received from all clients
MySQL.Connections.BytesSent(<i>id</i>)	bytes sent to all clients
MySQL.Connections.Current(<i>id</i>)	number of active connections
MySQL.Connections.CurrentPerc(<i>id</i>)	connection pool usage (%)
MySQL.Connections.Failed(<i>id</i>)	failed connection attempts
MySQL.Connections.Limit(<i>id</i>)	maximum possible number of simultaneous connections
MySQL.Connections.Max(<i>id</i>)	maximum number of simultaneous connections
MySQL.Connections.MaxPerc(<i>id</i>)	maximum connection pool usage (%)
MySQL.Connections.Total(<i>id</i>)	cumulative connection count
MySQL.InnoDB.BufferPool.Dirty(<i>id</i>)	InnoDB used buffer pool space in dirty pages
MySQL.InnoDB.BufferPool.DirtyPerc	InnoDB used buffer pool space in dirty pages (%)
MySQL.InnoDB.BufferPool.Free(<i>id</i>)	InnoDB free buffer pool space
MySQL.InnoDB.BufferPool.FreePerc	InnoDB free buffer pool space (%)
MySQL.InnoDB.BufferPool.Size(<i>id</i>)	InnoDB buffer pool size
MySQL.InnoDB.BufferPool.Used(<i>id</i>)	InnoDB used buffer pool space
MySQL.InnoDB.BufferPool.UsedPerc	InnoDB used buffer pool space (%)
MySQL.InnoDB.DiskReads(<i>id</i>)	InnoDB disk reads
MySQL.InnoDB.ReadCacheHitRatio(<i>i</i>	InnoDB read cache hit ratio (%)
MySQL.InnoDB.ReadRequest(<i>id</i>)	InnoDB read requests
MySQL.InnoDB.WriteRequest(<i>id</i>)	InnoDB write requests
MySQL.IsReachable(<i>id</i>)	is database reachable
MySQL.MyISAM.KeyCacheFree(<i>id</i>)	MyISAM key cache free space
MySQL.MyISAM.KeyCacheFreePerc	MyISAM key cache free space (%)

continues on next page

Table 3 – continued from previous page

Metric	Description
MySQL.MyISAM.KeyCacheReadHitl	MyISAM key cache read hit ratio (%)
MySQL.MyISAM.KeyCacheSize(<i>id</i>)	MyISAM key cache size
MySQL.MyISAM.KeyCacheUsed(<i>id</i>)	MyISAM key cache used space
MySQL.MyISAM.KeyCacheUsedPer	MyISAM key cache used space (%)
MySQL.MyISAM.KeyCacheWriteHit	MyISAM key cache write hit ratio (%)
MySQL.MyISAM.KeyDiskReads(<i>id</i>)	MyISAM key cache disk reads
MySQL.MyISAM.KeyDiskWrites(<i>id</i>)	MyISAM key cache disk writes
MySQL.MyISAM.KeyReadRequests(MyISAM key cache read requests
MySQL.MyISAM.KeyWriteRequests(MyISAM key cache write requests
MySQL.OpenFiles.Current(<i>id</i>)	open files
MySQL.OpenFiles.CurrentPerc(<i>id</i>)	open file pool usage (%)
MySQL.OpenFiles.Limit(<i>id</i>)	maximum possible number of open files
MySQL.Queries.Cache.HitRatio(<i>id</i>)	query cache hit ratio (%)
MySQL.Queries.Cache.Hits(<i>id</i>)	query cache hits
MySQL.Queries.Cache.Size(<i>id</i>)	query cache size
MySQL.Queries.ClientsTotal(<i>id</i>)	number of queries executed by clients
MySQL.Queries.Delete(<i>id</i>)	number of DELETE queries
MySQL.Queries.DeleteMultiTable(<i>id</i>)	number of multitable DELETE queries
MySQL.Queries.Insert(<i>id</i>)	number of INSERT queries
MySQL.Queries.Select(<i>id</i>)	number of SELECT queries
MySQL.Queries.Slow(<i>id</i>)	slow queries
MySQL.Queries.SlowPerc(<i>id</i>)	slow queries (%)
MySQL.Queries.Total(<i>id</i>)	number of queries
MySQL.Queries.Update(<i>id</i>)	number of UPDATE queries
MySQL.Queries.UpdateMultiTable(<i>id</i>)	number of multitable UPDATE queries
MySQL.Server.Uptime(<i>id</i>)	server uptime
MySQL.Sort.MergePasses(<i>id</i>)	sort merge passes
MySQL.Sort.MergeRatio(<i>id</i>)	sort merge ratio (%)
MySQL.Sort.Range(<i>id</i>)	number of sorts using ranges
MySQL.Sort.Scan(<i>id</i>)	number of sorts using table scans
MySQL.Tables.Fragmented(<i>id</i>)	fragmented tables
MySQL.Tables.Open(<i>id</i>)	open tables
MySQL.Tables.OpenLimit(<i>id</i>)	maximum possible number of open tables
MySQL.Tables.OpenPerc(<i>id</i>)	table open cache usage (%)
MySQL.Tables.Opened(<i>id</i>)	tables that have been opened
MySQL.TempTables.Created(<i>id</i>)	temporary tables created
MySQL.TempTables.CreatedOnDisk(temporary tables created on disk
MySQL.TempTables.CreatedOnDiskF	temporary tables created on disk (%)
MySQL.Threads.CacheHitRatio(<i>id</i>)	thread cache hit ratio (%)
MySQL.Threads.CacheSize(<i>id</i>)	thread cache size
MySQL.Threads.Created(<i>id</i>)	threads created
MySQL.Threads.Running(<i>id</i>)	threads running

24.6 PostgreSQL

NetXMS subagent for PostgreSQL monitoring. Monitors one or more instances of PostgreSQL servers and reports various database-related metrics.

PostgreSQL subagent requires PostgreSQL driver to be available in the system.

24.6.1 Pre-requisites

A PostgreSQL user with **CONNECT** right to at least one database on the server.

If the **PostgreSQL.DatabaseSize** metric should be monitored the user must have the **CONNECT** right to other databases on the server too.

Starting from the PostgreSQL version 10, the user must have the role **pg_monitor** assigned. Required role can be assigned to user with the following query:

```
GRANT pg_monitor TO user;
```

Where *user* is the user configured in PostgreSQL subagent for database access.

24.6.2 Configuration

Configuration of PostgreSQL subagent is done in agent configuration file (nxagentd.conf). One or multiple PostgreSQL server instances can be specified. In case of single server definition simply set all required parameters under `[pgsql]` section. In multi server configuration define each server instance under `pgsql/servers/<name>` section with unique `<name>` for each server. If no id provided `<name>` of the section will be used as a server id.

It is not necessary to configure connections to more than one database on the same PostgreSQL server instance.

Each server definition supports the following parameters:

Parameter	Description	Default value
Id	Server identifier. It will be used to address this server connection in parameters.	localdb - for single server definition last part of section name - for multi server definition
Database	Maintenance database name. This is a name of the database on the server the subagent is connected to.	postgres
Server	Name or IP of the PostgreSQL server. If the sever uses differnt than default port (5432) the <i>.port</i> must be added to the server name or IP.	127.0.0.1
ConnectionTTL	Time in seconds. When this time gets elapsed, connection to the DB is closed and reopened again.	3600
Login	User name for connecting to database.	netxms
Password	Database user password. When using INI format, remember to enclose password in double quotes ("password") if it contains # character. This parameter automatically detects and accepts password encrypted with <i>nxencpasswd</i> tool.	

Single server configuration example:

```
Subagent=pgsql.nsm

[pgsql]
Id=production
Server = 10.0.3.5
Database = database1
Login = user
Password = password
```

Multi server configuration example:

```
Subagent=pgsql.nsm

[pgsql/servers/production]
Server = 10.0.3.5
Database = database1
Login = user
Password = password

[pgsql/servers/testing]
Server = 10.0.3.6
Database = test_database
Login = user
Password = password
```

24.6.3 Provided Metrics

When loaded, PostgreSQL subagent adds two types of metrics to the agent.

Database server metrics are common for all databases on the server. These metrics require one argument which is server id from the configuration.

Database metrics are independent for each database on the server. These metrics require two arguments. The first one is server id from the configuration the second one is name of the database. If the second argument is missing the name of the maintenance database from the configuration is used.

Alternatively, these two arguments can be specified as one argument in following format: *datanase_name@server_id*. This format is returned by the PostgreSQL.AllDatabases list.

Following table shows the database server metrics:

Metric	Type	Description
PostgreSQL.IsReachable(<i>id</i>)	String	Is database server instance reachable
PostgreSQL.Version(<i>id</i>)	String	Database server version
PostgreSQL.Archiver.ArchivedCount(<i>id</i>)	Integer 64-bit	Number of WAL files that have been successfully archived
PostgreSQL.Archiver.FailedCount(<i>id</i>)	Integer 64-bit	Number of failed attempts for archiving WAL files
PostgreSQL.Archiver.IsArchiving(<i>id</i>)	String	Is archiving running
PostgreSQL.Archiver.LastArchivedAge(<i>id</i>)	Integer	Age of the last successful archive operation
PostgreSQL.Archiver.LastArchivedWAL(<i>id</i>)	String	Name of the last WAL file successfully archived
PostgreSQL.Archiver.LastFailedAge(<i>id</i>)	Integer	Age of the last failed archival operation
PostgreSQL.Archiver.LastFailedWAL(<i>id</i>)	String	Name of the WAL file of the last failed archival operation
PostgreSQL.BGWriter.BuffersAllocated(<i>id</i>)	Integer 64-bit	Cumulative number of buffers allocated
PostgreSQL.BGWriter.BuffersBackends(<i>id</i>)	Integer 64-bit	Cumulative number of buffers written directly by a backend

continues on next page

Table 4 – continued from previous page

Metric	Type	Description
PostgreSQL.BGWriter.BuffersBacken	Integer 64-bit	Cumulative number of times a backend had to execute its own fsync call
PostgreSQL.BGWriter.BuffersClean(<i>i</i>	Integer 64-bit	Cumulative number of buffers written by the background writer
PostgreSQL.BGWriter.BuffersCheckp	Integer 64-bit	Cumulative number of buffers written during checkpoints
PostgreSQL.BGWriter.CheckpointsR	Integer 64-bit	Cumulative number of requested checkpoints that have been performed
PostgreSQL.BGWriter.CheckpointsT	Integer 64-bit	Cumulative number of scheduled checkpoints that have been performed
PostgreSQL.BGWriter.CheckpointSy	Float	Total amount of time that has been spent in the portion of checkpoint processing where files are synchronized to disk, in milliseconds
PostgreSQL.BGWriter.CheckpointW	Float	Total amount of time that has been spent in the portion of checkpoint processing where files are written to disk, in milliseconds
PostgreSQL.BGWriter.MaxWrittenCl	Integer 64-bit	Cumulative number of times the background writer stopped a cleaning scan because it had written too many buffers
PostgreSQL.GlobalConnections.Auto	Integer	Maximal number of autovacuum backends
PostgreSQL.GlobalConnections.Total	Integer	Total number of connections
PostgreSQL.GlobalConnections.Total	Integer	Maximal number of connections
PostgreSQL.GlobalConnections.Total	Integer	Used connections (%)
PostgreSQL.Replication.InRecovery(<i>i</i>	String	Is recovery in progress (from version 9.6.0)
PostgreSQL.Replication.IsReceiver(<i>ia</i>	String	Is the server WAL receiver
PostgreSQL.Replication.Lag(<i>id</i>)	Integer	Replication lag in seconds (from version 10.0)
PostgreSQL.Replication.LagBytes(<i>id</i>)	Float	Replication lag in bytes (from version 10.0)
PostgreSQL.Replication.WALSenders	Integer 64-bit	Number of WAL senders
PostgreSQL.Replication.WALFiles(<i>id</i>	Integer 64-bit	Number of the WAL files (from version 10.0)
PostgreSQL.Replication.WALSize(<i>id</i>	Float	Size of the WAL files (from version 10.0)

Following table shows the database metrics:

Metric	Type	Description
PostgreSQL.DBConnections.Active(<i>ic</i> * <i>database</i>])	Integer	Number of backends for this database executing a query
PostgreSQL.DBConnections.Autovaci * <i>database</i>])	Integer	Number of autovacuum backends for this database

continues on next page

Table 5 – continued from previous page

Metric	Type	Description
PostgreSQL.DBCConnections.Fastpath(*database])	Integer	Number of backends for this database executing a fast-path function
PostgreSQL.DBCConnections.Idle(id*[,*database])	Integer	Number of backends for this database waiting for a new client command
PostgreSQL.DBCConnections.IdleInTr(*database])	Integer	Number of backends for this database in a transaction, but is not currently executing a query
PostgreSQL.DBCConnections.IdleInTr(*database])	Integer	Number of backends for this database in a transaction, but is not currently executing a query and one of the statements in the transaction caused an error
PostgreSQL.DBCConnections.OldestXID(*database])	Integer	Age of the oldest XID
PostgreSQL.DBCConnections.Total(id*[,*database])	Integer	Total number of backends for connections to this database
PostgreSQL.DBCConnections.Waiting(*database])	Integer	Number of waiting backends for this database
PostgreSQL.Locks.AccessExclusive(id*[,*database])	Integer 64-bit	Number of AccessExclusive locks for this database
PostgreSQL.Locks.AccessShare(id*[,*database])	Integer 64-bit	Number of AccessShare locks for this database
PostgreSQL.Locks.Exclusive(id*[,*database])	Integer 64-bit	Number of Exclusive locks for this database
PostgreSQL.Locks.RowExclusive(id*[,*database])	Integer 64-bit	Number of RowExclusive locks for this database
PostgreSQL.Locks.RowShare(id*[,*database])	Integer 64-bit	Number of RowShare locks for this database
PostgreSQL.Locks.Share(id*[,*database])	Integer 64-bit	Number of Share locks for this database
PostgreSQL.Locks.ShareRowExclusive(id*[,*database])	Integer 64-bit	Number of ShareRowExclusive locks for this database
PostgreSQL.Locks.ShareUpdateExclusive(id*[,*database])	Integer 64-bit	Number of ShareUpdateExclusive locks for this database
PostgreSQL.Locks.Total(id*[,*database])	Integer 64-bit	Total number of locks for this database
PostgreSQL.Stats.BlkWriteTime(id*[,*database])	Float	Cumulative time spent writing data file blocks by backends in this database, in milliseconds

continues on next page

Table 5 – continued from previous page

Metric	Type	Description
PostgreSQL.Stats.BlockReadTime(<i>id</i> *, * <i>database</i>])	Float	Cumulative time spent reading data file blocks by backends in this database, in milliseconds
PostgreSQL.Stats.BlocksRead(<i>id</i> *, * <i>database</i>])	Integer 64-bit	Cumulative number of disk blocks read in this database
PostgreSQL.Stats.BlocksHit(<i>id</i> *, * <i>database</i>])	Integer 64-bit	Cumulative number of times disk blocks were found already in the buffer cache
PostgreSQL.Stats.CacheHitRatio(<i>id</i> *, * <i>database</i>])	Float	Query cache hit ratio (%)
PostgreSQL.Stats.Conflicts(<i>id</i> *, * <i>database</i>])	Integer 64-bit	Cumulative number of queries canceled due to conflicts with recovery in this database (standby servers only)
PostgreSQL.Stats.DatabaseSize(<i>id</i> *, * <i>database</i>])	Integer 64-bit	Disk space used by the database
PostgreSQL.Stats.Deadlocks(<i>id</i> *, * <i>database</i>])	Integer 64-bit	Cumulative number of deadlocks detected in this database
PostgreSQL.Stats.ChecksumFailures(<i>id</i> *, * <i>database</i>])	Integer 64-bit	Cumulative number of data page checksum failures detected in this database (from version 12.0)
PostgreSQL.Stats.NumBackends(<i>id</i> *, * <i>database</i>])	Integer	Number of backends currently connected to this database
PostgreSQL.Stats.RowsDeleted(<i>id</i> *, * <i>database</i>])	Integer 64-bit	Cumulative number of rows deleted by queries in this database
PostgreSQL.Stats.RowsFetched(<i>id</i> *, * <i>database</i>])	Integer 64-bit	Cumulative number of rows fetched by queries in this database
PostgreSQL.Stats.RowsInserted(<i>id</i> *, * <i>database</i>])	Integer 64-bit	Cumulative number of rows inserted by queries in this database
PostgreSQL.Stats.RowsReturned(<i>id</i> *, * <i>database</i>])	Integer 64-bit	Cumulative number of rows returned by queries in this database
PostgreSQL.Stats.RowsUpdated(<i>id</i> *, * <i>database</i>])	Integer 64-bit	Cumulative number of rows updated by queries in this database
PostgreSQL.Stats.TempBytes(<i>id</i> *, * <i>database</i>])	Integer 64-bit	Total amount of data written to temporary files by queries in this database
PostgreSQL.Stats.TempFiles(<i>id</i> *, * <i>database</i>])	Integer 64-bit	Cumulative number of temporary files created by queries in this database

continues on next page

Table 5 – continued from previous page

Metric	Type	Description
PostgreSQL.Stats.TransactionCommit(<i>*database</i>)	Integer 64-bit	Cumulative number of transactions in this database that have been committed
PostgreSQL.Stats.TransactionRollback(<i>*database</i>)	Integer 64-bit	Cumulative number of transactions in this database that have been rolled back
PostgreSQL.Transactions.Prepared(<i>id *database</i>)	Integer 64-bit	Number of prepared transactions for this database

24.6.4 Lists

When loaded, PostgreSQL subagent adds the following lists to agent:

List	Description
PostgreSQL.DBServers	All configured servers (server ids).
PostgreSQL.Databases(<i>id</i>)	All databases on server identified by <i>id</i> .
PostgreSQL.AllDatabases	All databases on configured servers. The format of the list items is <i>database_name@server_id</i> .
PostgreSQL.DataTags(<i>id</i>)	All data tags for server identified by <i>id</i> . Used only for internal diagnostics.

24.6.5 Tables

When loaded, PostgreSQL subagent adds the following tables to agent:

Table	Description
PostgreSQL.Backends(<i>id</i>)	Connection backends on server identified by <i>id</i> .
PostgreSQL.Locks(<i>id</i>)	Locks on server identified by <i>id</i> .
PostgreSQL.PreparedTransactions(<i>id</i>)	Prepared transactions on server identified by <i>id</i> .

APPLICATION MONITORING

25.1 Process monitoring

Platform subagents support process monitoring. Process metrics have “Process.*” format. Metrics differ between different OS. Detailed description of each metric can be found in *List of supported metrics*.

25.2 Application Database Monitoring

For application database monitoring you can use database monitoring subagents or database query subagents. Information about database monitoring subagents can be found in *Database monitoring*. This chapter discusses only DBQuery subagents configuration and usage.

DBQuery subagent has 2 types of query execution: background - that periodically executes SQL query and provides result and error code as metrics and synchronous, when query is executed by request. Background query, however, can be also executed per request. Synchronously executed query can have parameters that are supplied along with requested metric. SQL queries are specified in the agent configuration or a full query can be supplied via `DB.Query()` metric.

For time consuming SQL requests it is highly recommended to use background execution. Heavy SQL can cause request timeout for synchronous execution.

25.2.1 Configuration file

General configuration parameters related to DBQuery subagent are set in **[DBQUERY]** section of agent's configuration file. The following parameters are supported:

Parameter	Format	Description
AllowEmptyResultSet	yes or no	If set to <code>yes</code> (default), agent returns empty metric value if database returns empty result. If set to <code>no</code> , agents returns error in case if query returns empty result.
Database	Semicolon-separated option list	Database connection information. Deprecated, specify database connection parameters in [DB-QUERY/Databases/id] sections
Query	<i>name:dbid:interval:query</i>	Define query scheduled for background execution. Can be specified multiple times to define multiple queries. Fields in query definition have the following meaning: <ul style="list-style-type: none"> <i>name</i> - Query name which will be used in metrics to retrieve collected data. <i>dbid</i> - Database connection ID <i>interval</i> - Polling interval in seconds. <i>query</i> - SQL query to be executed.
ConfigurableQuery	<i>name:dbid:description:query</i>	Define query for synchronous execution. Can be specified multiple times to define multiple queries. Fields in query definition have the following meaning: <ul style="list-style-type: none"> <i>name</i> - Query name which will be used in metrics to retrieve collected data. <i>dbid</i> - Database connection ID <i>description</i> - Description that will be shown in agents parameter description. <i>query</i> - SQL query to be executed. Bind variables are supported, question mark (?) placeholders in the query will be substituted with parameters supplied along with requested metric.

Database connection parameters are set in separate sections named **[DBQUERY/Databases/id]** where `id` is database connection id used to identify this connection in configuration parameters and agent metrics. The following parameters are supported:

Name	Status	Description
name	optional	Database name
DBDriverOptions	optional	Additional driver-specific parameters
driver	mandatory	Database driver name. Available drivers are: <ul style="list-style-type: none"> • db2 • informix • mssql • mysql • odbc • oracle • pgsql • sqlite
encryptedPassword	optional	Database password in encrypted form (use <i>nxencpasswd</i> command line tool to encrypt passwords). This option takes precedence over password option
login	optional	Login name
password	optional	Database password. Remember to enclose password in double quotes ("password") if it contains # character. This parameter automatically detects and accepts password encrypted with <i>nxencpasswd</i> tool.
server	optional	Database server name or IP address.

25.2.2 Configuration Example

```

MasterServers = netxms.demo
SubAgent = dbquery.nsm

[DBQUERY]
# Query1 will be executed every 60 seconds (be can be also executed on-demand via_
↪metric "query1"):
Query = query1:db1:60:SELECT f1 FROM table1

# Query2 will be executed on demand, one parameter should be supplied along with the_
↪metric
ConfigurableQuery = query2:db1:This query requires one parameter:SELECT f1 FROM_
↪table2 WHERE f2 LIKE ?

[DBQUERY/Databases/db1]
driver=pgsql
server=10.0.0.4
login=netxms
password=netxms1
name=test_db

```

25.2.3 Metrics

When loaded, DBQuery subagent adds the following metrics to agent:

Metric	Description
DB.Query(<i>dbid</i> , <i>query</i>)	Result of immediate execution of the query <i>query</i> in database identified by <i>dbid</i> . Database with given name must be defined in configuration file.
DB.QueryExecutionTime(<i>name</i>)	Last execution duration in milliseconds of the query <i>name</i> . Query with given name must be defined in configuration file. Added in version 4.4.3.
DB.QueryResult(<i>name</i>)	Last result of execution of the query <i>name</i> . Query with given name must be defined in configuration file.
DB.QueryStatus(<i>name</i>)	Status of last execution of the query <i>name</i> . Query with given name must be defined in configuration file. Value returned is native SQL error code.
DB.QueryStatusText(<i>name</i>)	Status of last execution of the query <i>name</i> as a text. Query with given name must be defined in configuration file.
<i>queryName</i>	Result of immediate execution of query <i>queryName</i> defined in agent config file with <code>Query=...</code>
<i>queryName</i> (<i>param1</i> , <i>param2</i> ...)	Result of immediate execution of query <i>queryName</i> defined in agent config file with <code>ConfigurableQuery=...</code> . Optional parameters <i>param1</i> , <i>param2</i> ... will be used as bind variables in the query.

25.2.4 Tables

When loaded, DBQuery subagent adds the following tables to agent:

Table	Description
DB.Query(<i>dbid</i> , <i>query</i>)	Result of immediate execution of the query <i>query</i> in database identified by <i>dbid</i> . Database with given name must be defined in configuration file
DB.QueryResult(<i>name</i>)	Last result of execution of the query <i>name</i> . Query with given name must be defined in configuration file
<i>queryName</i>	Result of immediate execution of query <i>queryName</i> defined in agent config file with <code>Query=...</code>
<i>queryName</i> (<i>param1</i> , <i>param2</i> ...)	Result of immediate execution of query <i>queryName</i> defined in agent config file with <code>ConfigurableQuery=...</code> . Optional parameters <i>param1</i> , <i>param2</i> ... will be used as bind variables in the query.

25.3 Log monitoring

Application logs can be added to monitoring. For log monitoring configuration refer to [Log monitoring](#) chapter.

25.4 External Metrics

It is possible to define External metrics that will get metric data from the script that is executed on the agent. This option can be used to get status from some command line tools or from self made scripts. Information about options and configuration is available in [Agent External Metrics](#) chapter.

ICMP PING

The following options exist to monitor systems using ICMP pings:

- ICMP response statistic collection
- Metrics provided by ping subagent

26.1 ICMP response statistic collection

NetXMS can periodically perform ICMP polls and calculate node availability statistics. This functionality can be controlled globally via the server configuration parameter `ICMP.CollectPollStatistics` or locally on each node. The ICMP polling interval and statistic calculation period (expressed in number of polls), timeout and ICMP packet size are configured via server configuration parameters. See for more details [Server configuration parameters](#).

ICMP requests are sent to the primary IP address of the node. Additional targets can be specified in the node properties. It is also possible to set interfaces of the node as targets by enabling *Collect ICMP response statistic for this interface* in the properties of the interface. Please note that enabling this for the interface that corresponds to the primary IP address will lead to pinging this address twice.

ICMP polling is performed either from the server, from a zone proxy if zoning is used, or from a specific proxy when this is configured in the node properties. The proxying agent should have the `ping.nsm` subagent enabled.

The results of the ICMP response statistic collection for primary IP address are visible in *Object Details -> Overview* and are available as internal metrics:

- `ICMP.ResponseTime.Average`
- `ICMP.PacketLoss`
- `ICMP.ResponseTime.Last`
- `ICMP.ResponseTime.Max`
- `ICMP.ResponseTime.Min`

The results of the ICMP response statistic collection for additional targets and interfaces are available as internal metrics:

- `ICMP.ResponseTime.Average(*)`
- `ICMP.PacketLoss(*)`
- `ICMP.ResponseTime.Last(*)`
- `ICMP.ResponseTime.Max(*)`
- `ICMP.ResponseTime.Min(*)`

For example, the `ICMP.PacketLoss(8.8.8.8)` internal metric will provide packet loss for the target with IP address 8.8.8.8.

No historical data is stored by default. It is necessary to configure DCIs using the above mentioned internal metric to store historical data.

26.2 Ping subagent

This subagent can be used to measure ICMP ping response times from one location to another. When loaded, the PING subagent adds a number of metrics to the agent. Measurements can be either requested by the server or scheduled by the agent itself.

26.2.1 Metrics requested by the server

Metric	Description
<code>Icmp.Ping(target, timeout, psize, dontfragmentflag, retrycount)</code>	<p>ICMP ping response time from <i>target</i>. Agent will send echo request as soon as it receives request for the value of the metric, and will return the response time for that particular request.</p> <p>Arguments:</p> <ul style="list-style-type: none"> • <i>target</i> should be an IP address or hostname. • <i>timeout</i> specifies timeout in milliseconds. This is an optional argument. If omitted, the value from the <i>Timeout</i> configuration parameter will be used. • <i>psize</i> specifies the packet size in bytes including the IP header. This is an optional argument. If omitted, the value from the <i>DefaultPacketSize</i> configuration parameter will be used. • <i>dontfragmentflag</i> defines if the don't fragment flag is set in ICMP requests. This is an optional argument. If omitted, the value from the <i>DefaultDoNotFragmentFlag</i> configuration parameter will be used. • <i>retrycount</i> defines the number of retries. This is an optional argument. If omitted, a default value of 1 is used. <p>Please note that while metrics scheduled by the agent just return the result of the background ping process, this metric waits for actual ping completion and then returns the result. Because of this behavior, it is not recommended to use the Icmp.Ping metric for regular monitoring but instead only for occasional tests. For instant monitoring, you should configure targets for background ping and use the Icmp.AvgPingTime or Icmp.LastPingTime metrics to retrieve results.</p>

26.2.2 Metrics scheduled by the agent

There is a number of metrics that are collected based on the background ping process scheduled by the agent (based on the "PacketRate" parameter).

Targets for these metrics can be either defined in the agent configuration file (using one or more "Target" parameters), or registered automatically on first request from server. If targets are registered automatically, the default packet size is used. The first request to a non-existing target will return "0" as a value. Automatically registered targets are automatically removed after a timeout, when the server stops requesting metrics for that target.

Single-value metrics

Metric	Description
Icmp.AvgPingTime(<i>target</i>)	Average ICMP ping response time from <i>target</i> for the last minute. The argument <i>target</i> can be either an IP address or a name specified in the Target configuration record (see below).
ICMP.MovingAvgPingTime(<i>target</i>)	Moving average of response time from <i>target</i> . Time period for moving average calculation is set by the <i>MovingAverageTimePeriod</i> agent configuration parameter (see below).
Icmp.LastPingTime(<i>target</i>)	Last ICMP ping response time from <i>target</i> .
ICMP.MaxPingTime(<i>target</i>)	Maximum ICMP ping response time from <i>target</i> for the last minute.
ICMP.MinPingTime(<i>target</i>)	Minimum ICMP ping response time from <i>target</i> for the last minute.
ICMP.CumulativeMaxPingTime(<i>target</i>)	Maximum encountered ICMP ping response time from <i>target</i> since that target was added.
ICMP.CumulativeMinPingTime(<i>target</i>)	Minimum encountered ICMP ping response time from <i>target</i> since that target was added.
Icmp.PacketLoss(<i>target</i>)	ICMP ping packet loss (in percents) for <i>target</i> for the last minute.
Icmp.PingStdDev(<i>target</i>)	Standard deviation of the response time for the <i>target</i> for last minute.
ICMP.Jitter(<i>target</i>)	Jitter of ICMP ping response time from <i>target</i> for last minute.
ICMP.MovingAvgJitter(<i>target</i>)	Moving average of response time jitter from <i>target</i> . Time period for moving average calculation is set by <i>MovingAverageTimePeriod</i> agent configuration parameter (see below).

Tables

Table	Description
Icmp.Targets	Table of configured ping targets. Columns: <ul style="list-style-type: none"> • IP address • Last response time (milliseconds) • Average response time (milliseconds) • Minimal response time (milliseconds) • Maximum response time (milliseconds) • Moving average response time (milliseconds) • Standard deviation of response time (milliseconds) • Jitter of response time (milliseconds) • Moving average jitter of response time (milliseconds) • Cumulative minimal response time (milliseconds) • Cumulative maximum response time (milliseconds) • Packet loss (percents) • Configured packet size • Name • DNS name • Automatic

Lists

List	Description
Icmp.Targets	List of configured ping target names

26.2.3 Configuration file

All configuration parameters related to the PING subagent should be placed into **[PING]** section of the configuration file of the agent. The following configuration parameters are supported:

Parameter	Format	Description	Default value
AutoConfigureTargets	<i>boolean</i>	Allow automatic registration of ICMP targets when the metrics for a new target are requested from the server.	yes
DefaultDoNotFragmentFlag	<i>boolean</i>	Default value for the Don't Fragment flag in ICMP requests.	no
DefaultPacketSize	<i>bytes</i>	Set default packet size to <i>bytes</i> .	46
MaxTargetInactivityTime	<i>seconds</i>	Timeout to remove an automatically registered ICMP target if the server stops requesting metrics for that target.	86400
MovingAverageTimePeriod	<i>seconds</i>	Set time period used for the moving average value calculation.	3600
PacketRate	<i>packets</i>	Set ping packet rate per minute. Allowed values are between 1 and 60 and values below or above will be adjusted automatically.	4
Target	<i>ip:name:psize</i>	Add target with IP address <i>ip</i> to the background ping target list and assign an optional name <i>name</i> to it. The target will be pinged using packets of <i>psize</i> bytes. The name and packet size fields are optional and can be omitted. This parameter can be given multiple times to add multiple targets.	<i>none</i>
ThreadPoolMaxSize	<i>threads</i>	Maximal number of threads in the thread pool of the agent that is serving scheduled ICMP measurements.	1024
ThreadPoolMinSize	<i>threads</i>	Minimal number of threads in the thread pool of the agent that is serving scheduled ICMP measurements.	1
Timeout	<i>milliseconds</i>	Set response timeout to <i>milliseconds</i> . Allowed values are between 500 and 5000 and values below or above will be adjusted automatically.	3000

Configuration example:

```
# This sample nxagentd.conf instructs agent to:
# 1. load the PING subagent
# 2. Ping target 10.0.0.1 with default size (46 bytes) packets and 10.0.0.2 with
↪ 1000 bytes packets
# 3. Timeout for ping set to 1 second and pings are sent 12 times per minute (each
↪ 5 seconds)

MasterServers = netxms.demo
SubAgent = ping.nsm

[PING]
Timeout = 1000
PacketRate = 12 # every 5 seconds
```

(continues on next page)

(continued from previous page)

```
Target = 10.0.0.1:target_1  
Target = 10.0.0.2:target_2:1000
```

Note

Response time of 10000 indicates timeout

HARDWARE(SENSOR) MONITORING

NetXMS has subagents that allow to monitor hardware sensors.

- lm-sensors - Can collect data from all sensors that are supported by [lm-sensors](#) drivers on Linux.
- DS18x20 - This subagent collects temperature data from ds18x20 sensors. Linux only.
- RPI - This subagent is created for Raspberry Pi. It can collect data from DHT22 sensor and get the status of any GPIO pin.

27.1 lm-sensors

This subagent can be used to read hardware status using the `lm_sensors` package.

27.1.1 Pre-requisites

The package `lm_sensors` should be installed and configured properly. The output of the `sensors` command should produce meaningful output (see example below).

```
alk@b08s02ur:~$ sensors
w83627dhg-isa-0290
Adapter: ISA adapter
Vcore:      +1.14 V  (min = +0.00 V, max = +1.74 V)
in1:        +1.61 V  (min = +0.05 V, max = +0.01 V)  ALARM
AVCC:       +3.31 V  (min = +2.98 V, max = +3.63 V)
VCC:        +3.31 V  (min = +2.98 V, max = +3.63 V)
in4:        +1.79 V  (min = +1.29 V, max = +0.05 V)  ALARM
in5:        +1.26 V  (min = +0.05 V, max = +1.67 V)
in6:        +0.10 V  (min = +0.26 V, max = +0.08 V)  ALARM
3VSB:       +3.30 V  (min = +2.98 V, max = +3.63 V)
Vbat:       +3.18 V  (min = +2.70 V, max = +3.30 V)
fan1:       3308 RPM (min = 1188 RPM, div = 8)
fan2:       6250 RPM (min = 84375 RPM, div = 8)  ALARM
fan3:        0 RPM  (min = 5273 RPM, div = 128)  ALARM
fan4:        0 RPM  (min = 10546 RPM, div = 128)  ALARM
fan5:        0 RPM  (min = 10546 RPM, div = 128)  ALARM
temp1:      +39.0°C  (high = +4.0°C, hyst = +1.0°C)  ALARM  sensor = diode
temp2:      +17.0°C  (high = +80.0°C, hyst = +75.0°C)  sensor = diode
temp3:     +124.5°C  (high = +80.0°C, hyst = +75.0°C)  ALARM  sensor = thermistor
cpu0_vid:   +2.050 V

coretemp-isa-0000
```

(continues on next page)

(continued from previous page)

```

Adapter: ISA adapter
Core 0:          +37.0°C  (high = +76.0°C, crit = +100.0°C)

coretemp-isa-0001
Adapter: ISA adapter
Core 1:          +37.0°C  (high = +76.0°C, crit = +100.0°C)

```

27.1.2 Parameters

When loaded, the lm_sensors subagent adds the following metrics:

Metric	Description
LM Sensors.Value(<i>chip, label</i>)	Current value returned by hardware sensor

27.1.3 Configuration file

All configuration parameters related to the m_sensors subagent should be placed into the ***LMSSENSORS** section of agent's configuration file. The following configuration parameters are supported:

Parameter	Format	Description	Default value
Use-Fahrenheit	Boolean	If set to "yes", all temperature reading will be converted to Fahrenheit	no
ConfigFile	String	Path to <code>sensors.conf</code>	none, use system default (usually <code>/etc/sensors3.conf</code>)

27.1.4 Configuration example

```

MasterServers = netxms.demo
SubAgent = lmsensors.nsm

[LMSSENSORS]
UseFahrenheit = yes
ConfigFile = /etc/sensors.netxms.conf

```

27.1.5 Sample usage

(based on output of "sensors" from Pre-requisites section)

```

alk@b08s02ur:~$ nxget netxms.demo 'LM Sensors.Value(coretemp-isa-0001,Core 1)'
38.000000
alk@b08s02ur:~$ nxget netxms.demo 'LM Sensors.Value(w83627dhg-isa-0290,AVCC)'
3.312000

```

27.2 DS18x20

This subagent collects temperature from DS18x20 sensor. Subagent available for Linux only. To use this subagent the 1-Wire driver should be installed.

27.2.1 Metrics

Metric	Type	Meaning
Sensor.Temperature(*)	Float	Sensor temperature

27.2.2 Configuration file

All configuration parameters related to the lm_sensors subagent should be placed into the ***DS18X20** section of the configuration file of the agent. The following configuration parameters are supported:

Parameter	Format	Description
Sensor	String	Sensor identification in format sensorName:uniqueID

27.2.3 Configuration example

```
MasterServers = netxms.demo
SubAgent = DS18X20.nsm

[DS18X20]
Sensor = sensorName:uniqueID123456788990
```

27.3 RPI

This subagent collects data from the Raspberry Pi DHT22 sensor as well as the status of GPIO pins.

27.3.1 Metrics

Metric	Type	Meaning
GPIO.PinState(pinNumber)	Integer	State of pin with given number. This pin number should be enabled in the agent configuration file.
Sensors.Humidity	Integer	Sensors data for humidity
Sensors.Temperature	Integer	Sensors data for temperature

27.3.2 Configuration file

All configuration parameters related to the lm_sensors subagent should be placed into the ***RPI** section of the configuration file of the agent. The following configuration parameters are supported:

Parameter	Format	Description
DisableDHT	Boolean	Disables dht22 sensor if <i>yes</i> . By default <i>no</i> .
EnabledPins	Comma separated list of numbers	List of pins that are enabled for status check.

27.3.3 Configuration example

```
MasterServers = netxms.demo
SubAgent = rpi.nsm

[RPI]
DisableDHT22 = no
EnabledPins = 1,4,5,8
```

27.4 MQTT

This is a subagent that can be used to collect data from devices and sensors that use the MQTT protocol for communication. The subagent can be used to connect to existing MQTT brokers, listen to user specified topics, map posted data to metrics and generate events.

There are two ways how to set up data collection for MQTT.

One approach is to specify an MQTT topic - agent metric mapping in agent configuration file. In this case DCIs are created with origin *NetXMS Agent*.

The other approach is to use the *MQTT* origin in DCI properties. The metric has the following format *broker_name:mqtt_topic*, where *broker_name* is name specified in the agent configuration file. The Agent which performs MQTT data collection is selected automatically. If the node is in a zone, the zone proxy is used. If a MQTT proxy is specified in the node properties, that will be used. With this approach there is no need to specify metrics in the agent configuration file - when the server requests mqtt topic for the first time, the agent subscribes to that topic.

27.4.1 Configuration file

These are configuration sections and parameters for the MQTT subagent:

Section	Parameters	Format	Description
[MQTT/Brokers/broker_name]	Hostname, Port, Login, Password	String	This section holds the data needed to connect to the MQTT broker
[MQTT/Brokers/broker_name/Events]	EVENT_NAME	String	This section is optional and allows to specify event that would be generated when MQTT topic gets new value
[MQTT/Brokers/broker_name/Metrics]	Metric Name	Dot separated string	This section is optional and sets mapping of data posted to MQTT topics to agent metrics

27.4.2 Configuration example

```
SubAgent = mqtt.nsm

[MQTT/Brokers/Local]
Hostname = 10.10.10.3
```

27.4.3 Configuration example with metric and event configuration

```
SubAgent = mqtt.nsm

[MQTT/Brokers/Office]
Hostname = mqtt.office.radensolutions.com

[MQTT/Brokers/Office/Events]
MQTT_METERHUB_RAW_DATA = "cmd/5C:CF:7F:25:79:D6/#"

[MQTT/Brokers/Office/Metrics]
MeterHub.Telemetry.RSSI = "tele/5C:CF:7F:25:79:D6/RSSI"
MeterHub.Telemetry.Time = "tele/5C:CF:7F:25:79:D6/TIME"
```

This configuration will connect to an MQTT broker Office at the Hostname. Whenever data is published to the topic `cmd/5C:CF:7F:25:79:D6/#`, the event `MQTT_METERHUB_RAW_DATA` will be triggered. It will also provide two metrics, `MeterHub.Telemetry.RSSI` and `MeterHub.Telemetry.Time` which will report data received on the topics `tele/5C:CF:7F:25:79:D6/RSSI` and `tele/5C:CF:7F:25:79:D6/TIME` respectively.

UPS MONITORING

There are two options to monitor a UPS: the first is through a USB or serial connection with help of a subagent and the second one is through the network with help of SNMP.

A subagent can be used for monitoring a UPS (Uninterruptible Power Supply) attached to a serial or USB port on a computer where the NetXMS agent is running. USB-attached devices are currently only supported on the Windows platform. Serial devices are supported on all platforms. One subagent can monitor multiple attached devices.

28.1 USB or serial UPS monitoring

You can monitor UPS devices attached to the hosts via serial cable or USB via the UPS subagent. Once you have your UPS attached to the host and the NetXMS agent installed, you should configure the UPS subagent. First, add the following line to the agents configuration file main section:

```
SubAgent = ups.nsm
```

Second, configure the attached UPS devices. Create a UPS section and for each UPS device attached to the host add a line in the following format:

```
Device = id:port:protocol
```

`id` is an arbitrary but unique number in the range 0 to 127, which is used to distinguish multiple UPS devices in further requests.

`device` is either the name of the serial port (e.g. *COM1:* or */dev/ttyS0*) or the serial number of the USB device. The keyword *ANY* can be used instead of an exact serial number to select the first available port.

`protocol` specifies which communication protocol should be used. Supported protocols are:

- APC
- BCMXCP - Some of the HP/Compaq, PowerWare, etc.
- MEGATEC
- METASYS
- MICRODOWELL
- USB - HID UPS devices (currently Windows only)

A sample configuration section for two devices attached via serial ports where one is an APC device (configured as device 0) and one is a HP device (configured as device 1):

```
# UPS subagent configuration section
[UPS]
Device = 0:/dev/ttyS0:APC
Device = 1:/dev/ttyS1:BCMXP
```

Once the UPS subagent is configured, you can start monitoring the UPS device status via metrics provided by it:

Metric Name	Type	Meaning
UPS.BatteryLevel ^(*)	Integer	Battery charge level in percents.
UPS.BatteryVoltage	Float	Current battery voltage.
UPS.ConnectionSta	Integer	Connection status between agent and device. Can have the following values: <ul style="list-style-type: none"> 0 - Agent is communication with the device 1 - Communication with the device has been lost
UPS.EstimatedRun	Integer	Estimated on-battery runtime in minutes.
UPS.Firmware(*)	String	Device's firmware version.
UPS.InputVoltage ^(*)	Float	Input line voltage.
UPS.LineFrequency	Integer	Input line frequency in Hz.
UPS.Load(*)	Integer	Device load in percents.
UPS.MfgDate(*)	String	Device manufacturing date.
UPS.Model(*)	String	Device model name.
UPS.NominalBatter	Float	Nominal battery voltage.
UPS.OnlineStatus ^(*)	Integer	Device online status. Can have the following values: <ul style="list-style-type: none"> 0 - Device is online. 1 - Device is on battery power. 2 - Device is on battery power and battery level is low.
UPS.OutputVoltage	Float	Output line voltage.
UPS.SerialNumber	String	Device's serial number.
UPS.Temperature ^(*)	Integer	Internal device temperature.

Please note that not all metrics are supported by all UPS devices. Many old or simple models will support only basic metrics like UPS.OnlineStatus. The most typical approach is to monitor UPS.OnlineStatus for going to 1 or 2, and then send notifications to administrators and shutdown affected hosts if needed. You can also monitor the UPS.EstimatedRuntime metric for the same purpose, if your device supports it.

28.2 SNMP UPS monitoring

Another option is to monitor the UPS using SNMP. NetXMS already includes MIBs for some UPSs, like APC UPS and the standard UPS MIB. The description for possible OIDs and some additional information for APC UPS configuration can be found on a [NetXMS wiki](#).

Please check [Import MIB](#) for MIB loading and [DCI configuration](#) for metric collection.

CLUSTER MONITORING

29.1 Introduction

Cluster monitoring provides aspects of monitoring needed in high availability setups. There is a special class of object in NetXMS - Cluster.

DCIs defined on a cluster object are automatically applied to its nodes. A cluster allows to aggregate data from its nodes, e.g. to calculate sum or average for a metric that is collected from all nodes. A cluster can adequately collect data from services as they move from one node to another, providing uninterrupted data collection.

JVM MONITORING

NetXMS has a Java plugin that allows to monitor the JVM. This subagent is build using JMX functionality.

30.1 Metrics

30.1.1 Single-value Metrics

Metric	Type	Meaning
JMX.ObjectAttribute(name,object,attribute,[item])	String	Get attribute of any connection, object. Optional attribute <i>item</i> is used when attribute is a list.
JMX.Memory.ObjectsPendingFinalization(name)	Unsigned integer	JVM objects pending finalization
JMX.Memory.Heap.Committed(name)	Unsigned integer 64	JVM committed heap memory
JMX.Memory.Heap.Current(name)	Unsigned integer 64	JVM current heap size
JMX.Memory.Heap.Init(name)	Unsigned integer 64	JVM initial heap size
JMX.Memory.Heap.Max(name)	Unsigned integer 64	JVM maximum heap size
JMX.Memory.NonHeap.Committed(name)	Unsigned integer 64	JVM committed non-heap memory
JMX.Memory.NonHeap.Current(name)	Unsigned integer 64	JVM current non-heap memory size
JMX.Memory.NonHeap.Init(name)	Unsigned integer 64	JVM initial non-heap memory size
JMX.Memory.NonHeap.Max(name)	Unsigned integer 64	JVM maximum non-heap memory size
JMX.Threads.Count(name)	Unsigned integer	JVM live threads count
JMX.Threads.DaemonCount(name)	Unsigned integer	JVM daemon threads count
JMX.Threads.PeakCount(name)	Unsigned integer	JVM peak number of threads
JMX.Threads.TotalStarted(name)	Unsigned integer	JVM total threads started
JMX.VM.BootClassPath(name)	String	JVM boot class path
JMX.VM.ClassPath(name)	String	JVM class path
JMX.VM.LoadedClassCount(name)	Unsigned integer	JVM loaded class count
JMX.VM.Name(name)	String	JVM name
JMX.VM.SpecVersion(name)	String	JVM specification version
JMX.VM.TotalLoadedClassCount(name)	Unsigned integer	JVM total loaded class count
JMX.VM.UnloadedClassCount(name)	Unsigned integer	JVM unloaded class count
JMX.VM.Uptime(name)	Unsigned integer	JVM uptime
JMX.VM.Vendor(name)	String	JVM vendor
JMX.VM.Version(name)	String	JVM version

30.1.2 Lists

Metric	Meaning
JMX.Domains(name)	List of JVM domains
JMX.Objects(name)	List of JVM objects
JMX.ObjectAttributes	List of JVM object's attributes

30.2 Configuration

It is required to define the java subagent and its configuration before JMX plugin configuration. More information about the Java subagent and its configuration can be found in the *Java subagent* section. JMS has only one configuration parameter “Server”. It is used to define the JMX server connection string.

JMS server connection string declaration options:

- name:url
- name:login@url
- name:login/password@url

30.2.1 Configuration example

In this example there are 2 JMS connections defined: *name* and *serverName2*.

```
MasterServers = netxms.demo
SubAgent=java.nsm

[JAVA]
jvm = /usr/lib/jvm/java-8-oracle/jre/lib/amd64/server/libjvm.so
Plugin = jmx.jar

[JMX]
Server=name:login/password@localhost
Server=serverName2:admin/pwd123@server1
```


HYPERVISOR MONITORING

NetXMS has subagents that allow to monitor hypervisors. This subagent is build using libvirt functionality. Due to the fact that libvirt is poorly supported on Windows platforms, vmgr subagent is not provided on Windows.

When installing NetXMS from packages, vmgr subagent is provided as a separate package named netxms-agent-vmgr. If building from source, ./configure should be ran with `--with-vmgr`.

31.1 Configuration

Configuration is separated into two parts: **vmgr** section defines all monitored hosts, and each host configuration is defined in separate section for each host.

Each host configuration should contain connection URL. Login and password parameters are optional. URL creation rules for each vitalization solution type can be found [in libvirt documentation](#).

Not all api functions are supported by all hypervisors in libvirt. See [libvirt API support matrix](#) for more information.

31.1.1 Configuration example

In this example two hosts are defined: **localESX1** and **test**. **localESX1** connection details are described in section **vmgr:localESX1** and **test** connection details are described in section **vmgr:test**.

```
MasterServers = netxms.demo
SubAgent = vmgr.nsm

[vmgr]
host = localESX1
host = test

[vmgr:localESX1]
Url = esx://root@10.5.0.21/?no_verify=1
Login = root
Password = password

[vmgr:test]
Url = test:///default
```

31.2 Provided Metrics

31.2.1 Single-value Metrics

Metric	Type	Description
VMGR.Host.CPU.Arch(hostName)	String	Host CPU architecture
VMGR.Host.CPU.MaxCount(hostName)	Unsigned integer	Host maximum virtual CPU count
VMGR.Host.FreeMemory(hostName)	Unsigned integer 64	Host free memory
VMGR.Host.MemorySize(hostName)	Unsigned integer 64	Host memory size
VMGR.Host.CPU.Model(hostName)	String	Host CPU model name
VMGR.Host.CPU.Frequency(hostName)	Unsigned integer	Host CPU frequency
VMGR.Host.ConnectionType(hostName)	String	Connection type
VMGR.Host.LibraryVersion(hostName)	Unsigned integer 64	Library version
VMGR.Host.ConnectionVersion(hostName)	Unsigned integer 64	Connection version
VMGR.VM.Memory.Used(hostName,vmName)	Unsigned integer 64	Memory currently used by VM
VMGR.VM.Memory.UsedPrec(hostName,vmName)	Unsigned integer	Percentage of currently memory usage by VM
VMGR.VM.Memory.Max(hostName,vmName)	Unsigned integer 64	Maximum VM available memory
VMGR.VM.CPU.Time(hostName,vmName)	Unsigned integer 64	Maximum VM CPU time

31.2.2 Tables

Metric	Description
VMGR.VM(hostName)	Connection VM table
VMGR.InterfaceList(hostName)	Connection interface list
VMGR.VMDisks(hostName,vmName)	VM Disks
VMGR.VMController(hostName,vmName)	VM Controllers
VMGR.VMInterface(hostName,vmName)	VM Interfaces
VMGR.VMVideo(hostName,vmName)	VM Video adapter settings
VMGR.Networks(hostName)	Networks table
VMGR.Storages(hostName)	Storages table

31.2.3 Lists

Metric	Description
VMGR.VMHost	List of hosts
VMGR.VMList(hostName)	List of VM for the host

ASTERISK MONITORING

NetXMS can be used to monitor health and performance of Asterisk PBX. All monitoring data collected and provided by subagent **asterisk.nsm**. One agent can collect data from multiple Asterisk systems.

32.1 Configuration

All Asterisk systems should be defined in subagent configuration. For simplified setup for single system monitoring subagent supports “local” system. Configuration for local system can be defined in **Asterisk** section of agent configuration file. For each additional system new section should be created in configuration file named **Asterisk/Systems/SystemName** (*SystemName* should be replaced with unique name). Each section can have the following parameters:

Parameter	Description	Default value
Hostname	DNS name or IP address of Asterisk PBX	127.0.0.1
Login	Login name	root
Password	Password	<i>empty</i>
Port	TCP port number for AMI connection	5038

It is also possible to configure subagent to periodically perform SIP registration tests. Each test should be configured in separate configuration section named **Asterisk/SIPRegistrationTests/TestName** for local system and **Asterisk/Systems/SystemName/SIPRegistrationTests/TestName** for other systems. *SystemName* and *TestName* should be replaced with unique system and test names respectively. Each test configuration section can have the following parameters:

Parameter	Description	Default value
Domain	Domain name used for registration	<i>empty</i>
Interval	Check interval in seconds	300
Login	SIP login name	netxms
Password	SIP password	netxms
Proxy	SIP proxy	sip:Asterisk IP address

32.1.1 Configuration Examples

Local system without SIP tests:

```
MasterServers = netxms.demo
SubAgent = asterisk.nsm
```

[Asterisk]

```
Login = root
Password = password1
```

Local system with two SIP tests:

```
MasterServers = netxms.demo
SubAgent = asterisk.nsm
```

[Asterisk]

```
Login = root
Password = password1
```

[Asterisk/SIPRegistrationTests/104]

```
Login = 104
Password = 12345
Domain = demo.netxms
```

[Asterisk/SIPRegistrationTests/115]

```
Login = 115
Password = 12345
Domain = demo.netxms
Interval = 60
```

Local system and remote system (named **Remote1**) on address 10.0.0.1 with one SIP test each:

```
MasterServers = netxms.demo
SubAgent = asterisk.nsm
```

[Asterisk]

```
Login = root
Password = password1
```

[Asterisk/SIPRegistrationTests/104]

```
Login = 104
Password = 12345
Domain = demo.netxms
```

[Asterisk/Systems/Remote1]

```
Hostname = 10.0.0.1
Login = root
Password = password1
```

[Asterisk/Systems/Remote1/SIPRegistrationTests/120]

```
Login = 120
Password = 12345
Domain = remote.netxms
```

32.2 Metrics

32.2.1 Single-value metrics

All metrics accept system name as first argument. Name for default local system is **LOCAL**. If system name is omitted local system is assumed. If system name is the only argument braces can be omitted as well.

Metric	Type	Meaning
Asterisk.AMI.Status(<i>system</i>)	Integer	AMI connection status (1 if AMI session is ready, 0 if not)
Asterisk.AMI.Version(<i>system</i>)	Integer	AMI version
Asterisk.Channels.Active(<i>system</i>)	Integer	Number of active channels
Asterisk.Channels.Busy(<i>system</i>)	Integer	Number of busy channels
Asterisk.Channels.Dialing(<i>system</i>)	Integer	Number of dialing channels
Asterisk.Channels.OffHook(<i>system</i>)	Integer	Number of off-hook channels
Asterisk.Channels.Reserved(<i>system</i>)	Integer	Number of reserved channels
Asterisk.Channels.Ringing(<i>system</i>)	Integer	Number of ringing channels
Asterisk.Channels.Up(<i>system</i>)	Integer	Number of up channels
Asterisk.Channels.CurrentCalls(<i>system</i>)	Integer	Number of currently active calls
Asterisk.Events.CallBarred(<i>system</i>)	Integer	Global cumulative counter of “call barred” events
Asterisk.Events.CallRejected(<i>system</i>)	Integer	Global cumulative counter of “call rejected” events
Asterisk.Events.ChannelUnavailable(<i>system</i>)	Integer	Global cumulative counter of “channel unavailable” events
Asterisk.Events.Congestion(<i>system</i>)	Integer	Global cumulative counter of “congestion” events
Asterisk.Events.NoRoute(<i>system</i>)	Integer	Global cumulative counter of “no route” events
Asterisk.Events.SubscriberAbsent(<i>system</i>)	Integer	Global cumulative counter of “subscriber absent” events
Asterisk.Peer.Events.CallBarred(<i>system, peer</i>)	Integer	Cumulative counter of “call barred” events for given peer
Asterisk.Peer.Events.CallRejected(<i>system, peer</i>)	Integer	Cumulative counter of “call rejected” events for given peer
Asterisk.Peer.Events.ChannelUnavailable(<i>system, peer</i>)	Integer	Cumulative counter of “channel unavailable” events for given peer
Asterisk.Peer.Events.Congestion(<i>system, peer</i>)	Integer	Cumulative counter of “congestion” events for given peer
Asterisk.Peer.Events.NoRoute(<i>system, peer</i>)	Integer	Cumulative counter of “no route” events for given peer
Asterisk.Peer.Events.SubscriberAbsent(<i>system, peer</i>)	Integer	Cumulative counter of “subscriber absent” events for given peer
Asterisk.Peer.RTCP.AverageJitter(<i>system, peer</i>)	Integer	Average jitter for given peer in milliseconds (moving average over last 180 measurements)
Asterisk.Peer.RTCP.AveragePacketLoss(<i>system, peer</i>)	Integer	Average packet loss for given peer (moving average over last 180 measurements)
Asterisk.Peer.RTCP.AverageRTT(<i>system, peer</i>)	Integer	Average round trip time in milliseconds for given peer (moving average over last 180 measurements)
Asterisk.Peer.RTCP.LastJitter(<i>system, peer</i>)	Integer	Last reported jitter for given peer in milliseconds

continues on next page

Table 1 – continued from previous page

Metric	Type	Meaning
Asterisk.Peer.RTCP.LastPacketLoss(<i>system, peer</i>)	Integer	Last reported packet loss for given peer
Asterisk.Peer.RTCP.LastRTT(<i>system, peer</i>)	Integer	Last reported round trip time in milliseconds for given peer
Asterisk.Peer.RTCP.MaxJitter(<i>system, peer</i>)	Integer	Maximum reported jitter for given peer in milliseconds
Asterisk.Peer.RTCP.MaxPacketLoss(<i>system, peer</i>)	Integer	Maximum reported packet loss for given peer
Asterisk.Peer.RTCP.MaxRTT(<i>system, peer</i>)	Integer	Maximum reported round trip time in milliseconds for given peer
Asterisk.Peer.RTCP.MinJitter(<i>system, peer</i>)	Integer	Minimum reported jitter for given peer in milliseconds
Asterisk.Peer.RTCP.MinPacketLoss(<i>system, peer</i>)	Integer	Minimum reported packet loss for given peer
Asterisk.Peer.RTCP.MinRTT(<i>system, peer</i>)	Integer	Minimum reported round trip time in milliseconds for given peer
Asterisk.SIP.Peer.Details(<i>system, peer, tag</i>)	String	Value of specific tag from SIPshowpeer AMI message
Asterisk.SIP.Peer.IPAddress(<i>system, peer</i>)	String	SIP peer IP address
Asterisk.SIP.Peer.Status(<i>system, peer</i>)	String	SIP peer status
Asterisk.SIP.Peer.Type(<i>system, peer</i>)	String	SIP peer type
Asterisk.SIP.Peer.UserAgent(<i>system, peer</i>)	String	SIP peer user agent information
Asterisk.SIP.Peer.VoiceMailbox(<i>system, peer</i>)	String	SIP peer voice mailbox information
Asterisk.SIP.Peers.Connected(<i>system</i>)	Integer	Number of connected SIP peers
Asterisk.SIP.Peers.Total(<i>system</i>)	Integer	Total count of configured SIP peers
Asterisk.SIP.Peers.Unknown(<i>system</i>)	Integer	Number of SIP peers in unknown state
Asterisk.SIP.Peers.Unmonitored(<i>system</i>)	Integer	Number of unmonitored SIP peers
Asterisk.SIP.Peers.Unreachable(<i>system</i>)	Integer	Number of unreachable SIP peers
Asterisk.SIP.RegistrationTest.ElapsedTime(<i>system, test</i>)	Integer	Elapsed time for last run of given registration test
Asterisk.SIP.RegistrationTest.Status(<i>system, test</i>)	Integer	Status of last run of given registration test
Asterisk.SIP.RegistrationTest.Timestamp(<i>system, test</i>)	Integer	Timestamp last run of given registration test as UNIX time (number of seconds since 1.1.1970 00:00:00 UTC)
Asterisk.SIP.TestRegistration(<i>system, login, password, domain</i>)	Integer	Status of ad-hoc registration
Asterisk.TaskProcessor.HighWatermark(<i>system, processor</i>)	Integer	High watermark for given task processor
Asterisk.TaskProcessor.LowWatermark(<i>system, processor</i>)	Integer	Low watermark for given task processor
Asterisk.TaskProcessor.MaxDepth(<i>system, processor</i>)	Integer	Maximum queue depth for given task processor
Asterisk.TaskProcessor.Processed(<i>system, processor</i>)	Integer	Number of processed tasks for given task processor
Asterisk.TaskProcessor.Queued(<i>system, processor</i>)	Integer	Number of queued tasks for given task processor
Asterisk.Version(<i>system</i>)	String	Asterisk version

32.2.2 Tables

All tables accept system name as first argument. Name for default local system is **LOCAL**. If system name is omitted local system is assumed. If system name is the only argument braces can be omitted as well.

Metric	Description
Asterisk.Channels(<i>system</i>)	Active channels
Asterisk.CommandOutput(<i>system</i> , <i>command</i>)	Output of given Asterisk console command
Asterisk.SIP.Peers(<i>system</i>)	SIP peers
Asterisk.SIP.RegistrationTests(<i>system</i>)	Configured SIP registration tests
Asterisk.TaskProcessors(<i>system</i>)	Task processors

32.2.3 Lists

All lists accept system name as first argument. Name for default local system is **LOCAL**. If system name is omitted local system is assumed. If system name is the only argument braces can be omitted as well.

Metric	Description
Asterisk.Channels(<i>system</i>)	Active channels
Asterisk.CommandOutput(<i>system</i> , <i>command</i>)	Output of given Asterisk console command
Asterisk.SIP.Peers(<i>system</i>)	SIP peers
Asterisk.SIP.RegistrationTests(<i>system</i>)	Configured SIP registration tests
Asterisk.Systems	Configured Asterisk systems
Asterisk.TaskProcessors(<i>system</i>)	Task processors

NETWORK TOPOLOGY

33.1 Introduction

NetXMS server automatically creates and maintains network model on different layers. All necessary information taken from ARP cache, routing tables, and switch forwarding database of managed nodes. Topology data provided by CDP, LLDP, and NDP (SONMP) protocols also used in building network model. Having network model instantly available allows NetXMS users to perform various network topology tasks much faster and easier.

Requirements to build network topology:

- All network equipment should be registered in NetXMS system
- Equipment should response to SNMP
- Equipment should have at least STP
- There will be more information if equipment will have LLDP or CDP

Manual topology poll can be started on the network equipment to have information about information availability.

Based on network topology network correlation is done. Network correlation reduce number of alerts and increase problem resolution speed.

Currently there are 3 states/events regarding connectivity:

- down (event SYS_NODE_DOWN) - when server cannot contact the node and has no topology information for event correlation or it is really problem with that node
- unreachable (SYS_NODE_UNREACHABLE) - when server knows that node cannot be contacted due to intermediate router/interface failure
- up (SYS_NODE_UP) - when node is reachable

So when node becomes unreachable, either SYS_NODE_DOWN or SYS_NODE_UNREACHABLE event is generated, depending on root cause. But when node became reachable again, SYS_NODE_UP being generated.

33.2 How topology information is built

FDB. From FDB table we take ports where only one mac address is present - this means that something is directly connected. If this device is present in NetXMS and it's mac address is known (we have agent on it, SNMP, or some other agent on that network communicated to that device and has IP-MAC pair in ARP table) - we have a peer.

LLDP. So if we have another switch connected, that switch is sending LLDP packets, the switch that we are polling receives these packets and saves information in LLDP table. We read this table and we know that there's a device with some LLDP ID connected to port X of our device. But we also need NetXMS to read that device via SNMP, in this case LLDP ID will be read and we will be able to match.

CDP. Similar to LLDP.

STP table on a switch has limited information - only about peers that are on the way to root LLDP switch. But we read that and can get peers from there.

Interfaces tab has *Peer Discovery Protocol* column which tells, how peer information was obtained.

For debug you can set debug tags poll.topology, topo.*, topology.* to level 7 - there will be some information in server log when topology poll is executed.

33.3 Find where node is connected

It is possible to find switch port where any given node is connected (sometimes called “connection point” in management client). To find out node’s connection point, right-click on node object, and select *Find switch port* in pop-up menu. Message box with search results will pop up, and if port is found, search results view will be opened (or updated if already open). Search results view looks like this:

Seq.	Node	Interface	MAC	IP	Switch	Port	Type
2	ATM001	unknown	00:1C:C0:79:9A:91	192.168.22.2	catalyst-2900-central	Fa0/24	indirect
1	betelgeuse	br0	00:1F:D0:A4:0B:FE	192.168.22.100	catalyst-2900-central	Fa0/7	indirect

Columns have the following meaning:

Seq.	Search result sequence number
Node	Name of end node object
Interface	Name of node's interface object
MAC	Interface's MAC address
IP	Interface's IP address
Switch	Name of switch node object
Port	Name of interface object representing switch port
Type	Connection type - direct or indirect. Direct connection type means that NetXMS server did not detect any other devices on same switch port, and most likely end node connected directly to the switch. Indirect means that some other devices was detected on same switch port. Virtual machines and virtual machine host will always be detected as indirect.

33.4 Find MAC address

It is possible to find location of any known MAC address in the network. To do this, select *Tools ▸ Find MAC address*. Results of a search will be displayed in the same results view. It is not necessary that node with given MAC address be managed by NetXMS server, but if it is, appropriate details will be displayed.

33.5 Find IP address

It is possible to find location of any known IP address in the network. To do this, select *Tools ▸ Find IP address*. Results of a search will be displayed in the same results view. It is not necessary that node with given IP address be managed by NetXMS server, but if it is, appropriate details will be displayed.

HARDWARE ASSET MANAGEMENT

Added in version 4.4.

NetXMS can store information about hardware assets organized as a hierarchical structure. Asset information is kept in Asset objects under Assets tree. There are Asset group objects which acts as folders.

Assets information attributes are defined globally in *Asset management schema*.

Assets can be linked to Nodes, Access Points, Chassis, Mobile Devices, Racks or Sensors in one-to-one relationship. Linking can be done either manually or automatically, based on serial number information or MAC address of primary network interface (MAC address is used only if serial number is not available).

When asset is linked to Node (or other type of object), Vendor, Model and IP Address fields in the asset can be automatically updated based on information on a Node. Asset fields can also be automatically filled in using Auto Fill Script.

34.1 Configuring Asset management schema

Configuration of information attributes which are present in assets is performed in *Configuration -> Asset management schema*. The schema is global.

The screenshot displays the NetXMS Management Client interface. The title bar indicates the user is 'admin@127.0.0.1'. The main window is divided into two panes. The left pane, titled 'Configuration', shows a tree view of various system components, with 'Asset management schema' selected. The right pane, titled 'Asset Management Schema', displays a table of attributes for the selected schema.

Name	Display Name	Data Type	Mandatory	Unique	Hidden	Autofill	Range min	Range max	System type
IpAddress	IP Address	IP address	No	No	No	No	0	0	IP address
MacAddress	MAC Address	MAC address	No	No	No	No	0	0	MAC address
Model		String	No	No	No	No	0	0	Model
ProcurementDate	Procurement Date	Date	Yes	No	No	No	0	0	None
Serial		String	No	No	No	No	0	0	Serial
Vendor		String	No	No	No	No	0	0	Vendor

To add a new attribute, select *New attribute...* from context menu or click + button on the toolbar. This will open asset attribute property editor:

The screenshot shows the 'Asset Attribute Properties' dialog box. The 'General' tab is selected in the sidebar. The main panel contains the following fields and options:

- Name:** A text field containing 'Serial'.
- Display name:** An empty text field.
- Data type:** A dropdown menu set to 'String'.
- System type:** A dropdown menu set to 'Serial'.
- Use limits:** An unchecked checkbox.
- Minimum length:** A numeric field set to '0'.
- Maximum length:** A numeric field set to '0'.
- Mandatory:** An unchecked checkbox.
- Unique:** An unchecked checkbox.
- Hidden:** An unchecked checkbox.

At the bottom of the dialog are two buttons: 'Cancel' and 'Apply and Close'.

Asset attribute properties has the following settings:

- Name - Should be unique and conform to NXSL naming convention. This name is used when accessing asset information from scripts.
- Display name - Optional, Name will be used if not filled in.
- Data type - The following data types are supported:
 - String - Maximum length 2000 characters
 - Integer - Int32
 - Number - Double
 - Boolean
 - Enum - Possible values are configured on *Enum Values* tab.
 - MAC Address
 - IP Address
 - UUID
 - Object Reference
 - Date
- System type - enables special processing depending on the selected type:
 - Serial - used for automatic linking. Asset will be automatically linked to node if value of this attribute matches serial number of that node.

- MAC Address - used for automatic linking. Asset will be automatically linked to node, if value of this attribute matches MAC Address on primary interface of that node (but only if node does not has Serial number)
- IP address - used to autofill. This attribute will be automatically created and filled with primary IP address of node (or other object) linked to this asset.
- Vendor - used to autofill. This attribute will be automatically created and filled with vendor value of node linked to this asset. Autofill is performed only once, once this attribute has a value, it will not be updated.
- Model - used to autofill. This attribute will be automatically created and filled with model value of node linked to this asset. Autofill is performed only once, once this attribute has a value, it will not be updated.

Processing is performed on node's (or other object's) configuration poll or when asset is linked.

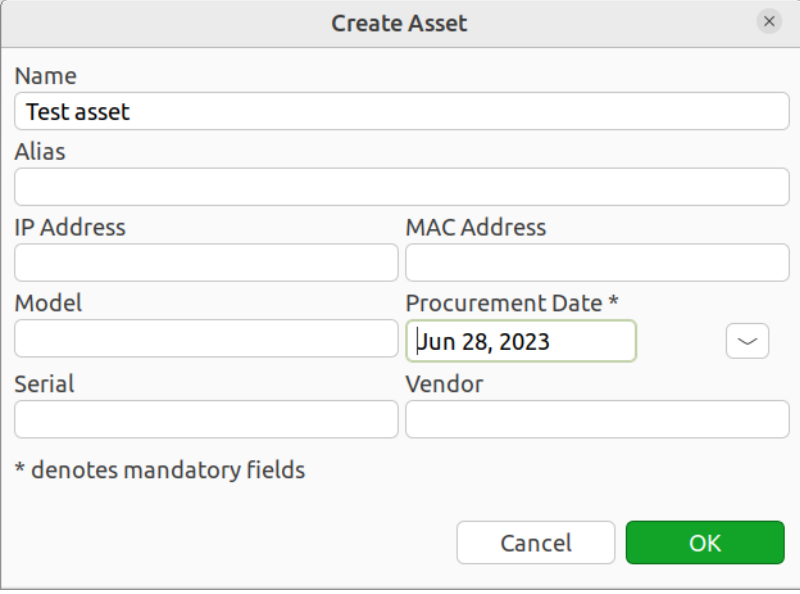
- Use limits - enables limits for attribute value. For String type minimum and maximum number of characters can be defined. For numeric types minimum and maximum value is defined.
- Mandatory - this attribute is mandatory.
- Unique - values for this attribute should be unique among all assets.
- Hidden - attribute is hidden from summary table displayed on asset groups.
- Auto Fill Script - NXSL script that performs auto-fill of asset property. Ignored, if System type is set.
- Enum Values - defines list of possible values for Enum data type. Display name is optional, if it's not filled in, Value is used.

34.2 Asset Creation

Assets are managed under *Assets* perspective. Hierarchical structure is built using Asset Group objects, Asset Root is the top object of the hierarchy.

To create a new Asset Group, select *Create->Asset Group* from context menu of Asset Root or Asset Group and provide asset group name.

To create a new Asset, select *Create->Asset* from context menu of Asset Root or Asset Group. Asset creation dialog will be displayed, with asset attributes configured in asset management schema:



The image shows a 'Create Asset' dialog box with the following fields and controls:

- Name:** Text field containing 'Test asset'.
- Alias:** Empty text field.
- IP Address:** Empty text field.
- MAC Address:** Empty text field.
- Model:** Empty text field.
- Procurement Date *:** Date picker showing 'Jun 28, 2023' with a dropdown arrow.
- Serial:** Empty text field.
- Vendor:** Empty text field.
- Legend:** '* denotes mandatory fields'.
- Buttons:** 'Cancel' and 'OK' buttons at the bottom right.

Name and mandatory attributes should be filled in, the rest of attributes can be left empty.

34.3 Asset Linking

To link asset to node (or other type of object), select *Link to...* from context menu of asset and choose a node. If that node already has an asset linked, a warning message will be displayed.

Linking can also be performed by selecting *Link to asset...* from context menu of node (or other type of object) and choosing an asset. If that asset already has a node linked, a warning message will be displayed.

To unlink, select *Unlink* from asset context menu or *Unlink from asset* from node context menu.

BUSINESS SERVICES

35.1 Introduction

In a nutshell, Business Services is a tool for availability monitoring of logical services. Company email, web site, server farm, call center - all are examples of logical services. Moreover, the services can be combined together to define a “broader” logical service. Company email, web site, name server and firewall all can be referred to as “Company Internet Services” and monitored for availability as a whole. So if the name server goes down then the “Company Internet Services” do not function properly as a whole. This feature can be used both for internal QA and external Service Level Agreement (SLA) monitoring.

35.2 Business service object

35.2.1 Business Service

Business Services represented with service checks and a tree-like hierarchy of other business services. For each service in the hierarchy, NetXMS keeps track of all downtime cases so later user can request calculation of availability percentage for required time period. To check availability at any particular level, select Business Service object in the *Object Browser*, choose *Availability* tab and select time period.

Business service contains two NXSL scripts in configuration: for object automatic binding and for DCI automatic binding. Those scripts can be used to automatically populate Business service with resources that require monitoring. Service checks can be automatically created and also removed if “Auto remove” filter option is selected.

35.2.2 Service check

Service check is a test whose result is used to define the state of the service. There can be 3 types of checks: DCI check, object check and NXSL script. Service check can have one of statuses: OK, Failed or Degraded. Degraded status means that object or DCI status is not Normal, but is less worse than threshold for this check, this state will not change state of business service to failed and will not affect availability percentage.

DCI check

DCI check is based on the status of DCI. DCI status is calculated from the status of threshold (if it is active) and severity of active threshold. DCI check has its own status threshold starting from which check is counted as failed. Threshold can be set separately for each check. If default value is chosen, value of “BusinessServices.Check.Threshold.DataCollection” server configuration variable is used.

Object check

Object check is based on object status. Object check has its own status threshold starting from which check is counted as failed. Threshold can be set separately for each check. If default value is chosen, value of “BusinessServices.Check.Threshold.Objects” server configuration variable is used.

NXSL script check

NXSL script check either returns success (the test result ok) or failure (the service has failed). For success “true” should be returned, and “false” for failure. In addition failure reason can be returned from the script - script should return textual with the reason, this is interpreted as failed check.

There are the following special variables which can be used in NXSL scripts for service checks:

- \$object - points to the object for which the check is executed
- \$node - points to the current node for which the check is executed. Will be null, if the object, for which the check is executed is not a node.
- \$service - the business service this check belongs to

35.3 Business service prototype

To avoid manually defining of the same business service multiple times (for multiple clients or infrastructure items) you can create business service prototype. The principle behind business service prototype is very similar to DCI instance discovery. There is instance discovery options and script to filter it. For instances that passed the filter business services are created. In object and DCI auto-apply scripts of created business services information about instance value and id of business service prototype are available.

35.4 Configuration and usage

For both configuration and monitoring use *Business Service* perspective.

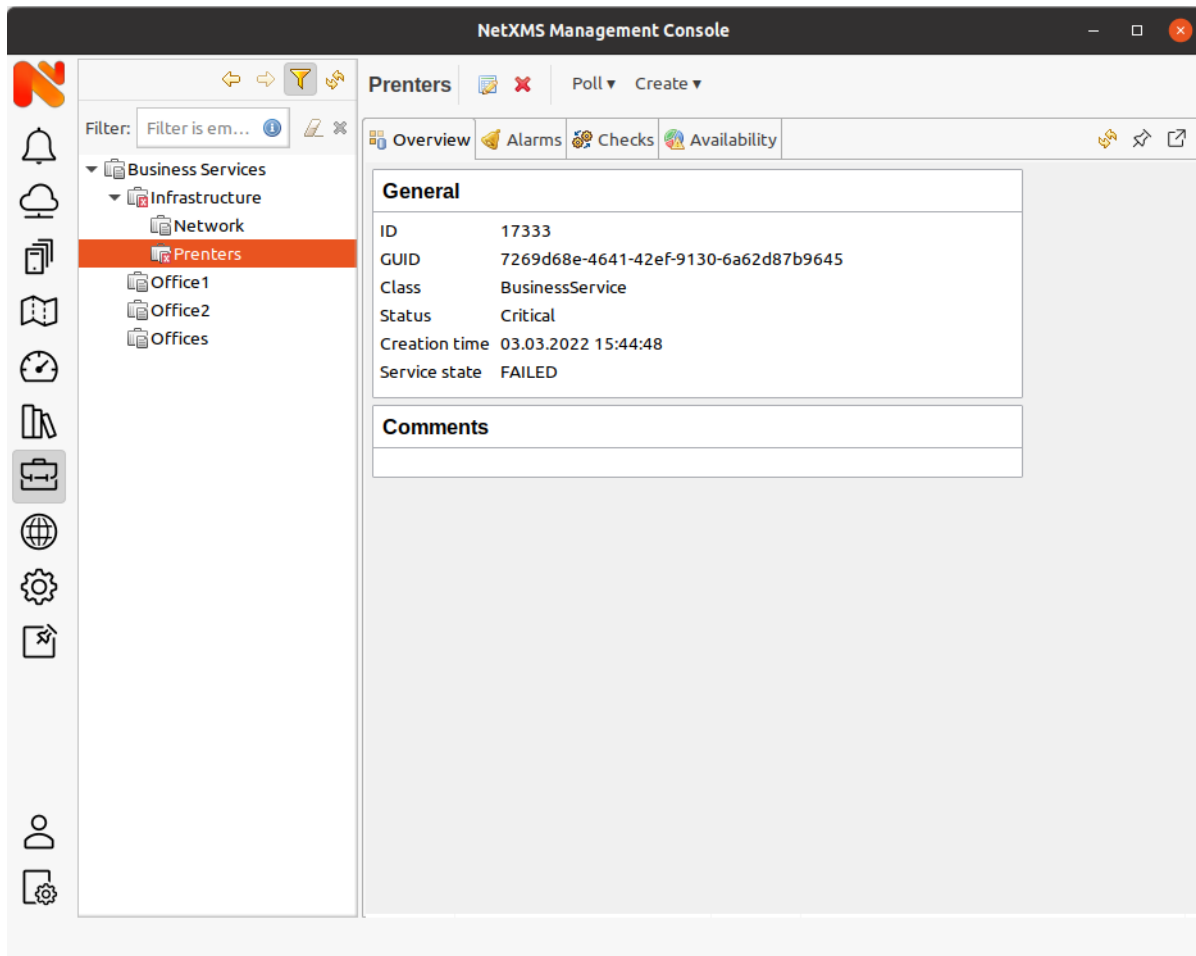


Fig. 1: Business service perspective

35.4.1 Configuration

To define a new service select *Create business service* from the context menu in *Object Browser* and enter the service name. Then in newly created service you may want to define checks or define check auto apply scripts in business service properties.

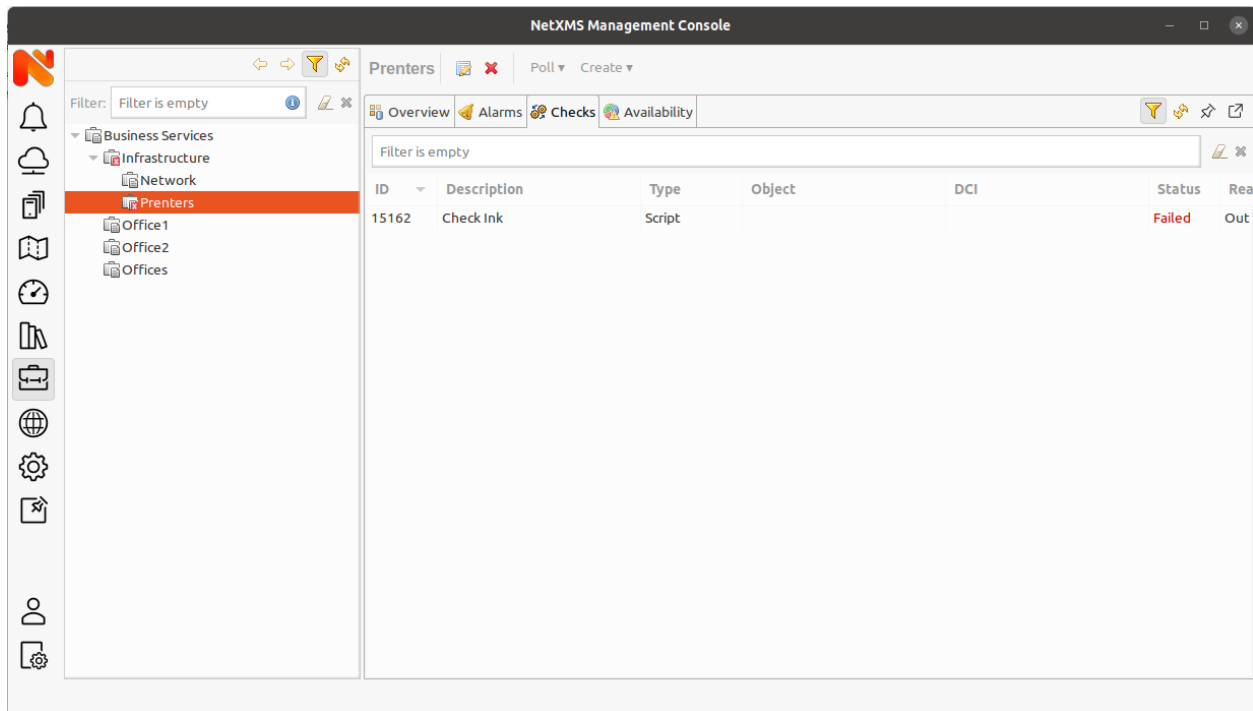


Fig. 2: Business service checks

Business service prototype is defined the same way, but it is also required to configure Instance Discovery method.

35.4.2 Monitoring

Business service availability for exact period can be checked using *Availability* tab. It has predefined time ranges and a date selector for arbitrary date range. A list of problems occurred for a business service is also shown with detailed information, start time, end time and reason.

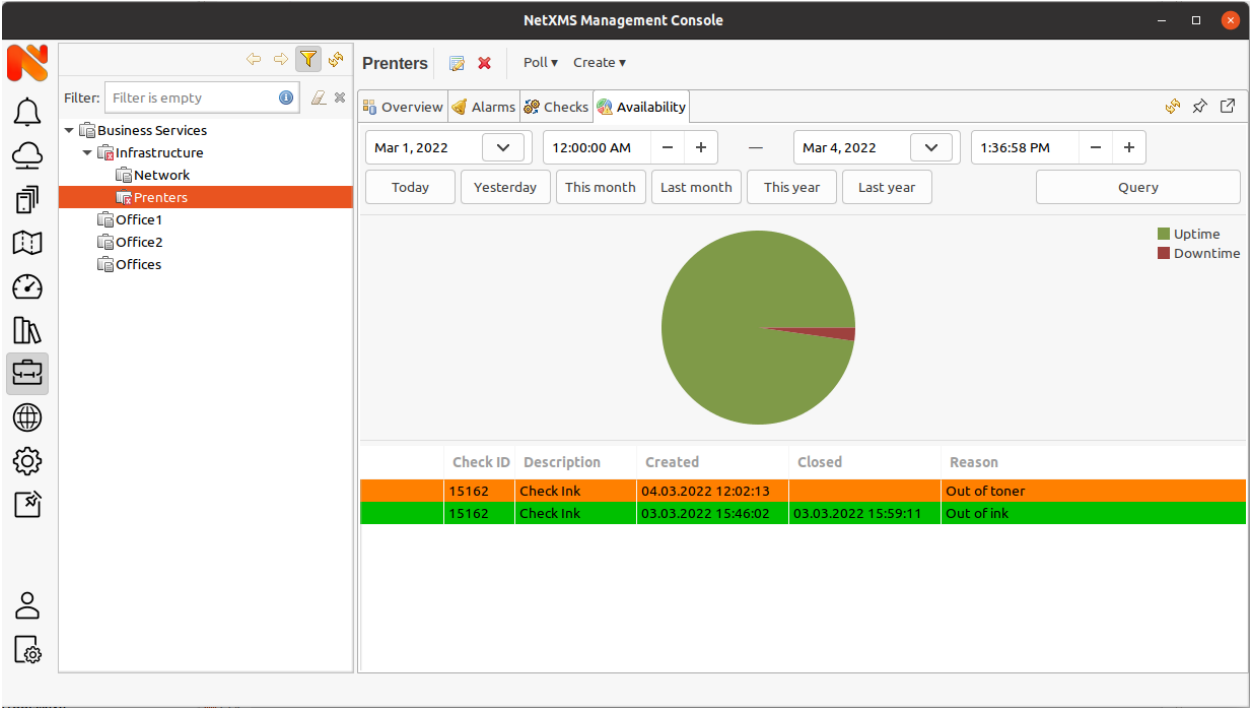


Fig. 3: Availability pie chart and details

REMOTE FILE MANAGEMENT

36.1 Agent file management

36.1.1 Introduction

This section describes possibilities to manage files on remote nodes using agent and required configuration for it.

36.1.2 Required Configuration

Subagent configuration

To do any manipulations with files on a node it is required to load filemng subagent and configure accessible paths. It provides possibility to upload, download, delete, move and rename files.

All configuration parameters related to filemng subagent should be placed into **[filemgr]** section of agent's configuration file. The following configuration parameters are supported:

Parameter	Description
Root-Folder	Path to the folder which should be exposed. If “;ro” is appended to path - agent will reject any write operations with this folder

Agent's configuration file example:

```
MasterServers = netxms.demo
SubAgent = filemgr.nsm

[filemgr]
RootFolder = /home/zev # read/write access
RootFolder = /home/zev/etc # read/write access
RootFolder = /logs;ro # read only access
```

Access rights

To view File Manager View it's enough to have “Read” access to node.

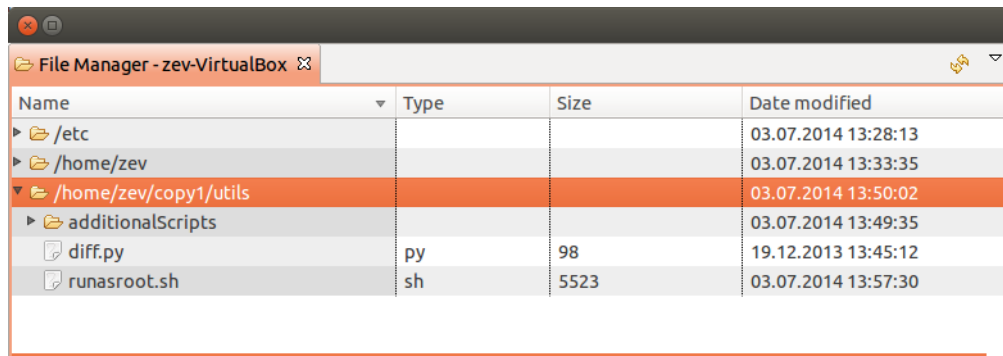
To download files from file manager of through multiple file download there should be “Download file” access for this node and for multiple download “Read server files” access.

To upload file from subagent there should be “Upload file” access for this node.

For moving, renaming and deleting files from node it is required “Manage files” access to node.

36.1.3 File Manager view

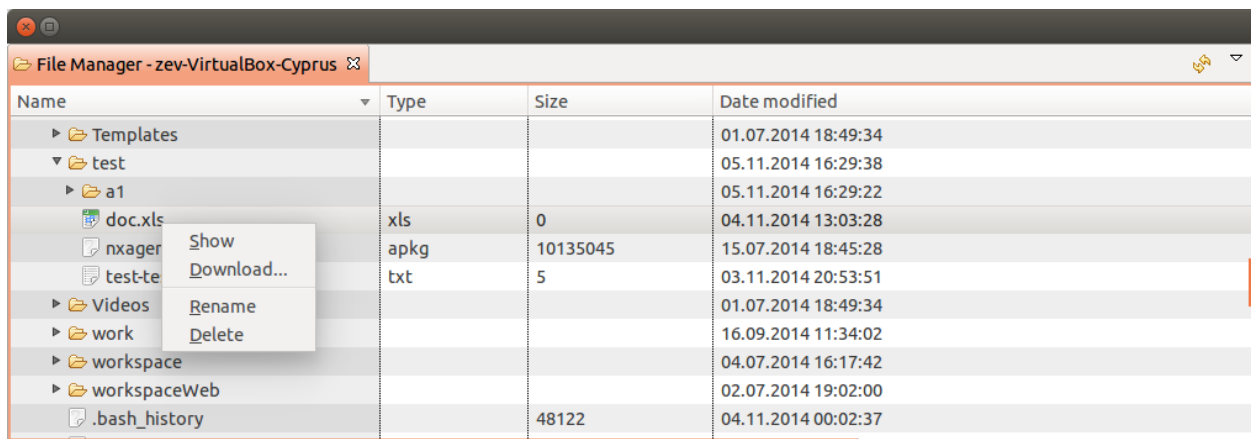
For each configured node it is possible to open File Manager. It will display all configured root folders and allow to browse into these folders.



Name	Type	Size	Date modified
▶ /etc			03.07.2014 13:28:13
▶ /home/zev			03.07.2014 13:33:35
▼ /home/zev/copy1/Utils			03.07.2014 13:50:02
▶ additionalScripts			03.07.2014 13:49:35
diff.py	py	98	19.12.2013 13:45:12
runasroot.sh	sh	5523	03.07.2014 13:57:30

File menu

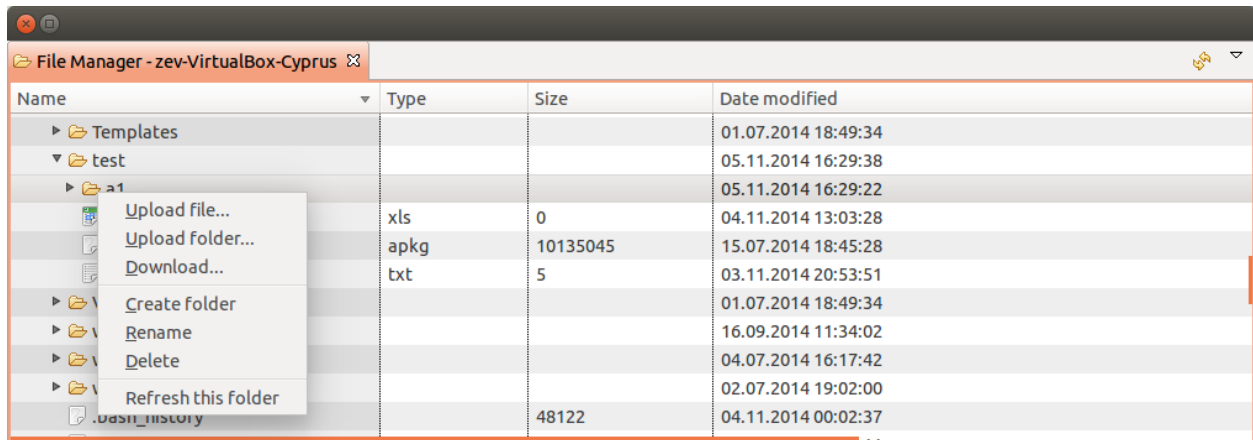
- Download... : downloads file to selected folder on local computer
- Show : shows file with tail option 'on'
- Rename : renames file
- Delete : deletes file



Name	Type	Size	Date modified
▶ Templates			01.07.2014 18:49:34
▼ test			05.11.2014 16:29:38
▶ a1			05.11.2014 16:29:22
doc.xls	xls	0	04.11.2014 13:03:28
nxager	apkg	10135045	15.07.2014 18:45:28
test-te	txt	5	03.11.2014 20:53:51
▶ Videos			01.07.2014 18:49:34
▶ work			16.09.2014 11:34:02
▶ workspace			04.07.2014 16:17:42
▶ workspaceWeb			02.07.2014 19:02:00
.bash_history		48122	04.11.2014 00:02:37

Folder menu

- Upload file... : uploads local file to selected folder in view
- Upload folder... : uploads local folder to selected folder in view (not supported on web client)
- Download... : download folder to selected folder on local computer (on web client will be advised to save as a zip of the selected folder)
- Rename : renames folder
- Delete : deletes folder and all it's content
- Refresh this folder : refreshes content of selected folder in view



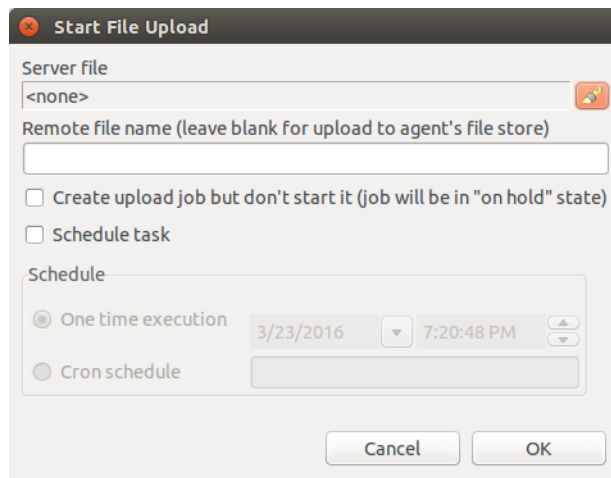
Other options

- It is possible to move files and folders with help of drag and drop.
- To refresh all view should be used view refresh button (not form folder menu). But in this case all expanded folders will be closed.

36.2 Advanced File Management

There are options to run multiple file upload to agents, file upload jobs on hold and scheduled file upload jobs. All this options are available uploading file from server to agent. That means that before upload file should be uploaded to server for instruction check *Upload file on server* section.

Advanced file upload can be accessed selecting required nodes (can be selected more than one with help of 'Ctrl' key) and in object menu selecting *Upload file*....



Job configuration:

- File that should be uploaded on the agent(s).
- Remote file path (If destination will not be set then as a destination will be taken from agent's config parameter 'FileStore'). If path is set agent will check if there is access to this folder. Access is configured by *filemgr* subagent, check *Agent file management*.

- Job can be created “on hold”. This mean that job will be created, but not started. After creation it can be manually started selecting job in *Server Jobs* view and clicking *Unhold*.
- Other option is to schedule file upload job. It can scheduled to be executed once at exact time (*One time execution*) or to be executed according to schedule(*Cron schedule*). See [Cron format](#) for supported cron format options.

Result of file upload job can be checked in *Server Jobs* view. It can be accessed by clicking *View* ▶ *Server Jobs*.

36.3 Server File Management

36.3.1 Access Rights

There are 2 access rights that can be granted:

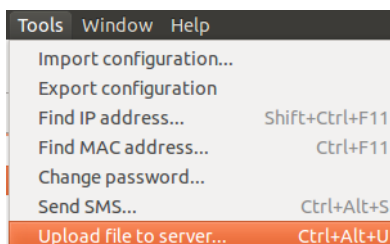
- Read server files : possibility to see files that are download on server
- Manage server files : possibility to remove or upload on server files

36.3.2 Upload file on server

It can be done in “Server File List” view

File name	File type	File size	Modification day
Booboo.wav	wav	24613	17.02.2014 18:29:51
failure1.wav	wav	16508	17.02.2014 18:29:58
fallen.wav	wav	19492	17.02.2014 18:30:15
screenshot.png	png	294441	06.02.2014 14:33:53

or “Tools”->”Upload file to server...”.



PACKAGE MANAGEMENT

37.1 Introduction

The package management functionality can upload and execute installers via the NetXMS agent. This allows to perform centralized upgrade of the NetXMS agent, to install other software or upload and extract archive files onto target systems.

To access package management, open the *Configuration* perspective and select *Packages*. Software packages are first uploaded to the NetXMS server. In order to do this, select *Upload to server* and select a file.

For some types of packages, the additional dialog *Edit Package Metadata* is displayed. This allows to specify additional metadata for a package. Whenever possible, metadata information is filled in automatically based on information contained in file name.

You can open the metadata editor by double-clicking on a package in the list. In the metadata editor *Name*, *Version* and *Description* are just informative fields, they are not used in package processing.

Platform denotes for which platforms a package is applicable. The actual platform of a node is compared to this field as string value using wildcard characters. Two wildcard characters are supported: * - represents zero, one or multiple characters. ? - represents any single character. Setting *Platform* to * would mean any platform. `Linux*` would mean both 32 and 64 bit Linuxes.

Type defines package type. This defines how the agent should process the package when installing it. The meaning of the *Command* field depends on the package type. See information in the table below.

The following types of package files are supported by package management:

Package type	Extension	Description
NetXMS Agent Package (agent-installer)	.apkg	<i>Command</i> is not used by this package type.
Debian/Ubuntu Package	.deb	<i>Command</i> contains additional parameters passed to /usr/bin/dpkg
Executable	.exe	<i>Command</i> is optional. If specified, it sets the actual command executed by agent. <code>\${file}</code> macro will be replaced by actual file name.
Windows Installer Package	.msi	<i>Command</i> contains additional parameters passed to Windows installer API
Windows Installer Patch	.msp	<i>Command</i> contains additional parameters passed to Windows installer API
Windows Update Package	.msu	<i>Command</i> contains additional parameters passed to wusa.exe
Red Hat Package	.rpm	<i>Command</i> contains additional parameters passed to /usr/bin/rpm
NetXMS Package Info	.npi	Deprecated type of metadata file for NetXMS Agent Package.
Compressed TAR Archive	.tgz, .tar.gz	<i>Command</i> is optional. If specified, it defines the path the archive should be extracted to.
ZIP Archive	.zip	<i>Command</i> is optional. If specified, it defines the path the archive should be extracted to.

To deploy a package, select one or several nodes from *Infrastructure services* or *Entire Network*. You can also select containers or subnets. Right-click on the selected items and select *Deploy package....* Select the package and click *OK*.

During the package deployment process, the server will request the platform name from agent and check if it matches *Platform* from the package metadata. The deployment process is shown in the *Package deployment monitor* tab that is visible on all relevant containers, subnets and nodes.

REPORTING

Reporting module is an optional component, build on top of well known [JasperReports](#) library, which can produce pixel-perfect documents in variety of formats based on historical data collected by NetXMS.

Reporting module is a separate process that communicates with NetXMS and handles execution and rendering of reports.

Report generation is two step process: first step is to collect and process input data, then render output files in desired format. This separation exist for a reason: unlike rendering step, data collection could take hours to complete and it make no sense to repeat same processing process to render Excel file instead of PDF. When first step is finished, all processed information is saved into intermediate file on the reporting server and available for rendering at any time (e.g. user can render and download report from last year, even if source data is already purged).

Reports execution and rendering can be initiated both manually and on schedule.

38.1 User Interface

All reporting-related operations are available in Management Client in a separate *Reporting* perspective. Perspective contains two main areas - list of available reports on the left and report details view on the right. Details view show information about currently selected report.

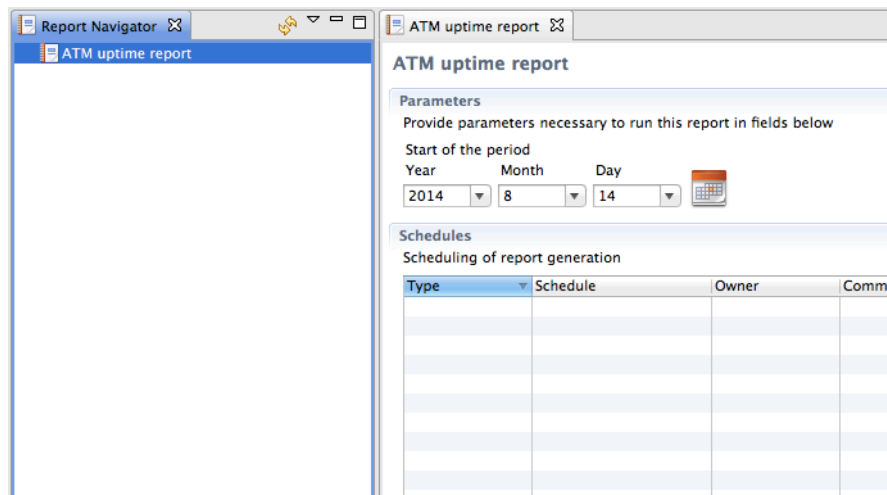


Fig. 1: Reporting perspective.

Details view contains tree main areas: *Parameters*, *Schedules*, and *Results*.

38.1.1 Parameters

Parameters
Provide parameters necessary to run this report in fields below

Start of the period

Year	Month	Day
2014	8	14

Fig. 2: Execution parameters for report (in this example: *Start date*)

In this section, user can set all input parameters required for report execution, for example data range or list of objects which should be included in the report. List of required parameters is extracted from report definition file and can be empty, if particular report do not require any input data to operate.

38.1.2 Schedules

Each report can have one or more schedules, which define when it should be executed, and optionally rendered. Reporting server can also notify users that new report is executed and available for download, or send resulting file as an attachment.

Schedules
Scheduling of report generation

Type	Schedule	Owner	Comments
daily	08:15	admin	

[Add Schedule](#)

Fig. 3: List of scheduled executions

To add new schedule, click on *Add Schedule* down below, this will open schedule editor.

Report Execution Schedule

type filter text

General

Notifications

General

☒ Once
 ☐ Daily
 ☐ Weekly
 ☐ Monthly

14/08/2014 11:11:32

Cancel OK

Fig. 4: Schedule editor with two tabs, *General* and *Notifications*

General tab contains four scheduling options:

1. *Once* - execute report once at specified date and time
2. *Daily* - execute report every day at specified time
3. *Weekly* - execute report every week on selected days of week at specified time

4. *Monthly* - execute report every month on selected days at specified time

Fig. 5: *Notifications* tab of Schedule editor

Notification tab allows to control email notifications and report delivery to list of recipients. To enable notifications, select *Send notification on job completion* checkbox.

If checkbox *Attach rendered report* checkbox is enabled, report will be rendered into selected format and attached to notification email.

38.1.3 Results section

Results		
The following execution results are available for rendering		
Execution Time	Started by	Status
14.08.2014 11:59:09	admin	Success
14.08.2014 08:15:28	admin	Success

Fig. 6: List of generated reports

This section contains list of all generated reports, which are stored on the server and can be rendered on request. To render report in desired format, right click on the record and select *Render to PDF* or *Render to Excel*.

If report is no longer needed, right click on record and select *Delete* to completely remove it from server.

38.2 Installation

On Linux platforms where packages are provided reporting module is available in `netxms-reporting` package.

On Windows reporting module is a part of NetXMS server installer. Java 11 or later is required by reporting module.

38.3 Configuration

38.3.1 NetXMS Server

NetXMS server maintain persistent connection with reporting server on `localhost:4710`, but it can be changed in configuration.

Configuration Parameter	Description	Default Value
EnableReportingServer	Boolean on/off switch which enable integration	0
ReportingServerHostname	IP address or hostname of the reporting server	localhost
ReportingServerPort	Port number of the reporting server	4710

NetXMS server connects and maintains connection to reporting server on the given hostname and port. Via this connection reporting server receives all necessary configuration and database credentials that are needed for operation.

38.3.2 Reporting Server

Reporting module has so-called workspace directory which contains report definitions (in “definitions” subdirectory) and intermediate report data (in “output” subdirectory).

On Linux for reporting module installed from packages workspace directory is `/var/lib/netxms/nxreportd`.

If `$NETXMS_HOME` environment variable is set, workspace directory is `$NETXMS_HOME/var/lib/nxreportd`.

On Windows workspace directory is located `var\nxreportd` in NetXMS installation folder, for default installation location it's `C:\NetXMS\var\nxreportd`.

38.3.3 Report definitions

Report definitions are .jar files prepared by Jaspersoft® Studio. During operation reporting server scans workspace/definitions directory for *.jar files. Each file is unpacked into it's own folder based on jar name (e.g. “report1.jar” will be unpacked into “report1”). Each archive should contain at least one file - “main.jrxml”, which is main report definition. It can also contain subreports, images - or anything else, supported by Jasper Reports. Any additional resources should be referenced using paths relative to root folder of unpacked report, which is set as additional parameter “SUBREPORT_DIR” (e.g. “\${SUBREPORT_DIR}/logo.png”).

Archive can also contain java code, which will be used as data provider (instead of querying SQL database). Reporting server will try to load class “report.DataSource”, which should implement interface “com.radensolutions.reporting.custom.NXCLDataSource” (attached sample: Event Processing Policy). Query string language in jrxml should be set to “nxcl” (default - SQL).

Simplest way to create jar files are using Maven, empty project is provided in samples archive. Running “mvn package” will produce complete jar file in “target” directory.

IMAGE LIBRARY

All images used on maps or as rack, chassis or chassis module images must be uploaded to the Image Library first. It is possible to upload, delete and update images. They can be organized in categories.

Image Library

Name	MIME type	Protected	GUID
▼ Network Objects			
ATM	image/png	Yes	1ddb76a3-a05f-4a42-acda-2202...
HSM	image/png	Yes	b314cf44-b2aa-478e-b23a-73bc...
Node	image/png	Yes	904e7291-ee3f-41b7-8132-2bd...
Printer	image/png	Yes	f5214d16-1ab1-4577-bb21-063...
Router	image/png	Yes	bacde727-b183-4e6c-8dca-ab02...
Server	image/png	Yes	ba6ab507-f62d-4b8f-824c-ca9d...
Service	image/png	Yes	092e4b35-4e7c-42df-b9b7-d580...
Switch	image/png	Yes	f9105c54-8dcf-483a-b387-b458...
Unknown	image/png	Yes	7cd999e9-fbe0-45c3-a695-f845...
▼ Rack Images			
h3c-4500-series	image/png	No	4775168e-2ada-410a-9866-8f0d...
msa2012_back_panel	image/png	No	a0b69f86-9557-40c5-8148-663...
msa2012_controller	image/png	No	722d70fe-db84-4ba0-aeab-6837...
msa2012_front	image/png	No	ba82dd64-6e6c-4b03-92f8-debb...

msa2012_front

1676 x 297

Tips:

- Images on maps are displayed without scaling.

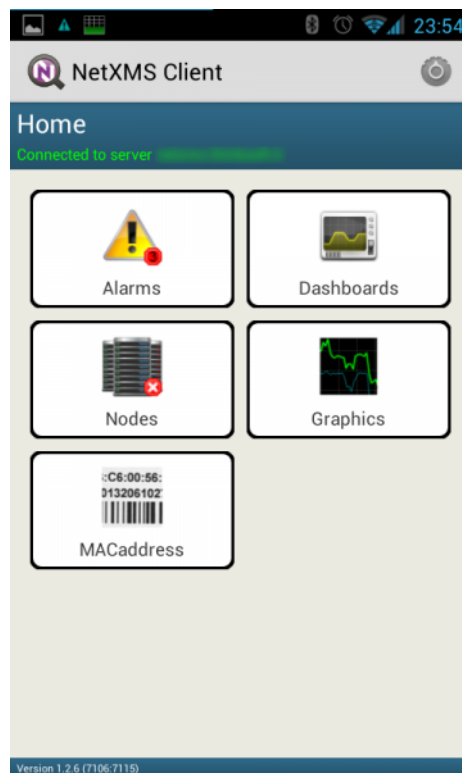
MOBILE CLIENT

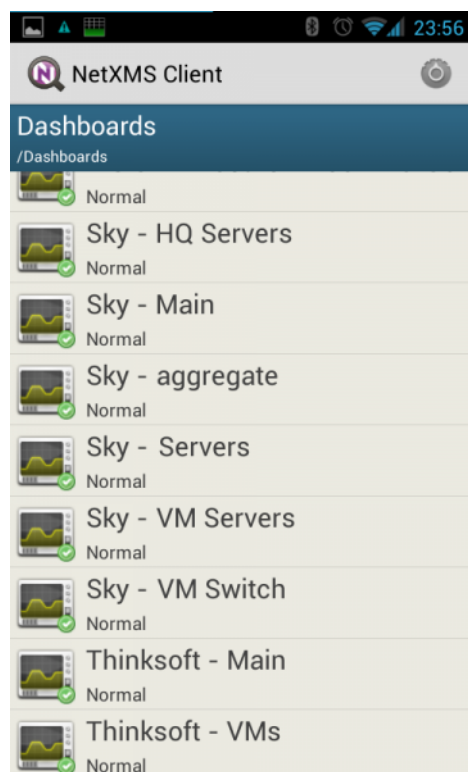
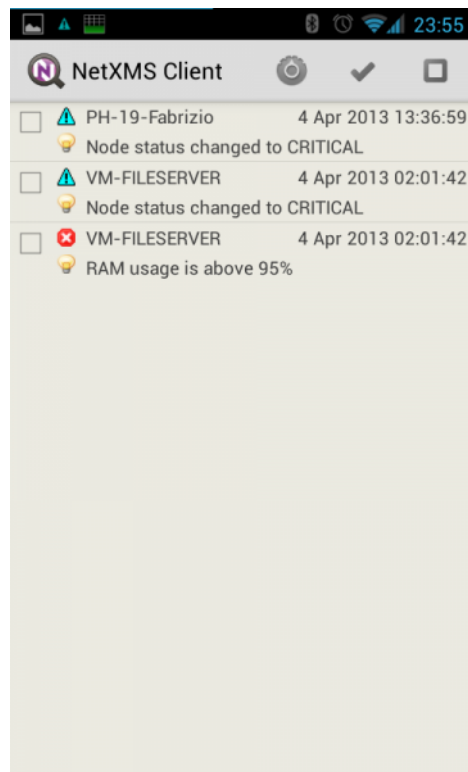
NetXMS mobile client is a monitoring tool for Android devices running version 2.2. and later.

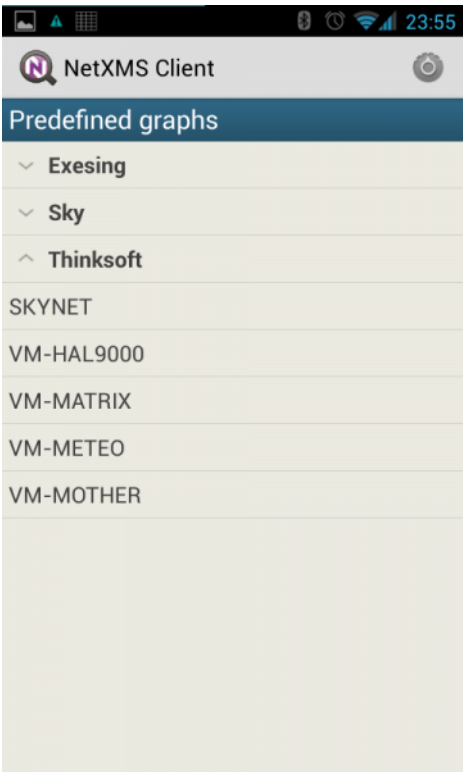
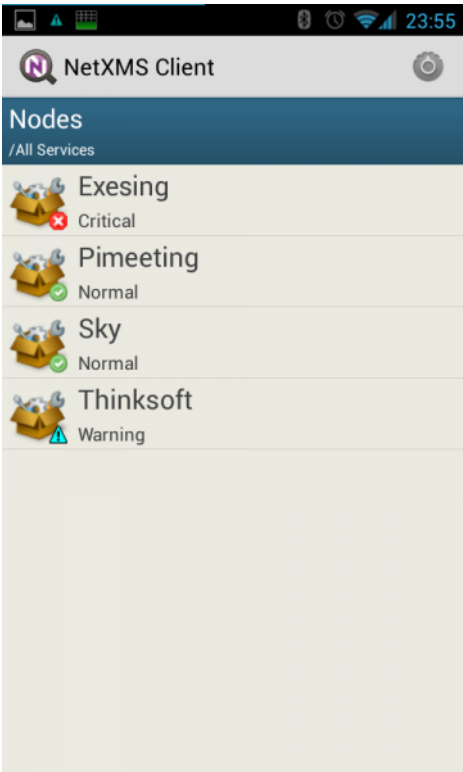
Currently, only a small subset of the functions present in the Desktop/Web edition are implemented, mainly read/only operations. The next paragraphs briefly describes each section.

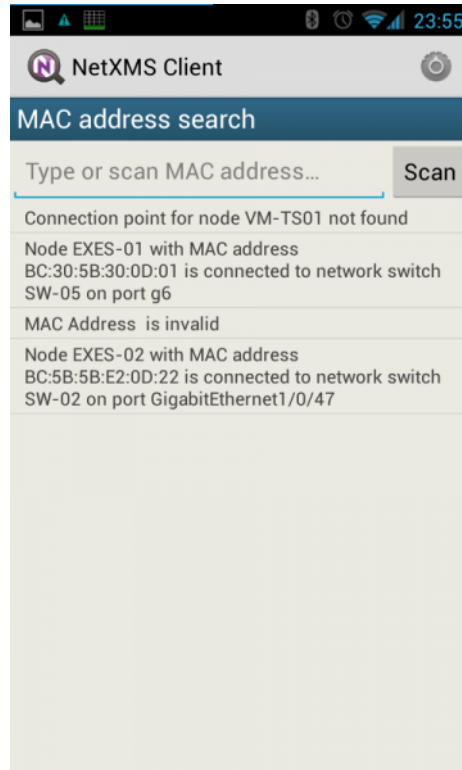
40.1 Main window

Here you can see how appears the main window and the underneath levels.









From the main window it is possible to get access to the following menu items:

- *Settings*: select this item to configure the client.
- *Reconnect*: select this item to force a reconnection to the server to gather new collected data.
- *Disconnect & Exit*: select this item to stop the client and exit from the app.

Underneath levels have menu that are context dependent, a detailed description can be found in each section.

40.2 Alarms

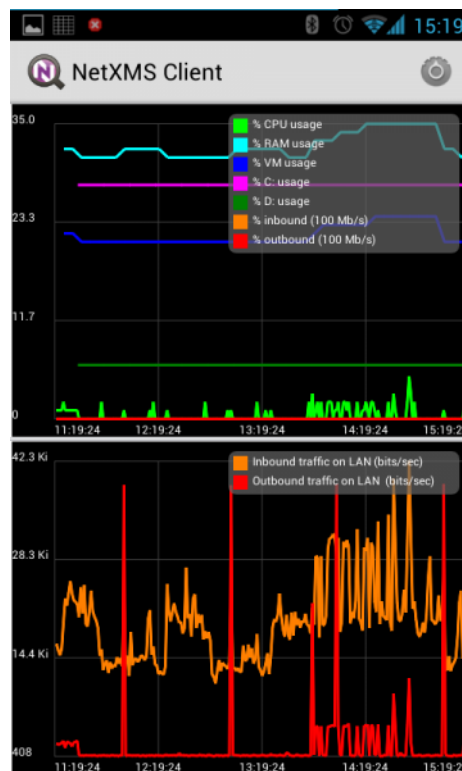
Alarms section is used to list and manage all pending alarms, eventually filtered on a particular node/container. Through this view it is possible to manage alarms:

- **Actions:**
 - *Acknowledge*: acknowledge the alarm.
 - *Sticky acknowledge*: sticky acknowledge the alarm.
 - *Resolve*: resolve the alarm.
 - *Terminate*: terminate the alarm.
 - *View last values*: jump to the node info section to view the last values for the node that generated the alarm.
- **Sort:**
 - *Sort by severity ascending*: sort list using event severity as criteria, ascending.
 - *Sort by severity descending*: sort list using event severity as criteria, descending.

- *Sort by date ascending*: sort list using date of event as criteria, ascending.
- *Sort by date descending*: sort list using date of event as criteria, descending.
- *Sort by node name ascending*: sort list using node name that generated the event as criteria, ascending.
- *Sort by node name descending*: sort list using node name that generated the event as criteria, descending.
- *Select all*: select all the alarms from the list
- *Unselect all*: clear any selection of alarms from the list

40.3 Dashboard

Dashboards are defined by administrator and allow to combine any available visualization components with data from multiple sources in order to create high-level views to see network (or parts of it) health at a glance. Not all elements are currently available for the mobile client, dashboards are properly refreshed according to their schedule. Due to dashboard size, keep in mind that Smartphones cannot be the best device to show them, a tablet is much more suitable device. Here an example:

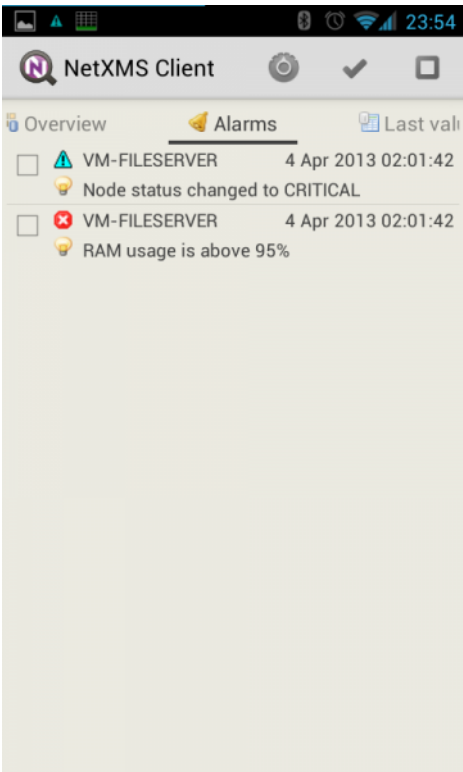
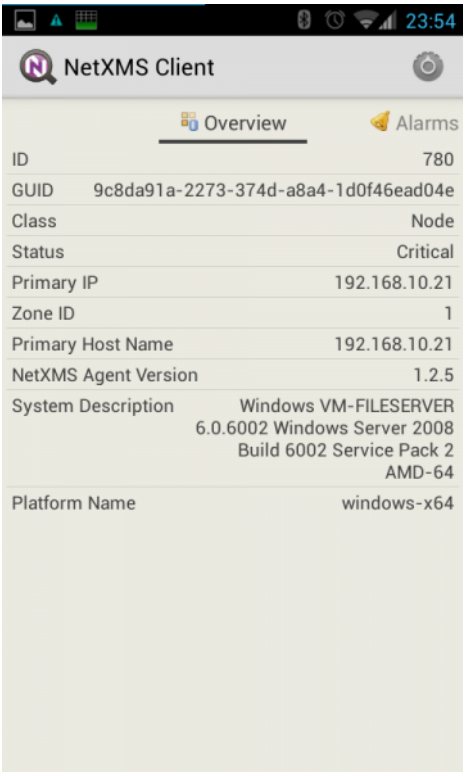


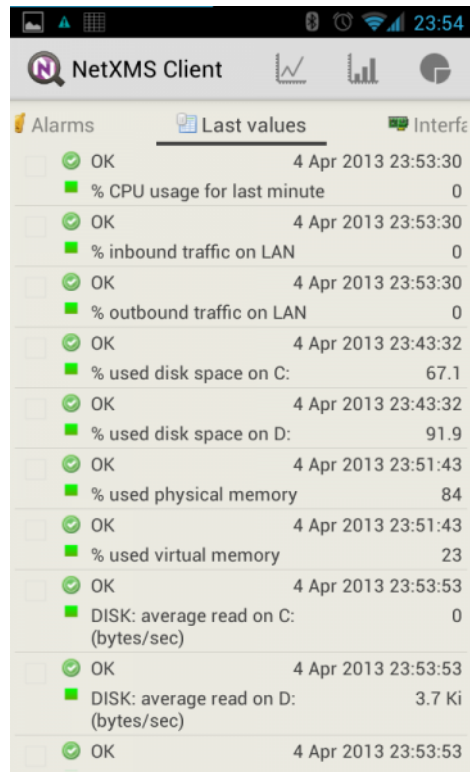
40.4 Nodes

This section is used to list and manage all nodes (all network infrastructure monitored by NetXMS are represented as a set of objects. Each object represents one physical or logical entity, or group of them). Objects can be organized into hierarchical structure, the Nodes section is used to explore them. In the right bottom corner of the icon there is a symbol that indicates the status of the node/container following the same symbology used on the desktop client. Clicking on a container will show the items inside, continuing to click up to an object will show a set of swipeable pages:

- *Overview*: here are presented the main info associated to this node, such as the name, the primary IP, the status, etc.

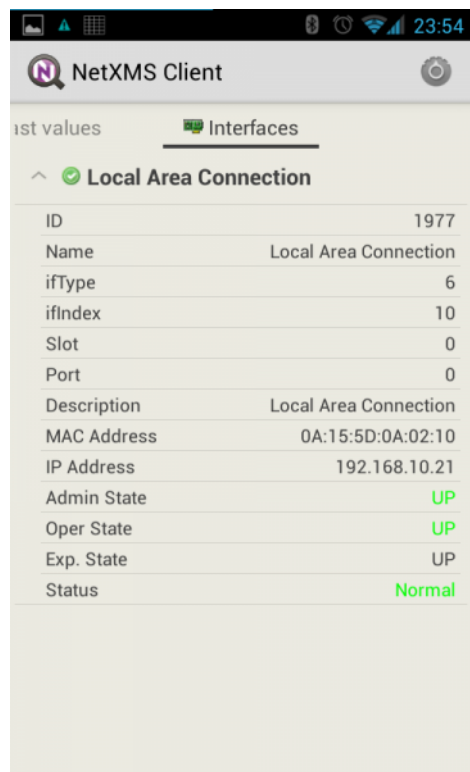
- *Alarms*: here are presented the list of pending alarms (if any) for this node, with the possibility to manage them with the following commands:
 - *Actions*:
 - * *Acknowledge*: acknowledge the alarm.
 - * *Sticky acknowledge*: sticky acknowledge the alarm.
 - * *Resolve*: resolve the alarm.
 - * *Terminate*: terminate the alarm.
 - * *View last values*: jump to the node info section to view the last values for the node that generated the alarm.
 - *Select all*: select all the alarms from the list
 - *Unselect all*: clear any selection of alarms from the list
- *Last values*: here are presented the DCI collected for this node, as well as the possibility to draw the following graphics (for one or more values):
 - *Last half hour*: draw one or more line graphs for the last half hour collected values
 - *Last hour*: draw one or more line graphs for the last hour collected values
 - *Last two hours*: draw one or more line graphs for the last two hours collected values
 - *Last four hours*: draw one or more line graphs for the last four hours collected values
 - *Last day*: draw one or more line graphs for the last day collected values
 - *Last week*: draw one or more line graphs for the last week collected values
 - *Bar chart*: draw a bar chart with the last collected value
 - *Pie chart*: draw a pie chart with the last collected value
- *Interfaces*: here are presented all the interfaces associated to this node. For each interface it is possible to instruct the following commands:
 - *Manage*: interface will be put in manage state
 - *Unmanage*: interface will be put in unmanaged state
 - *Change expected state*: change the expected interface state, possible values:
 - * *UP*: interface expected state will be put in UP state
 - * *DOWN*: interface expected state will be put in DOWN state
 - * *IGNORE*: interface expected state will be put in IGNORE state
- *Find switch port*: will start the search for a connection point (if available)





The screenshot shows the 'Alarms' tab in the NetXMS Client. It displays a list of system alarms, each with a status icon (green checkmark for OK), a description, a timestamp, and a value.

Status	Description	Timestamp	Value
OK	% CPU usage for last minute	4 Apr 2013 23:53:30	0
OK	% inbound traffic on LAN	4 Apr 2013 23:53:30	0
OK	% outbound traffic on LAN	4 Apr 2013 23:53:30	0
OK	% used disk space on C:	4 Apr 2013 23:43:32	67.1
OK	% used disk space on D:	4 Apr 2013 23:43:32	91.9
OK	% used physical memory	4 Apr 2013 23:51:43	84
OK	% used virtual memory	4 Apr 2013 23:51:43	23
OK	DISK: average read on C: (bytes/sec)	4 Apr 2013 23:53:53	0
OK	DISK: average read on D: (bytes/sec)	4 Apr 2013 23:53:53	3.7 Ki
OK		4 Apr 2013 23:53:53	

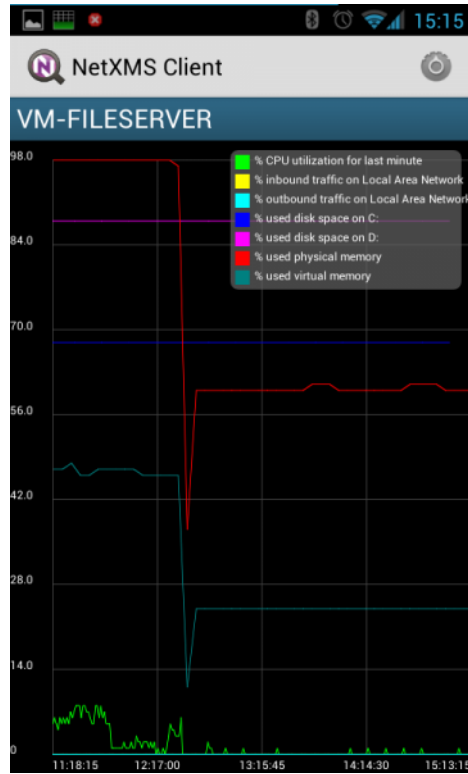


The screenshot shows the 'Interfaces' tab in the NetXMS Client. It displays details for the 'Local Area Connection' interface.

Property	Value
ID	1977
Name	Local Area Connection
ifType	6
ifIndex	10
Slot	0
Port	0
Description	Local Area Connection
MAC Address	0A:15:5D:0A:02:10
IP Address	192.168.10.21
Admin State	UP
Oper State	UP
Exp. State	UP
Status	Normal

40.5 Graphics

Predefined graphics are defined by administrator and can be used to view collected data in a graphical form (as a line chart). Currently, the mobile client doesn't autorefresh the content of the graphic selected. Here an example of a predefined graphs:



40.6 MACaddress

This section is used to list previously searched MAC addresses or to start a new search by scanning a barcode value (this feature needs the installation of Barcode Scanner from Zxing Team - freely available on the Google Play), by input it manually or by getting it directly from a node via the "Find Switch port" command.

40.7 Settings

This section is used to configure the behavior of the client.

40.8 Global settings

- *Autostart on boot*: check to automatically start the agent on boot (to be effective, app must not be moved to SD card).

40.9 Connection

40.9.1 Parameters

Allows selecting the parameters used to connect to the server:

- *Server*: address of the server (IP or name).
- *Port*: port of the server (default 4701).
- *User name*: username to connect to the server.
- *Password*: password to connect to the server.
- *Encrypt connection*: when selected challenges an encryption strategy with the server (depending on supported/configured providers).

40.9.2 Scheduler

Enables the possibility to define periodic connections to the server. If the scheduler is not enabled the app will try to connect to the server every time it detects a new connection (data or WiFi) and remains always connected as far as the connection remains active:

- *Enable scheduler*: check this to enable the scheduler.
- *Frequency (min)*: amount of time, in minutes, that has to elapse between each tentative of connection to the server to send the gathered info.
- *Duration (min)*: amount of time, in minutes, that has to elapse before disconnect from the server.
- **Daily scheduler: provides the ability to define a “one range” daily on which the agent is operational. Out of the specified range the app will not try to connect to the server to gather the new events:**
 - *Daily activation on*: start time for daily activation.
 - *Daily activation off*: stop time for daily activation.

40.10 Notifications

40.10.1 Connection status

This section is to manage the notifications related to the connection status.

- **Notification behavior: defines which kind of action should trigger notifications to the user. Possible options:**
 - Never: ignore connection status
 - When connected: notify when connection is successful
 - When disconnected: notify when connection is unsuccessful
 - Always: notify either connection successful and connection unsuccessful
- *Toast notification*: provides connection notification via “toast” , behavior is defined by “Notification behavior”.

- *Icon notification*: provides connection notification via icon in the status bar, behavior is defined by “Notification behavior”.

40.10.2 Alarms

- *Alarms notification*: select to enable alarms notification in the status bar.
- *Alarms sound by severity*: for each of the following categories:
 - *Normal*
 - *Warning*
 - *Minor*
 - *Major*
 - *Critical*

40.11 Interface

40.11.1 Multipliers

Allows to select the preferred multipliers to be used to show values. Allowed options: * *None*: do not apply multiplier, values are extended. * *Decimal*: applies a decimal multiplier (power of 10, e.g. 1000 -> 1K, 1000000 -> 1M, ...) * *Binary*: applies a binary multiplier (power of 2, e.g. 1024 -> 1Ki, 1048576 -> 1Mi, ...)

40.11.2 Graph text size

Allows to set the text size to be used for axis labels (if the default value is too small for high density devices).

40.11.3 Show legend in graphs

Allows to select to show or not the legend in the top right angle of the graphs. Since legend can be intrusive, especially when there are several lines plotted, user can select to disable the legend.

WEB API/REST API

41.1 Introduction

The NetXMS WebAPI is being developed to support larger integration possibilities for the NetXMS server and is based on the RESTful philosophy. API calls are REST-like (although not purely RESTful) and uses JSON for data exchange. The API currently supports Grafana integration and some additional parameters for integration. The NetXMS WebAPI is currently in very early development!

Information about Grafana configuration can be found [here](#).

41.2 Installation

41.2.1 Requirements

- A running instance of the NetXMS server.
- Access to a web server.

41.2.2 Setup

1. Download netxms-websvc-VERSION.war (example: netxms-websvc-2.2.15.war) file from <http://www.netxms.org/download> page.
2. Copy the downloaded .war file to your web server.

By default localhost address is used to connect to NetXMS Server. To specify server address or other parameters, create a `nxapisrv.properties` file and place it in the property file location of your web server. File should have parameters in ini format: NAME=VALUE. The following parameters are supported:

- netxms.server.address
- netxms.server.enableCompression
- netxms.server.port
- session.timeout

Configuration example:

```
netxms.server.address=server.office.radensolutions.com
netxms.server.port=44701
```

41.3 Implemented functionality

41.3.1 Authentication

Login

Any user account configured in NetXMS can be used to authenticate to Rest API, however this user should have access right to objects that will be requested through the API.

There are 3 implemented options of authentication:

1. Basic authentication for Rest API session creation, more information can be found on [Wikipedia](#)
2. Through POST request for Rest API session creation
3. Through POST request to allow external software user authentication using NetXMS user accounts. To be able to login using this authentication type, user account should have “External tool integration account” access right set.

Creating Rest API session:

Request type: **POST**

JSON data:

```
{ "login": "admin", "password": "netxms" }
```

Request path: *API_HOME*/sessions

Return data:

On success server will set cookie *session_handle* and json with session GUID and server version. When performing subsequent requests, session GUID should be provided in *Session-Id*: field of request's header or the cookie should be passed.

Performing external authentication:

Request type: **POST**

JSON data:

```
{ "login": "admin", "password": "netxms" }
```

Request path: *API_HOME*/authenticate

Return data:

The API will return a 200 response if the credentials are correct, a 400 response if either login or password is not provided or 401 if the provided credentials are incorrect.

Authentication used to gain Rest API session.

Logout

To log out request with given session ID.

Request type: **DELETE**

Request path: *API_HOME*/sessions/{**sid**}

Return data:

The API will return a 200 response if log out succeed.

41.3.2 Objects

Get multiple objects with filters

Request to get all objects available to this user or to get objects that fulfill filter requirements and are available to this user.

Request type: **GET**

Request path: *API_HOME*/objects

Filter options:

- *area=geographical area*
- *class=comma-separated class list*
- *name=pattern or regex, if useRegex=true*
- *parent=parent object id*
- *topLevelOnly=boolean - select top level objects only. false by default*
- *useRegex=boolean - treat name and custom attribute value as regex. false by default*
- *zone=comma-separated list of zone UINs*
- *@custom_attribute_name=pattern or regex, if useRegex=true*

Return data:

Will return filtered objects or all objects available to user.

Get object by id

Request to get exact object identified by ID or GUID.

Request type: **GET**

Request path: *API_HOME*/objects/{**object-id**}

Return data:

Object information identified by provided ID or GUID.

Create object

Request to create new object.

Request type: **POST**

JSON data:

JSON object can contain fields from 2 filed entities:

- *Creation fields*
- *Modification fields*

Minimal JSON for node creation under “Infrastructure Services” object:

```
{ "objectType": 2, "name": "testNode", "parentId": 2, "primaryName": "10.5.0.12"
  ↪ }
```

Minimal JSON for container creation under “Infrastructure Services” object:

```
{ "objectType": 5, "name": "New container", "parentId": 2 }
```

Request path: *API_HOME*/objects

Return data:

New object ID.

```
{ "id": 15130 }
```

Update object

Request to update object.

Request type: **PATCH**

Request path: *API_HOME*/objects/{**object-id**}

JSON data:

JSON object can contain *Modification fields*.

Fields that are not set will not be updated. Array elements will be replaced fully (if new array does not contain old elements - they will be deleted).

Json to update object's custom attributes (json should contain all custom attributes, attributes that are not part of JSON will be deleted):

```
{
  "customAttributes": {
    "test attr2": {
      "value": "new value"
    },
    "test attr": {
      "value": "new value"
    }
  }
}
```

Get object by id

Request to delete object.

Request type: **DELETE**

Request path: *API_HOME*/objects/{**object-id**}

Return data:

Object information identified by provided ID or GUID.

Creation fields

This list represents all fields that are object creation fields. Note that this is common list for any type of object.

Field name	Type	Comment
objectType	Integer	Possible options: <ul style="list-style-type: none"> • SUBNET: 1 • NODE: 2 • INTERFACE: 3 • NETWORK: 4 • CONTAINER: 5 • ZONE: 6 • SERVICEROOT: 7 • TEMPLATE: 8 • TEMPLATEGROUP: 9 • TEMPLATEROOT: 10 • NETWORKSERVICE: 11 • VPNCONNECTOR: 12 • CONDITION: 13 • CLUSTER: 14 • OBJECT_BUSINESSSERVICE_PROTOTYPE: 15 • NETWORKMAPROOT: 19 • NETWORKMAPGROUP: 20 • NETWORKMAP: 21 • DASHBOARDROOT: 22 • DASHBOARD: 23 • BUSINESSSERVICEROOT: 27 • BUSINESSSERVICE: 28 • NODELINK: 29 • SLMCHECK: 30 • MOBILEDEVICE: 31 • RACK: 32 • ACCESSPOINT: 33 • CHASSIS: 35 • DASHBOARDGROUP: 36 • SENSOR: 37
name	String	Object name
parentId	Long	Parent object id this object to be created under
comments	String	Object comment
creationFlags	Integer	Bit flags for object creation. Possible options: <ul style="list-style-type: none"> • DISABLE ICMP: 0x0001 • DISABLE NXCP: 0x0002 • DISABLE SNMP: 0x0004 • CREATE UNMANAGED: 0x0008 • ENTER MAINTENANCE: 0x0010 • AS ZONE PROXY: 0x0020 • DISABLE ETHERNET IP: 0x0040 • SNMP SETTINGS LOCKED: 0x0080 • EXTERNAL GATEWAY: 0x0100
primaryName	String	Node primary name (IP address or dns name)
agentPort	Integer	Node agent port
snmpPort	Integer	Node SNMP port
etherNetIpPort	Integer	Node ethernetIP port

continues on next page

Table 1 – continued from previous page

Field name	Type	Comment
sshPort	Integer	Node ssh port
ipAddress	String	Interface IP address
agentProxyId	Long	Node agent proxy id
snmpProxyId	Long	Node SNMP proxy id
etherNetIpProxyId	Long	Node ethernetIP proxy id
icmpProxyId	Long	Node ICMP proxy id
sshProxyId	Long	Node ssh proxy id
mapType	Integer	Network map type
seedObjectIds	Long[]	Network map seed objects
zoneUIN	Integer	Subnet/Node/Zone zone UIN
serviceType	Integer	Network service types: <ul style="list-style-type: none"> • CUSTOM: 0 • SSH: 1 • POP3: 2 • SMTP: 3 • FTP: 4 • HTTP: 5 • HTTPS: 6 • TELNET: 7
ipPort	Integer	Network Service IP port
request	String	Network Service request
response	String	Network Service response
linkedNodeId	Long	Linked object for Node Link object
template	Boolean	If service check object is template
macAddress	String	Interface or sensor MAC address
ifIndex	Integer	Interface index
ifType	Integer	Interface type
module	Integer	Interface module number
port	Integer	Interface port
physicalPort	Boolean	IF interface has physical port
createStatusDci	Boolean	IF status DCI should be created for network service
deviceId	String	Mobile device ID
height	Integer	Rack height
controllerId	Long	Chassis controller node id
sshLogin	String	Node ssh login
sshPassword	String	Node password
deviceClass	Integer	Sensor device class
vendor	String	Sensor vendor
commProtocol	Integer	Sensor communication protocol
xmlConfig	String	Sensor XML config
xmlRegConfig	String	Sensor XML registration config
serialNumber	String	Sensor serial number
deviceAddress	String	Sensor device address
metaType	String	Sensor meta type
description	String	Sensor description
sensorProxy	Long	Sensor proxy node id

continues on next page

Table 1 – continued from previous page

Field name	Type	Comment
instanceDiscoveryMethod	Business service instance discovery method	Possible values: <ul style="list-style-type: none"> • IDM_AGENT_LIST - 1 • IDM_AGENT_TABLE - 2 • IDM_SCRIPT - 5

Modification fields

Note

Starting from version 4 isAutoBindEnabled and isAutoUnbindEnabled replaced by autoBindFlags

Field name	Type	Comment
name	String	
primaryName	String	
alias	String	
nameOnMap	String	
acl	<i>AccessListElement</i> []	inheritAccessRights should be provided in the same request
inheritAccessRights	Boolean	acl should be provided in the same request
customAttributes	JSON object {String: <i>CustomAttribute</i> }	Object name is custom attribute name and value is in <i>CustomAttribute</i> object
autoBindFilter	String	
version	Integer	
description	String	
agentPort	Integer	
agentSecret	String	
agentProxy	Long	
snmpPort	Integer	
snmpVersion	String	Node SNMP version: <ul style="list-style-type: none"> • V1 • V2C • V3 • DEFAULT
snmpAuthMethod	Integer	snmpAuthName, snmpAuthPassword, snmpPrivPassword, snmpPrivMethod should be provided in the same request
snmpPrivMethod	Integer	snmpAuthName, snmpAuthPassword, snmpPrivPassword, snmpAuthMethod should be provided in the same request
snmpAuthName	String	snmpAuthPassword, snmpPrivPassword, snmpAuthMethod, snmpPrivMethod should be provided in the same request
snmpAuthPassword	String	snmpAuthName, snmpPrivPassword, snmpAuthMethod, snmpPrivMethod should be provided in the same request

continues on next page

Table 2 – continued from previous page

Field name	Type	Comment
snmpPrivPassword	String	snmpAuthName, snmpAuthPassword, snmpAuthMethod, snmpPrivMethod should be provided in the same request
snmpProxy	Long	
icmpProxy	Long	
trustedNodes	Long[]	
geolocation	<i>Geolocation</i>	
mapBackground	String	UUID. mapBackgroundLocation, mapBackgroundLocation, mapBackgroundZoom, mapBackgroundColor should be provided in the same request.
mapBackgroundLocation	<i>Geolocation</i>	mapBackground, mapBackgroundLocation, mapBackgroundZoom, mapBackgroundColor should be provided in the same request.
mapBackgroundZoom	Integer	mapBackground, mapBackgroundLocation, mapBackgroundLocation, mapBackgroundColor should be provided in the same request.
mapBackgroundColor	Integer	mapBackground, mapBackgroundLocation, mapBackgroundLocation, mapBackgroundZoom should be provided in the same request.
mapImage	String	UUID
columnCount	Integer	
script	String	
activationEvent	Integer	
deactivationEvent	Integer	
sourceObject	Long	
activeStatus	Integer	
inactiveStatus	Integer	
drillDownObjectId	Long	
pollerNode	Long	
requiredPolls	Integer	
serviceType	Integer	
ipProtocol	Integer	
ipPort	Integer	
ipAddress	String	Network service IP address
request	String	Network service IP request
response	String	Network service IP response
objectFlags	Integer	Object flags specific for each object. Possible values can be found in NXSL documentation under each object. (Example: Node flags) objectFlagsMask should be provided in the same request.
objectFlagsMask	Integer	Bitmask that defines which bits in objectFlags will have effect. objectFlags should be provided in the same request.
ifXTablePolicy	Integer	
reportDefinition	String	
networkList	String[]	IP address list
statusCalculationMethod	Integer	
statusPropagationMethod	Integer	

continues on next page

Table 2 – continued from previous page

Field name	Type	Comment
fixedPropagatedStatus	String	Object status: <ul style="list-style-type: none"> • NORMAL • WARNING • MINOR • MAJOR • CRITICAL • UNKNOWN • UNMANAGED • DISABLED • TESTING
statusShift	Integer	
statusTransformation	ObjectStatus[]	Object status mapping list. Possible values: <ul style="list-style-type: none"> • NORMAL • WARNING • MINOR • MAJOR • CRITICAL • UNKNOWN • UNMANAGED • DISABLED • TESTING
statusSingleThreshold	Integer	
statusThresholds	Integer[]	
expectedState	Integer	
linkColor	Integer	
connectionRouting	Integer	
discoveryRadius	Integer	
height	Integer	
filter	String	
peerGatewayId	Long	
localNetworks	String[]	VPN networks IP address. remoteNetworks should be provided in the same request.
remoteNetworks	String[]	VPN networks IP address. localNetworks should be provided in the same request.
postalAddress	<i>PostalAddress</i>	
agentCacheMode	String	Possible values: <ul style="list-style-type: none"> • DEFAULT • ON • OFF
agentCompressionMode	String	Possible values: <ul style="list-style-type: none"> • DEFAULT • ENABLED • DISABLED

continues on next page

Table 2 – continued from previous page

Field name	Type	Comment
mapObjectDisplayMode	String	Possible values: <ul style="list-style-type: none"> • ICON • SMALL_LABEL • LARGE_LABEL • STATUS • FLOOR_PLAN
physicalContainerObjectId	Long	
rackImageFront	String	UUID. rackImageRear, rackPosition, rackHeight, rackOrientation should be provided in the same request.
rackImageRear	String	UUID. rackImageFront, rackPosition, rackHeight, rackOrientation should be provided in the same request.
rackPosition	Short	rackImageFront, rackImageRear, rackHeight, rackOrientation should be provided in the same request.
rackHeight	Short	rackImageFront, rackImageRear, rackPosition, rackOrientation should be provided in the same request.
rackOrientation	String	Possible values: <ul style="list-style-type: none"> • FILL • FRONT • REAR rackImageFront, rackImageRear, rackPosition, rackHeight should be provided in the same request.
dashboards	Long[]	
rackNumberingTopBottom	Boolean	
controllerId	Long	
chassisId	Long	
sshProxy	Long	
sshLogin	String	
sshPassword	String	
sshPort	Integer	
sshKeyId	Integer	
zoneProxies	Long[]	
urls	ObjectUrl[]	
seedObjectIds	Long[]	
macAddress	String	Sensor mac address
deviceClass	Integer	
vendor	String	
serialNumber	String	
deviceAddress	String	
metaType	String	
sensorProxy	Long	
xmlConfig	String	
snmpPorts	String[]	
responsibleUsers	Long[]	

continues on next page

Table 2 – continued from previous page

Field name	Type	Comment
icmpStatCollectionMode	String	Possible values: <ul style="list-style-type: none"> • DEFAULT • ON • OFF
icmpTargets	String[]	ICMP ping targets IP addresses
chassisPlacement	String	
etherNetIPPort	Integer	
etherNetIPProxy	Long	
certificateMappingMethod	String	Possible values: <ul style="list-style-type: none"> • SUBJECT • PUBLIC_KEY • COMMON_NAME • TEMPLATE_ID certificateMappingData should be provided in the same request.
certificateMappingData	String	certificateMappingMethod should be provided in the same request.
categoryId	Integer	
geoLocationControlMode	GeoLocationControlMode	Possible values: <ul style="list-style-type: none"> • NO_CONTROL • RESTRICTED_AREAS • ALLOWED_AREAS
geoAreas	long[]	
instanceDiscoveryMethod	Business service instance discovery method	Possible values: <ul style="list-style-type: none"> • IDM_AGENT_LIST - 1 • IDM_AGENT_TABLE - 2 • IDM_SCRIPT - 5
instanceDiscoveryData	Business service instance discovery data	
instanceDiscoveryFilter	Business service instance discovery data filtering script	
autoBindFilter2	Second binding script used for DCI binding. Currently used in business service	
autoBindFlags	Auto bind bit flags	First script is currently used for object bind/unbind, second for dci bind/unbind. Possible values: <ul style="list-style-type: none"> • First script for auto bind is enabled - 0x0001 • First script for auto unbind is enabled - 0x0002 • Second script for auto bind is enabled - 0x0004 • Second script for auto unbind is enabled - 0x0008

continues on next page

Table 2 – continued from previous page

Field name	Type	Comment
objectStatusThreshold	Business service default threshold for auto created object checks	Possible values: <ul style="list-style-type: none"> • Default - 0 • Warning - 1 • Minor - 2 • Major - 3 • Critical - 4
dcStatusThreshold	Business service default threshold for auto created DCI checks	Possible values: <ul style="list-style-type: none"> • Default - 0 • Warning - 1 • Minor - 2 • Major - 3 • Critical - 4
sourceNode	Id of source node for business service instance discovery methods	

GeoLocation fields

Field name	Type	Comment
type	Integer	Available options: <ul style="list-style-type: none"> • UNSET: 0 • MANUAL: 1 • GPS: 2 • NETWORK: 3
latitude	Double	
longitude	Double	
accuracy	int	Location accuracy in meters
timestamp	Integer	UNIX timestamp

AccessListElement fields

Field name	Type	Comment
userId	Long	
accessRights	Integer	Bit flag field. Available options: <ul style="list-style-type: none"> • OBJECT ACCESS READ: 0x00000001 • OBJECT ACCESS MODIFY: 0x00000002 • OBJECT ACCESS CREATE: 0x00000004 • OBJECT ACCESS DELETE: 0x00000008 • OBJECT ACCESS READ ALARMS: 0x00000010 • OBJECT ACCESS ACL: 0x00000020 • OBJECT ACCESS UPDATE ALARMS: 0x00000040 • OBJECT ACCESS SEND EVENTS: 0x00000080 • OBJECT ACCESS CONTROL: 0x00000100 • OBJECT ACCESS TERM ALARMS: 0x00000200 • OBJECT ACCESS PUSH DATA: 0x00000400 • OBJECT ACCESS CREATE ISSUE: 0x00000800 • OBJECT ACCESS DOWNLOAD: 0x00001000 • OBJECT ACCESS UPLOAD: 0x00002000 • OBJECT ACCESS MANAGE FILES: 0x00004000 • OBJECT ACCESS MAINTENANCE: 0x00008000 • OBJECT ACCESS READ AGENT: 0x00010000 • OBJECT ACCESS READ SNMP: 0x00020000 • OBJECT ACCESS SCREENSHOT: 0x00040000

CustomAttributes fields

Field name	Type	Comment
value	String	Attribute value
flags	Long	Available options: <ul style="list-style-type: none"> • INHERITABLE: 1

PostalAddress fields

Field name	Type	Comment
country	String	
city	String	
streetAddress	String	
postcode	String	

Bind object

Request to bind object to container. Container id is specified in URL, object id in JSON.

Request type: **POST**

JSON data:

Bind object to object in URL:

```
{ "id": 15130 }
```

Request path: *API_HOME*/objects/{**object-id**}/bind

Bind node to

Request to bind object under container. Container id is specified in JSON, object id in URL.

Request type: **POST**

JSON data:

Bind object in URL to “Infrastructure service”:

```
{ "id": 2 }
```

Request path: *API_HOME*/objects/{**object-id**}/bind-to

Unbind node

Request to unbind object from container. Container id is specified in URL, object id in JSON.

Request type: **POST**

JSON data:

Unbind object from container in URL:

```
{ "id": 15130 }
```

Request path: *API_HOME*/objects/{**object-id**}/unbind

UnbindFrom node

Request to unbind object from container. Container id is specified in JSON, object id in URL.

Request type: **POST**

JSON data:

Unbind object in URL from “Infrastructure service”:

```
{ "id": 2 }
```

Request path: *API_HOME*/objects/{**object-id**}/unbind-from

Poll object

Create object poll request

Request type: **POST**

JSON data:

```
{ "type": "status" }
```

One of the following poll types:

- configuration full
- configuration
- discovery
- interface
- status
- topology

Request path: *API_HOME*/objects/{**object-id**}/polls

Return data:

Will return UUID of request, that should be used to get request output and request type.

```
{ "id": 15130,
  "type": "status" }
```

Get object poll data

Get object poll request data

Request type: **GET**

Request path: *API_HOME*/objects/{**object-id**}/polls/output/{**request-UUID**}

Return data:

Will return request output data.

```
{ "streamId": 0,
  "completed": false,
  "message": "Poll request accepted..." }
```

Change object zone

Added in version 4.4.4.

Request to move object to new zone. Zone UIN is specified in JSON, object id in URL.

Request type: **POST**

JSON data:

Move object specified in URL to “Default” zone:

```
{ "zoneUIN": 0 }
```

Request path: *API_HOME*/objects/{**object-id**}/change-zone

41.3.3 Business Services

Get checks

Request all business service checks

Request type: **GET**

Request path: *API_HOME*/objects/{**object-id**}/checks

Create new check

Create new business service check

Request type: **POST**

Request path: *API_HOME*/objects/{**object-id**}/checks

JSON data:

Create new script business service check:

```
{
  "checkType": "SCRIPT",
  "description": "Web created script",
  "script": "return OK;",
  "objectId": 0,
  "dcId": 0,
  "threshold": 0
}
```

Update existing check

Update existing business service check

Request type: **PUT**

Request path: *API_HOME*/objects/{**object-id**}/checks/**check-id**

JSON data:

Update existing business service check to object check with object ID “166”:

```
{
  "checkType": "OBJECT",
  "description": "Web created script",
  "script": "return OK;",
  "objectId": 166,
  "dcId": 0,
  "threshold": 0
}
```

Delete existing check

Delete existing business service check

Request type: **DELETE**

Request path: *API_HOME*/objects/{**object-id**}/checks/**check-id**

Get tickets

Get ticket list for given time range.

Request type: **GET**

Request path: *API_HOME*/objects/{**object-id**}/tickets

Time range can be requested in 2 ways.

First option is back from now with given parameters:

- *timeUnit=Type of time range. Possible values: MINUTE, HOUR, DAY*
- *timeRage=Range in given units*

Second option is fixe time range:

- *start=UNIX timestamp*
- *end=UNIX timestamp*

Get uptime

Get uptime for given time range.

Request type: **GET**

Request path: *API_HOME*/objects/{**object-id**}/uptime

Time range can be requested in 2 ways.

First option is back from now with given parameters:

- *timeUnit=Type of time range. Possible values: MINUTE, HOUR, DAY*
- *timeRage=Range in given units*

Second option is fixe time range:

- *start=UNIX timestamp*
- *end=UNIX timestamp*

41.3.4 Alarms

Full scope of currently active alarms can be obtained or object specific list.

Get multiple alarms with filters

Request to get all active alarms available to this user or to get active alarms that fulfill filter requirements and are available to this user.

Request type: **GET**

Request path: *API_HOME*/alarms

Filter options:

- alarm=*list of alarm states. Possible values: outstanding, acknowledged, resolved*
- createdBefore=*UNIX timestamp*
- createdAfter=*UNIX timestamp*
- objectId=*ID or related object*
- objectGuid=*GUID or related object*
- includeChildObjects=*boolean. Set to true to get alarms of container child objects*
- resolveReferences=*resolve IDs into human readable data*
- updatedBefore=*UNIX timestamp*
- updatedAfter=*UNIX timestamp*

Return data:

Will return filtered active alarms or all active alarms available to user.

Alarm by id

Request to get an alarm by it's ID.

Request type: **GET**

Request path: *API_HOME*/alarms/{**alarm-id**}

Return data:

Will return alarm specified by ID.

41.3.5 Data collection configuration

Get data collection configuration

Request type: **GET**

Request path: *API_HOME*/objects/{**object-id**}/data-collection

Filter options (all are case-insensitive):

- dciName=*text that name should contain*
- dciNameRegexp=*regular expression for name*
- dciDescription=*text that description should contain*
- dciDescriptionRegexp=*regular expression for description*

Return data:

Will return data collection configuration.

Create DCI

Request type: **POST**

Request path: *API_HOME*/objects/{**object-id**}/data-collection

JSON data:

Create new DCI (name, description and valueType are obligatory fields):


```
{
  "name": "Agent.Version",
  "description": "Version of agent",
  "origin": "AGENT",
  "pollingInterval": "120",
  "pollingScheduleType": "1",
  "retentionType": "1",
  "retentionTime": "60",
  "valueType": "single"
}
```

Note

valueType should be one of the following: * single * table

Update DCI

Request to get last values of DCI identified by ID for exact object identified by ID or GUID.

Request type: **PUT**

Request path: *API_HOME*/objects/{**object-id**}/data-collection/{**dci-id**}

JSON data:

Update existing DCI setting custom polling interval and custom retention time (name and description are obligatory fields):

```
{
  "name": "Agent.Version",
  "description": "Version of agent",
  "pollingInterval": "120",
  "pollingScheduleType": "1",
  "retentionType": "1",
  "retentionTime": "60"
}
```

41.3.6 DCI data**DCI values**

Request to get last values of DCI identified by ID for exact object identified by ID or GUID.

Request type: **GET**

Request path: *API_HOME*/objects/{**object-id**}/data-collection/{**dci-id**}/values

Filter options:

- from=*requested period start time as unix timestamp*
- to=*requested period end time as unix timestamp*
- timeInterval=*requested time interval in seconds*
- itemCount=*number of items to be returned*

Return data:

Will return DCI values for requested node limited by filters.

DCI last value

Request to get last value of DCI identified by ID for exact object identified by ID or GUID.

Request type: **GET**

Request path: *API_HOME*/objects/{**object-id**}/data-collection/{**dci-id**}/last-value

Filter options:

- rowsAsObjects=*true or false. Determines how table DCI is returned*

Return data:

Will return last value of DCI.

Object last values

Request to get DCI last values of object.

Request type: **GET**

Request path: *API_HOME*/objects/{**object-id**}/last-values

Filter options (all are case-insensitive):

- dciName=*text that name should contain*
- dciNameRegexp=*regular expression for name*
- dciDescription=*text that description should contain*
- dciDescriptionRegexp=*regular expression for description*

Return data:

Will return DCI last values of object.

Query last values

Request type: **GET**

Request path: *API_HOME*/objects/{**object-id**}/data-collection//query?query=**{filter string}**

Filter string options:

- NOT *negation of following filtering parameter*
- Description
- GUID
- Id
- Name
- PollingInterval
- RetentionTime
- SourceNode

Example filter string:

Name:FileSystem.UsedPerc PollingInterval:60

Adhoc summary table

Option to get last values for multiple nodes(for all nodes under provided container) for the same DCIs. Required DCIs and container are provided in request.

Request type: **POST**

Request path: *API_HOME*/summary-table/ad-hoc

POST request JSON

```
{
  "baseObject": "ContainerName",
  "columns": [
    {
      "columnName": "Free form name that will be used in return table for this column",
      "dciName": "Name of DCI, that will be used for filtering"
    },
    {
      "columnName": "Name2",
      "dciName": "DCIName2"
    }
  ]
}
```

Return data:

Will return adhoc summary table configured accordingly to request json.

41.3.7 Object tools

List of available object tools

Request to object tools available to specified object.

Request type: **GET**

Request path: *API_HOME*/objects/{**object-id**}/object-tools

Execute object tool

Request to object tools available to specified object.

Request type: **POST**

Request path: *API_HOME*/objects/{**object-id**}/object-tools

JSON data:

```
{
  "toolData": {
    "id": "1234",
    "inputFields": {
      "field1": "value1",
      "field2": "1000"
    }
  }
}
```

Return data:

Will return JSON with UUID and toolId. UUID can be supplied to this endpoint (with GET request) to view object tool output: *API_HOME*/objects/{**object-id**}/object-tools/output/{**uuid**}. With POST request to the same endpoint execution of object tool can be stopped.

41.3.8 Persistent storage

Get all persistent storage variables

Request to get all persistent storage variables available to this user.

Request type: **GET**

Request path: *API_HOME*/persistent-storage

Return data:

Will return all persistent storages in “key”:”value” format.

Get persistent storage variable by key

Request to get persistent storage value by key.

Request type: **GET**

Request path: *API_HOME*/persistent-storage/{**key**}

Return data:

Will return corresponding persistent storages value in “value”:”value” format.

Create persistent storage variable

Request to create new persistent storage variable.

Request type: **POST**

JSON data:

JSON object should contain two fields: key and value.

```
{ "key": "a" }  
{ "value": "10" }
```

Request path: *API_HOME*/persistentstorage

Return data:

Will return newly created persistent storages in “key”:”value” format.

Update persistent storage variable

Request to update specified persistent storage variable value.

Request type: **PUT**

JSON data:

JSON object should contain one field: new value.

```
{ "value": "10" }
```

Request path: *API_HOME*/persistentstorage/{key}

Return data:

Will return updated persistent storages in “key”:”value” format.

Delete persistent storage variable

Request to delete persistent storage variable.

Request type: **DELETE**

Request path: *API_HOME*/persistentstorage/{key}

41.3.9 User agent notifications

TODO

41.3.10 Push DCI data

Request to push values for one or multiple DCIs. Node and DCI can be specified either by id or by name. If both id and name are provided, id has priority.

Request type: **POST**

JSON data:

To send value for one DCI JSON object should contain the following:

```
{
  "nodeId" : 10,
  "dciId" : 20,
  "value" : "Value"
}
```

Or, alternatively using node and DCI names:

```
{
  "nodeName" : "Node name",
  "dciName" : "DCI name",
  "value" : "Value"
}
```

To send value for several DCIs JSON object should contain an array:

```
[
  {
    "nodeId" : 10,
    "dciId" : 20,
    "value" : "Value"
  },
  {
    "nodeName" : "Node name",
    "dciName" : "DCI name",
    "value" : "Value"
  }
]
```

Request path: *API_HOME*/pushData

41.3.11 Predefined graphs

TODO

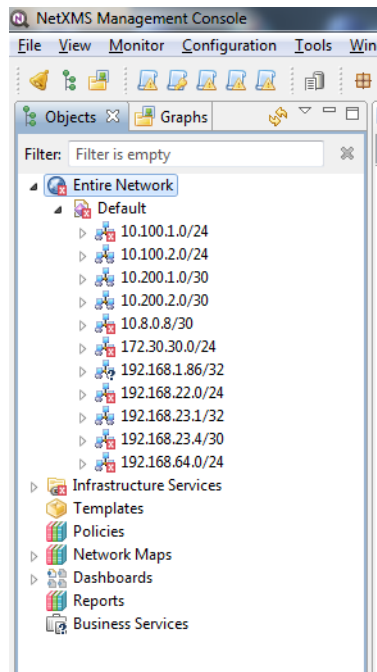
ADVANCED TOPICS

42.1 Zones

As NetXMS server keeps track of an IP topology, it is important to maintain the configuration in which IP addresses do not overlap and that two IP addresses from same subnet are really within one subnet. Sometimes, however, it is needed to monitor multiple sites with overlapping IP address ranges. To correctly handle such situation, zoning must be used. Zone in NetXMS is a group of IP subnets which form non-overlapping IP address space. There is always zone 0 which contains subnets directly reachable by management server. For all other zones server assumes that subnets within that zones are not reachable directly, and proxy must be used.

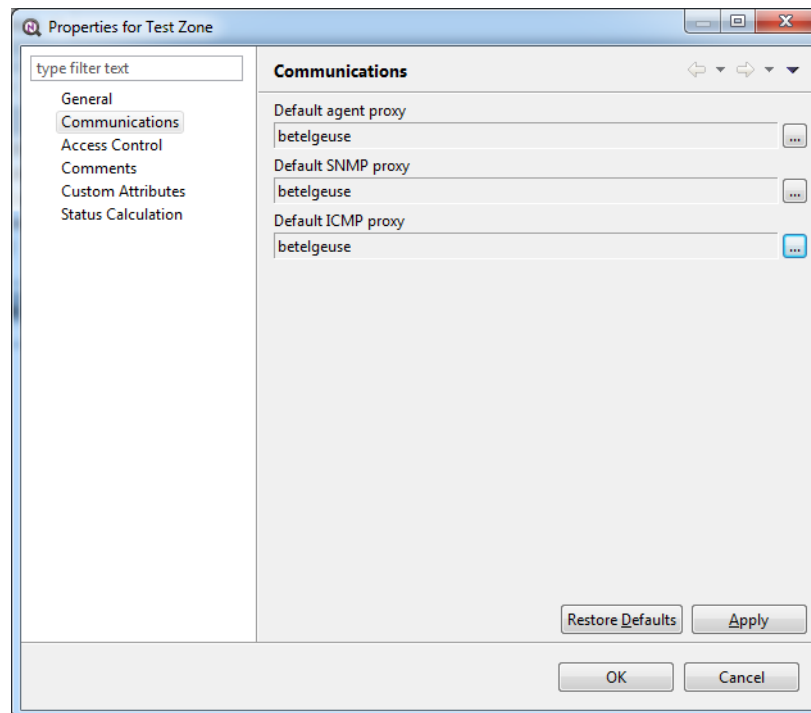
42.1.1 Enable Zoning

Zoning support is off by default. To turn it on you must set server's configuration variable `EnableZoning` to 1 and restart server. After restart, server will create default zone with UIN (unique identification number) 0 and put all existing subnets into that zone. Subnet tree will look like this:



42.1.2 Setting communication options for zones

Server have to know proxy nodes to be able to communicate with nodes in remote zones. Default proxy settings for all nodes in the zone can be set on Communications page in zone object properties:



On this page you can set default proxy node for NetXMS agents, SNMP, and ICMP. Note that proxy node must be in default zone and must have primary IP reachable by NetXMS server.

42.1.3 Moving nodes between zones

To move existing node to another zone, select *Change zone* from nodes context menu, then select target zone in zone selection dialog that will appear. After move to another zone, server will immediately do configuration poll on the node.

42.1.4 Integration with external HelpDesk

NetXMS provides possibility to create issues in external helpdesk system directly from NetXMS management client, based on pending alarms. In this situation NetXMS and external helpdesk system will have synchronized issue workflow.

For now integration is done only with JIRA.

42.1.5 JIRA Module

This module provide integration between NetXMS and JIRA.

Required NetXMS configuration

For NetXMS is required to configure server parameters and restart the server.

Parameter name	Description
HelpDeskLink	For JIRA integration should be set to “jira.hdlink” (without quotes)
Jira.IssueType	Name of the JIRA issue type, which will be used by NetXMS. Sample value: “Task” (without quotes)
Jira.Login	Login of the JIRA user(This user should exist in JIRA system with with permissions to create issues in project(JiraProjectCode) and comment on own issues)
Jira.Password	Password of the JIRA user
Jira.ProjectCode	Project Key in JIRA. (Project should exist)
Jira.ProjectComponent	Jira project component. (Project should exist)
Jira.ResolvedStatus	Comma separated list of issue status codes indicating that issue is resolved. Default is “Done”.
Jira.ServerURL	URL of JIRA installation. Example: “ http://localhost:8080/jira ”. Please note, that trailing slash (“/”) should be removed!
Jira.Webhook.Path	Path part of Jira webhook URL (must start with /). Example: “/jira-webhook”.
Jira.Webhook.Port	Jira webhook listener port (0 to disable webhook). Default: “8008”.

Note

Starting from version 4.1.283 NetXMS version Webhook can be used for Jira to NetXMS integration. Not a jira plugin.

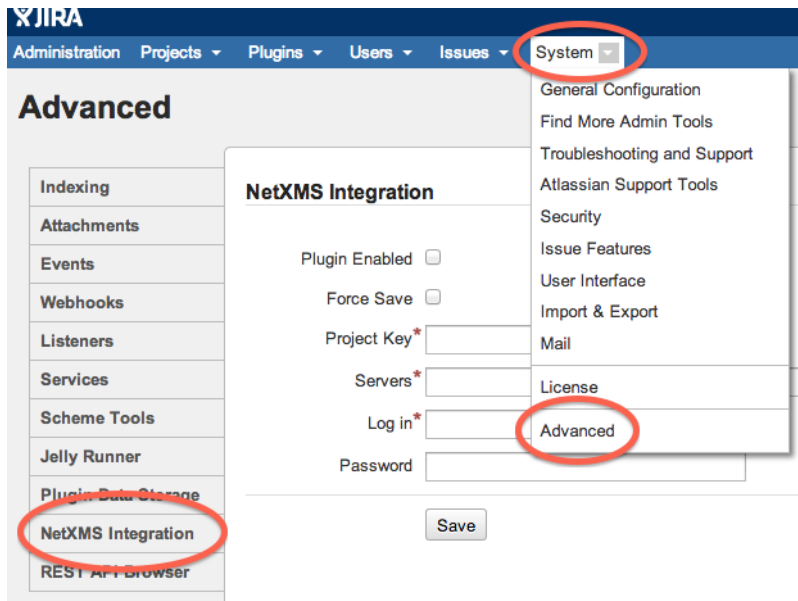
If all configuration was successfully done after retester in console should be present:

```
[25-Apr-2014 14:16:07.894] [INFO ] Helpdesk link module JIRA (version 1.2.14) loaded...
↪successfully
```

Required JIRA configuration

NetXMS JIRA plugin should be deployed to JIRA and configured. REST API should be enabled in JIRA configuration (enabled in default configuration).

To access configuration page for the plugin, go to “System → Advanced” and select “NetXMS Integration” tab:



Possible configuration options:

1. “Plugin Enabled” — global on/off switch, plugin completely cease any activity when turned off (default).
2. “Force Save” — by default, plugin will verify configuration before saving (connectivity to all servers, credentials). This checkbox allows to bypass this step completely and save configuration even if one of more NetXMS servers are rejecting provided credentials or do not respond at all)
3. “Project Key” — Key of the project, where issues from NetXMS will be created. This key will be also used in workflow operations — plugin will process events related to this project:

Project list		
Name	Key	URL
Demonstration	DEMO	No URL

4. “Servers” — addresses of up to a 3 NetXMS servers, can be either IP address or hostname.
5. “Log In” — user login in NetXMS (User should exist in NetXMS with Read, View Alarms, Acknowledge Alarms, Terminate Alarms to all nodes)
6. “Password” — user password in NetXMS

Plugin will verify configuration and provide feedback. If one or more NetXMS servers are not responding (e.g. they are not configured yet), you can select “Force Save” to overrule verification process and save configuration.

Workflow configuration

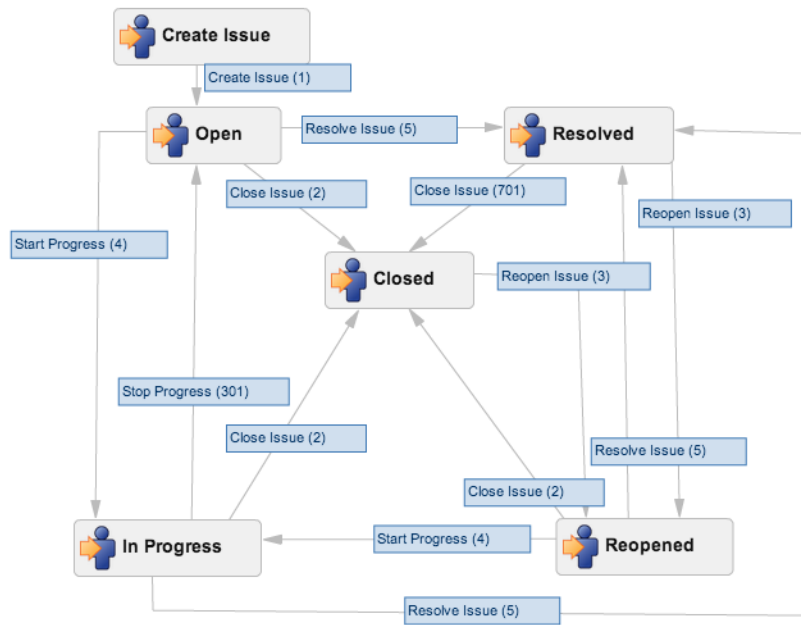
Since JIRA workflow can be much more sophisticated than alarm states in NetXMS, JIRA Administrator should decide which workflow transition should change NetXMS alarm state.

NetXMS supports four alarm states:

1. Outstanding — initial state, can’t be set from JIRA side

2. Acknowledged — operator is aware of the problem and it's in progress (“Acknowledge” action)
3. Resolved — problem is resolved but alarm stays in the list until verified and terminated by supervisor (“Resolve” action)
4. Terminated — problem is resolved and verified, alarm is removed from the list (“Terminate” action)

Sample workflow (JIRA default workflow):

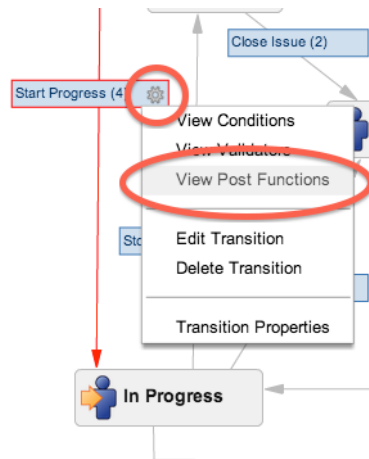


Sample mapping:

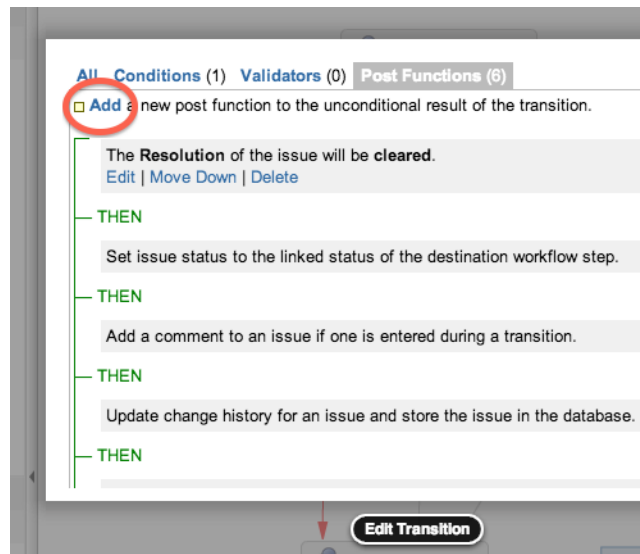
Transition	NetXMS post-function action
Start Progress	Acknowledge
Resolve Issue	Resolve
Close Issue	Terminate
<i>All other transitions</i>	<i>Ignored</i>

Configure workflow in JIRA:

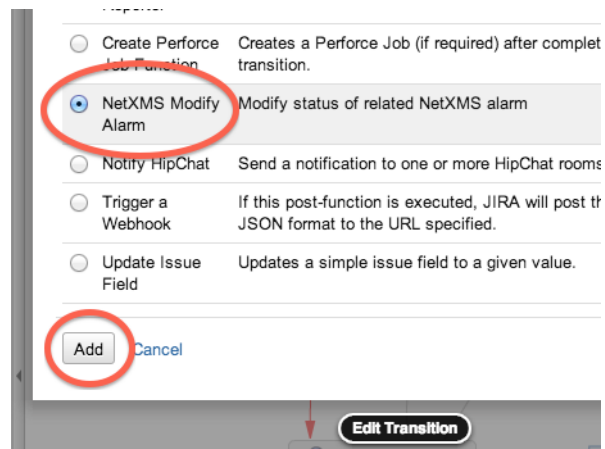
1. Create new Workflow Schema if required
2. Copy existing or create new Workflow
3. Assign Workflow to the project, where NetXMS will create issues
4. Modify transitions to call plugin's post-function and change related alarm in NetXMS
 - a. Click on a “cog” icon on a transition and select “View Post Functions”:



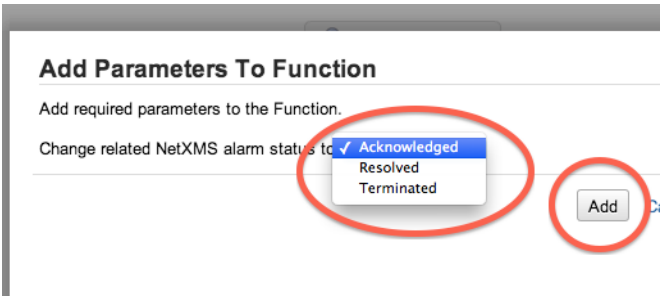
- b. Click on “Add a new post function to the unconditional result of the transition”:



- c. Select “NetXMS Modify Alarm” and click “Add”:



- d. Select desired alarm action (Acknowledge / Resolve / Terminate) and click “Add”:



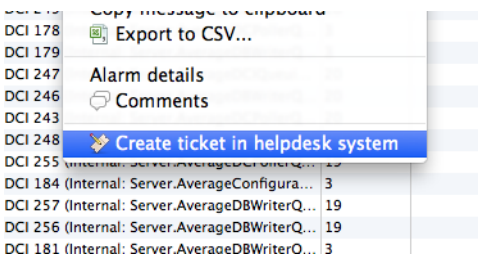
- e. Repeat for all required transitions
- 5. Publish workflow changes

Ticket creation

Tickets are created from from alarms manually. To create ticket user should have “Create helpdesk tickets” access for required objects.

Steps to create ticket:

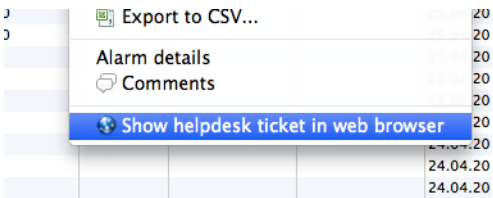
- 1. Right click on alarm in NetXMS and select “Create ticket in helpdesk system”:



- 2. In a moment, issue will be created and Helpdesk ID will be show in corresponding column:

	Count	Comments	Helpdesk ID	
ageConfigura...	21		DEMO-8	
ageDCIQueul...	21			
ageDBWriterQ...	21			

- 3. Right click on the alarm and select “Show helpdesk ticket in web browser” to navigate to the issue in JIRA:



42.2 Hooks

Sometimes it is required to add some additional functionality after poll, object creation or other action - for this purpose hooks were created. Hook is manually created script in *Script Library* that is executed at a special condition like end of the poll or interface creation.

More about poll types and purposes can be found [there](#) and about script creation [there](#).

To be recognized as a hook script should have special name. It should be named according to convention: Hook::*hook_name*.

Example: Hook::ConfigurationPoll

Full list of hooks:

Hook name	Description	Parameters	Return value
Hook::StatusPoll	Hook that is executed at the end of status poll	\$object - current object, one of 'NetObj' subclasses \$node - current object if it is 'Node' class	none
Hook::ConfigurationPoll	Hook that is executed at the end of configuration poll	\$object - current object, one of 'NetObj' subclasses \$node - current object if it is 'Node' class	none
Hook::InstancePoll	Hook that is executed after instance discovery poll.	\$object - current object, one of 'NetObj' subclasses \$node - current object if it is 'Node' class	none
Hook::TopologyPoll	Hook that is executed at the end of topology poll	\$node - current node, object of 'Node' type	none
Hook::CreateInterface	Hook that is executed after new interface is created.	\$node - current node, object of 'Node' type \$i - current interface, object of 'Interface' type	true/false - boolean - whether interface should be created
Hook::AcceptNewNode	This hook is executed by discovery process, after a new node is found and it's checked that no node with give IP address is present in the system and before any network discovery filters.	\$ipAddr - IP address of the node being processed \$ipNetMask - netmask of the node being processed \$macAddr - MAC address of the node being processed \$zoneUIN - zone UIN of the node being processed	true/false - boolean - whether node should be created
Hook::DiscoveryPoll	Hook that is executed at the end of discovery poll	\$node - current node, object of 'Node' type	none
Hook::PostObjectCreate	Hook that is executed after object is created	\$object - current object, one of 'NetObj' subclasses \$node - current object if it is 'Node' class	none
Hook::CreateSubnet	Hook that is executed on subnet creation	\$node - current node, object of 'Node' class \$i - current subnet, object of 'Subnet' class	true/false - boolean - whether subnet should be created
Hook::UpdateInterface	Hook that is executed at the end of interface update	\$node - current node, object of 'Node' type \$interface - current interface, object of 'Interface' type	none

continues on next page

Table 1 – continued from previous page

Hook name	Description	Parameters	Return value
Hook::EventProcessor	Hook that is executed for each event prior to it's processing by Event Processing Policies.	\$object - event source object, one of 'NetObj' subclasses \$node - event source object if it is 'Node' class \$event - event being processed (object of 'Event' class)	none
Hook::AlarmStateChange	Hook that is executed on alarm state change (alarm gets acknowledged, resolved or terminated)	\$alarm - alarm being processed (object of 'Alarm' class)	none
Hook::UnboundTunnelOpened	Hook that is executed when tunnel connection is established, but not bound to a node.	\$tunnel - incoming tunnel information (object of 'Tunnel' class)	none
Hook::BoundTunnelOpened	Hook that is executed when tunnel connection bound to a node is established.	\$node - node this tunnel was bound to (object of 'Node' class) \$tunnel - incoming tunnel information (object of 'Tunnel' class)	none
Hook::LDAPSynchronization	Hook executed for each LDAP record (user or group) during LDAP synchronization.	\$ldapObject - LDAP object being synchronized (object of 'LDAPObject' class)	true/false - boolean - whether processing of this LDAP record should continue
Hook::Login	Hook executed prior to user login	\$user - user object (object of 'User' class) \$session - session object (object of 'ClientSession' class)	true/false - boolean - whether login for this session should continue

Usually hooks are used for automatic actions that need to be done on node. For example automatic remove change of expected state of interface depending on some external parameters.

42.3 Troubleshooting

42.3.1 Resetting “system” user password

Warning

Server (“netxmsd”) should be stopped while performing password reset operation!

Passwords in NetXMS are stored in hashed, not-reversible way, so there are no way to recover it, but it can be reset. Use following procedure to reset password and unlock account:

1. stop netxmsd
2. run “nxdmng reset-system-account” to unlock “system” account and change it's password to default (“netxms”).

3. start netxmsd
4. login as “system” using password “netxms”
5. In user manager change password for any admin user account
6. login as admin user and disable “system” user account

42.3.2 Enable Crash Dump Generation

When running on Windows server is capable of creating crash dumps. To enable crash dump generation, add the following options to netxmsd.conf file:

```
CreateCrashDumps = yes
DumpDirectory = path
```

DumpDirectory must point to directory writable by server process. After each crash server will create two files: info and mdmp. Info file contains basic information about crash, server version, and call stack of current thread. Mdmp file is a minidump which can be read and analyzed using debugger.

42.3.3 Force Crash Dump Creation

It is possible to force creation of crash dump. To do that you'll need access to server debug console. You can access it using `nxadm` tool or via *Tools ▶ Server Console* menu in management client. Once in server debug console, you can run command `dump` or `raise access`. First command works only on Windows and will produce process dump without stopping it. Second command will cause access violation exception which will lead to process crash and crash dump generation.

42.3.4 SNMP Device not recognized as SNMP-capable

Common issues:

1. Invalid community string or credentials
2. Access control on the device or firewall prevent connections from NetXMS server
3. Device do not support `System (.1.3.6.1.2.1.1)` or `Interfaces (.1.3.6.1.2.1.2)` MIBs, which are used to detect SNMP-capable devices. To override OIDs used for detection, set node's custom attribute `snmp.testoid` to any OID supported by device.

42.4 Automatic actions on a new node

On a new node creation is generated `SYS_NODE_ADDED` event. So any automatic actions that should be done on a node can be done by creating *EPP* rule on on this event, that will run script. In such way can be done node bind to container, template auto apply and other automatic actions.

42.5 Autologin for Management Client

It is possible to connect management client (`nxmc`) or web management client to server automatically without login dialog. This chapter describes additional command line options and URL parameters for that.

42.5.1 Desktop Management Client

Command line option	Description
-auto	Connect to server automatically without login dialog
-dashboard=dashboard	Automatically open given dashboard after login (either dashboard object ID or name can be specified)
-login=login	Set login name
-password=password	Set password, default is empty
-server=address	Set server name or IP address

For example, to connect management client to server 10.0.0.2 as user guest with empty password, use command

```
nxmc -auto -server=10.0.0.2 -login=guest
```

42.5.2 Web Management Client

URL parameters	Description
auto	Connect to server automatically without login dialog
dashboard=dashboard	Automatically open given dashboard after login (either dashboard object ID or name can be specified)
login=login	Set login name
password=password	Set password, default is empty
server=address	Set server name or IP address

For example, to connect web management console to server 10.0.0.2 as user guest with empty password and open dashboard called “SystemOverview”, use URL

```
http://server/nxmc?auto&server=10.0.0.2&login=guest&dashboard=SystemOverview
```

42.6 NetXMS data usage in external products

NetXMS provides next options to use data in other applications:

- Use [autologin](#) and dashboard name in URL to add dashboard to your company documentation(where URL usage is possible).
- Use [Grafana](#) for graph creation and further usage
- Get data through [Web API](#)

42.7 Find Object

Management client has an option to filter objects by defined by user criteria. Filter can be access by *Tools->Find Object*. Filter can be used in two different modes: filter and query.

42.7.1 Filter

Filter will search object using class filter, zone filter, IP range and search string that will be checked for each object in all it's text fields (name, comments, custom attributes, Location, etc.).

42.7.2 Query

There can be written any script that will be executed on all objects and if script returns true - object will be shown in the resulting table. There can be used the same syntax as for *Object query* Dashboard element, but variables will not be added as additional columns for table in this case.

42.8 Audit log forwarding

42.8.1 Syslog

NetXMS allows to forward audit log to another syslog server to have all data in one place.

Next configuration parameters should be set in order to forward audit log to external syslog server:

Name	Description
ExternalAuditFacility	Syslog facility to be used in audit log records sent to external server.
ExternalAuditPort	UDP port of external syslog server to send audit records to.
ExternalAuditServer	External syslog server to send audit records to. If set to “none”, external audit logging is disabled.
ExternalAuditSeverity	Syslog severity to be used in audit log records sent to external server.
ExternalAuditTag	Syslog tag to be used in audit log records sent to external server.

42.8.2 LEEF

LEEF server module provides functionality to send audit log to IBM Security QRadar. The Log Event Extended Format (LEEF) is a customized event format for IBM Security QRadar. More about it can be found [there](#).

LEEF server module should be enabled in server configuration file by adding “Module=leef.nxm” line to `netxmsd.conf` file.

Additionally to module configuration “LEEF” section should be added with required configurations.

Name	Description
Server	Server address
Port	Server port
EventCode	LEEF event code
RFC5424Timestamp	“No” if RFC5424 Timestamp format should not be used (default value is Yes)
Facility	Facility as facility in syslog
Severity	Severity as severity in syslog
Product	LEEF product field, by default will be “NetXMS”
ProductVersion	LEEF product version field, by default will be server version
Vendor	LEEF vendor field, default it “Raden Solutions”
Separator	LEEF separator character as a char or in numeric format: “xHH”, where HH is hexadecimal digit

Additional fields can be configured in ExtraData sub section in the same key=value format.

Example:

```
[LEEF]
Server = 127.0.0.1
Port = 514
Facility = 13
Severity = 5
EventCode =
Separator = ^

[LEEF/ExtraData]
key = value
key2 = value2
```

42.9 Custom housekeeping scripts

To customize housekeeper operations it's possible to use custom scripts. Scripts are executed in the end of housekeeping process. Due to security considerations scripts are stored on server file system in `<DataDirectory>/housekeeper` folder, where `<DataDirectory>` is path to server data directory (see `DataDirectory` parameter in *Server configuration file (netxmsd.conf)* for more information). Multiple scripts can be present in the mentioned folder.

Two types of scripts are supported:

- SQL (files with .sql extension) - file containing SQL queries. SQL query can take multiple lines, end of query is denoted with semicolon (;) character
- NXSL (files with .nxsl extension) - file contains *NXSL* script. In addition to all standard NXSL functionality, `SQLQuery()` NXSL function is supported, allowing SQL query execution to the database.

To implement custom deletion of DCI and Table DCI data built-in deletion of this data can be disabled by setting server configuration parameter `Housekeeper.DisableCollectedDataCleanup`.

42.10 Fanout drivers

NetXMS has concept of fanout driver, which enable collected data sending to an additional database.

42.10.1 InfluxDB

To enable InfluxDB fanout driver, add `PerfDataStorageDriver=influxdb` to `netxmsd.conf` file. Driver configuration is specified in `[InfluxDB]` section.

Name	Description
Bucket	Bucket name.
EnableUnsignedType	Enable (true) or disable (false) unsigned data type. If disabled, values for DCIs with unsigned data types will be sent as signed type. Default: <i>false</i> .
Database	Database name. Default value is <i>netxms</i> .
Hostname	Hostname. Default is <i>localhost</i> .
MaxCacheWaitTime	Maximum time in ms before cache being flushed. Default is <i>30000</i> .
Password	Password.
Port	Network port number
Protocol	Options are: <i>udp</i> , <i>api-v1</i> and <i>api-v2</i> . Default is <i>udp</i> .
QueueFlushThreshold	Cache will be flushed when reaching this size (in bytes). Default: <i>32768</i>
Queues	Number of queues for parallel operation. Default: <i>1</i> .
QueueSizeLimit	Upper limit on queue size in bytes. If queue reaches this size, data will be dropped. Default: <i>4194304</i> .
Token	Authentication token.
ValidateValues (from 5.1.2)	If true, driver will validate values according to DCI data type, and drop invalid values (invalid numbers, out-of-range values). Default: <i>false</i>
CorrectValues	If both ValidateValues and CorrectValues set to true, instead of dropping values that did not pass validation, correct values will be sent to InfluxDB instead. Unparsable numbers will be set to last parsable part (for example, 123abc will be sent as 123), out-of-range values will be sent as maximal or minimal possible value. Default: <i>false</i>

Configuration example:

```
PerfDataStorageDriver=influxdb

[InfluxDB]
Protocol=api-v2
Organization=netxms
Bucket=netxms
Token=MJzXfwcNm7uEu4mL31S-iVjZ-DJO9pPbCuDl90XotOS3TyY9VkVMoDr5o4u4w8opucyZ2-
↳MwcrpfC2ymbcj2Q==
```

Details of operation

Field key is made from DCI's metric name (except for SNMP and internal "Dummy" DCIs where description is used). Space characters are removed, *:-, #* characters are replaced with *_*, ** is replaced with */*.

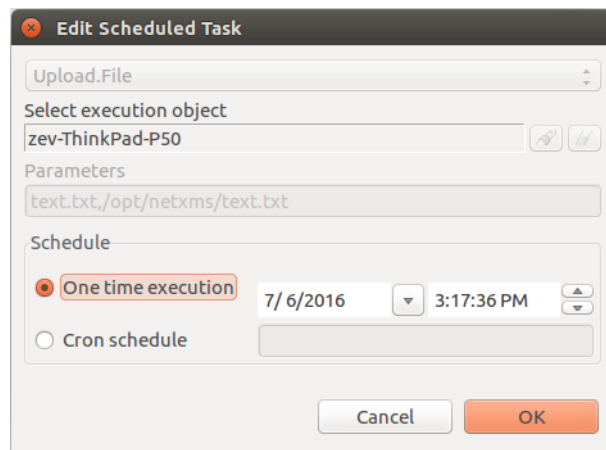
Empty DCI values are not sent.

If custom attribute named *ignore_influxdb* (with any value) exists on a node, this node will be excluded from export. Also, if a DCI has Related Object set to an interface and this interface has *ignore_influxdb* custom attribute, this DCI will be ignored.

If there is custom attribute on the node or on related object with name starting with *tag_*, it's name (excluding *tag_* part) and value will be used as tag. There can be several such custom attributes.

SCHEDULED TASKS

NetXMS provides the option to schedule different tasks. Each task has its own parameter count and type. The only common parameter is the node on which task will be executed. The schedule time can be set in two ways: as a one time schedule or as a cron task (see [Cron format](#) for supported cron format options).



Information about available tasks can be found there:

1. *File Upload*
2. *Script Execution*
3. *Maintenance*

43.1 File Upload

The task is named *Upload.File*. This task uploads a file from the server to the agent. The file to be uploaded must exist at the server file storage. Task can be created in the *Schedules* view or in the *Upload file...* dialog.

Parameters:

1. File name that should be uploaded
2. Path and file name where this file should be uploaded on the agent

Example: Warning-C.wav,/destination/location/Warning-C.wav

43.2 Script Execution

The task is named *Execute.Script*. This task executes a script from the library. The selected node is set as the *\$node* variable in the script.

Parameters:

1. Server script name

43.3 Package deploy

The task is named *Agent.DeployPackage*. This task schedules package deployment via agent which has been created in Configuration -> Packages section. The task handler *Agent.DeployPackage* expects parameter string as set of key=value entries separated by semicolons. Currently only one key is supported - "package".

Parameters:

1. Package ID

Scheduled Tasks						
Filter is empty						
ID	Schedule Type	Object	Parameters	Timer key	Execution time	Execution time description
69	Agent.DeployPackage	VM Windows 10	package=4		10.10.2024 15:51:48	Exactly at 10.10.2024 15:51:48

Packages							
Filter is empty							
ID	Name	Type	Version	Platform	File	Command	Description
4	nxagent	agent-ins...	5.0.8	windows-x64	nxagent-5.0.8-x64.exe		NetXMS Agent for Windows

43.4 Maintenance

The tasks are named *Maintenance.Enter* and *Maintenance.Leave*. These tasks turn on and turn off maintenance mode for selected node. More about maintenance mode can be found [there](#).

These tasks do not require parameters.

43.5 Access Rights

Access right for schedules can be separated into two parts. Rights that are required to create, edit and delete tasks and rights that are required to schedule the exact task type. Task can be created by the user or by the system.

Overall access rights:

Access right	Description
Manage user scheduled tasks	Option to add, view, edit, delete users' tasks
Manage own scheduled tasks	Option to add, view, edit, delete tasks created by this user
Manage all scheduled tasks	Option to add, view, edit, delete tasks created by user and system

Task specific access rights:

Schedule type	Required access right
File Upload	Schedule file upload task
Script Execution	Schedule script task
Maintenance	Schedule object maintenance

For some tasks like *File.Upload* there is an additional check if the user has permissions to upload the file to this node and if there is access to the specific folder. Access rights like this are checked during task execution, not during scheduling. If the user does not have access, then the task will fail.

SCRIPTING

44.1 NXSL

44.1.1 Overview

In many parts of the system, fine tuning can be done by using NetXMS built-in scripting language called NXSL (stands for NetXMS Scripting Language). NXSL was designed specifically to be used as embedded scripting language within NetXMS, and because of this has some specific features and limitations. Most notable is very limited access to data outside script boundaries - for example, from NXSL script you cannot access files on server, nor call external programs, nor even access data of the node object other than script is running for without explicit permission. NXSL is interpreted language - scripts first compiled into internal representation (similar to byte code in Java), which is then executed inside NXSL Virtual Machine. Language syntax and available functions can be found in [NXSL documentation](#).

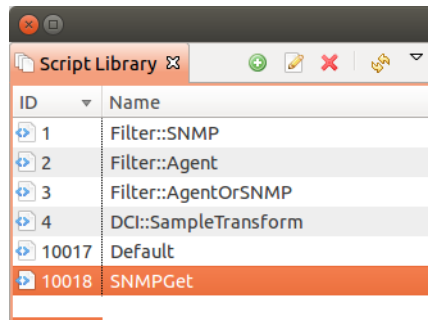
List of places where NXSL scripting is used

- Script library
- DCI transformation scripts
- DCI instance filter script
- DCI scripted threshold
- DCI summary table object filter script
- Container, template, cluster auto-bind script
- SNMP trap transformation script
- EPP filter script
- EPP inline script actions
- Map object filter script
- Map link styling script
- Dashboard scripted chart
- Dashboard status indicator
- Context dashboard auto-bind script
- Business service scripted check
- Business service DCI auto apply script
- Business service object auto apply script
- Business service prototype instance filter script

- Asset attribute auto fill script
- Object query
- Agent configuration filter script
- Condition status calculation script
- Custom housekeeping scripts (see *Custom housekeeping scripts*)

44.1.2 Scripting library

Script Library is used to store scripts that can be afterwards executed as macros, part of other script or from debug server console. Scripts can be added, deleted and modified in in this view.



Usage

Scripts from Script Library can be accessed as:

1. a macros %[scriptName]
2. used in action of type “Execute NXSL script”
3. executed from DCIs with “Script” source
4. functions can be called from other scripts either by using “import scriptName” and calling functions by name, or without import, by calling “scriptName::functionName”
5. executed from server debug console “execute scriptName”
6. scripts having name starting with “Hook::” are executed automatically, e.g. “Hook::ConfigurationPoll” is being run on each node’s configuration poll

Note

All parameters provided to script are accessible via \$ARGS array. The other option to use parameters is to specify *main()* function in the script and define parameters in it’s definition.

44.1.3 Execute Server Script

This view allows to execute arbitrary script. Script can be manually created just before execution, and saved afterwards, can be taken from the script library or modified script can be used from the script library and saved or saved as afterwards. If this view is opened on a node, then in the script \$node variable is available with node object. All parameters provided to script, like \$node, \$object, \$isCluster, \$ARGV, etc, are accessible via \$ARGS array. Please refer to [NXSL Guide](#) for more information.

The screenshot displays the NetXMS Management Client interface. The top bar shows the user 'admin@::1' and the current host 'sw-mgmt.office.radensolutions.com'. The left sidebar contains a tree view of 'Infrastructure Services' with 'All' selected. The main panel shows the 'Execute Script' window for the 'SNMPGet' script. The script parameters are set to 'System description, .1.3.6.1.2.1.1.0'. The source code of the script is visible, and the output section shows the execution results.

SNMPGet
Script from library

Parameters (comma-separated list)
System description, .1.3.6.1.2.1.1.0

Source

```

1 println(F"Name: {$node->name}");
2 println(F"Arguments: {$ARGS}");
3
4
5 transport = CreateSNMPTransport($node);
6
7 if (transport == null)
8 {
9     println("Failed to create SNMP transport, exit");
10    return 1;
11 }
12
13 value = SNMPGetValue(transport, $ARGS[2]); //" .1.3.6.1.2.1.1.0"
14 if (value == null)
15 {
16     println("Failed to issue SNMP GET request");
17     return 2;
18 }
19 else
20 {
21     println(F"{$ARGS[1]}: {value}"); //System description
22     return 0;
23 }

```

Output

```

Name: sw-mgmt.office.radensolutions.com
Arguments: [System description, .1.3.6.1.2.1.1.0]
Failed to create SNMP transport, exit

*** FINISHED ***

Result: 1

```

44.2 NXShell

NXShell is based on Jython and provide access to NetXMS Java API using interactive shell. NXShell binary comes with server distribution suite and can be run from shell or crontab. NXShell is also build as single jar file, which includes all required libraries.

Download: <http://www.netxms.org/download/nxshell-VERSION.jar> (example: <http://www.netxms.org/download/nxshell-5.0.8.jar>)

44.2.1 Usage

NXShell binary gets installed in \$NETXMS_HOME directory, for example /usr/bin/nxshell. As of version 5.1, nxshell launcher accepts command line -r or --properties= for providing path to nxshell properties file.

Usage: nxshell [OPTIONS] [script]

Options:

- C, --classpath <path>** Additional Java class path.
- D, --debug** Show additional debug output (use twice for extra output).
- h, --help** Display this help message.
- H, --host <hostname>** Specify host name or IP address. Could be in host:port form.
- j, --jre <path>** Specify JRE location.
- n, --no-sync** Do not synchronize objects on connect.
- p, --port <port>** Specify TCP port for connection. Default is 4701.
- P, --password <password>** Specify user's password. Default is empty.
- r, --properties <file>** File with additional Java properties.
- t, --token <token>** Login to server using given authentication token.
- u, --user <user>** Login to server as user. Default is "admin".
- v, --version** Display version information.

There are two options of this jar usage:

1. it can be started as interactive shell:

```
java -jar nxshell-5.0.8.jar
```

2. it can be started with the script name as a first parameter. Then it will just execute this script and exit.
Example:

```
java -jar nxshell-5.0.8.jar test.py
```

When NXShell is started, it tries to get server IP, login and password from Java properties. In interactive mode, user will be asked for details, otherwise default values will be used.

Start as interactive shell, with IP and Login provided (password will be asked):

```
java -Dnetxms.server=127.0.0.1 -Dnetxms.login=admin -jar nxshell-5.0.8.jar
```

Properties

These properties should be set with JVM's "-D" option. Please make sure that all "-D" options are before "-jar".

Parameter	Default Value
netxms.server	127.0.0.1
netxms.login	admin
netxms.password	netxms
netxms.encryptSession	true

44.2.2 Scripting

For details on API please refer to javadoc at <http://www.netxms.org/documentation/javadoc/latest/>.

NXShell provide user with already connected and synchronized session to simplify scripting. Most required packages are imported as well to minimize typing.

Global Variables

Variable	Type	Notes
session	org.netxms.client.NXCSession	
s	org.netxms.client.NXCSession	Alias for "session"

Helper Functions

Example

More examples can be found on a [NetXMS wiki](#).

```

parentId = objects.GenericObject.SERVICEROOT # Infrastructure Services root
cd = NXObjectCreationData(objects.GenericObject.OBJECT_CONTAINER, "Sample Container",
    ↪ parentId);
containerId = session.createObject(cd) # createObject return ID of newly created_
    ↪ object
print '"Sample Container" created, id=%d' % (containerId, )

flags = NXObjectCreationData.CF_DISABLE_ICMP | \
        NXObjectCreationData.CF_DISABLE_NXCP | \
        NXObjectCreationData.CF_DISABLE_SNMP
for i in xrange(0, 5):
    name = "Node %d" % (i + 1, )
    cd = NXObjectCreationData(objects.GenericObject.OBJECT_NODE, name, containerId);
    cd.setCreationFlags(flags);
    cd.setPrimaryName("0.0.0.0") # Create node without IP address
    nodeId = session.createObject(cd)
    print '"%s" created, id=%d' % (name, nodeId)

```


HIGH AVAILABILITY SETUP

45.1 Infrastructure

45.1.1 Production

IP/hostname: netxms-prod

PostgreSQL version: 14.3

PostgreSQL systemd service name: postgresql-14.service

PostgreSQL data directory: /u0fs1/pg-data/14

PostgreSQL port: 5432

NetXMS installation prefix: /opt/netxms

NetXMS system service names: netxmsd.service, nxagentd.service, nxreportd.service

45.1.2 DR

IP/hostname: netxms-dr

PostgreSQL version: 14.2

PostgreSQL systemd service name: postgresql-14.service

PostgreSQL data directory: /u0fs1/pg-data/14

PostgreSQL port: 5432

NetXMS installation prefix: /opt/netxms

NetXMS system service names: netxmsd.service, nxagentd.service, nxreportd.service

45.2 Switchover procedure

Switchover steps:

1. Confirm which node is currently active
 1. The process “netxmsd” should be running only on active node (check with “ps” or “pgrep”)
 2. Run “pg_replica_state” to get the current state of the database on this server. The active node will be marked as “Sender / Primary”.
2. Stop netxmsd on active node:
 1. Run “systemctl stop netxmsd”

2. Make sure it is stopped (with “ps” or “pgrep”)
3. Switch active database instance to standby (read-only) mode:
 1. Run “`sudo -u postgres touch /u0fs1/pg-data/14/standby.signal`”
 2. Run “`systemctl restart postgresql-14`”
 3. Check logs (`/u0fs1/pg-data/14/log/postgresql-*.log`), it should contain records:
 1. “starting PostgreSQL...”
 2. “consistent recovery state reached at...”
 3. “database system is ready to accept read only connections”
4. Promote another node as new PostgreSQL sender node:
 1. On second node run `sudo -u postgres psql -c 'select pg_promote()'`
 2. Check log file for following records:
 1. “...received promote request”
 2. “selected new timeline ID: ...”
 3. “archive recovery complete”
 4. “database system is ready to accept connections” (non-readonly!)
5. Start netxmsd on another node

The switchover procedure is identical when switching from PROD to DR and from DR to PROD.

45.3 Failover procedure

Follow the switchover procedure from item 4 onwards.

45.4 Failover recovery

Once a failed server (which was sender before the failover) is up and running, you need to switch it to replica mode.

1. Stop PostgreSQL (“`systemctl stop postgresql-14`”) on the failed node
2. Run “`sudo -u postgres touch /u0fs1/pg-data/14/standby.signal`” to switch it to replica mode
3. Unwind this DB instance to the state where it is in sync with the current sending server:

```
run    sudo    -u    postgres    /usr/pgsql-14/bin/pg_rewind    -target-pgdata=/u0fs1/pg-data/14    -source-server="host=ACTIVE_DB user=postgres password=PASSWORD"
```

ACTIVE_DB should point to the current sender instance (netxms-prod or netxms-dr).

4. Start PostgreSQL instance with “`systemctl start postgresql-14`”
5. Check logs and make sure that the database is started and it is in read only mode. Once recovery is completed, a switchover procedure might be performed

46.1 Cron format

Record has five fields, separated by spaces: minute, hour, day of month, month, and day of week. In DCI Collection Schedule only, an optional the sixth field can be specified for resolution in seconds (this is a non-standard extension which is not compatible with a regular cron format).

Allowed values and special characters for each field are:

Field	Allowed values	Allowed special characters
minute	0 - 59	*, - /
hour	0 - 23	*, - /
day of month	1 - 31	*, - / L
month	1 - 12	*, - /
day of week	0 - 7 (0 and 7 is Sunday)	*, - / L
seconds (for DCI collection only, optional)	0 - 59 (0 - unlimited for %)	*, - / %

A field may be an asterisk (*), which always stands for “any”.

Commas (,) are used to separate items of a list. For example, using 1, 3, 4 in the 5th field (day of week) means Mondays, Wednesdays and Fridays.

Hyphens (–) define ranges. For example, using 6–8 in 4th field (month) means June, July and August.

Slashes (/) can be combined with ranges to specify step values. For example, */5 in the minutes field indicates every 5 minutes. If a step value does not evenly divide it’s range, there will be an inconsistent “short” period at the end of time-unit.

L stands for “last”. When used in the day-of-week field, it allows to specify constructs such as “the last Friday” (“5L”) of a given month. In the day-of-month field, it specifies the last day of the month.

The sixth field (but not others) supports additional stepping syntax with a percent sign (%), which means that the step in seconds calculated in absolute seconds since the Unix epoch (00:00:00 UTC, 1st of January, 1970). It’s not recommended to use seconds in custom schedules as your main data collection strategy though. Use seconds only if it is absolutely necessary.

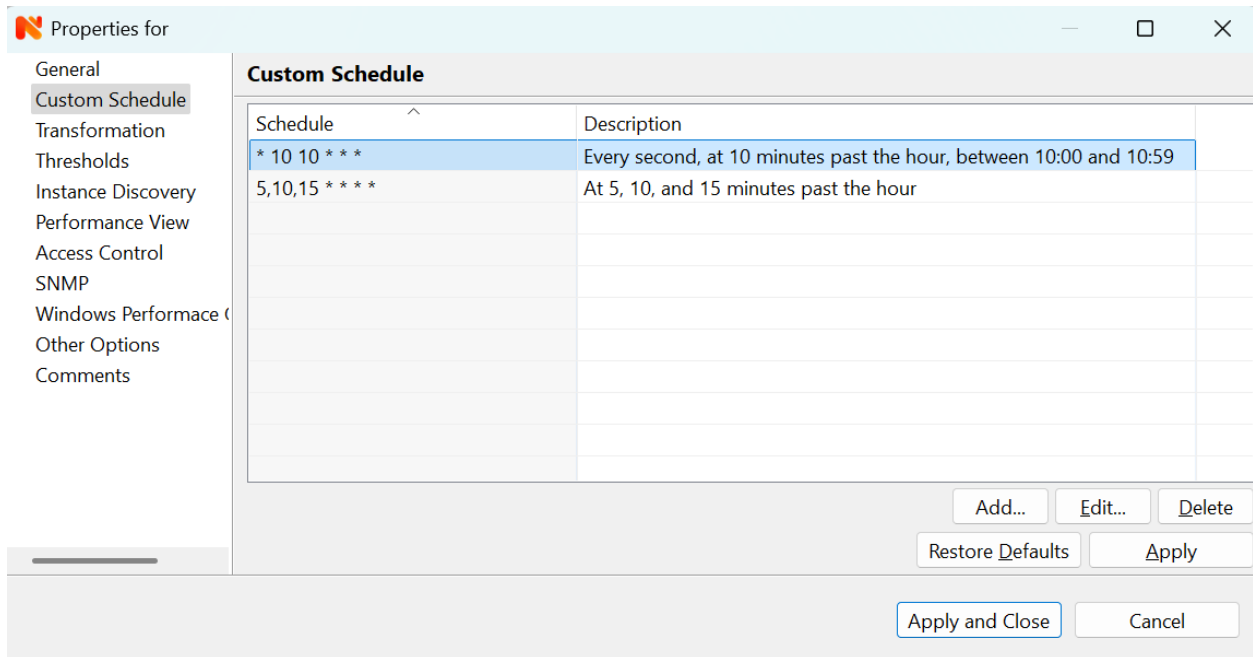


Fig. 1: DCI configuration custom schedule property page

46.1.1 Examples

Run five minutes after midnight, every day:

```
5 0 * * *
```

Run at 14:15 on the first day of every month:

```
15 14 1 * *
```

Run every 5 minutes:

```
*/5 * * * *
```

Run every minute on 10th second:

```
* * * * * 10
```

Run twice a minute (on seconds 0 and 45):

```
* * * * * */45
```

Run every 45 seconds from Monday till Friday:

```
* * * * 1-5 */45
```

46.2 SMS Drivers

Deprecated since version 3.0.

SMS driver functionality replaces by notification channel functionality. More can be found in [Notification channels](#) section.

46.3 Agent configuration file (nxagentd.conf)

Parameter	Description	Default Value
Action	Define action, which can be later executed by management server. Parameters to the action can be provided from the server. They can be accessed as \$1, \$2... variables. On Windows platform system process execution API's CreateProcess() is used to run the command, it will search in PATH, but the command should be with file extension, e.g. <code>command.exe</code> . For more information please check Agent Actions .	No defaults
ActionShellExec	Same as Action, but on Windows platform agent will use shell to execute command instead of normal process creation. There is no difference between Action and ActionShellExec on UNIX platforms. Parameters to the action can be provided from the server. They can be accessed as \$1, \$2... variables. For more information please check Agent Actions .	No defaults
AppAgent	The registered name of application with built in subagent library that can be as subagent by agent.	No defaults
AutoStartUserAgent	Enable (yes) or disable (no) automatic start of User Support Application (Windows only). If enabled, Agent will check on it's start, if User Support Application is running in each user session and will start it if needed. For this to work, Agent should be started under local SYSTEM user.	no
BackgroundLogWriter	Enable (yes) or disable (no) log writer as separate background thread. Has no effect if logging is done through syslog or Windows Event Log.	no
CodePage	Code page used by NetXMS agent. Has no effect on Windows or if agent was compiled without iconv support.	Depends on your system, usually ISO8859-1
ConfigIncludeDir	Folder containing additional configuration files. This parameter can only be specified in master configuration file and will be ignored if found in additional configuration files or configuration policy.	See Additional configuration files for information on default value.
ControlServers	A list of management servers, which can execute actions on agent and change agent's config. Hosts listed in this parameter also have read access to the agent. Both IP addresses and DNS names can be used. Multiple servers can be specified in one line, separated by commas. If this parameter is used more than once, servers listed in all occurrences will have access to agent.	Empty list
CreateCrashDumps	Enable (yes) or disable (no) creation of agent's crash dumps. Windows only	yes

continues on next page

Table 1 – continued from previous page

Parameter	Description	Default Value
DataDirectory	Directory where additional agent files (log file monitoring policy files, agent configuration policy files, user agent configuration, local agent database, etc) will be stored. This parameter can only be specified in master configuration file and will be ignored if found in additional configuration files or configuration policy.	UNIX-like systems: If \$NETXMS_HOME environment variable is set: \$NETXMS_HOME/var/lib/netxms, otherwise /var/lib/netxms. Windows: 'AppData'\nxagentd where 'AppData' is AppData folder for the user account under which NetXMS agent is started. If agent runs under local SYSTEM user account, data directory is C:\Windows\System32\config\systemprofile\AppData\Local\nxagentd.
DailyLogFileSuffix	Log file name suffix used when LogRotationMode is set to 1 (daily), can contain <code>strftime(3C)</code> macros	%Y%m%d
DebugLevel	Set agent debug logging level (0 - 9). Value of 0 turns off debugging, 9 enables very detailed logging. Can also be set with command line “-D<level>” option.	0
DebugTags	Set agent debug logging level (0 - 9) for exact log tag or log tag mask. Value of 0 turns off debugging, 9 enables very detailed logging. Configuration should look like <code>debugTag:logLevel</code> (like <code>db.conn:6</code>). To configure multiple log tags, you should use multiple DebugTags parameters or write them coma separated (like <code>proc.spexec:8,tunnel.*:4,db.conn:6</code>).	
DefaultExecutionTimeout	Timeout in milliseconds for external metric and external command execution. This value will be used for external metrics and external commands if <code>ExternalCommandTimeout</code> or <code>ExternalMetricTimeout</code> not set explicitly.	5000
DisableIPv4	Disables (yes) or enables(no) IPv4 support.	no
DisableIPv6	Disables (yes) or enables(no) IPv6 support.	no
DumpDirectory	Directory for storing crash dumps (Windows only).	C:\
EnableActions	Enable (yes) or disable (no) action execution by agent.	yes
EnableArbitraryCommandExecution	Not yet implemented. Enables server to run any shell command on the agent without specifying it as action in agent's config file. Enabling this adds <code>System.Execute</code> action (and also <code>System.ExecuteInAllSessions</code> on Windows).	no

continues on next page

Table 1 – continued from previous page

Parameter	Description	Default Value
EnabledCiphers	Controls what ciphers agent can use for connection encryption. A value for this parameter is a cipher code. To enable more than one cipher, the codes should be summed up. Possible cipher codes: <ul style="list-style-type: none"> • 1 - “AES-256” • 2 - “BLOWFISH-256” • 4 - “IDEA” • 8 - “3DES” • 16 - “AES-128” • 32 - “BLOWFISH-128” Example (enable AES-256 and IDEA): EnabledCiphers = 5	63
EnableControlConnector	Enables named pipe used by the agent to receive shutdown and delayed restart commands. A command is sent by another instance of agent, launched with -k or -K parameter. Used on Windows during upgrade process.	yes
EnableProxy	Enable (yes) or disable (no) agent proxy functionality.	no
EnableModbusProxy	Enable (yes) or disable (no) Modbus-TCP proxy functionality.	no
EnablePushConnector	Enables named pipe / unix socket used by the agent to receive data sent by nxapush command line tool.	yes
EnableSNMPPProxy	Enable (yes) or disable (no) SNMP proxy functionality.	no
EnableSNMPTrapProxy	Enable (yes) or disable (no) SNMP Trap proxy functionality.	no
EnableSSLTrace	Enable (yes) or disable (no) additional debug messages from SSL library.	no
EnableSubagentAutoload	Enable (yes) or disable (no) automatic loading of subagent(s) depending on the platform on which the agent is running.	yes
EnableSyslogProxy	Enable (yes) or disable (no) Syslog proxy functionality.	no
EnableTCPProxy	Enable TCP proxy functionality that allows to forward TCP connections inside the connection between NetXMS server and agent. Connection can be established from Management Client when using URL and Local Command Object Tools. It's also possible to use this functionality from third party applications, Java utility called TcpProxyApp that forwards local ports is provided as an example.	no
EnableWatchdog	Enable (yes) or disable (no) automatic agent restart in case of unexpected shutdown.	no
EnableWebServiceProxy	Enable (yes) or disable (no) web service data collection proxy functionality.	no

continues on next page

Table 1 – continued from previous page

Parameter	Description	Default Value
ExecTimeout	Deprecated, replaced by <code>DefaultExecutionTimeout</code>	
ExternalCommandTimeout	External process execution timeout for external commands (actions) in milliseconds. Value of <code>DefaultExecutionTimeout</code> will be used if this parameter is not set.	
ExternalList	Add list handled by external command. To add multiple lists, you should use multiple <code>ExternalList</code> entries.	No defaults
ExternalMasterAgent	ID that is checked when external subagent connects to master agent. Should have same value as <code>ExternalSubagent</code> parameter in external subagent configuration file.	No defaults
ExternalMetric	Adds metric handled by external command. To add multiple metrics, you should use multiple <code>ExternalMetric</code> entries. On Windows platform system process execution API's <code>CreateProcess()</code> is used to run the command, it will search in <code>PATH</code> , but the command should be with file extension, e.g. <code>command.exe</code> .	No defaults
ExternalMetricProvider	Specifies external command and execution interval after semicolon (;). External command returns a number of metrics and their values. Metrics are cached by the agent and returned to server per request. Command should return data in <code>metric=value</code> format each pair in new line.	No defaults
ExternalMetricProvider-Timeout	Timeout in milliseconds for external metric provider and background-pollled external table execution	30000
ExternalMetricShellExec	<code>ExternalMetricShellExec</code> has same meaning as <code>ExternalMetric</code> with exception that agent will use shell to execute specified command instead of system process execution API. This difference presented only on Windows system, on other systems <code>ExternalMetric</code> and <code>ExternalMetricShellExec</code> behaves identically.	No defaults
ExternalMetricTimeout	Timeout in milliseconds for external metrics. Value of <code>DefaultExecutionTimeout</code> will be used if this parameter is not set.	
ExternalParameter	Deprecated, replaced by <code>ExternalMetric</code>	
ExternalParameterProvider	Deprecated, replaced by <code>ExternalMetricProvider</code>	
ExternalParametersProvider	Deprecated, replaced by <code>ExternalMetricProvider</code>	
ExternalParameterProvider-Timeout	Deprecated, replaced by <code>ExternalMetricProviderTimeout</code>	
ExternalParameter-ShellExec	Deprecated, replaced by <code>ExternalMetricShellExec</code>	

continues on next page

Table 1 – continued from previous page

Parameter	Description	Default Value
ExternalSubagent	ID of external subagent. Should be same as ExternalMasterAgent in master agent configuration file.	No defaults
ExternalTable	Adds table metric handled by external command. To add multiple parameters, you should use multiple ExternalTable entries. See <i>Agent External Metrics</i> for more information.	No defaults
FileStore	Directory to be used for storing files uploaded by management server(s). It's value is set to environment variable NETXMS_FILE_STORE that is available to all processes launched by agent.	/tmp on UNIX C:\ on Windows
FullCrashDumps	Enable (yes) or disable (no) full crash dump generation. Windows only	no
GroupId	GroupId under which NetXMS agent is started (Unix only). See also UserId parameter.	No defaults
ListenAddress	IP address that the agent should listen on. If 0.0.0.0 or * is specified as listen address, agent will listen on all available IP addresses.	0.0.0.0
ListenPort	TCP port to be used for incoming requests.	4700
LogFile	Agent's log file. To write log to syslog (or Event Log on Windows), use {syslog} as file name.	/var/log/nxagentd on UNIX syslog on Windows
LogHistorySize	Defines how many old log files should be kept after log rotation.	4
LogRotationMode	Define log rotation mode. Possible values are: <ul style="list-style-type: none"> • 0 - No rotation; • 1 - Daily rotation (log will be rotated every midnight); • 2 - Rotation by size (log will be rotated when it's size will exceed value defined by MaxLogSize parameter). 	2
LogUnresolvedSymbols	If set to yes, all dynamically resolved symbols, which failed to be resolved, will be logged.	no
LongRunningQueryThreshold	Expressed in milliseconds. If a query to agent's local database or DBQuery subagent query takes longer then this time, the query will be logged to agent log file.	250
MasterServers	List of management servers, which have full access to agent. Hosts listed in this group can upload files to agent and initiate agent upgrade, as well as perform any task allowed for hosts listed in Servers and ControlServers. Both IP addresses and DNS names can be used. Multiple servers can be specified in one line, separated by commas. If this parameter is used more than once, servers listed in all occurrences will have access to agent.	Empty list
MaxLogSize	Maximum log size, in bytes. When log file reaches this limit, log rotation occurs. Use 0 to disable log rotation. This parameter supports (K, M, G, T suffixes).	16M

continues on next page

Table 1 – continued from previous page

Parameter	Description	Default Value
MaxSessions	Maximum number of simultaneous communication sessions. Possible value can range from 2 to 1024.	32
OfflineDataExpirationTime	Applicable only if Agent Cache Mode is on. Defines the duration (in days) for how collected data will be stored in agent's database if there is no connection to NetXMS server.	10
PlatformSuffix	String to be added as suffix to the value of <code>System.PlatformName</code> parameter.	Empty string
RequireAuthentication	If set to yes, a host connected to an agent has to provide correct shared secret before issuing any command.	no
RequireEncryption	If set to yes, a host connected to an agent will be forced to use encryption, and if encryption is not supported by a remote host, the connection will be dropped. If an agent was compiled without encryption support, this parameter has no effect.	no
ServerConnection	IP address or host name of NetXMS server for tunnel agent connection. Several such parameters can be present, in this case agent will establish tunnel connection to more then one server.	No defaults
[ServerConnection]	Section with parameters for for tunnel agent connection. Several such sections can be present. See Agent to server connection for more information.	No defaults
Servers	A list of management servers, which have read access to this agent. Both IP addresses and DNS names can be used. Multiple servers can be specified in one line, separated by commas. If this parameter is used more than once, servers listed in all occurrences will have access to agent.	Empty list
SessionIdleTimeout	Communication session idle timeout in seconds. If an agent will not receive any command from peer within the specified timeout, session will be closed.	60
SharedSecret	Agent's shared secret used for remote peer authentication. If <code>RequireAuthentication</code> set to no, this parameter has no effect.	admin
EncryptedSharedSecret	Agent's shared secret used for remote peer authentication, encrypted using "nxencpasswd -a". If <code>RequireAuthentication</code> set to no, this parameter has no effect.	
SNMPProxyThreadPool-Size	SNMP proxy thread pool size	128
SNMPTimeout	Timeout in milliseconds for SNMP requests sent by agent	3000
SNMPTrapListenAddress	Interface address which should be used by server to listen for incoming SNMP trap connections. Use value 0.0.0.0 or * to use all available interfaces.	*
SNMPTrapPort	Port that will be used to listen SNMP traps	162

continues on next page

Table 1 – continued from previous page

Parameter	Description	Default Value
StartupDelay	Number of seconds that agent should wait on startup before start servicing requests. This parameter can be used to prevent false reports about missing processes or failed services just after monitored system startup.	0
SubAgent	Subagent to load. To load multiple subagents, you should use multiple SubAgent parameters. Subagents will be loaded in the same order as they appear in configuration file.	No defaults
SyslogListenPort	Listening port number for syslog proxy functionality.	514
SystemName	If tunnel agent connection is used, the system appears in <i>Agent Tunnel Manager</i> under that name.	localhost is used by default
TrustedRootCertificate	Path to file or folder with root certificate used to verify certificate chain in tunnel connection.	See Agent to server connection for information on default locations
TunnelKeepaliveInterval	Interval (in seconds) between keepalive packets over tunnel agent connection.	30
UserAgentExecutable	Name of User Support Application executable used by AutoStartUserAgent and UserAgent-Watchdog parameters.	nxuseragent.exe
UserAgentWatchdog	Enable (yes) or disable (no) automatic restart of User Support Application (Windows only). If enabled, Agent will check once per minute, if User Support Application is running in each user session and will start it if needed. For this to work, Agent should be started under local SYSTEM user.	no
UserId	Username under which NetXMS agent is started (Unix only). See also GroupId parameter.	No defaults
VerifyServerCertificate	Perform server certificate chain verification when establishing tunnel connection. See Agent to server connection for more information.	no
WaitForProcess	If specified, an agent will pause initialization until given process starts.	No defaults
WriteLogAsJson	Enable (yes) or disable (no) writing log file in JSON format.	no
ZoneUIN	Allows to set agent's zone explicitly. This can be useful when agent forwards syslog or SNMP traps of devices, that belong to a particular zone. Agent will include zone UIN along with the trap message that will allow correct matching of traps.	No defaults

Note

All boolean parameters understand “Yes/No”, “On/Off” and “True/False” values.

46.4 Server configuration file (netxmsd.conf)

Parameter	Description	Default Value
AuditLogKey	Key for audit log entry signing using HMAC .	Empty string
BackgroundLogWriter	Enables separate thread that writes log in blocks.	no
CodePage	Code page used by NetXMS server. Has no effect on Windows or if server was compiled without iconv support.	Depends on your system, usually ISO8859-1
CreateCrashDumps	Control creation of server's crash dumps. Possible values: yes or no. Has effect only on Windows platforms.	no
CRL	Certificate revocation list - path to local file or http/https url. Supports and autodetects PEM and DER formats. Multiple such entries can be present in the configuration file.	No default value
DailyLogFileSuffix	Log file name suffix used when <code>LogRotationMode</code> is set to 1 (daily), can contain <code>strftime(3C)</code> macros	%Y%m%d
DataDirectory	Directory where server looks for compiled MIB files, keep server encryption key, etc.	On UNIX-like platforms: 'prefix'/var/lib/netxms. 'prefix' is set during build configuration with --prefix='prefix' parameter. If that parameter was not specified during build, /usr/local is used. If installed from .deb packages: /var/lib/netxms. On Windows: 'Installation folder'\netxms\var where 'Installation folder' is the folder to which NetXMS server is installed.
DBCachedConfigurationTables	Cache configuration tables to in-memory sqlite database to speed up server startup	yes
DBDriver	Database driver to be used.	No default value
DBDriverOptions	Additional driver-specific parameters.	Empty string
DBDrvParams	Deprecated, replaced by <code>DBDriverOptions</code>	Empty string
DBLogin	Database user name.	netxms
DBName	Database name (not used by ODBC driver).	netxms_db
DBPassword	Database user's password. When using INI configuration file format, remember to enclose password in double quotes ("password") if it contains # character.	Empty password
DBEncryptedPassword	Hashed password, as produced by "nxencpass"	none
DBSchema	Schema name	not set
DBServer	Database server (ODBC source name for ODBC driver).	localhost
DBSessionSetup-SQLScript	Path to a plain text file containing a list of SQL commands which will be executed on every new database connection, including initial connection on server startup.	Empty string
DebugLevel	Set server debug logging level (0 - 9). Value of 0 turns off debugging, 9 enables very detailed logging. Can also be set with command line <code>-D <level></code> option.	0

continues on next page

Table 2 – continued from previous page

Parameter	Description	Default Value
DebugTags	Set server debug logging level (0 - 9) for exact log tag or log tag mask. Value of 0 turns off debugging, 9 enables very detailed logging. Configuration should look like <code>debugTag:logLevel</code> (like <code>agent.tunnel.*:4</code>). To configure multiple log tags, you should use multiple DebugTags parameters or write them coma separated (like <code>crypto.*:8,agent.tunnel.*:4</code>).	Empty string
DefaultThreadStackSize	Advanced feature, please contact support prior to changing. This parameter supports (K, M, G, T suffixes).	1M
DumpDirectory	Directory for storing crash dumps.	"/" or "C:"
FullCrashDumps	Write full crash dump instead of minidump (Windows only)	no
InternalCACertificate	Path to file of server CA certificate. This certificate is used to issue agent certificates. InternalCACertificate parameter also implies that this certificate is trusted by the server when checking agent certificate validity.	Empty string
InternalCACertificateKey	Private key of server CA certificate. Can be omitted if key is included in server certificate file.	Empty string
InternalCACertificatePassword	Password of server CA certificate. Can be omitted if certificate does not use password.	Empty string
LibraryDirectory	Defines location of library folder where drivers (ndd files) are stored. It's highly recommended not to use this parameter.	Empty string
ListenAddress	Interface address which should be used by server to listen for incoming connections. Use value 0.0.0.0 or * to use all available interfaces.	0.0.0.0
LogFile	Server's log file. To write log to syslog (or Event Log on Windows), use {syslog} as file name.	{syslog}
LogHistorySize	Number rotated files to keep, older will be discarded	4
LogRotationMode	Define log rotation mode. Possible values are: <ul style="list-style-type: none"> • 0 - No rotation; • 1 - Daily rotation (log will be rotated every midnight); • 2 - Rotation by size (log will be rotated when it's size will exceed value defined by MaxLogSize parameter). 	2
MaxClientMessageSize	Advanced feature, please contact support prior to changing. This parameter supports (K, M, G, T suffixes).	4M
MaxClientSessions	Maximum number of client sessions.	256
MaxLogSize	Maximum log file size in bytes, used only if LogRotationMode is set to 2. This parameter supports (K, M, G, T suffixes).	16M
Module	Additional server module to be loaded at server startup. You can use more then one Module parameters to load multiple modules.	No default value

continues on next page

Table 2 – continued from previous page

Parameter	Description	Default Value
PeerNode	IP address of peer node in high availability setup. If there is lock in the database with this address indicated, server process will communicate to agent and server on that address to ensure the server is not running prior to removing the database lock.	No default value
PerfDataStorageDriver	Name of fanout driver used to send collected data to an additional database. Multiple such parameters can be specified in the configuration file to specify multiple drivers. See Fanout drivers for more information.	Empty string
ProcessAffinityMask	Sets a processor affinity mask for the netxmssd process (Windows only). A process affinity mask is a bit vector in which each bit represents a logical processor on which the threads of the process are allowed to run. See this MSDN article for more details.	0xFFFFFFFF
ServerCertificate	Path to file of server certificate for agent tunnel connections. This certificate is used to issue agent certificates. ServerCertificate parameter also implies that this certificate is trusted by the server when checking agent certificate validity.	Empty string
ServerCertificateKey	Private key of server certificate. Can be omitted if key is included in server certificate file.	Empty string
ServerCertificatePassword	Password of server certificate. Can be omitted if certificate does not use password.	Empty string
StartupSQLScript	Path to a plain text file containing a list of SQL commands which will be executed on server startup.	Empty string
TrustedCertificate	Certificate issued by certificate authority or self-signed CA certificate. If certificate chain for server certificate is longer, all upper level certificates should be added to configuration file by adding multiple TrustedCertificate entries.	Empty string
TunnelCertificate	Path to file of server certificate for agent tunnel connections.	Empty string
TunnelCertificateKey	Private key of server tunnel certificate. Can be omitted if key is included in server certificate file.	Empty string
TunnelCertificatePassword	Password of server tunnel certificate. Can be omitted if certificate does not use password.	Empty string
WriteLogAsJson	Write server log in JSON format.	no

Note

All boolean parameters accept “Yes/No”, “On/Off” and “True/False” values.

46.5 Server configuration parameters

These parameters can be changed in *Configuration ▶ Server Configuration*

Parameter	Description	Default Value	Restart Re-quired
ActionExecution-Log.RetentionTime	Retention time in days for the records in server action execution log. All records older then specified will be deleted by housekeeping process.	90	No
Agent.CommandTimeout	Timeout in milliseconds for commands sent to agent. If agent did not respond to command within this time, command considered as failed.	4000	Yes
Agent.DefaultCacheMode	Default agent cache mode	Off	Yes
Agent.DefaultEncryption	Set the default encryption policy for communications with agents: 0 - encryption disabled, 1 - allowed, 2 - preferred, 3 - required.	Allowed	Yes
Agent.DefaultAgentProtocol	Default agent protocol compression mode	Enabled	No
Agent.EnableRegistration	Enable/disable agents self-registration.	true	No
Agent.RestartWaitTime	Period of time (in seconds) after agent restart for which server will not perform status, configuration, and other polls on the agent.	0	No
Agent.Upgrade.NumberOfThreads	The number of threads used to perform agent upgrades (i.e. maximum number of parallel upgrades).	10	No
Agent.Upgrade.WaitTime	Maximum wait time in seconds for agent restart after upgrade. If agent cannot be contacted after this time period, upgrade process is considered as failed.	600	No
AgentPolicy.MaxFileSize	Maximum file size for exported files in agent policies. Files larger then this size will not be included when exporting configuration to .xml.	16777215	Yes
AgentTunnels.Certificates.ReissueInterval	Interval in days for newly issued agent certificates.	30	Yes
AgentTunnels.Certificates.ValidityPeriod	Validity period in days for newly issued agent certificates.	90	Yes
AgentTunnels.ListenPort	TCP port number to listen on for incoming agent tunnel connections	4703	Yes
AgentTunnels.NewNodesContainer	Name of the container where nodes that were created automatically for unbound tunnels will be placed. If several containers with that name are present, it is not guaranteed, which container will be selected. If empty, such nodes will be created in infrastructure services root.		No
AgentTunnels.TLS.MinVersion	Minimal version of TLS protocol used on agent tunnel connection.	1.2	No
AgentTunnels.UnboundTunnelTimeout	Unbound agent tunnels inactivity timeout. If tunnel has not been bound or closed after that timeout, action defined by AgentTunnels.UnboundTunnelTimeoutAction parameter will be taken.	3600	No
AgentTunnels.UnboundTunnelTimeoutAction	Action to be taken when unbound agent tunnel timeout expires.	Reset Tunnel	No
Alarms.DeleteAlarmsOnDeletion	Enable/disable automatic alarm removal of an object when it is deleted.	true	No
Alarms.EnableTimedAcknowledgment	Enable/disable ability to acknowledge an alarm for a specific time.	true	Yes
Alarm.HistoryRetentionPeriod	Number of days the server keeps alarm history in the database.	180	No

continues on next page

Table 3 – continued from previous page

Parameter	Description	Default Value	Restart Re- quired
Alarms.IgnoreHelpdesk	If set, alarm helpdesk state will be ignored when resolving or terminating.	false	No
Alarms.ResolveExpiration	Expiration time (in seconds) for resolved alarms. If set to non-zero, resolved and untouched alarms will be terminated automatically after given timeout.	0	No
Alarm.StrictStatusFlow	This parameter describes if alarm status flow should be strict (alarm can be terminated only after it was resolved).	false	No
Alarms.SummaryEmail	Enable/disable alarm summary emails. Summary emails will be sent via notification channel specified in DefaultNotificationChannel.SMTP.Html server configuration parameter.	false	No
Alarms.SummaryEmail	A semicolon separated list of e-mail addresses to which the alarm summary will be sent.		No
Alarms.SummaryEmail	Schedule for sending alarm summary e-mails in cron format. See Cron format for supported cron format options.	0 0 * * *	No
As-setChangeLog.Retention	Retention time in days for the records in asset change log. All records older then specified will be deleted by housekeeping process.	90	No
Audit-Log.External.Facility	Syslog facility to be used in audit log records sent to external server.	13	Yes
Audit-Log.External.Port	UDP port of external syslog server to send audit records to.	514	Yes
Audit-Log.External.Server	External syslog server to send audit records to. If set to “none”, external audit logging is disabled.	none	Yes
Audit-Log.External.Severity	Syslog severity to be used in audit log records sent to external server.	5	Yes
Audit-Log.External.Tag	Syslog tag to be used in audit log records sent to external server.	netxmsd-audit	Yes
Audit-Log.External.UseUTF8	Changes audit log encoding to UTF-8	false	No
Audit-Log.RetentionTime	Retention time in days for the records in audit log. All records older than specified will be deleted by housekeeping process.	90	No
Beacon.Hosts	Comma-separated list of hosts to be used as beacons for checking NetXMS server network connectivity. Either DNS names or IP addresses can be used. This list is pinged by NetXMS server and if none of the hosts have responded, server considers that connection with network is lost and generates specific event.		Yes
Beacon.PollingInterval	Interval in milliseconds between beacon hosts polls.	1000	Yes
Beacon.Timeout	Timeout in milliseconds to consider beacon host unreachable.	1000	Yes
BlockInactiveUserAccounts	Inactivity time after which user account will be blocked (0 to disable blocking).	0	No
BusinessServices.Check.AutobindC	Class filter for automatic creation of business service checks.	AccessPoint, Cluster, In- terface, Net- workService, Node	No

continues on next page

Table 3 – continued from previous page

Parameter	Description	Default Value	Restart Re-quired
BusinessServices.Check.Threshold.I	Default threshold for business DCI service checks	Warning	No
BusinessServices.Check.Threshold.O	Default threshold for business service object checks	Warning	No
BusinessServices.History.Retention	Retention time for business service historical data	90	No
CAS.AllowedProxies	Comma-separated list of allowed CAS (Central Authentication Service) proxies.		No
CAS.Host	CAS server DNS name or IP address.	localhost	No
CAS.Port	CAS server TCP port number.	8443	No
CAS.Service	Service to validate (usually NetXMS web UI URL).	https://127.0.0.1/nxmc	No
CAS.TrustedCACert	File system path to CAS server trusted CA certificate.		No
CAS.ValidateURL	URL for service validation on CAS server.	/cas/serviceValidate	No
CertificateActionLog.RetentionTime	Retention time in days for certificate action log. All records older then specified will be delete by housekeeping process.	370	No
Client.AlarmList.Display	Maximum alarm count that will be displayed on <i>Alarm Browser</i> page. Alarms that exceed this count will not be shown.	4096	No
Client.DashboardDataExport	Enable/disable data interpolation in dashboard data export.	true	Yes
Client.DefaultConsoleDateFormat	Default format to display date for GUI.	dd.MM.yyyy	No
Client.DefaultConsoleShortTimeFormat	Default short time display format for GUI.	HH:mm	No
Client.DefaultConsoleLongTimeFormat	Default long time display format for GUI.	HH:mm:ss	No
Client.KeepAliveInterval	Interval in seconds between sending keep alive packets to connected clients.	60	Yes
Client.ListenerPort	The server port for incoming client connections (such as management client).	4701	Yes
Client.MinVersion	The minimum client version allowed to connection to this server.		No
Client.MinViewRefresh	Minimal interval between view refresh in milliseconds (hint for client).	300	No
Client.ObjectBrowser.AutoFilter	Enable/disable object browser's filter applying as user types (if disabled, user has to press ENTER to apply filter).	true	No
Client.ObjectBrowser.FilterDelay	Delay (in milliseconds) between typing in object browser's filter and applying it to object tree.	300	No
Client.ObjectBrowser.MinFilterLength	Minimal length of filter string in object browser required for automatic apply.	1	No
Client.TileServerURL	The base URL for the tile server used to draw maps.	http://tile.netxms.org/osm/	No
DataCollection.ApplyDCIFromTemplate	Enable applying all DCIs from a template to the node, including disabled ones.	true	Yes
DataCollection.DefaultDCIPollingInterval	Default polling interval for newly created DCI (in seconds).	60	No
DataCollection.DefaultDCIRetentionTime	Default retention time for newly created DCI (in days).	30	No
DataCollection.InstancePollingInterval	Instance polling interval (in seconds).	600	Yes

continues on next page

Table 3 – continued from previous page

Parameter	Description	Default Value	Restart Re-quired
DataCollection.InstanceRetentionT	Default retention time (in days) for missing DCI instances.	7	No
DataCollection.OfflineDataRelevant	Time period in seconds within which received offline data still relevant for threshold validation	86400	Yes
DataCollection.OnDCIDelete.Tern	Enable/disable automatic termination of related alarms when data collection item is deleted.	true	No
DataCollection.ScriptErrorReportIn	Minimal interval (seconds) between reporting errors in data collection related script.	86400	No
DataCollection.StartupDelay	Enable/disable randomized data collection delays on server startup for more even server load distribution.	false	Yes
DataCollection.TemplateRemovalC	Setting up grace period (in days) for removing templates from target.	0	No
DataCollection.ThresholdRepeatIn	System-wide interval in seconds for resending threshold violation events. Value of 0 disables event resending.	0	Yes
DBConnection-Pool.BaseSize	Number of connections to the database created on the server startup.	10	Yes
DBConnection-Pool.CooldownTime	Inactivity time (in seconds) after which database connection will be closed.	300	Yes
DBConnection-Pool.MaxLifetime	Maximum lifetime (in seconds) for a database connection.	14400	Yes
DBConnection-Pool.MaxSize	Maximum number of connections in the connection pool.	30	Yes
DB-Writer.BackgroundWorl	Number of background workers for DCI data writer.	1	Yes
DB-Writer.DataQueues	Number of queues for DCI data writer.	1	Yes
DB-Writer.HouseKeeperInt	Controls if server should block background write of collected performance data while housekeeper deletes expired records. Auto enables this feature is server is running on MsSQL database.	Auto	No
DB-Writer.InsertParallelism	Degree of parallelism for INSERT statements executed by DCI data writer (only valid for TimescaleDB).	1	Yes
DB-Writer.MaxQueueSize	Maximum size for DCI data writer queue (0 to disable size limit). If writer queue size grows above that threshold any new data will be dropped until queue size drops below threshold again.	0	No
DB-Writer.MaxRecordsPer	Maximum number of records per one SQL statement for delayed database writes	100	Yes
DB-Writer.MaxRecordsPer	Maximum number of records per one transaction for delayed database writes	1000	Yes
DB-Writer.RawDataFlushIn	Interval between writes of accumulates war DCI data to database.	30	Yes
DB-Writer.UpdateParallelis	Degree of parallelism for UPDATE statements executed by raw DCI data writer.	1	Yes
DefaultNotification-Channel.SMTP.Html	Default notification channel for SMTP HTML formatted messages.	SMTP-HTML	No
DefaultNotification-Channel.SMTP.Text	Default notification channel for SMTP text formatted messages.	SMTP-Text	No

continues on next page

Table 3 – continued from previous page

Parameter	Description	Default Value	Restart Re-quired
EnableISCListener	Enable/disable Inter-Server Communications Listener.	false	Yes
Events.Correlation.Topo	Enable/disable topology based event correlation.	true	No
Events.DeleteEventsOfI	Enable/disable automatic event removal of an object when it is deleted.	true	No
Events.LogRetentionTir	Retention time in days for the records in event log. All records older than specified will be deleted by housekeeping process.	90	No
Events.Processor.PoolSi	Number of threads for parallel event processing.	1	Yes
Events.Processor.Queue	Queue selector for parallel event processing. In parallel processing server ensures that events having same queue selector will be processed in one queue.	%z	Yes
Events.ReceiveForward	Enable/disable reception of events forwarded by another NetXMS server. Please note that for external event reception ISC listener should be enabled as well.	false	No
EventStorm.Duration	Time period for events per second to be above threshold that defines event storm condition.	15	Yes
EventStorm.EnableDete	Enable/disable event storm detection.	false	Yes
EventStorm.EventsPerS	Threshold for number of events per second that defines event storm condition.	1000	Yes
Geoloca- tion.History.RetentionT	Retention time in days for object's geolocation history. All records older then specified will be deleted by housekeeping process.	90	No
HelpDeskLink	Helpdesk driver name. If "none", then no helpdesk driver is in use.	none	Yes
House- keeper.DisableCollectec	Disable automatic cleanup of collected DCI data during housekeeper run.	false	No
House- keeper.StartTime	Time when housekeeper starts. Housekeeper deletes expired log recored and DCI data as well as cleans removed objects.	02:00	Yes
House- keeper.Throttle.HighWa	If database writer queue length (in queue elements) exceeds this number, housekeeper process is paused.	250000	No
House- keeper.Throttle.LowWa	If housekeeper got paused due to DB writer queue reaching Housekeeper.Throttle.HighWatermark, it will resume operation when DB writer queue becomes lower then this setting.	50000	No
ICMP.CollectPollStatist	Collect ICMP poll statistics for all nodes by default. See ICMP ping chapter for information.	1	No
ICMP.PingSize	Size of ICMP packets (in bytes, excluding IP header size) used for status polls.	46	Yes
ICMP.PingTimeout	Timeout for ICMP ping used for status polls (in milliseconds).	1500	Yes
ICMP.PollingInterval	Interval between ICMP statistic collection polls (in seconds)	60	No
ICMP.StatisticPeriod	Time period for collecting ICMP statistics (in number of polls).	60	No
Jira.IssueType	Jira issue type	Task	No
Jira.Login	Jira login	netxms	No
Jira.Password	Jira password		No
Jira.ProjectCode	Jira project code	NETXMS	No
Jira.ProjectComponent	Jira project component		No
Jira.ResolvedStatus	Comma separated list of issue status codes indicating that issue is resolved.		No
Jira.ServerURL	The URL of Jira server	http://localhost	No

continues on next page

Table 3 – continued from previous page

Parameter	Description	Default Value	Restart Re-quired
Jira.Webhook.Path	Path part of Jira webhook URL (must start with /).	/jira-webhook	Yes
Jira.Webhook.Port	Jira webhook listener port (0 to disable webhook).	8008	Yes
JobRetryCount	Maximum number of job execution retries.	5	No
LDAP.ConnectionString	The LdapConnectionString configuration parameter may be a comma- or whitespace-separated list of URIs containing only the schema, the host, and the port fields. Apart from ldap, other (non-standard) recognized values of the schema field are ldaps (LDAP over TLS), ldapi (LDAP over IPC), and cldap (connectionless LDAP). If other fields are present, the behavior is undefined. Format: schema://host:port. For more information refer to Integration with LDAP chapter.	ldap://localhost:389	No
LDAP.GroupClass	Specifies which object class represents group objects. If found entry will not be of a user or group class, it will be ignored.		No
LDAP.GroupUniqueId	Unique identifier for LDAP group object. If not set, LDAP users are identified by DN.		No
LDAP.Mapping.Description	The name of an attribute whose value will be used as a user's description.		No
LDAP.Mapping.Email	The name of an attribute whose value will be used as a user's email.	displayName	No
LDAP.Mapping.FullName	The name of an attribute whose value will be used as a user's full name.	displayName	No
LDAP.Mapping.GroupLoginName	The name of an attribute whose value will be used as group's login name		No
LDAP.Mapping.PhoneNumber	The name of an attribute whose value will be used as group's phone number		No
LDAP.Mapping.UserName	The name of an attribute whose value will be used as a user's login name.	displayName	No
LDAP.NewUserAuthMethod	Authentication method to be set for user object created by LDAP synchronization process.	LDAP password	No
LDAP.PageSize	The maximum amount of records that can be returned in one search page.	1000	No
LDAP.SearchBase	The DN of the entry at which to start the search.		No
LDAP.SearchFilter	A string representation of the filter to apply in the search.		No
LDAP.SyncInterval	The synchronization interval (in minutes) between the NetXMS server and the LDAP server. If the parameter is set to 0, no synchronization will take place.	0	No
LDAP.SyncUser	User login for LDAP synchronization		No
LDAP.SyncUserPassword	User password for LDAP synchronization		No
LDAP.UserClass	The object class which represents user objects. If the found entry is not of user or group class, it will be ignored.		No
LDAP.UserDeleteAction	This parameter specifies what should be done while synchronization with deleted from LDAP user/group. 0 - if user should be just deleted from NetXMS DB. 1 - if it should be disabled. If it is chosen to disable user, then on LDAP sync user will be disabled and it's description will be change on "LDAP entry was deleted." Afterwards this user/group can be detached from LDAP and enabled if it is required or just deleted manually.	Disable user	No

continues on next page

Table 3 – continued from previous page

Parameter	Description	Default Value	Restart Re-quired
LDAP.UserUniqueId	Unique identifier for LDAP user object. If not set, LDAP users are identified by DN.		No
LongRunningQuery-Threshold	Threshold in milliseconds to report long running SQL queries (0 to disable). Queries are logged to NetXMS server log file on debug level 3.	0	Yes
MaintenanceJournal.RetentionTime	Retention time in days for maintenance journal entries. All records older then specified will be deleted by housekeeping process.	1826	No
MobileDeviceListenerPort	Listener port for connections from NetXMS mobile agent.	4747	Yes
NetworkDeviceDrivers.BlackList	Comma separated list of blacklisted network device drivers.		Yes
NetworkDiscovery.ActiveDiscovery.BlockSize	Size of address block to which ICMP ping requests are sent simultaneously during active discovery.	1024	No
NetworkDiscovery.ActiveDiscovery.EnableSnmp	Enable/disable SNMP probing during active network discovery. If enabled, server will send SNMP requests to detect devices that respond to SNMP, but not to ICMP pings.	true	No
NetworkDiscovery.ActiveDiscovery.EnableTcp	Enable/disable TCP probing during active network discovery. If enabled, server will try to establish TCP connection to list of well-known ports to detect devices that are not responding to ICMP pings. This setting is changed by Network Discovery Configuration GUI	false	No
NetworkDiscovery.ActiveDiscovery.Interval	Pause in milliseconds between scanning of blocks during active discovery. Together with BlockSize this allows to slow down active discovery if network equipment treats large number of ICMP requests as flood.	0	No
NetworkDiscovery.ActiveDiscovery.IntervalSecs	Interval in seconds between active network discovery polls. This setting is changed by Network Discovery Configuration GUI	7200	No
NetworkDiscovery.ActiveDiscovery.Schedule	Active network discovery poll schedule in cron format. This setting is changed by Network Discovery Configuration GUI		No
NetworkDiscovery.DisableProtocolProbing	Disable probing discovered addresses for NetXMS agent.	false	No
NetworkDiscovery.DisableProtocolProbingEthernetIP	Disable probing discovered addresses for Ethernet/IP support.	false	No
NetworkDiscovery.DisableProtocolProbingSnmp1	Disable SNMP version 1 when probing discovered addresses for SNMP support.	false	No
NetworkDiscovery.DisableProtocolProbingSnmp2	Disable SNMP version 2 when probing discovered addresses for SNMP support.	false	No
NetworkDiscovery.DisableProtocolProbingSnmp3	Disable SNMP version 3 when probing discovered addresses for SNMP support.	false	No
NetworkDiscovery.DisableProtocolProbingSsh	Disable probing discovered addresses for SSH support.	false	No
NetworkDiscovery.EnableParallelProcessing	Enable/disable parallel processing of discovered addresses.	false	No
NetworkDiscovery.Filter.Flags	Discovery filter settings. This setting is changed by Network Discovery Configuration GUI	0	No

continues on next page

Table 3 – continued from previous page

Parameter	Description	Default Value	Restart Re-quired
NetworkDiscovery.Filter.Script	Name of discovery filter script from script library. This setting is changed by Network Discovery Configuration GUI	none	No
NetworkDiscovery.MergeDuplicateNodes	Enable/disable merging of duplicate nodes (that may be created due to parallel processing of discovered addresses).	false	No
NetworkDiscovery.PassiveDiscovery.Interval	Interval in seconds between passive network discovery polls. This setting is changed by Network Discovery Configuration GUI	900	No
NetworkDiscovery.Type	Defines enabled modes of network discovery. This setting is changed by Network Discovery Configuration GUI	Disabled	No
NetworkDiscovery.UseDNSNameForDiscovery	Enable/disable the use of DNS name instead of IP address as primary name for newly discovered nodes. If enabled, server will do back resolve of IP address, and then resolve obtained name back to IP address. Only if this IP address will match the original one, DNS name will be used.	false	No
NetworkDiscovery.UseFQDNForNodes	Enable/disable the use of fully qualified domain names as primary names for newly discovered nodes.	true	Yes
NetworkDiscovery.UseSNMPTraps	This parameter defines if trap information should be used for new node discovery.	false	Yes
NetworkDiscovery.UseSyslog	Enable/disable use of syslog messages for new node discovery.	false	Yes
NotificationChannels.MaxRetryCount	Maximum count of retries to send a message for all notification channels.	3	No
NotificationLog.RetentionTime	Retention time in days for the records in notification log. All records older then specified will be deleted by housekeeper process.	90	No
NXSL.EnableContainer	Enable/disable server-side NXSL functions for containers (such as CreateContainer, BindObject, etc.).	true	No
NXSL.EnableFileIOFunctions	Enable/disable server-side NXSL functions for file I/O (such as OpenFile, DeleteFile, etc.).	false	No
Objects.AccessPoints.Container	Enable/disable container auto binding for access points.	false	No
Objects.AccessPoints.Template	Enable/disable template auto apply for access points.	false	No
Objects.Assets.AllowDeletion	Enable/disable deletion of linked assets.	false	No
Objects.AutobindOnConfigurationPolls	Enable/disable automatic object binding on configuration polls.	true	No
Objects.AutobindPollingInterval	Interval in seconds between automatic object binding polls.	3600	No
Objects.Clusters.Container	Enable/disable container auto binding for clusters.	false	No
Objects.Clusters.Template	Enable/disable template auto apply for clusters.	false	No
Objects.Conditions.PollingInterval	Interval in seconds between polling (re-evaluating) of condition objects.	60	Yes
Objects.ConfigurationPollingInterval	Interval in seconds between configuration polls.	3600	Yes

continues on next page

Table 3 – continued from previous page

Parameter	Description	Default Value	Restart Re-quired
Ob-jects.DeleteUnreachable	Delete nodes which were unreachable for a number of days specified by this parameter. If this parameter is set to 0 then unreachable nodes will never be deleted.	0	Yes
Objects.EnableZoning	Enable/disable zoning support.	true	Yes
Ob-jects.Interfaces.DefaultIf	Default expected state for new interface objects.	AUTO	No
Ob-jects.Interfaces.Enable8	Globally enable or disable checking of 802.1x port state during status poll.	true	No
Ob-jects.Interfaces.NamePa	Custom name pattern for interface objects. This field supports macros. E.g. if set to %n%{suffix}, interface name will be composed from original name and node's custom attribute suffix.		No
Ob-jects.Interfaces.UseAlia	Control usage of interface aliases (or descriptions). Possible values are: <ul style="list-style-type: none"> • 0 - Always use name (Don't use aliases) • 1 - Use aliases instead of names, when possible • 2 - Concatenate alias and name to form interface object name • 3 - Concatenate name and alias to form interface object name 	Don't use aliases	No
Ob-jects.Interfaces.UseIfX	Enable/disable the use of SNMP ifXTable instead of ifTable for interface configuration polling. See SNMP for more information.	true	No
Ob-jects.MobileDevices.Co	Enable/disable container auto binding for mobile devices.	false	No
Ob-jects.MobileDevices.Te	Enable/disable template auto apply for mobile devices.	false	No
Ob-jects.NetworkMaps.Def	Default background color for new network map objects (as RGB value).	0xffffffff	No
Ob-jects.Nodes.CapabilityE	Grace period (in seconds) for capability expiration after node recovered from unreachable state.	3600	No
Ob-jects.Nodes.CapabilityE	Time (in seconds) before capability (NetXMS Agent, SNMP, Ethernet/IP, etc) expires if node is not responding for requests via appropriate protocol.	604800	No
Ob-jects.Nodes.FallbackTo	Enable/disable fallback to server's local resolver if node address cannot be resolved via zone proxy.	false	No
Ob-jects.Nodes.ResolveDN	Enable/disable resolve DNS to IP on status poll.	Never	No
Ob-jects.Nodes.ResolveDN	Number of status polls between resolving primary host name to IP, if Objects.Nodes.ResolveDNToIPOnStatusPoll set to "Always".	0	No
Ob-jects.Nodes.ResolveNar	Resolve node name using DNS, SNMP system name, or host name if current node name is it's IP address.	true	No
Ob-jects.Nodes.Resolver.Ac	Address family hint for node DNS name resolver.	None	No
Ob-jects.Nodes.SyncNames	Enable/disable synchronization of node names with DNS on each configuration poll.	false	No

continues on next page

Table 3 – continued from previous page

Parameter	Description	Default Value	Restart Re-quired
Ob-jects.PollCountForStatus	The number of consecutive unsuccessful polls required to declare interface as down.	1	Yes
Ob-jects.ResponsibleUsers.	Allowed tags for responsible users (Comma-separated list).		No
Ob-jects.Security.CheckTru	Enable/disable trusted objects checks for cross-object access.	false	No
Ob-jects.Sensors.Container.	Enable/disable container auto binding for sensors.	false	No
Ob-jects.Sensors.Template/	Enable/disable template auto apply for sensors.	false	No
Ob-jects.StatusCalculation.C	Default algorithm for calculation object status from it's DCIs, alarms and child objects. Possible values are: <ul style="list-style-type: none"> • Most critical • Single threshold. Threshold value is defined by StatusSingleThreshold parameter. • Multiple thresholds. Threshold values are defined by StatusThresholds parameter. 	Most critical	Yes
Ob-jects.StatusCalculation.I	Value for status propagation if "StatusPropagationAlgorithm" server configuration parameter is set to "2 - Fixed".	0	Yes
Ob-jects.StatusCalculation.I	Default algorithm for status propagation (how object's status is affected by it's child object statuses). Possible values are: <ul style="list-style-type: none"> • Unchanged • Fixed. Status value is defined by FixedStatusValue parameter. • Relative with offset. Offset value is defined by StatusShift parameter. • Translated. Status translation is defined by Status-Translation parameter. 	Unchanged	Yes
Ob-jects.StatusCalculation.S	Status shift value for Relative propagation algorithm.	0	Yes
Ob-jects.StatusCalculation.S	Threshold value (in %) for Single threshold status calculation algorithm.	75	Yes
Ob-jects.StatusCalculation.S	Threshold values for Multiple thresholds status calculation algorithm. Every byte (from left to right) of this hex number express threshold values for warning, minor, major and critical statuses.	503C2814 (80%, 60%, 40%, 20%)	Yes
Ob-jects.StatusCalculation.S	Values for Translated status propagation algorithm. Every byte (from left to right) of this hex number defines status translation for Warning, Minor, Major and Critical statuses. Status values are: <ul style="list-style-type: none"> • 1 - Warning • 2 - Minor • 3 - Major • 4 - Critical 	01020304	Yes

continues on next page

Table 3 – continued from previous page

Parameter	Description	Default Value	Restart Re-quired
Ob-jects.StatusPollingInterval	Interval in seconds between status polls.	60	Yes
Ob-jects.Subnets.DefaultSubnetMask	Default mask for synthetic IPv6 subnets.	24	No
Ob-jects.Subnets.DefaultSubnetPrefix	Default mask for synthetic IPv6 subnets.	64	No
Ob-jects.Subnets.DeleteEmpty	Enable/disable automatic deletion of subnet objects that have no nodes within. When enabled, empty subnets will be deleted by housekeeping process.	false	Yes
Objects.SyncInterval	Interval in seconds between writing object changes to the database.	60	Yes
RA-DIUS.AuthMethod	RADIUS authentication method to be used (PAP, CHAP, MS-CHAPv1, MS-CHAPv2).	PAP	No
RA-DIUS.NASIdentifier	Value for NAS-Identifier attribute in RADIUS request (will not be sent if empty)	none	No
RADIUS.NumRetries	The number of retries for RADIUS authentication.	5	No
RADIUS.Port	Port number used for connection to primary RADIUS server.	1645	No
RA-DIUS.SecondaryPort	Port number used for connection to secondary RADIUS server.	1645	No
RA-DIUS.SecondarySecret	Shared secret used for communication with secondary RADIUS server.	netxms	No
RA-DIUS.SecondaryServer	Host name or IP address of secondary RADIUS server.	none	No
RADIUS.Secret	Shared secret used for communication with primary RADIUS server.	netxms	No
RADIUS.Server	Host name or IP address of primary RADIUS server.	none	No
RA-DIUS.ServiceType	Value for Service-Type attribute in RADIUS request. Value of 0 will exclude service type from request attributes.	8	No
RADIUS.Timeout	Timeout in seconds for requests to RADIUS server	3	No
ReportingServer.Enable	Enable/disable reporting server	false	Yes
ReportingServer.Hostname	The hostname of the reporting server.	127.0.0.1	Yes
ReportingServer.Port	The port of the reporting server.	4710	Yes
Scheduler.TaskRetentionTime	Period (in seconds) after which non-recurring scheduled tasks (e.g. Maintenance enter / Maintenance leave) are deleted.	86400	No
Server.AllowedCiphers	A bitmask for encryption algorithms allowed in the server (sum of the values to allow multiple algorithms at once): <ul style="list-style-type: none"> • 1 - AES256 • 2 - Blowfish-256 • 4 - IDEA • 8 - 3DES • 16 - AES128 • 32 - Blowfish-128 	63	Yes
Server.Color	Identification color for this server. Used in status bar of management client.		No

continues on next page

Table 3 – continued from previous page

Parameter	Description	Default Value	Restart Re-quired
Server.CommandOutput	Time (in seconds) to wait for output of a local command object tool.	60	No
Server.EscapeLocalCon	Enable/disable TAB and new line characters replacement by t n r in execute command on management server action.	false	No
Server.ImportConfigura	Import configuration (templates, events, object tools, etc) on server startup. Configuration is imported from files located on NetXMS server in share/templates. Missing elements are identified by GUID.	Only missing elements	Yes
Server.MessageOfTheD	Message to be shown when a user logs into the client.		No
ServerName	Name of this server. Displayed in status bar of management client.		No
Server.RoamingMode	Enable/disable roaming mode for server (when server can be disconnected from one network and connected to another or IP address of the server can change)	true	No
Server.Security.2FA.Tr	TTL (in seconds) for 2FA trusted device.	0	No
Server.Security.CaseIns	Enable/disable case insensitive login names.	false	Yes
Server.Security.Extende	Enable/disable extended access control in log queries. When enabled, server will check user's access to objects and only select those log records where user has read access to related object. Please note that enabling this option can cause slow and inefficient SQL queries depending on number of objects and actual access right assignment.	false	No
Server.Security.GraceL	Number of times a user can login if password has been expired.	5	No
Server.Security.Intruder	Number of incorrect password attempts after which a user account is temporarily locked.	0	No
Server.Security.Intruder	Duration of user account temporarily lockout (in minutes) if allowed number of incorrect password attempts was exceeded.	30	No
Server.Security.MinPas	Default minimum password length for a NetXMS user. The default applied only if per-user setting is not defined.	0	No
Server.Security.Passwor	Set of flags to enforce password complexity (see Password Policy for more details).	0	No
Server.Security.Passwor	Password expiration time in days. If set to 0, password expiration is disabled.	0	No
Server.Security.Passwor	Number of previous passwords to keep. Users are not allowed to set password if it matches one from previous passwords list.	0	No
Server.Security.Restrict	If enabled, restrict access to local server debug console (via nxagm command line tool) only to authenticated users with server debug console access rights.	true	No
SNMP.Codepage	Default server SNMP codepage		No
SNMP.Discovery.Separ	Use separate SNMP request for each test OID.	0	No
SNMP.EngineId	Server SNMP engine ID.	80:00:DF:4B:05:20	Yes
SNMP.RequestTimeout	Timeout in milliseconds for SNMP requests sent by NetXMS server.	1500	Yes
SNMP.RetryCount	Number of retries for SNMP requests sent by NetXMS server.	3	Yes

continues on next page

Table 3 – continued from previous page

Parameter	Description	Default Value	Restart Re-quired
SNMP.Traps.AllowVarl	Allows/disallows conversion of SNMP trap OCTET STRING varbinds into hex strings if they contain non-printable characters.	1	No
SNMP.Traps.Enable	Enable/disable SNMP trap processing. A dedicated thread will be created if set to true.	true	Yes
SNMP.Traps.ListenerPo	Port used for SNMP traps.	162	Yes
SNMP.Traps.LogAll	Log all SNMP traps (even those received from addresses not belonging to any known node).	false	No
SNMP.TrapLogRetenti	The time (in days) how long SNMP trap logs are retained.	90	No
SNMP.Traps.ProcessUr	Enable/disable processing of SNMP traps received from un-managed nodes.	false	No
SNMP.Traps.RateLimit	Time period (in seconds) for SNMP traps per second to be above threshold that defines SNMP trap flood condition.	15	No
SNMP.Traps.RateLimit	Threshold for number of SNMP traps per second that defines SNMP trap flood condition. Detection is disabled if 0 is set.	0	No
SNMP.Traps.SourcesIn	Search all zones to match trap/syslog source address to node.	false	Yes
Sys-log.AllowUnknownSou	Enable or disable processing of syslog messages from unknown sources.	false	No
Syslog.Codepage	Default server syslog codepage.		No
Syslog.EnableListener	Enable/disable receiving of syslog messages.	0	Yes
Syslog.EnableStorage	Enable/disable local storage of received syslog messages in NetXMS database.	true	No
Sys-log.IgnoreMessageTime	Ignore timestamp received in syslog messages and always use server time.	false	No
Syslog.ListenPort	UDP port used by built-in syslog server.	514	Yes
Sys-log.NodeMatchingPolic	Node matching policy for built-in syslog daemon. Possible values are: <ul style="list-style-type: none"> IP,then hostname - syslog message source IP address, then hostname Hostname,then IP - hostname, then syslog message source IP address 	IP,then hostname	Yes
Syslog.RetentionTime	Retention time in days for stored syslog messages. All messages older than specified will be deleted by housekeeping process.	90	No
Thread-Pool.Agent.BaseSize	This parameter represents base thread pool size for threads that receive data, traps, events, etc from agents. This is minimal number of threads that will always run.	32	Yes
Thread-Pool.Agent.MaxSize	This parameter represents maximum thread pool size for threads that receive data, traps, events, etc from agents. In case of high load on existing threads server will increase number of threads up to this value. When load come back to normal, number of threads will be automatically decreased to base size.	256	Yes
Thread-Pool.DataCollector.Bas	This parameter represents base thread pool size for data collector threads. This is minimal number of threads that will always run.	10	Yes

continues on next page

Table 3 – continued from previous page

Parameter	Description	Default Value	Restart Re-quired
Thread-Pool.DataCollector.Max	This parameter represents maximum thread pool size for data collector threads. In case of high load on existing threads server will increase number of threads up to this value. When load come back to normal, number of threads will be automatically decreased to base size.	250	Yes
Thread-Pool.Discovery.BaseSize	This parameter represents base thread pool size for network discovery threads. This is minimal number of threads that will always run.	8	Yes
Thread-Pool.Discovery.MaxSize	This parameter represents maximum thread pool size for network discovery threads. In case of high load on existing threads server will increase number of threads up to this value. When load come back to normal, number of threads will be automatically decreased to base size.	64	Yes
Thread-Pool.Main.BaseSize	This parameter represents base thread pool size for threads that perform general system tasks. This is minimal number of threads that will always run.	8	Yes
Thread-Pool.Main.MaxSize	This parameter represents maximum thread pool size for threads that perform general system tasks. In case of high load on existing threads server will increase number of threads up to this value. When load come back to normal, number of threads will be automatically decreased to base size.	256	Yes
Thread-Pool.Poller.BaseSize	This parameter represents base thread pool size for threads that perform all types of polls: Status poll, Configuration poll, etc. except DCI collection. This is minimal number of threads that will always run.	10	Yes
Thread-Pool.Poller.MaxSize	This parameter represents maximum thread pool size for threads that perform all types of polls: Status poll, Configuration poll, etc. except DCI collection. In case of high load on existing threads server will increase number of threads up to this value. When load come back to normal, number of threads will be automatically decreased to base size.	250	Yes
Thread-Pool.Scheduler.BaseSize	This parameter represents base thread pool size for scheduler threads. This is minimal number of threads that will always run.	1	Yes
Thread-Pool.Scheduler.MaxSize	This parameter represents maximum thread pool size for scheduler threads. In case of high load on existing threads server will increase number of threads up to this value. When load come back to normal, number of threads will be automatically decreased to base size.	64	Yes
Thread-Pool.Syncer.BaseSize	This parameter represents base thread pool size for threads that perform object synchronization to the database. This is minimal number of threads that will always run.	1	Yes
Thread-Pool.Syncer.MaxSize	This parameter represents maximum thread pool size for threads that perform object synchronization to the database. In case of high load on existing threads server will increase number of threads up to this value. When load come back to normal, number of threads will be automatically decreased to base size. Value of 1 will disable pool creation.	1	Yes

continues on next page

Table 3 – continued from previous page

Parameter	Description	Default Value	Restart Re-quired
Topology.AdHocRequest.ExpirationInterval	Ad-hoc network topology request expiration time. Server will use cached result of previous request if it is newer than given interval.	900	No
Topology.DefaultDiscoveryRequests	Default number of hops from seed node to be added to topology map.	5	No
Topology.PollingInterval	Interval in seconds between topology polls.	1800	Yes
Topology.RoutingTableUpdateInterval	Interval in seconds between reading routing table from node.	300	Yes
UserAgent.DefaultMessageRetentionTime	Default user agent message retention time (in minutes).	10800	No
UserAgent.RetentionTime	User agent message historical data retention time (in days).	30	No
WindowsEvents.EnableStorage	Enable/disable local storage of received Windows events in NetXMS database.	true	No
WindowsEvents.LogRetentionTime	Retention time in days for records in Windows event log. All records older than specified will be deleted by housekeeping process.	90	No

46.6 Bundled Subagents

46.7 Command line tools

NetXMS provide some additional command line tools. Each tool serves its own purpose.

46.7.1 Database Manager

This is tool used to make manipulations with NetXMS database.

Usage: `nxdmgrp [options] <command>`

Valid commands are:

background-convert	Convert collected data to TimescaleDB format in background
background-upgrade	Run pending background upgrade procedures
batch <file>	Run SQL batch file
check	Check database for errors
check-data-tables	Check database for missing data tables
convert	Convert standard PostgreSQL schema to TimescaleDB schema
export <file>	Export database to file
get <name>	Get value of server configuration variable
import <file>	Import database from file
init [<type>]	Initialize database. If type is not provided it will be deduced from driver name.
migrate <source>	Migrate database from given source
reset-system-account	Unlock user “system” and reset it’s password to default (“netxms”). Warning: server (“netxmsd”) should be stopped while performing password reset operation! See Resetting “system” user password for detailed procedure.
set <name> <value>	Set value of server configuration variable
unlock	Forced database unlock
upgrade	Upgrade database to new version

Valid options are:

-c <config>	Use alternate configuration file. Default is {search}
-C <dba>	Create database and user before initialization using provided DBA credentials
-d	Check collected data (may take very long time).
-D	Migrate only collected data.
-e <table>	Exclude specific table from export, import, or migration.
-E	Fail check if fix required
-f	Force repair - do not ask for confirmation.
-F <syntax>	Fallback database syntax to use if not set in metadata.
-h	Display help and exit.
-I	MySQL only - specify TYPE=InnoDB for new tables.
-L <log>	Migrate only specific log.
-m	Improved machine readability of output.
-M	MySQL only - specify TYPE=MyISAM for new tables.
-N	Do not replace existing configuration value (“set” command only).
-o	Show output from SELECT statements in a batch.
-P	Pause after error.
-q	Quiet mode (don’t show startup banner).
-s	Skip collected data during export, import, conversion, or migration.
-S	Skip collected data during export, import, or migration and do not clear or create data tables.
-t	Enable trace mode (show executed SQL queries).
-T <recs>	Transaction size for migration.
-v	Display version and exit.
-x	Ignore collected data import/migration errors
-X	Ignore SQL errors when upgrading (USE WITH CAUTION!!!)
-Y <table>	Migrate only given table.
-Z <log>	Exclude specific log from export, import, or migration.

Database initialization

```
nxdbmgr init
```

Used to initialize the database for the first time. Database and user should already exist. Database name and credentials are taken from server configuration file.

Check database for errors

It's recommended to check database for errors when performing server upgrade or after server process has crashed or was killed. Server process should be stopped when performing the check. To perform the check, execute the following command:

```
nxdbmgr check
```

Unlocking database

When NetXMS server process or nxdbmgr starts, it makes a record in the database meaning that it locked this database and no other server process should work with it. This prevents situations when due to incorrect configuration two server processes connect to same database, as this would corrupt data in the database.

When server process or nxdbmgr stops, it would remove the lock. However, if process was not able to stop correctly, the lock could stay in the database and manual unlocking using nxdbmgr might be needed. The procedure is the following:

- 1) Make sure that server process is not running, e.g. on Linux you can check by running:

```
ps aux | grep netxmsd
```

- 2) Unlock database by running:

```
nxdbmgr unlock
```

Database migration

nxdbmgr allows to migrate NetXMS database between different database management systems supported by NetXMS (e.g. from MySQL to Postgres). This also allows to migrate the database from one host to another.

Migration is only possible when NetXMS server process is stopped. It is recommended to perform database check prior to migration with the help of `nxdbmgr check` command.

Connection parameters and credentials for DESTINATION database are taken from server configuration file (or from arbitrary configuration file specified with `-c` option).

Connection parameters and credentials for SOURCE database are taken from same format configuration file that is provided as nxdbmgr parameter.

Destination database should be initialized prior to migration by running `nxdbmgr init`.

To migrate the whole database:

```
nxdbmgr migrate netxmsd-source-db.conf
```

Note

You may need to use full path to .conf file

Migration can also be performed as two-step process - on the first step only configuration data is transferred, then server is started on the new database and collected data and logs are transferred in the background. First step:

```
nxdbmgr -s -Z all migrate netxmsd-source-db.conf
```

After completion and starting server on the new database, run below two commands to transfer collected data and logs:

```
nxdbmgr -D migrate netxmsd-source-db.conf  
nxdbmgr -S -L all migrate netxmsd-old.conf
```

In-place conversion from Postgres to Timescale

nxdbmgr allows to perform in-place conversion from standard PostgreSQL schema to TimescaleDB schema. This is irreversible operation. It's strongly recommended to have database backup prior to running this. Conversion is only possible when NetXMS server process is stopped.

To convert the whole database:

```
nxdbmgr convert
```

Conversion can also be performed in two steps. First step requires server process to be stopped, log tables are converted during that step. Then server can be started and second step - conversion of tables with collected data can be performed. First step:

```
nxdbmgr -s convert
```

After completion and starting server, run the second step:

```
nxdbmgr background-convert
```

Database export and import

nxdbmgr allows convenient way to export and import database. To ensure export data consistency, NetXMS server should be stopped. In large deployments export may take long time.

```
nxdbmgr export mysql_backup.sql
```

It is possible to export configuration without collected DCI data and logs and this can be achieved with -s and -Z switches. Use -e switch to exclude specific tables from export.

```
nxdbmgr -s -Z all -e hardware_inventory -e software_inventory export plsql_  
↪ backup.sql
```

For database import similar syntax and switches apply. One can export full database, but import only configuration or exclude any specific table.

```
nxdbmgr -e tdata_237 import plsql_backup.sql
```

46.7.2 nxaction

nxaction - command line tool used to execute preconfigured actions on NetXMS agent

Usage: nxaction <host> [<options>] <action> [<action args>]

Options:

Source	Description
-D level	Set debug level (0..9 or off, default is off).
-e policy	Set encryption policy. Possible values are: 0 = Encryption disabled; 1 = Encrypt connection only if agent requires encryption; 2 = Encrypt connection if agent supports encryption; 3 = Force encrypted connection; Default value is 1.
-h	Display help and exit.
-K file	Specify server's key file (default is /var/lib/netxms/.server_key).
-o	Show action's output.
-O port	Proxy agent's port number. Default is 4700.
-p port	Agent's port number. Default is 4700.
-s secret	Shared secret for agent authentication.
-S secret	Shared secret for proxy agent authentication.
-v	Display version and exit.
-w seconds	Set command timeout (default is 5 seconds).
-W seconds	Set connection timeout (default is 30 seconds).
-X addr	Use proxy agent at given address.

Example:

```
$ nxaction 127.0.0.1 Agent.Restart
Action executed successfully
```

Note

you can use `nxget -l 127.0.0.1 Agent.ActionList` to query list of available actions from agent

46.7.3 nxadm

Nxadm is used for server console access and script execution; provides built-in commands for server debugging.

Usage:

- `nxadm [-u <login>] [-P|-p <password>] -c <command>`
- `nxadm [-u <login>] [-P|-p <password>] -i`
- `nxadm [-u <login>] [-P|-p <password>] [-r] -s <script>`
- `nxadm -P`
- `nxadm -p <db password>`

Options:

Source	Description
-c <command>	Execute given command at server debug console and disconnect.
-i	Connect to server debug console in interactive mode.
-h	Display help and exit.
-p <password>	Provide database password for server startup or user's password for console access.
-P	Provide database password for server startup or user's password for console access (password read from terminal).
-r	Use script's return value as exit code.
-s <script>	Execute given NXSL script and disconnect.
-u name	User name for authentication.
-v	Display version and exit.

Example

```
$ nxadm -u admin -p admin -i

NetXMS Server Remote Console V5.1.1 Ready
Enter "help" for command list

netxmsd: help
Valid commands are:
at +<sec> <script> [<params>] - Schedule one time script execution task
at <schedule> <script> [<params>] - Schedule repeated script execution task
clear - Show list of valid component names for
↳clearing
clear <component> - Clear internal data or queue for given
↳component
dbcp reset - Reset database connection pool
debug [<level>|off] - Set debug level (valid range is 0..9)
debug [<debug tag> <level>|off|default] - Set debug level for a particular debug
↳tag
debug sql [on|off] - Turn SQL query trace on or off
down - Shutdown NetXMS server
exec <script> [<params>] - Executes NXSL script from script library
exit - Exit from remote session
kill <session> - Kill client session
get <variable> - Get value of server configuration
↳variable
help - Display this help
hkrun - Run housekeeper immediately
ldapsync - Synchronize ldap users with local user
↳database
log <text> - Write given text to server log file
logmark - Write marker ***** MARK ***** to
↳server log file
ping <address> - Send ICMP echo request to given IP
↳address
poll <type> <node> - Initiate node poll
raise <exception> - Raise exception
scan <range start> <range end> [proxy <id>|zone <uin>] [discovery]
↳Manual active discovery scan for given
(continues on next page)
```


(continued from previous page)

```

↪range. Without 'discovery' parameter prints results only
set <variable> <value>           - Set value of server configuration↵
↪variable
show arp <node>                   - Show ARP cache for node
show authtokens                   - Show user authentication tokens
show components <node>            - Show physical components of given node
show dbcp                         - Show active sessions in database↵
↪connection pool
show dbstats                      - Show DB library statistics
show discovery ranges             - Show state of active network discovery↵
↪by address range
show ep                           - Show event processing threads statistics
show fdb <node>                   - Show forwarding database for node
show flags                       - Show internal server flags
show heap details                 - Show detailed heap information
show heap summary                 - Show heap usage summary
show index <index>                - Show internal index
show modules                      - Show loaded server modules
show ndd                          - Show loaded network device drivers
show objects [<filter>]           - Dump network objects to screen
show pe                           - Show registered prediction engines
show pollers                      - Show poller threads state information
show queues                      - Show internal queues statistics
show routing-table <node>         - Show cached routing table for node
show sessions                    - Show active client sessions
show stats                       - Show global server statistics
show syncer                      - Show syncer statistics
show tasks                       - Show background tasks
show threads [<pool>]             - Show thread statistics
show topology <node>             - Collect and show link layer topology for↵
↪node
show tunnels                     - Show active agent tunnels
show users                       - Show users
show version                     - Show NetXMS server version
show vlans <node>                - Show cached VLAN information for node
show watchdog                    - Display watchdog information
tcping <address> <port>           - TCP ping on given address and port
tp loadtest <pool> <tasks>        - Start test tasks in given thread pool
trace <node1> <node2>             - Show network path trace between two nodes
tunnel bind <tunnel> <node>       - Bind agent tunnel to node
tunnel unbind <node>             - Unbind agent tunnel from node

```

Almost all commands can be abbreviated to 2 or 3 characters

You can use the following shortcuts to execute `command` from history:

```

!!      - Execute last command
!<N>   - Execute Nth command from history
!-<N>  - Execute Nth command back from last one

```

46.7.4 nxaevent

This tool can be used to push events to NetXMS server via local NetXMS agent.

Usage:

- `nxaevent [OPTIONS] event_code [parameters]`
- `nxaevent [OPTIONS] -n event_code [name=parameter ...]`

Source	Description
-h, -help	Display this help message.
-n, -named-parameters	Parameters are provided in named format: name=value.
-o, -object <id>	Send event on behalf of object with given id.
-q, -quiet	Suppress all messages.
-t, -timestamp-unix <time>	Specify timestamp for event as UNIX timestamp.
-T, -timestamp-text <time>	Specify timestamp for event as YYYYMMDDhhmmss.
-v, -verbose	Enable verbose messages. Add twice for debug
-V, -version	Display version information.

Send event to server via agent:

```
nxaevent MY_APP_EVENT

nxaevent -n MY_APP_EVENT state=UP desc="Application started"
```

46.7.5 nxalarm

nxalarm is command line alarm management utility.

Usage: `nxalarm [<options>] <server> <command> [<alarm_id>]`

Commands:

Source	Description
ack <id>	Acknowledge alarm
add-comment <id> <text>	Add comment to alarm
get-comments <id>	Get comments of alarm
list	List active alarms
open <id>	Open helpdesk issue from alarm
resolve <id>	Resolve alarm
terminate <id>	Terminate alarm

Options:

Source	Description
-c	Codepage (default is ISO8859-1)
-D	Turn on debug mode.
-e	Encrypt session (for compatibility only, session is always encrypted).
-h	Display help and exit.
-o <format>	Output format for list (see below).
-P <password>	Specify user's password. Default is empty password.
-s	Sticky acknowledge (only for "ack" command).
-S <minutes>	Sticky acknowledge with timeout (only for "ack" command).
-u <user>	Login to server as <user>. Default is "guest".
-v	Display version and exit.
-w <seconds>	Specify command timeout (default is 3 seconds).

Output format string syntax:

- %a - Primary IP address of source object
- %A - Primary host name of source object
- %c - Repeat count
- %d - Related DCI ID
- %e - Event code
- %E - Event name
- %h - Helpdesk state as number
- %H - Helpdesk state as text
- %i - Source object identifier
- %I - Alarm identifier
- %m - Message text
- %n - Source object name
- %s - Severity as number
- %S - Severity as text
- %x - Alarm state as number
- %X - Alarm state as text
- %% - Percent sign

Default format is %I %S %H %m

Examples

List alarms:

```
nxalarm -u admin -P adminpasswd 127.0.0.1 list
```

Resolve alarm:

```
nxalarm -u admin -P adminpasswd 127.0.0.1 resolve 226875
```

46.7.6 nxap

nxap - command line tool used to manage agent policies

Usage:

- nxap [<options>] -l <host>
- nxap [<options>] -u <guid> <host>

Options:

Source	Description
-l	List policies.
-u <guid>	Uninstall policy.

Common options:

Source	Description
-D level	Set debug level (0..9 or off, default is off).
-e policy	Set encryption policy. Possible values are: 0 = Encryption disabled; 1 = Encrypt connection only if agent requires encryption; 2 = Encrypt connection if agent supports encryption; 3 = Force encrypted connection; Default value is 1.
-h	Display help and exit.
-K file	Specify server's key file (default is /var/lib/netxms/.server_key).
-O port	Proxy agent's port number. Default is 4700.
-p port	Agent's port number. Default is 4700.
-s secret	Shared secret for agent authentication.
-S secret	Shared secret for proxy agent authentication.
-v	Display version and exit.
-w seconds	Set command timeout (default is 5 seconds).
-W seconds	Set connection timeout (default is 30 seconds).
-X addr	Use proxy agent at given address.

Example

List agent policies:

```
nxap 127.0.0.1 -l
```

46.7.7 nxappget

nxappget - command line tool for reading metrics from application agents

Usage: nxappget agent_name metric_name

Options:

Source	Description
-V, --version	Display version information.
-h, --help	Display this help message.
-v, --verbose	Enable verbose messages. Add twice for debug
-q, --quiet	Suppress all messages.

46.7.8 nxapush

This tool has same usage as nxpush, but it sends data through local agent.

When new version of NetXMS is released - version of server protocol is changed. Change of version affects on server communication with other tools like nxpush. So after each server update nxpush tool also should be updated. In case of usage nxapush - only agent should be updated as this tool uses agent protocol to send data.

Usage:

- `nxapush [OPTIONS] [@batch_file] [values]`
- `nxapush [OPTIONS] -`

Options:

Source	Description
-h, --help	Display this help message.
-l, --local-cache	Push to agent's local cache.
-o, --object <id>	Push data on behalf of object with given id.
-q, --quiet	Suppress all messages.
-s, --statsite	Use statsite sink format.
-t, --timestamp-unix <time>	Specify timestamp for data as UNIX timestamp.
-T, --timestamp-text <time>	Specify timestamp for data as YYYYMMDDhhmmss.
-v, --verbose	Enable verbose messages. Add twice for debug
-V, --version	Display version information.

Note

- Values should be given in format `dci=value` or (if statsite sink format is selected): `dci|value|timestamp` where dci can be specified by it's name
- Name of batch file cannot contain character = (equality sign)
- Use - character in place of values to read from standard input

Examples

Push two values:

```
nxapush PushParam1=1 PushParam2=4
```

Push values from file:

```
nxapush @file
```

46.7.9 nxencpasswd

This tool can be used to obfuscate passwords stored in server and agent configuration files as well as various places in the system, e.g. ssh passwords, notification channel passwords, etc.

Usage:

- `nxencpasswd [<options>] <login> [<password>]`
- `nxencpasswd [<options>] -a [<password>]`

Options:

Source	Description
-a	Encrypt agent's secret.
-h	Display help and exit.
-v	Display version and exit.

Note

If password is not provided it will be requested from terminal.

46.7.10 nxevent

Nxevent is installed with NetXMS client distribution. Sends events to server using client protocol. On Linux is provided by netxms-client package.

Usage:

- `nxevent [<options>] <server> <event> [<param_1> [... <param_N>]]`
- `nxevent [<options>] -n <server> <event> [name=parameter [... name=parameter]]`

Options:

Source	Description
-c	Codepage (default is ISO8859-1).
-C <count>	Repeat event sending given number of times.
-d	Turn on debug mode.
-e	Encrypt session (for compatibility only, session is always encrypted).
-h	Display help and exit.
-i <interval>	Repeat event sending with given interval in milliseconds.
-n	Parameters are provided in named format (name=value).
-o <id>	Specify source object ID.
-P <password>	Specify user's password. Default is empty password.
-S	Skip protocol version check (use with care).
-T <tag>	User tag to be associated with the message. Default is empty.
-u <user>	Login to server as <user>. Default is "guest".
-v	Display version and exit.
-w <seconds>	Specify command timeout (default is 3 seconds).

Example

Send event to server:

```
nxevent -u admin -P adminpassword 127.0.0.1 MY_APP_EVENT
nxevent -u admin -P adminpassword 127.0.0.1 MY_APP_EVENT state=UP desc=
↪ "Application started"
```

46.7.11 nxget

This tool is intended to get values of *Metric* from NetXMS agent.

Usage: nxget [<options>] <host> [<metric> [<metric> ...]]

Where *host* is the name or IP address of the host running NetXMS agent; and *metric* is a metric, list or table name, depending on given options. When metric is requested without explicitly specifying metric type (table or list), nxget attempts to get values trying types in the following order: single-value metric, table, list.

Valid options for nxget

Option	Description
-b	Batch mode - get all parameters listed on command line.
-C	Get agent's configuration file
-d delimiter	Print table content as delimited text.
-D level	Set debug level (default is 0).
-e policy	Set encryption policy. Possible values are: 0 = Encryption disabled; 1 = Encrypt connection only if agent requires encryption; 2 = Encrypt connection if agent supports encryption; 3 = Force encrypted connection; Default value is 1.
-E file	Take screenshot. First parameter is file name, second (optional) is session name.
-f	Do not try lists and tables if requested metric does not exist.
-F	Get information about given file set. Each parameter is separate file name.
-h	Display help and exit.
-i seconds	Get specified parameter(s) continuously with given interval.
-I	Get list of supported parameters.
-K file	Specify server's key file (default is /opt/netxms/var/lib/netxms/.server_key).
-l	Requested parameter is a list.
-n	Show parameter's name in result.
-N addr	Check state of network service at given address.
-o proto	Protocol number to be used for service check.
-O port	Proxy agent's port number. Default is 4700.
-p port	Agent's port number. Default is 4700.
-P port	Network service port (to be used with -N option).
-r string	Service check request string.
-R string	Service check expected response string.
-s secret	Shared secret for authentication.
-S secret	Shared secret for proxy agent authentication.

continues on next page

Table 4 – continued from previous page

Option	Description
-t type	Set type of service to be checked. Possible types are - custom, ssh, pop3, smtp, ftp, http, https, telnet.
-T	Requested parameter is a table.
U	Get list of active user sessions.
-v	Display version and exit.
-w seconds	Set command timeout (default is 5 seconds).
-W seconds	Set connection timeout (default is 30 seconds).
-X addr	Use proxy agent at given address.
-Y	Read remote system time.

Examples

Get value of *Agent.Version* metric from agent at host 10.0.0.2:

```
nxget 10.0.0.2 Agent.Version
```

Get list of supported parameters from agent at host 10.0.0.2:

```
nxget 10.0.0.2 -I
```

Get list of supported lists from agent at host 10.0.0.2:

```
nxget 10.0.0.2 Agent.SupportedLists -l
```

Get list of supported tables from agent at host 10.0.0.2:

```
nxget 10.0.0.2 Agent.SupportedTables -l
```

Get value of *Agent.Uptime* and *System.Uptime* metrics in one request, with output in metric = value form:

```
nxget -bn 10.0.0.2 Agent.Uptime System.Uptime
```

Get agent configuration file from agent at host 10.0.0.2:

```
nxget -C 10.0.0.2
```

Get value of *System.PlatformName* metric from agent at host 10.0.0.2, connecting via proxy agent at 172.16.1.1:

```
nxget -X 172.16.1.1 10.0.0.2 System.PlatformName
```

Get value of *Agent.AcceptedConnections* enum from agent at host 10.0.0.10, forcing use of encrypted connection:

```
nxget -e 3 -l 10.0.0.10 Agent.AcceptedConnections
```

Check POP3 service at host 10.0.0.4 via agent at host 172.16.1.1:

```
nxget -S 10.0.0.4 -t 2 -r user:pass 172.16.1.1
```


Useful lists for debugging purpose

List name	Description
Agent.ActionList	List of defined actions
Agent.SubAgentList	List of loaded subagents
Agent.SupportedLists	List of supported lists
Agent.SupportedParameters	List of supported parameters
Agent.SupportedPushParameters	List of supported push parameters
Agent.SupportedTables	List of supported table parameters
Agent.ThreadPools	List of thread pools

46.7.12 nxmibc

nxmibc - cli tool for mib file management. Adding MIB files should be performed using management client, see: [Import MIB](#). This tool should not be normally used.

Usage: nxmibc [options] source1 ... sourceN

Options:

Option	Description
-a	Compile all input files (continue after file parsing errors)
-d <dir>	Include all MIB files from given directory to compilation
-e <ext>	Specify file extensions (default extension: "mib")
-m	Produce machine-readable output
-o <file>	Set output file name (default is netxms.mib)
-P	Pause before exit
-r	Scan sub-directories
-s	Strip descriptions from MIB objects
-u	Do not compress output file
-z	Compress output file

Note

compression is ON by default, so option -z effectively does nothing and left only for backward compatibility.

Example

Compile and compress mib file:

```
nxmibc -d /usr/share/netxms/mibs -o /var/lib/netxms/netxms.mib -z
```

46.7.13 nxpush

nxpush is a command line tool used to push DCI values to NetXMS server.

There are different options how this tool can be used:

- with help of this tool data collected with different monitoring system can be pushed also to netxms
- can be used on nodes where agent can not be installed(not the case for nxapush)
- can be used on nodes behind NAT with no port forwarding option

Usage: nxpush [OPTIONS] [server] [@batch_file] [values]

Options:

Option	Description
-b, --batchsize <size>	Batch size (default is to send all data in one batch).
-c, --codepage <page>	Codepage (default is ISO8859-1).
-e, --encrypt	Encrypt session (for compatibility only, session is always encrypted).
-h, --help	Display this help message.
-H, --host <host>	Server address.
-P, --password <password>	Specify user's password. Default is empty.
-q, --quiet	Suppress all messages.
-S, --skip-version-check	Skip protocol version check (use with care).
-t, --timestamp-unix <time>	Specify timestamp for data as UNIX timestamp.
-T, --timestamp-text <time>	Specify timestamp for data as YYYYMMDDhhmmss.
-u, --user <user>	Login to server as user. Default is "guest".
-v, --verbose	Enable verbose messages. Add twice for debug.
-V, --version	Display version information.

Note

- Values should be given in the following format: dci=value where DCI can be specified by ID or name and node by ID, object name, DNS name, or IP address. If you wish to specify node by DNS name or IP address, you should prefix it with @ character
- First parameter will be used as "host" if -H/--host is unset
- Name of batch file cannot contain character = (equality sign)

Examples:

Push two values to server 10.0.0.1 as user "sender" with password "passwd". Values will be pushed to node with ID 104, first to DCI with ID 4567, second to DCI with metric "PushParam":

```
nxpush -H 10.0.0.1 -u sender -P passwd 104:4567=1 104:PushParam=4
```

Push values from file to server 10.0.0.1 as user "guest" without password:

```
nxpush 10.0.0.1 @file
```

Required server configurations are described there: [Push metrics](#)

46.7.14 nxscript

nxscript - command line utility for script management.

Usage: nxscript [options] script [arg1 [... argN]]

Options:

Option	Description
-5	Convert given script to NXSL version 5
-b	Input is a binary file
-c	Compile only
-C <count>	Run script multiple times
-d	Dump compiled script code
-e <name>	Entry point
-E	Show expression variables on exit
-m	Show memory usage information
-M	Show program metadata
-o <file>	Write compiled script
-r	Print script return value
-R	Show list of required modules
-t	Enable instruction trace

Example

Convert script to NXSL version 5:

```
nxscript -5 file.nxsl
```

46.7.15 nxsnmpget

This tool can be used to get *SNMP Metric* from node.

Usage: nxsnmpget [<options>] <host> <variables>

Options:

Option	Description
-a <method>	Authentication method for SNMP v3 USM. Valid methods are MD5, SHA1, SHA224, SHA256, SHA384, SHA512
-A <passwd>	User's authentication password for SNMP v3 USM
-c <string>	Community string. Default is "public"
-C <codepage>	Codepage for remote system
-e <method>	Encryption method for SNMP v3 USM. Valid methods are DES and AES
-E <passwd>	User's encryption password for SNMP v3 USM
-h	Display help and exit
-i <seconds>	Repeat request with given interval in seconds
-p <port>	Agent's port number. Default is 161
-u <user>	User name for SNMP v3 USM
-v <version>	SNMP version to use (valid values is 1, 2c, and 3)
-w <seconds>	Request timeout (default is 3 seconds)
-x	Show raw value in hex

Example

Get system description for given IP:

```
nxsnmpget -c public -v 2c 127.0.0.1 .1.3.6.1.2.1.1.1.0
```

46.7.16 nxsnmpset

nxsnmpset - command line tool used to set parameters on SNMP agent

Usage: nxsnmpset [<options>] <host> <variable>[@<type>] <value>

Options:

Option	Description
-a <method>	Authentication method for SNMP v3 USM. Valid methods are MD5, SHA1, SHA224, SHA256, SHA384, SHA512
-A <passwd>	User's authentication password for SNMP v3 USM
-B	Provided value is a base64 encoded raw value
-c <string>	Community string. Default is "public"
-e <method>	Encryption method for SNMP v3 USM. Valid methods are DES and AES
-E <passwd>	User's encryption password for SNMP v3 USM
-h	Display help and exit
-H	Provided value is a raw value encoded as hexadecimal string
-p <port>	Agent's port number. Default is 161
-t <type>	Specify variable's data type. Default is octet string.
-u <user>	User name for SNMP v3 USM
-v <version>	SNMP version to use (valid values is 1, 2c, and 3)
-w <seconds>	Request timeout (default is 3 seconds)

Note

You can specify data type either as number or in symbolic form. Valid symbolic representations are following:

- INTEGER
- STRING
- OID
- IPADDR
- COUNTER32
- GAUGE32
- TIMETICKS
- COUNTER64
- UINT32

46.7.17 nxsnmpwalk

nxsnmpwalk - command line tool used to retrieve parameters from SNMP agent

Usage: nxsnmpwalk [<options>] <host> <start_oid>

Options:

Option	Description
-a <method>	Authentication method for SNMP v3 USM. Valid methods are MD5, SHA1, SHA224, SHA256, SHA384, SHA512
-A <passwd>	User's authentication password for SNMP v3 USM
-c <string>	Community string. Default is "public"
-C <codepage>	Codepage for remote system
-e <method>	Encryption method for SNMP v3 USM. Valid methods are DES and AES
-E <passwd>	User's encryption password for SNMP v3 USM
-h	Display help and exit
-n <name>	SNMP v3 context name
-p <port>	Agent's port number. Default is 161
-u <user>	User name for SNMP v3 USM
-v <version>	SNMP version to use (valid values is 1, 2c, and 3)
-w <seconds>	Request timeout (default is 3 seconds)

Example

Get system description for given IP:

```
nxsnmpwalk -c public -v 2c 127.0.0.1 .1.3.6.1.2.1.1.1
```

46.7.18 nxupload

nxupload - command line tool used to upload files to NetXMS agent

Usage: nxupload [<options>] <host> <file>

Tool options:

Option	Description
-C <options>	Set package deployment options or command line (depending on package type)
-d <file>	Fully qualified destination file name
-i	Start installation of uploaded package.
-q	Quiet mode.
-t <type>	Set package type (default is "executable").
-u	Start agent upgrade from uploaded package.
-z	Compress data stream with LZ4.
-Z	Compress data stream with DEFLATE.

Common options:

Option	Description
-D level	Set debug level (0..9 or off, default is off).
-e policy	Set encryption policy. Possible values are: 0 = Encryption disabled; 1 = Encrypt connection only if agent requires encryption; 2 = Encrypt connection if agent supports encryption; 3 = Force encrypted connection; Default value is 1.
-h	Display help and exit.
-K file	Specify server's key file (default is /var/lib/netxms/.server_key).
-O port	Proxy agent's port number. Default is 4700.
-p port	Agent's port number. Default is 4700.
-s secret	Shared secret for agent authentication.
-S secret	Shared secret for proxy agent authentication.
-v	Display version and exit.
-w seconds	Set command timeout (default is 5 seconds).
-W seconds	Set connection timeout (default is 30 seconds).

Example

Upload file to agent's data directory:

```
nxupload localhost test_script.sh
```

46.7.19 nxwsget

nxwsget - command line tool used to query web services via NetXMS agent. Such agent needs to have *EnableWebServiceProxy=yes* in its configuration.

Usage: nxwsget [<options>] <host> <URL> <path> [<path> ...]

Options:

Option	Description
-a auth	HTTP authentication type. Valid methods are “none”, “basic”, “digest”, “ntlm”, “bearer”, “any”, or “anysafe”. Default is “none”.
-c	Do not verify service certificate.
-C	Do not verify certificate’s name against host.
-d data	Request data.
-D level	Set debug level (default is 0).
-e policy	Set encryption policy. Possible values are: 0 = Encryption disabled; 1 = Encrypt connection only if agent requires encryption; 2 = Encrypt connection if agent supports encryption; 3 = Force encrypted connection; Default value is 1.
-F	Follow location header that the server sends as part of a 3xx response.
-h	Display help and exit.
-H header	HTTP header (can be used multiple times).
-i seconds	Query service continuously with given interval.
-K file	Specify server’s key file (default is /var/lib/netxms/.server_key).
-l	Requested parameter is a list.
-L login	Web service login name.
-m method	HTTP request method. Valid methods are GET, POST, PUT, PATCH, DELETE.
-O port	Proxy agent’s port number. Default is 4700.
-p port	Agent’s port number. Default is 4700.
-P passwod	Web service password.
-r seconds	Cache retention time.
-s secret	Shared secret for agent authentication.
-S secret	Shared secret for proxy agent authentication.
-t	Use text parsing.
-v	version
-w seconds	Set command timeout (default is 5 seconds).
-W seconds	Set connection timeout (default is 30 seconds).
-X addr	Use proxy agent at given address.

Example

```
nxwsget 127.0.0.1 "http://api.open-notify.org/astros.json" .number
```

46.8 List of supported metrics

In this chapter will be described Agent and OS Subagent provided metrics.

46.8.1 Single value metrics

Agent.AcceptedConnections

Data type: Unsigned Integer

Supported Platforms: Windows, Linux, Solaris, AIX, H\$1-\$3X, FreeBSD, NetBSD, OpenBSD

Cumulative counter of connections accepted by agent

Agent.AcceptErrors

Data type: Unsigned Integer

Supported Platforms: Windows, Linux, Solaris, AIX, HP-UX, FreeBSD, NetBSD, OpenBSD

Cumulative counter of agent's accept() system call errors

Agent.ActiveConnections

Data type: Unsigned Integer

Supported Platforms: Windows, Linux, Solaris, AIX, HP-UX, FreeBSD, NetBSD, OpenBSD

Number of active connections to agent

Agent.AuthenticationFailures

Data type: Unsigned Integer

Supported Platforms: Windows, Linux, Solaris, AIX, HP-UX, FreeBSD, NetBSD, OpenBSD

Cumulative counter of failed AUTH commands (due to invalid secret)

Agent.ConfigurationServer

Data type: String

Supported Platforms: Windows, Linux, Solaris, AIX, HP-UX, FreeBSD, NetBSD, OpenBSD

Configuration server address set on agent startup.

Agent.FailedRequests

Data type: Unsigned Integer

Supported Platforms: Windows, Linux, Solaris, AIX, HP-UX, FreeBSD, NetBSD, OpenBSD

Cumulative counter of requests with errors in processing (others than unsupported metrics)

Agent.GeneratedTraps

Note

Deprecated

Data type: Unsigned Integer 64-bit

Supported Platforms: Windows, Linux, Solaris, AIX, HP-UX, FreeBSD, NetBSD, OpenBSD

Number of traps generated by agent

Agent.IsSubagentLoaded(*)

Data type: Integer

Parameters:

1. subagent name

Supported Platforms: Windows, Linux, Solaris, AIX, HP-UX, FreeBSD, NetBSD, OpenBSD

Check if given subagent is loaded. Return 1 if loaded and 0 if not.

Agent.LastTrapTime**Note**

Deprecated

Data type: Unsigned Integer 64-bit

Supported Platforms: Windows, Linux, Solaris, AIX, HP-UX, FreeBSD, NetBSD, OpenBSD

Timestamp of last generated trap

Agent.IsUserAgentInstalled

Data type: Unsigned Integer

Supported Platforms: Windows, Linux, Solaris, AIX, HP-UX, FreeBSD, NetBSD, OpenBSD

Check if user support application is installed

Agent.LocalDatabase.FailedQueries

Data type: Unsigned Integer

Supported Platforms: Windows, Linux, Solaris, AIX, HP-UX, FreeBSD, NetBSD, OpenBSD

Agent local database: failed queries

Agent.LocalDatabase.LongRunningQueries

Data type: Unsigned Integer

Supported Platforms: Windows, Linux, Solaris, AIX, HP-UX, FreeBSD, NetBSD, OpenBSD

Agent local database: long running queries

Agent.LocalDatabase.Status

Data type: Unsigned Integer

Supported Platforms: Windows, Linux, Solaris, AIX, HP-UX, FreeBSD, NetBSD, OpenBSD

Agent local database: status

Agent.LocalDatabase.TotalQueries

Data type: Unsigned Integer

Supported Platforms: Windows, Linux, Solaris, AIX, HP-UX, FreeBSD, NetBSD, OpenBSD

Agent local database: total queries executed

Agent.LogFile.Status

Data type: Unsigned Integer

Supported Platforms: Windows, Linux, Solaris, AIX, HP-UX, FreeBSD, NetBSD, OpenBSD

Agent log status

Agent.Notification.QueueSize

Data type: Unsigned Integer

Supported Platforms: Windows, Linux, Solaris, AIX, HP-UX, FreeBSD, NetBSD, OpenBSD

Agent notification queue size

Agent.ProcessedRequests

Data type: Unsigned Integer

Supported Platforms: Windows, Linux, Solaris, AIX, HP-UX, FreeBSD, NetBSD, OpenBSD

Cumulative counter of successfully processed requests

Agent.Registrar

Data type: String

Supported Platforms: Windows, Linux, Solaris, AIX, HP-UX, FreeBSD, NetBSD, OpenBSD

Registrar server address set on agent startup

Agent.RejectedConnections

Data type: Unsigned Integer

Supported Platforms: Windows, Linux, Solaris, AIX, HP-UX, FreeBSD, NetBSD, OpenBSD

Cumulative counter of connections rejected due to authentication failure

Agent.SentTraps

Note
Deprecated

Data type: Unsigned Integer 64-bit

Supported Platforms: Windows, Linux, Solaris, AIX, HP-UX, FreeBSD, NetBSD, OpenBSD

Number of traps successfully sent to server

Agent.SourcePackageSupport

Data type: Integer

Supported Platforms: Windows, Linux, Solaris, AIX, HP-UX, FreeBSD, NetBSD, OpenBSD

Non-zero if system is capable of building agent from source

Agent.SupportedCiphers

Data type: String

Supported Platforms: Windows, Linux, Solaris, AIX, HP-UX, FreeBSD, NetBSD, OpenBSD

List of ciphers supported by agent

Agent.SyslogProxy.IsEnabled

Data type: Unsigned Integer

Supported Platforms: Windows, Linux, Solaris, AIX, HP-UX, FreeBSD, NetBSD, OpenBSD

Check if syslog proxy is enabled

Agent.SyslogProxy.ReceivedMessages

Data type: Unsigned Integer 64-bit

Supported Platforms: Windows, Linux, Solaris, AIX, HP-UX, FreeBSD, NetBSD, OpenBSD

Number of syslog messages received by agent

Agent.ThreadPool.ActiveRequests(*)

Data type: Unsigned Integer

Supported Platforms: Windows, Linux, Solaris, AIX, HP-UX, FreeBSD, NetBSD, OpenBSD

Parameters:

1. Thread pool name. Possible options: MAIN, AGENT, POLLERS, SCHEDULER

Count of active requests for specified agent thread pool.

Agent.ThreadPool.CurrSize(*)

Data type: Unsigned Integer

Supported Platforms: Windows, Linux, Solaris, AIX, HP-UX, FreeBSD, NetBSD, OpenBSD

Parameters:

1. Thread pool name. Possible options: MAIN, AGENT, POLLERS, SCHEDULER

Current size of specified agent thread pool.

Agent.ThreadPool.Load(*)

Data type: Unsigned Integer

Supported Platforms: Windows, Linux, Solaris, AIX, HP-UX, FreeBSD, NetBSD, OpenBSD

Parameters:

1. Thread pool name. Possible options: MAIN, AGENT, POLLERS, SCHEDULER

Current load of specified agent thread pool. It's active requests divided by current thread count in percent.

Agent.ThreadPool.LoadAverage(*)

Data type: Unsigned Integer

Supported Platforms: Windows, Linux, Solaris, AIX, HP-UX, FreeBSD, NetBSD, OpenBSD

Parameters:

1. Thread pool name. Possible options: MAIN, AGENT, POLLERS, SCHEDULER
2. *optional* Normalization flag. If it is set to 1, then the value is divided to max thread count.

Active request moving average load of specified agent thread pool for last minute.

Agent.ThreadPool.LoadAverage5(*)

Data type: Unsigned Integer

Supported Platforms: Windows, Linux, Solaris, AIX, HP-UX, FreeBSD, NetBSD, OpenBSD

Parameters:

1. Thread pool name. Possible options: MAIN, AGENT, POLLERS, SCHEDULER
2. *optional* Normalization flag. If it is set to 1, then the value is divided to max thread count.

Active request moving average of specified agent thread pool for last 5 minutes.

Agent.ThreadPool.LoadAverage15(*)

Data type: Unsigned Integer

Supported Platforms: Windows, Linux, Solaris, AIX, HP-UX, FreeBSD, NetBSD, OpenBSD

Parameters:

1. Thread pool name. Possible options: MAIN, AGENT, POLLERS, SCHEDULER
2. *optional* Normalization flag. If it is set to 1, then the value is divided to max thread count.

Active request moving average load of specified agent thread pool for last 15 minutes.

Agent.ThreadPool.MaxSize(*)

Data type: Unsigned Integer

Supported Platforms: Windows, Linux, Solaris, AIX, HP-UX, FreeBSD, NetBSD, OpenBSD

Parameters:

1. Thread pool name. Possible options: MAIN, AGENT, POLLERS, SCHEDULER

Maximum size of specified agent thread pool.

Agent.ThreadPool.MinSize(*)

Data type: Unsigned Integer

Supported Platforms: Windows, Linux, Solaris, AIX, HP-UX, FreeBSD, NetBSD, OpenBSD

Parameters:

1. Thread pool name. Possible options: MAIN, AGENT, POLLERS, SCHEDULER

Maximum size of specified agent thread pool.

Agent.ThreadPool.Usage(*)

Data type: Unsigned Integer

Supported Platforms: Windows, Linux, Solaris, AIX, HP-UX, FreeBSD, NetBSD, OpenBSD

Parameters:

1. Thread pool name. Possible options: MAIN, AGENT, POLLERS, SCHEDULER

Current usage of specified agent thread pool. The value is equal to current thread count divided by max thread count in percent.

Agent.TimedOutRequests

Data type: Unsigned Integer

Supported Platforms: Windows, Linux, Solaris, AIX, HP-UX, FreeBSD, NetBSD, OpenBSD

Cumulative counter of timed out requests

Agent.UnsupportedRequests

Data type: Unsigned Integer

Supported Platforms: Windows, Linux, Solaris, AIX, HP-UX, FreeBSD, NetBSD, OpenBSD

Cumulative counter of requests for unsupported metrics

Agent.Uptime

Data type: Unsigned Integer

Supported Platforms: Windows, Linux, Solaris, AIX, HP-UX, FreeBSD, NetBSD, OpenBSD

Number of seconds since agent start

Agent.Version

Data type: String

Supported Platforms: Windows, Linux, Solaris, AIX, HP-UX, FreeBSD, NetBSD, OpenBSD

Agent's version

Disk.Avail(*)

TODO

Disk.AvailPerc(*)

TODO

Disk.Free(*)

TODO

Disk.FreePerc(*)

TODO

Disk.Total(*)

TODO

Disk.Used(*)

TODO

Disk.UsedPerc(*)

TODO

File.Content(*)

Data type: String

Supported Platforms: Windows, Linux, Solaris, AIX, HP-UX, FreeBSD, NetBSD, OpenBSD

Parameters:

1. Path - it specifies path to file

Returns first line of file content (but no more then 255 characters). Only servers which are in MasterServers in agent configuration file have access to this metric.

The following macros are supported in path and pattern parameters:

- Environment variables as \${ENV_VAR_NAME}
- `strftime(3C)` macros
- Text inside `braces` will be executed as a command and first line of output will be taken (only for servers which are in MasterServers in agent configuration file)

File.Count(*)

Data type: Unsigned Integer

Supported Platforms: Windows, Linux, Solaris, AIX, HP-UX, FreeBSD, NetBSD, OpenBSD

Parameters:

1. Path is the only mandatory argument. It specifies base directory for search.
2. Pattern - If pattern is given, only files whose names matched against it will be counted. Since version 3.8.314 it's possible to invert the mask by prefixing this parameter with "!". In this case files NOT matching the mask will be counted.
3. Recursive - determines if agent should count files in subdirectories. To enable recursion, use values `1` or `true`.
4. Size filter. If parameter < 0 , only files with size less than `abs(value)` will match. If parameter > 0 , only files with size greater than value will match.
5. Age filter. If parameter < 0 , only files created after now - `abs(value)` will match. If parameter > 0 , only files created before now - value will match.

Number of files in directory

The following macros are supported in path and pattern parameters:

- Environment variables as \${ENV_VAR_NAME}
- `strftime(3C)` macros
- Text inside `braces` will be executed as a command and first line of output will be taken (only for servers which are in MasterServers in agent configuration file)

File.FolderCount(*)

Data type: Unsigned Integer

Supported Platforms: Windows, Linux, Solaris, AIX, HP-UX, FreeBSD, NetBSD, OpenBSD

Parameters:

1. Path is the only mandatory argument. It specifies base directory for search.
2. Pattern - If pattern is given, only folders whose names matched against it will be counted.
3. Recursive - determines if agent should count folders in subdirectories. To enable recursion, use values 1 or true.
4. Size filter. If parameter < 0, only folders with size less than abs(value) will match. If parameter > 0, only folders with size greater than value will match.
5. Age filter. If parameter < 0, only folders created after now - abs(value) will match. If parameter > 0, only folders created before now - value will match.

Number of folders in directory

File.Hash.CRC32(*)

Data type: Unsigned Integer

Supported Platforms: Windows, Linux, Solaris, AIX, HP-UX, FreeBSD, NetBSD, OpenBSD

Parameters:

1. Path - it specifies path to file

CRC32 hash of given file

The following macros are supported in path parameter:

- Environment variables as \${ENV_VAR_NAME}
- `strftime(3C)` macros
- Text inside `braces` will be executed as a command and first line of output will be taken (only for servers which are in MasterServers in agent configuration file)

File.Hash.MD5(*)

Data type: String

Supported Platforms: Windows, Linux, Solaris, AIX, HP-UX, FreeBSD, NetBSD, OpenBSD

Parameters:

1. Path - it specifies path to file

MD5 hash of given file

The following macros are supported in path parameter:

- Environment variables as \${ENV_VAR_NAME}
- `strftime(3C)` macros
- Text inside `braces` will be executed as a command and first line of output will be taken (only for servers which are in MasterServers in agent configuration file)

File.Hash.SHA1(*)

Data type: String

Supported Platforms: Windows, Linux, Solaris, AIX, HP-UX, FreeBSD, NetBSD, OpenBSD

Parameters:

1. Path - it specifies path to file

SHA1 hash of given file

The following macros are supported in path parameter:

- Environment variables as `${ENV_VAR_NAME}`
- `strftime(3C)` macros
- Text inside ``` braces will be executed as a command and first line of output will be taken (only for servers which are in MasterServers in agent configuration file)

File.Size(*)

Data type: Unsigned Integer 64-bit

Supported Platforms: Windows, Linux, Solaris, AIX, HP-UX, FreeBSD, NetBSD, OpenBSD

Parameters:

1. Path is the only mandatory argument. It specifies either single file or base directory for calculation.
2. If pattern is given, only files whose names matched against it will be counted.
3. Recursive determines if agent should count files in subdirectories. To enable recursion, use values `1` or `true`.
4. Size filter. If parameter `< 0`, only files with size less than `abs(value)` will match. If parameter `> 0`, only files with size greater than value will match.
5. Age filter. If parameter `< 0`, only files created after now - `abs(value)` will match. If parameter `> 0`, only files created before now - value will match.

Size in bytes of single file or all files in given directory.

The following macros are supported in path and pattern parameters:

- Environment variables as `${ENV_VAR_NAME}`
- `strftime(3C)` macros
- Text inside ``` braces will be executed as a command and first line of output will be taken (only for servers which are in MasterServers in agent configuration file)

File.Time.Access(*)

Data type: Unsigned Integer 64-bit

Supported Platforms: Windows, Linux, Solaris, AIX, HP-UX, FreeBSD, NetBSD, OpenBSD

Parameters:

1. Path - it specifies path to file

File's last access time in seconds since epoch (1 Jan 1970 00:00:00 UTC)

The following macros are supported in path parameter:

- Environment variables as `${ENV_VAR_NAME}`
- `strftime(3C)` macros
- Text inside ``` braces will be executed as a command and first line of output will be taken (only for servers which are in MasterServers in agent configuration file)

File.Time.Change(*)

Data type: Unsigned Integer 64-bit

Supported Platforms: Windows, Linux, Solaris, AIX, HP-UX, FreeBSD, NetBSD, OpenBSD

Parameters:

1. Path - it specifies path to file

File's last status change time in seconds since epoch (1 Jan 1970 00:00:00 UTC)

The following macros are supported in path parameter:

- Environment variables as \${ENV_VAR_NAME}
- `strftime(3C)` macros
- Text inside ` braces will be executed as a command and first line of output will be taken (only for servers which are in MasterServers in agent configuration file)

File.Time.Modify(*)

Data type: Unsigned Integer 64-bit

Supported Platforms: Windows, Linux, Solaris, AIX, HP-UX, FreeBSD, NetBSD, OpenBSD

Parameters:

1. Path - it specifies path to file

File's last modification time in seconds since epoch (1 Jan 1970 00:00:00 UTC)

The following macros are supported in path parameter:

- Environment variables as \${ENV_VAR_NAME}
- `strftime(3C)` macros
- Text inside ` braces will be executed as a command and first line of output will be taken (only for servers which are in MasterServers in agent configuration file)

File.Type(*)

Data type: Unsigned Integer

Supported Platforms: Windows, Linux, Solaris, AIX, HP-UX, FreeBSD, NetBSD, OpenBSD

Parameters:

1. Path - it specifies path to file

Type of a file or directory. Returns one of the following values:

- 0 - file does not exist
- 1 - file is a directory
- 2 - file is a device
- 3 - file is a regular file
- 4 - file is of other type

The following macros are supported in path parameter:

- Environment variables as \${ENV_VAR_NAME}
- `strftime(3C)` macros

- Text inside ` braces will be executed as a command and first line of output will be taken (only for servers which are in MasterServers in agent configuration file)

FileSystem.Avail(*)

Data type: Unsigned Integer 64-bit

Supported Platforms: Linux, Solaris, AIX, HP-UX, FreeBSD, NetBSD, OpenBSD

Parameters:

1. Mountpoint, device name (linux only) or disk name (for Windows)

Available space on file system in bytes

FileSystem.AvailNodes(*)

TODO

FileSystem.AvailNodesPerc(*)

TODO

FileSystem.AvailPerc(*)

Data type: Float

Supported Platforms: Linux, Solaris, AIX, HP-UX, FreeBSD, NetBSD, OpenBSD

Parameters:

1. Mountpoint, device name (linux only) or disk name (for Windows)

Percentage of available space on file system

FileSystem.Free(*)

Data type: Unsigned Integer 64-bit

Supported Platforms: Windows, Linux, Solaris, AIX, HP-UX, FreeBSD, NetBSD, OpenBSD

Parameters:

1. Mountpoint, device name (linux only) or disk name (for Windows)

Free space on file system in bytes

FileSystem.FreeNodes(*)

TODO

FileSystem.FreeNodesPerc(*)

TODO

FileSystem.FreePerc(*)

Data type: Float

Supported Platforms: Windows, Linux, Solaris, AIX, HP-UX, FreeBSD, NetBSD, OpenBSD

Parameters:

1. Mountpoint, device name (linux only) or disk name (for Windows)

Percentage of free space on file system

FileSystem.Total(*)

Data type: Unsigned Integer 64-bit

Supported Platforms: Windows, Linux, Solaris, AIX, HP-UX, FreeBSD, NetBSD, OpenBSD

Parameters:

1. Mountpoint, device name (linux only) or disk name (for Windows)

Total number of bytes on file system

FileSystem.TotalNodes(*)

TODO

FileSystem.Type(*)

Data type: String

Supported Platforms: Windows, Linux, Solaris, AIX, HP-UX, FreeBSD, NetBSD, OpenBSD

Parameters:

1. Mountpoint or disk name (for Windows)

Type of file system

FileSystem.Used(*)

Data type: Unsigned Integer 64-bit

Supported Platforms: Windows, Linux, Solaris, AIX, HP-UX, FreeBSD, NetBSD, OpenBSD

Parameters:

1. Mountpoint, device name (linux only) or disk name (for Windows)

Used space on file system in bytes

FileSystem.UsedNodes(*)

TODO

FileSystem.UsedNodesPerc(*)

TODO

FileSystem.UsedPerc(*)

Data type: Float

Supported Platforms: Windows, Linux, Solaris, AIX, HP-UX, FreeBSD, NetBSD, OpenBSD

Parameters:

1. Mountpoint, device name (linux only) or disk name (for Windows)

Percentage of used space on file system

DRBD.ConnState(*)

TODO

DRBD.DataState(*)

TODO

DRBD.DeviceState(*)

TODO

DRBD.PeerDataState(*)

TODO

DRBD.PeerDeviceState(*)

TODO

DRBD.Protocol(*)

TODO

DRBD.Version.API

TODO

DRBD.Version.Driver

TODO

DRBD.Version.Protocol

TODO

Hardware.Baseboard.Manufacturer

TODO

Hardware.Baseboard.Product

TODO

Hardware.Baseboard.SerialNumber

TODO

Hardware.Baseboard.Type

TODO

Hardware.Baseboard.Version

TODO

Hardware.Battery.Capacity(*)

TODO

Hardware.Battery.Chemistry(*)

TODO

Hardware.Battery.Location(*)

TODO

Hardware.Battery.ManufactureDate(*)

TODO

Hardware.Battery.Manufacturer(*)

TODO

Hardware.Battery.Name(*)

TODO

Hardware.Battery.SerialNumber(*)

TODO

Hardware.Battery.Voltage(*)

TODO

Hardware.MemoryDevice.Bank(*)

TODO

Hardware.MemoryDevice.ConfiguredSpeed(*)

TODO

Hardware.MemoryDevice.FormFactor(*)

TODO

Hardware.MemoryDevice.Location(*)

TODO

Hardware.MemoryDevice.Manufacturer(*)

TODO

Hardware.MemoryDevice.MaxSpeed(*)

TODO

Hardware.MemoryDevice.PartNumber(*)

TODO

Hardware.MemoryDevice.SerialNumber(*)

TODO

Hardware.MemoryDevice.Size(*)

TODO

Hardware.MemoryDevice.Type(*)

TODO

Hardware.Processor.Cores(*)

TODO

Hardware.Processor.CurrentSpeed(*)

TODO

Hardware.Processor.Family(*)

TODO

Hardware.Processor.Manufacturer(*)

TODO

Hardware.Processor.MaxSpeed(*)

TODO

Hardware.Processor.PartNumber(*)

TODO

Hardware.Processor.SerialNumber(*)

TODO

Hardware.Processor.Socket(*)

TODO

Hardware.Processor.Threads(*)

TODO

Hardware.Processor.Type(*)

TODO

Hardware.Processor.Version(*)

TODO

Hardware.System.MachineId

Data type: String

Supported Platforms: Windows, Linux, Solaris, AIX, HP-UX, FreeBSD

Unique machine identifier.

Hardware.System.Manufacturer

Data type: String

Supported Platforms: Windows, Linux, Solaris, AIX, HP-UX, FreeBSD

System manufacturer.

Hardware.System.Product

Data type: String

Supported Platforms: Windows, Linux, Solaris, AIX, HP-UX, FreeBSD

Product name.

Hardware.System.ProductCode

Data type: String

Supported Platforms: Windows, Linux, Solaris, AIX, HP-UX, FreeBSD

Product code.

Hardware.System.SerialNumber

Data type: String

Supported Platforms: Windows, Linux, Solaris, AIX, HP-UX, FreeBSD

System serial number.

Hardware.System.Version

Data type: String

Supported Platforms: Windows, Linux, Solaris, AIX, HP-UX, FreeBSD

System version.

Hardware.WakeUpEvent

TODO

Hypervisor.Type

TODO

Hypervisor.Version

TODO

Net.Interface.AdminStatus(*)

Data type: Integer

Supported Platforms: Windows, Linux, Solaris, AIX, HP-UX, FreeBSD, NetBSD, OpenBSD

Parameters:

1. Interface name or interface index. Index can be obtained from `Net.InterfaceList` list.

Network interface administrative status (1 = enabled, 2 = disabled, 3 = testing)

Net.Interface.BytesIn(*)

Data type: Counter32

Supported Platforms: Windows, Linux, Solaris, AIX, HP-UX, FreeBSD, NetBSD, OpenBSD

Parameters:

1. Interface name or interface index. Index can be obtained from `Net.InterfaceList` list.

Number of input bytes on interface

Net.Interface.BytesIn64(*)

Data type: Counter64

Supported Platforms: Windows, Linux, FreeBSD

Parameters:

1. Interface name or interface index. Index can be obtained from `Net.InterfaceList` list.

Number of input bytes on interface

Net.Interface.BytesOut(*)

Data type: Counter32

Supported Platforms: Windows, Linux, Solaris, AIX, HP-UX, FreeBSD, NetBSD, OpenBSD

Parameters:

1. Interface name or interface index. Index can be obtained from `Net.InterfaceList` list.

Number of output bytes on interface

Net.Interface.BytesOut64(*)

Data type: Counter64

Supported Platforms: Windows, Linux, FreeBSD

Parameters:

1. Interface name or interface index. Index can be obtained from `Net.InterfaceList` list.

Number of output bytes on interface

Net.Interface.Description(*)

Data type: String

Supported Platforms: Windows, Linux, Solaris, AIX, HP-UX

Parameters:

1. Interface name or interface index. Index can be obtained from `Net.InterfaceList` list.

Description of interface

Net.Interface.InErrors(*)

Data type: Counter32

Supported Platforms: Windows, Linux, Solaris, AIX, HP-UX, FreeBSD, NetBSD, OpenBSD

Parameters:

1. Interface name or interface index. Index can be obtained from `Net.InterfaceList` list.

Number of input errors on interface

Net.Interface.InErrors64(*)

Data type: Counter64

Supported Platforms: Windows, Linux, FreeBSD

Parameters:

1. Interface name or interface index. Index can be obtained from `Net.InterfaceList` list.

Number of input errors on interface

Net.Interface.Link(*)

Data type: Integer

Supported Platforms: Windows, Linux, Solaris, AIX, HP-UX, FreeBSD, NetBSD, OpenBSD

Parameters:

1. Interface name or interface index. Index can be obtained from `Net.InterfaceList` list.

Link status of interface

Net.Interface.MTU(*)

Data type: Integer

Supported Platforms: Windows, AIX, HP-UX

Parameters:

1. Interface name or interface index. Index can be obtained from `Net.InterfaceList` list.

Net.Interface.OperStatus(*)

Data type: Integer

Supported Platforms: Windows, Linux, Solaris, HP-UX, FreeBSD, NetBSD, OpenBSD

Parameters:

1. Interface name or interface index. Index can be obtained from `Net.InterfaceList` list.
Network interface operational status (0 = down, 1 = up)

Net.Interface.OutErrors(*)

Data type: Counter32

Supported Platforms: Windows, Linux, Solaris, AIX, HP-UX, FreeBSD, NetBSD, OpenBSD

Parameters:

1. Interface name or interface index. Index can be obtained from `Net.InterfaceList` list.
Number of output errors on interface

Net.Interface.OutErrors64(*)

Data type: Counter64

Supported Platforms: Windows, Linux, FreeBSD

Parameters:

1. Interface name or interface index. Index can be obtained from `Net.InterfaceList` list.
Number of output errors on interface

Net.Interface.PacketsIn(*)

Data type: Counter32

Supported Platforms: Windows, Linux, Solaris, AIX, HP-UX, FreeBSD, NetBSD, OpenBSD

Parameters:

1. Interface name or interface index. Index can be obtained from `Net.InterfaceList` list.
Number of input packets on interface

Net.Interface.PacketsIn64(*)

Data type: Counter64

Supported Platforms: Windows, Linux, FreeBSD

Parameters:

1. Interface name or interface index. Index can be obtained from `Net.InterfaceList` list.
Number of input packets on interface

Net.Interface.PacketsOut(*)

Data type: Counter32

Supported Platforms: Windows, Linux, Solaris, AIX, HP-UX, FreeBSD, NetBSD, OpenBSD

Parameters:

1. Interface name or interface index. Index can be obtained from `Net.InterfaceList` list.
Number of output packets on interface

Net.Interface.PacketsOut64(*)

Data type: Counter64

Supported Platforms: Windows, Linux, FreeBSD

Parameters:

1. Interface name or interface index. Index can be obtained from `Net.InterfaceList` list.

Number of output packets on interface

Net.Interface.Speed(*)

Current interface working speed in bits per second.

Data type: UInt32

Supported Platforms: Windows, Linux, FreeBSD, Solaris, AIX, HP-UX

Parameters:

1. Interface name or interface index. Index can be obtained from `Net.InterfaceList` list.

Net.IP.Forwarding

Data type: Int32

Supported Platforms: Windows, Linux, HP-UX, FreeBSD, NetBSD, OpenBSD

IP forwarding status (1 = forwarding, 0 = not forwarding)

Net.IP6.Forwarding

Data type: Int32

Supported Platforms: Linux, HP-UX, FreeBSD, NetBSD, OpenBSD

IPv6 forwarding status (1 = forwarding, 0 = not forwarding)

Net.IP.NextHop(*)

Data type: String

Supported Platforms: Windows, Linux, Solaris, AIX, HP-UX, FreeBSD, NetBSD, OpenBSD

Next hop for given destination address according to host's routing table

Net.RemoteShareStatus(*)

Data type: Int32

Supported Platforms: Windows

Parameters:

1. Correct UNC path
2. Domain
3. Login
4. Password

Status of remote shared resource

Net.RemoteShareStatusText(*)

Data type: String

Supported Platforms: Windows

Parameters:

1. Correct UNC path
2. Domain
3. Login
4. Password

Status of remote shared resource as text

Net.Resolver.AddressByName(*)

Data type: String

Supported Platforms: Windows, Linux, Solaris, AIX, HP-UX, FreeBSD, NetBSD, OpenBSD

Parameters:

1. Name to resolve

Resolves host name to IP address

Net.Resolver.NameByAddress(*)

Data type: String

Supported Platforms: Windows, Linux, Solaris, AIX, HP-UX, FreeBSD, NetBSD, OpenBSD

Parameters:

1. Address to resolve

Resolves IP address to host name

PDH.CounterValue(*)

Data type: UInt32

Supported Platforms: Windows

Parameters:

1. Counter path. It should start with single backslash character and not include machine name.
2. Optional second argument specifies if counter requires two samples to calculate value (typical example of such counters is CPU utilization). Two samples will be taken if this argument is set to 1.

Current value of given PDH counter.

PDH.Version

Data type: UInt32

Supported Platforms: Windows

Version of PDH.DLL (as returned by PdhGetDllVersion() call).

PhysicalDisk.Capacity(*)

Data type: Unsigned Integer 64-bit

Supported Platforms: Linux, Windows

Parameters:

1. Physical disk name. Run `smartctl --scan` (on Linux) or `C:\NetXMS\bin\smartctl.exe --scan` (on Windows) to see list of available disk names.

Capacity in bytes of provided hard disk.

PhysicalDisk.DeviceType(*)

Data type: String

Supported Platforms: Linux, Windows

Parameters:

1. Physical disk name

Device type of provided hard disk.

PhysicalDisk.Firmware(*)

Data type: String

Supported Platforms: Linux, Windows

Parameters:

1. Physical disk name

Firmware version of provided hard disk.

PhysicalDisk.Model(*)

Data type: String

Supported Platforms: Linux, Windows

Parameters:

1. Physical disk name

Model of provided hard disk.

PhysicalDisk.PowerCycles(*)

Data type: Unsigned integer

Supported Platforms: Linux, Windows

Parameters:

1. Physical disk name

Number of power cycles of provided hard disk.

PhysicalDisk.PowerOnTime(*)

Data type: Unsigned integer

Supported Platforms: Linux, Windows

Parameters:

1. Physical disk name

Power on time of provided hard disk.

PhysicalDisk.SerialNumber(*)

Data type: String

Supported Platforms: Linux, Windows

Parameters:

1. Physical disk name

Serial number of provided hard disk.

PhysicalDisk.SmartAttr(*)

Data type: String

Supported Platforms: Linux, Windows

Parameters:

1. Physical disk name
2. SMART attribute name

PhysicalDisk.SmartStatus(*)

Data type: Integer

Supported Platforms: Linux, Windows

Parameters:

1. Physical disk name

Status of provided hard disk reported by SMART.

PhysicalDisk.Temperature(*)

Data type: Integer

Supported Platforms: Linux, Windows

Parameters:

1. Physical disk name

Temperature of provided hard disk.

Process.Count(*)

Data type: Int32

Supported Platforms: Windows, Linux, Solaris, AIX, HP-UX, FreeBSD, NetBSD, OpenBSD

Parameters:

1. Process name

Number of processes with given name

Process.CountEx(*)

Data type: Int32

Supported Platforms: Windows, Linux, Solaris, FreeBSD, NetBSD, AIX

Parameters:

1. Process name.
2. Optional parameter that accepts process's command line regular expression, that should match cmd argument. If not set it means "match any".
3. Optional parameter that accepts process's owner username regular expression. If not set it means "match any".
4. Optional parameter that accepts process's main window title regular expression. If not set it means "match any". Process's window title can be checked only on Windows platform.

Number of processes matching filter

Process.CPUTime(*)

Data type: Counter64

Supported Platforms: Windows, Linux, Solaris, AIX, HP-UX, FreeBSD, NetBSD

Parameters:

1. Process name
2. Function - is the function that is used to measure data in case if there are more than one process with given name. By default it is used sum function. This parameter can have this options:
 - min - minimal value among all processes named proc
 - max - maximal value among all processes named proc
 - avg - average value for all processes named proc
 - sum - sum of values for all processes named proc
3. Optional parameter that accepts process's command line regular expression, that should match cmd argument. If not set it means "match any".
4. Optional parameter that accepts process's owner username regular expression. If not set it means "match any".
5. Optional parameter that accepts process's main window title regular expression. If not set it means "match any". Process's window title can be checked only on Windows platform.

Total execution time for process

Process.GDIObjects(*)

Data type: Unsigned Integer 64-bit

Supported Platforms: Windows

Parameters:

1. Process name
2. Function - is the function that is used to measure data in case if there are more than one process with given name. By default it is used sum function. This parameter can have this options:
 - min - minimal value among all processes named proc
 - max - maximal value among all processes named proc
 - avg - average value for all processes named proc
 - sum - sum of values for all processes named proc
3. Optional parameter that accepts process's command line regular expression, that should match cmd argument. If not set it means "match any".
4. Optional parameter that accepts process's main window title regular expression. If not set it means "match any". Process's window title can be checked only on Windows platform.

GDI objects used by process

Process.Handles(*)

Data type: Int32

Supported Platforms: Windows, Linux, Solaris, AIX

Parameters:

1. Process name
2. Function - is the function that is used to measure data in case if there are more than one process with given name. By default it is used sum function. This parameter can have this options:
 - min - minimal value among all processes named proc
 - max - maximal value among all processes named proc
 - avg - average value for all processes named proc
 - sum - sum of values for all processes named proc
3. Optional parameter that accepts process's command line regular expression, that should match cmd argument. If not set it means "match any".
4. Optional parameter that accepts process's owner username regular expression. If not set it means "match any".
5. Optional parameter that accepts process's main window title regular expression. If not set it means "match any". Process's window title can be checked only on Windows platform.

Number of handles in process with given name

Process.IO.OtherB(*)

Data type: Unsigned Integer 64-bit

Supported Platforms: Windows

Parameters:

1. Process name
2. Function - is the function that is used to measure data in case if there are more than one process with given name. By default it is used sum function. This parameter can have this options:
 - min - minimal value among all processes named proc
 - max - maximal value among all processes named proc
 - avg - average value for all processes named proc
 - sum - sum of values for all processes named proc
3. Optional parameter that accepts process's command line regular expression, that should match cmd argument. If not set it means "match any".
4. Optional parameter that accepts process's main window title regular expression. If not set it means "match any". Process's window title can be checked only on Windows platform.

Process.IO.OtherOp(*)

Data type: Unsigned Integer 64-bit

Supported Platforms: Windows

Parameters:

1. Process name
2. Function - is the function that is used to measure data in case if there are more than one process with given name. By default it is used sum function. This parameter can have this options:
 - min - minimal value among all processes named proc
 - max - maximal value among all processes named proc
 - avg - average value for all processes named proc
 - sum - sum of values for all processes named proc
3. Optional parameter that accepts process's command line regular expression, that should match cmd argument. If not set it means "match any".
4. Optional parameter that accepts process's main window title regular expression. If not set it means "match any". Process's window title can be checked only on Windows platform.

Process.IO.ReadB(*)

Data type: Unsigned Integer 64-bit

Supported Platforms: Windows

Parameters:

1. Process name
2. Function - is the function that is used to measure data in case if there are more than one process with given name. By default it is used sum function. This parameter can have this options:

- min - minimal value among all processes named proc
 - max - maximal value among all processes named proc
 - avg - average value for all processes named proc
 - sum - sum of values for all processes named proc
3. Optional parameter that accepts process's command line regular expression, that should match cmd argument. If not set it means "match any".
 4. Optional parameter that accepts process's main window title regular expression. If not set it means "match any". Process's window title can be checked only on Windows platform.

Process.IO.ReadOp(*)

Data type: Unsigned Integer 64-bit

Supported Platforms: Windows, AIX, HP-UX

Parameters:

1. Process name
2. Function - is the function that is used to measure data in case if there are more than one process with given name. By default it is used sum function. This parameter can have this options:
 - min - minimal value among all processes named proc
 - max - maximal value among all processes named proc
 - avg - average value for all processes named proc
 - sum - sum of values for all processes named proc
3. Optional parameter that accepts process's command line regular expression, that should match cmd argument. If not set it means "match any".
4. Optional parameter that accepts process's main window title regular expression. If not set it means "match any". Process's window title can be checked only on Windows platform.

Process.IO.WriteB(*)

Data type: Unsigned Integer 64-bit

Supported Platforms: Windows

Parameters:

1. Process name
2. Function - is the function that is used to measure data in case if there are more than one process with given name. By default it is used sum function. This parameter can have this options:
 - min - minimal value among all processes named proc
 - max - maximal value among all processes named proc
 - avg - average value for all processes named proc
 - sum - sum of values for all processes named proc
3. Optional parameter that accepts process's command line regular expression, that should match cmd argument. If not set it means "match any".
4. Optional parameter that accepts process's main window title regular expression. If not set it means "match any". Process's window title can be checked only on Windows platform.

Process.IO.WriteOp(*)

Data type: Unsigned Integer 64-bit

Supported Platforms: Windows, AIX, HP-UX

Parameters:

1. Process name
2. Function - is the function that is used to measure data in case if there are more than one process with given name. By default it is used sum function. This parameter can have this options:
 - min - minimal value among all processes named proc
 - max - maximal value among all processes named proc
 - avg - average value for all processes named proc
 - sum - sum of values for all processes named proc
3. Optional parameter that accepts process's command line regular expression, that should match cmd argument. If not set it means "match any".
4. Optional parameter that accepts process's main window title regular expression. If not set it means "match any". Process's window title can be checked only on Windows platform.

Process.KernelTime(*)

Data type: Counter64

Supported Platforms: Windows, Linux, Solaris, AIX, HP-UX, NetBSD

Parameters:

1. Process name
2. Function - is the function that is used to measure data in case if there are more than one process with given name. By default it is used sum function. This parameter can have this options:
 - min - minimal value among all processes named proc
 - max - maximal value among all processes named proc
 - avg - average value for all processes named proc
 - sum - sum of values for all processes named proc
3. Optional parameter that accepts process's command line regular expression, that should match cmd argument. If not set it means "match any".
4. Optional parameter that accepts process's owner username regular expression. If not set it means "match any".
5. Optional parameter that accepts process's main window title regular expression. If not set it means "match any". Process's window title can be checked only on Windows platform.

Total execution time in kernel mode for process

Process.MemoryUsage(*)

Data type: Float

Supported Platforms: Windows, Linux, Solaris, AIX, FreeBSD

Parameters:

1. Process name
2. Function - is the function that is used to measure data in case if there are more than one process with given name. By default it is used sum function. This parameter can have this options:
 - min - minimal value among all processes named proc
 - max - maximal value among all processes named proc
 - avg - average value for all processes named proc
 - sum - sum of values for all processes named proc
3. Optional parameter that accepts process's command line regular expression, that should match cmd argument. If not set it means "match any".
4. Optional parameter that accepts process's owner username regular expression. If not set it means "match any".
5. Optional parameter that accepts process's main window title regular expression. If not set it means "match any". Process's window title can be checked only on Windows platform.

Percentage of total physical memory used by process

Process.PageFaults(*)

Data type: Counter64

Supported Platforms: Windows, Linux, Solaris, AIX, HP-UX, NetBSD

Parameters:

1. Process name
2. Function - is the function that is used to measure data in case if there are more than one process with given name. By default it is used sum function. This parameter can have this options:
 - min - minimal value among all processes named proc
 - max - maximal value among all processes named proc
 - avg - average value for all processes named proc
 - sum - sum of values for all processes named proc
3. Optional parameter that accepts process's command line regular expression, that should match cmd argument. If not set it means "match any".
4. Optional parameter that accepts process's owner username regular expression. If not set it means "match any".
5. Optional parameter that accepts process's main window title regular expression. If not set it means "match any". Process's window title can be checked only on Windows platform.

Page faults for process

Process.RSS(*)

Alias to Process.WkSet(*)

Process.Syscalls(*)

Data type: UInt64

Supported Platforms: Solaris

Parameters:

1. Process name
2. Function - is the function that is used to measure data in case if there are more than one process with given name. By default it is used sum function. This parameter can have this options:
 - min - minimal value among all processes named proc
 - max - maximal value among all processes named proc
 - avg - average value for all processes named proc
 - sum - sum of values for all processes named proc
3. Optional parameter that accepts process's command line regular expression, that should match cmd argument. If not set it means "match any".
4. Optional parameter that accepts process's main window title regular expression. If not set it means "match any". Process's window title can be checked only on Windows platform.

Number of system calls made by process

Process.Threads(*)

Data type: Int32

Supported Platforms: Linux, Solaris, AIX, HP-UX, FreeBSD, NetBSD

Parameters:

1. Process name
2. Function - is the function that is used to measure data in case if there are more than one process with given name. By default it is used sum function. This parameter can have this options:
 - min - minimal value among all processes named proc
 - max - maximal value among all processes named proc
 - avg - average value for all processes named proc
 - sum - sum of values for all processes named proc
3. Optional parameter that accepts process's command line regular expression, that should match cmd argument. If not set it means "match any".
4. Optional parameter that accepts process's owner username regular expression. If not set it means "match any".
5. Optional parameter that accepts process's main window title regular expression. If not set it means "match any". Process's window title can be checked only on Windows platform.

Number of threads in process

Process.UserObjects(*)

Data type: UInt64

Supported Platforms: Windows

Parameters:

1. Process name
2. Function - is the function that is used to measure data in case if there are more than one process with given name. By default it is used sum function. This parameter can have this options:
 - min - minimal value among all processes named proc
 - max - maximal value among all processes named proc
 - avg - average value for all processes named proc
 - sum - sum of values for all processes named proc
3. Optional parameter that accepts process's command line regular expression, that should match cmd argument. If not set it means "match any".
4. Optional parameter that accepts process's main window title regular expression. If not set it means "match any". Process's window title can be checked only on Windows platform.

USER objects used by process

Process.UserTime(*)

Data type: Counter64

Supported Platforms: Windows, Linux, Solaris, AIX, HP-UX, NetBSD

Parameters:

1. Process name
2. Function - is the function that is used to measure data in case if there are more than one process with given name. By default it is used sum function. This parameter can have this options:
 - min - minimal value among all processes named proc
 - max - maximal value among all processes named proc
 - avg - average value for all processes named proc
 - sum - sum of values for all processes named proc
3. Optional parameter that accepts process's command line regular expression, that should match cmd argument. If not set it means "match any".
4. Optional parameter that accepts process's owner username regular expression. If not set it means "match any".
5. Optional parameter that accepts process's main window title regular expression. If not set it means "match any". Process's window title can be checked only on Windows platform.

Total execution time in user mode for process

Process.VMRegions(*)

Data type: Int32

Supported Platforms: Linux

Parameters:

1. Process name
2. Function - is the function that is used to measure data in case if there are more than one process with given name. By default it is used sum function. This parameter can have this options:
 - min - minimal value among all processes named proc
 - max - maximal value among all processes named proc
 - avg - average value for all processes named proc
 - sum - sum of values for all processes named proc
3. Optional parameter that accepts process's command line regular expression, that should match cmd argument. If not set it means "match any".
4. Optional parameter that accepts process's owner username regular expression. If not set it means "match any".
5. Optional parameter that accepts process's main window title regular expression. If not set it means "match any". Process's window title can be checked only on Windows platform.

Number of mapped virtual memory regions within process with given name

Process.VMSize(*)

Data type: Int64

Supported Platforms: Windows, Linux, Solaris, AIX, HP-UX, FreeBSD, NetBSD

Parameters:

1. Process name
2. Function - is the function that is used to measure data in case if there are more than one process with given name. By default it is used sum function. This parameter can have this options:
 - min - minimal value among all processes named proc
 - max - maximal value among all processes named proc
 - avg - average value for all processes named proc
 - sum - sum of values for all processes named proc
3. Optional parameter that accepts process's command line regular expression, that should match cmd argument. If not set it means "match any".
4. Optional parameter that accepts process's owner username regular expression. If not set it means "match any".
5. Optional parameter that accepts process's main window title regular expression. If not set it means "match any". Process's window title can be checked only on Windows platform.

Virtual memory used by process

Process.WkSet(*)

Data type: Int64

Supported Platforms: Windows, Linux, Solaris, HP-UX, FreeBSD, NetBSD

Parameters:

1. Process name
2. Function - is the function that is used to measure data in case if there are more than one process with given name. By default it is used sum function. This parameter can have this options:
 - min - minimal value among all processes named proc
 - max - maximal value among all processes named proc
 - avg - average value for all processes named proc
 - sum - sum of values for all processes named proc
3. Optional parameter that accepts process's command line regular expression, that should match cmd argument. If not set it means "match any".
4. Optional parameter that accepts process's owner username regular expression. If not set it means "match any".
5. Optional parameter that accepts process's main window title regular expression. If not set it means "match any". Process's window title can be checked only on Windows platform.

Physical memory used by process

System.AppAddressSpace

Data type: UInt32

Supported Platforms: Windows

Address space available to applications (MB)

System.BIOS.Date

Data type: String

Supported Platforms: Windows, Linux, Solaris, FreeBSD

BIOS date.

System.BIOS.Vendor

Data type: String

Supported Platforms: Windows, Linux, Solaris, FreeBSD

BIOS vendor.

System.BIOS.Version

Data type: String

Supported Platforms: Windows, Linux, Solaris, FreeBSD

BIOS version.

System.ConnectedUsers

Data type: Int32

Supported Platforms: Windows, Linux

Number of users connected to system

System.CPU.Count

Data type: Int32

Supported Platforms: Windows, Linux, Solaris, AIX, FreeBSD, NetBSD, OpenBSD, MacOS

Number of CPUs in the system

System.CPU.LoadAvg

Data type: Float

Supported Platforms: Windows, Linux, Solaris, AIX, HP-UX, FreeBSD, NetBSD, OpenBSD, MacOS

CPU load average for last minute

Note

On Windows this metric is provided by winperf subagent

System.CPU.LoadAvg5

Data type: Float

Supported Platforms: Windows, Linux, Solaris, AIX, HP-UX, FreeBSD, NetBSD, OpenBSD, MacOS

CPU load average for last 5 minutes

Note

On Windows this metric is provided by winperf subagent

System.CPU.LoadAvg15

Data type: Float

Supported Platforms: Windows, Linux, Solaris, AIX, HP-UX, FreeBSD, NetBSD, OpenBSD, MacOS

CPU load average for last 15 minutes

Note

On Windows this metric is provided by winperf subagent

System.CPU.Usage

Data type: Float

Supported Platforms: Windows, Linux, Solaris, AIX, HP-UX, MacOS

Average CPU usage for last minute (percents, all CPUs)

Note

On Windows this metric is provided by winperf subagent

System.CPU.Usage(*)

Data type: Float

Supported Platforms: Windows, Linux, Solaris, AIX, MacOS

Parameters:

1. Zero-based index of CPU.

Average CPU usage for last minute (percents, specific CPU)

Note

On Windows this metric is provided by winperf subagent

System.CPU.Usage5

Data type: Float

Supported Platforms: Windows, Linux, Solaris, AIX, HP-UX, MacOS

Average CPU usage for last 5 minutes (percents, all CPUs)

Note

On Windows this metric is provided by winperf subagent

System.CPU.Usage5(*)

Data type: Float

Supported Platforms: Windows, Linux, Solaris, AIX, MacOS

Parameters:

1. Zero-based index of CPU.

Average CPU usage for last 5 minutes (percents, specific CPU)

Note

On Windows this metric is provided by winperf subagent

System.CPU.Usage15

Data type: Float

Supported Platforms: Windows, Linux, Solaris, AIX, HP-UX, MacOS

Average CPU usage for last 15 minutes (percents, all CPUs)

Note

On Windows this metric is provided by winperf subagent

System.CPU.Usage15(*)

Data type: Float

Supported Platforms: Windows, Linux, Solaris, AIX, MacOS

Parameters:

1. Zero-based index of CPU.

Average CPU usage for last 15 minutes (percents, specific CPU)

Note

On Windows this metric is provided by winperf subagent

System.CPU.Usage.Idle

Data type: Float

Supported Platforms: Linux, AIX, MacOS

Average CPU usage (IDLE) for last minute (percents, all CPUs)

System.CPU.Usage.Idle(*)

Data type: Float

Supported Platforms: Linux, AIX, MacOS

Parameters:

1. Zero-based index of CPU.

Average CPU usage (IDLE) for last minute (percents, specific CPU)

System.CPU.Usage5.Idle

Data type: Float

Supported Platforms: Linux, AIX, MacOS

Average CPU usage (IDLE) for last 5 minutes (percents, all CPUs)

System.CPU.Usage5.Idle(*)

Data type: Float

Supported Platforms: Linux, AIX, MacOS

Parameters:

1. Zero-based index of CPU.

Average CPU usage (IDLE) for last 5 minutes (percents, specific CPU)

System.CPU.Usage15.Idle

Data type: Float

Supported Platforms: Linux, AIX, MacOS

Average CPU usage (IDLE) for last 15 minutes (percents, all CPUs)

System.CPU.Usage15.Idle(*)

Data type: Float

Supported Platforms: Linux, AIX, MacOS

Parameters:

1. Zero-based index of CPU.

Average CPU usage (IDLE) for last 15 minutes (percents, specific CPU)

System.CPU.Usage.IOWait

Data type: Float

Supported Platforms: Linux, AIX

Average CPU usage (IOWAIT) for last minute (percents, all CPUs)

System.CPU.Usage.IOWait(*)

Data type: Float

Supported Platforms: Linux, AIX

Parameters:

1. Zero-based index of CPU.

Average CPU usage (IOWAIT) for last minute (percents, specific CPU)

System.CPU.Usage5.IOWait

Data type: Float

Supported Platforms: Linux, AIX

Average CPU usage (IOWAIT) for last 5 minutes (percents, all CPUs)

System.CPU.Usage5.IOWait(*)

Data type: Float

Supported Platforms: Linux, AIX

Parameters:

1. Zero-based index of CPU.

Average CPU usage (IOWAIT) for last 5 minutes (percents, specific CPU)

System.CPU.Usage15.IOWait

Data type: Float

Supported Platforms: Linux, AIX

Average CPU usage (IOWAIT) for last 15 minutes (percents, all CPUs)

System.CPU.Usage15.IOWait(*)

Data type: Float

Supported Platforms: Linux, AIX

Parameters:

1. Zero-based index of CPU.

Average CPU usage (IOWAIT) for last 15 minutes (percents, specific CPU)

System.CPU.Usage.IRQ

Data type: Float

Supported Platforms: Linux

Average CPU usage (IRQ) for last minute (percents, all CPUs)

System.CPU.Usage.IRQ(*)

Data type: Float

Supported Platforms: Linux

Parameters:

1. Zero-based index of CPU.

Average CPU usage (IRQ) for last minute (percents, specific CPU)

System.CPU.Usage5.IRQ

Data type: Float

Supported Platforms: Linux

Average CPU usage (IRQ) for last 5 minutes (percents, all CPUs)

System.CPU.Usage5.IRQ(*)

Data type: Float

Supported Platforms: Linux

Parameters:

1. Zero-based index of CPU.

Average CPU usage (IRQ) for last 5 minutes (percents, specific CPU)

System.CPU.Usage15.IRQ

Data type: Float

Supported Platforms: Linux

Average CPU usage (IRQ) for last 15 minutes (percents, all CPUs)

System.CPU.Usage15.IRQ(*)

Data type: Float

Supported Platforms: Linux

Parameters:

1. Zero-based index of CPU.

Average CPU usage (IRQ) for last 15 minutes (percents, specific CPU)

System.CPU.Usage.Nice

Data type: Float

Supported Platforms: Linux, MacOS

Average CPU usage (NICE) for last minute (percents, all CPUs)

System.CPU.Usage.Nice(*)

Data type: Float

Supported Platforms: Linux, MacOS

Parameters:

1. Zero-based index of CPU.

Average CPU usage (NICE) for last minute (percents, specific CPU)

System.CPU.Usage5.Nice

Data type: Float

Supported Platforms: Linux, MacOS

Average CPU usage (NICE) for last 5 minutes (percents, all CPUs)

System.CPU.Usage5.Nice(*)

Data type: Float

Supported Platforms: Linux, MacOS

Parameters:

1. Zero-based index of CPU.

Average CPU usage (NICE) for last 5 minutes (percents, specific CPU)

System.CPU.Usage15.Nice

Data type: Float

Supported Platforms: Linux, MacOS

Average CPU usage (NICE) for last 15 minutes (percents, all CPUs)

System.CPU.Usage15.Nice(*)

Data type: Float

Supported Platforms: Linux, MacOS

Parameters:

1. Zero-based index of CPU.

Average CPU usage (NICE) for last 15 minutes (percents, specific CPU)

System.CPU.Usage.SoftIRQ

Data type: Float

Supported Platforms: Linux

Average CPU usage (SOFTIRQ) for last minute (percents, all CPUs)

System.CPU.Usage.SoftIRQ(*)

Data type: Float

Supported Platforms: Linux

Parameters:

1. Zero-based index of CPU.

Average CPU usage (SOFTIRQ) for last minute (percents, specific CPU)

System.CPU.Usage5.SoftIRQ

Data type: Float

Supported Platforms: Linux

Average CPU usage (SOFTIRQ) for last 5 minutes (percents, all CPUs)

System.CPU.Usage5.SoftIRQ(*)

Data type: Float

Supported Platforms: Linux

Parameters:

1. Zero-based index of CPU.

Average CPU usage (SOFTIRQ) for last 5 minutes (percents, specific CPU)

System.CPU.Usage15.SoftIRQ

Data type: Float

Supported Platforms: Linux

Average CPU usage (SOFTIRQ) for last 15 minutes (percents, all CPUs)

System.CPU.Usage15.SoftIRQ(*)

Data type: Float

Supported Platforms: Linux

Parameters:

1. Zero-based index of CPU.

Average CPU usage (SOFTIRQ) for last 15 minutes (percents, specific CPU)

System.CPU.Usage.Steal

Data type: Float

Supported Platforms: Linux

Average CPU usage (STEAL) for last minute (percents, all CPUs)

System.CPU.Usage.Steal(*)

Data type: Float

Supported Platforms: Linux

Parameters:

1. Zero-based index of CPU.

Average CPU usage (STEAL) for last minute (percents, specific CPU)

System.CPU.Usage5.Steal

Data type: Float

Supported Platforms: Linux

Average CPU usage (STEAL) for last 5 minutes (percents, all CPUs)

System.CPU.Usage5.Steal(*)

Data type: Float

Supported Platforms: Linux

Parameters:

1. Zero-based index of CPU.

Average CPU usage (STEAL) for last 5 minutes (percents, specific CPU)

System.CPU.Usage15.Steal

Data type: Float

Supported Platforms: Linux

Average CPU usage (STEAL) for last 15 minutes (percents, all CPUs)

System.CPU.Usage15.Steal(*)

Data type: Float

Supported Platforms: Linux

Parameters:

1. Zero-based index of CPU.

Average CPU usage (STEAL) for last 15 minutes (percents, specific CPU)

System.CPU.Usage.System

Data type: Float

Supported Platforms: Linux, AIX, MacOS

Average CPU usage (SYSTEM) for last minute (percents, all CPUs)

System.CPU.Usage.System(*)

Data type: Float

Supported Platforms: Linux, AIX, MacOS

Parameters:

1. Zero-based index of CPU.

Average CPU usage (SYSTEM) for last minute (percents, specific CPU)

System.CPU.Usage5.System

Data type: Float

Supported Platforms: Linux, AIX, MacOS

Average CPU usage (SYSTEM) for last 5 minutes (percents, all CPUs)

System.CPU.Usage5.System(*)

Data type: Float

Supported Platforms: Linux, AIX, MacOS

Parameters:

1. Zero-based index of CPU.

Average CPU usage (SYSTEM) for last 5 minutes (percents, specific CPU)

System.CPU.Usage15.System

Data type: Float

Supported Platforms: Linux, AIX, MacOS

Average CPU usage (SYSTEM) for last 15 minutes (percents, all CPUs)

System.CPU.Usage15.System(*)

Data type: Float

Supported Platforms: Linux, AIX, MacOS

Parameters:

1. Zero-based index of CPU.

Average CPU usage (SYSTEM) for last 15 minutes (percents, specific CPU)

System.CPU.Usage.User

Data type: Float

Supported Platforms: Linux, AIX, MacOS

Average CPU usage (USER) for last minute (percents, all CPUs)

System.CPU.Usage.User(*)

Data type: Float

Supported Platforms: Linux, AIX, MacOS

Parameters:

1. Zero-based index of CPU.

Average CPU usage (USER) for last minute (percents, specific CPU)

System.CPU.Usage5.User

Data type: Float

Supported Platforms: Linux, AIX, MacOS

Average CPU usage (USER) for last 5 minutes (percents, all CPUs)

System.CPU.Usage5.User(*)

Data type: Float

Supported Platforms: Linux, AIX, MacOS

Parameters:

1. Zero-based index of CPU.

Average CPU usage (USER) for last 5 minutes (percents, specific CPU)

System.CPU.Usage15.User

Data type: Float

Supported Platforms: Linux, AIX, MacOS

Average CPU usage (USER) for last 15 minutes (percents, all CPUs)

System.CPU.Usage15.User(*)

Data type: Float

Supported Platforms: Linux, AIX, MacOS

Parameters:

1. Zero-based index of CPU.

Average CPU usage (USER) for last 15 minutes (percents, specific CPU)

System.CPU.VendorId

Data type: String

Supported Platforms: Windows, Linux, FreeBSD

CPU vendor ID.

System.CurrentTime

Data type: Float

Supported Platforms: Windows, Linux

Current system time

System.CurrentTime.ISO8601.Local

Data type: String

Supported Platforms: Windows, Linux

Current system local time in ISO 8601 format

System.CurrentTime.ISO8601.UTC

Data type: String

Supported Platforms: Windows, Linux

Current system UTC time in ISO 8601 format

System.HandleCount

Data type: Int32

Supported Platforms: Windows, Linux, Solaris, AIX

Total handles count at the moment

System.Hostname

Data type: String

Supported Platforms: Windows, Linux, Solaris, AIX, HP-UX, FreeBSD, NetBSD, OpenBSD

Host name

System.IO.BytesReadRate

Data type: Int64

Supported Platforms: Linux, Solaris, AIX, HP-UX

Average number of bytes read per second for last minute

System.IO.BytesReadRate(*)

Data type: Int64

Supported Platforms: Linux, Solaris, AIX, HP-UX

Parameters:

1. Device name

Average number of bytes read per second on specific device for last minute

System.IO.BytesWriteRate

Data type: Int64

Supported Platforms: Linux, Solaris, AIX, HP-UX

Average number of bytes written per second for last minute

System.IO.BytesWriteRate(*)

Data type: Int64

Supported Platforms: Linux, Solaris, AIX, HP-UX

Parameters:

1. Device name

Average number of bytes written per second on specific device for last minute

System.IO.DiskQueue

Data type: Float

Supported Platforms: Windows, Linux, Solaris, AIX, HP-UX

Average disk queue length for last minute

Note

On Windows this metric is provided by winperf subagent

System.IO.DiskQueue(*)

Data type: Float

Supported Platforms: Linux, Solaris, AIX, HP-UX

Parameters:

1. Device name

Average disk queue length for last minute for specific device

System.IO.DiskTime

Data type: Float

Supported Platforms: Windows, Linux

Average disk busy time for last minute (percents)

Note

On Windows this metric is provided by winperf subagent

System.IO.DiskTime(*)

Data type: Float

Supported Platforms: Linux

Parameters:

1. Device name

Average disk busy time for last minute for specific device (percents)

System.IO.ReadRate

Data type: Float

Supported Platforms: Linux, Solaris, AIX, HP-UX

Average number of read operations per second for last minute

System.IO.ReadRate(*)

Data type: Float

Supported Platforms: Linux, Solaris, AIX, HP-UX

Parameters:

1. Device name

Average number of read operations per second on specific device for last minute

System.IO.TransferRate

Data type: Float

Supported Platforms: AIX, HP-UX

Average number of data transfers per second for last minute

System.IO.TransferRate(*)

Data type: Float

Supported Platforms: AIX, HP-UX

Parameters:

1. Device name

Average number of data transfers per second on specific device for last minute

System.IO.OpenFiles

Data type: Int32

Supported Platforms: HP-UX

Number of open files

System.IO.WaitTime

Data type: UInt32

Supported Platforms: AIX, HP-UX

Average I/O wait time in milliseconds for last minute

System.IO.WaitTime(*)

Data type: UInt32

Supported Platforms: AIX, HP-UX

Parameters:

1. Device name

Average I/O wait time on specific device in milliseconds for last minute

System.IO.WriteRate

Data type: Float

Supported Platforms: Linux, Solaris, AIX, HP-UX

Average number of write operations per second for last minute

System.IO.WriteRate(*)

Data type: Float

Supported Platforms: Linux, Solaris, AIX, HP-UX

Parameters:

1. Device name

Average number of write operations per second on specific device for last minute

System.IsRestartPending

Data type: Integer

Supported Platforms: Windows

Indicator of pending system restart. Returns 1 when there are pending file renaming or deletion operations that cannot be immediately completed by the system because the files are currently in use.

System.IsVirtual

Data type: Integer

Supported Platforms: Windows, Linux, Solaris, AIX, HP-UX, FreeBSD, NetBSD, OpenBSD

Virtual system indicator. Returns 1 if system is virtual, 0 if not.

System.KStat(*)

Data type: Undefined

Supported Platforms: Solaris

Parameters:

1. Module
2. Instance
3. Name
4. Statistic

Solaris kstat data. More information can be found in kstat man.

System.Memory.Physical.Available

Data type: UInt64

Supported Platforms: Linux

Available physical memory in bytes

System.Memory.Physical.AvailablePerc

Data type: Float

Supported Platforms: Linux

Percentage of available physical memory

System.Memory.Physical.Buffers

Data type: UInt64

Supported Platforms: Linux

Physical memory used for buffers.

System.Memory.Physical BuffersPerc

Data type: Float

Supported Platforms: Linux

Percentage of physical memory used for buffers.

System.Memory.Physical.Cached

Data type: UInt64

Supported Platforms: Linux

Physical memory used for cache.

System.Memory.Physical.CachedPerc

Data type: Float

Supported Platforms: Linux

Percentage of physical memory used for cache.

System.Memory.Physical.Free

Data type: UInt64

Supported Platforms: Windows, Linux, Solaris, AIX, HP-UX, FreeBSD, NetBSD, OpenBSD

Free physical memory in bytes

System.Memory.Physical.FreePerc

Data type: UInt

Supported Platforms: Windows, Linux, Solaris, AIX, HP-UX, FreeBSD

Percentage of free physical memory

System.Memory.Physical.Total

Data type: UInt64

Supported Platforms: Windows, Linux, Solaris, AIX, HP-UX, FreeBSD, NetBSD, OpenBSD

Total amount of physical memory in bytes

System.Memory.Physical.Used

Data type: UInt64

Supported Platforms: Windows, Linux, Solaris, AIX, HP-UX, FreeBSD, NetBSD, OpenBSD

Used physical memory in bytes

System.Memory.Physical.UsedPerc

Data type: Float

Supported Platforms: Windows, Linux, Solaris, AIX, HP-UX, FreeBSD

Percentage of used physical memory

System.Memory.Swap.Free

Data type: UInt64

Supported Platforms: Linux, AIX, HP-UX, FreeBSD, NetBSD, OpenBSD

Free swap space in bytes

System.Memory.Swap.FreePerc

Data type: Float

Supported Platforms: Linux, AIX, HP-UX, FreeBSD

Percentage of free swap space

System.Memory.Swap.Total

Data type: UInt64

Supported Platforms: Linux, AIX, HP-UX, FreeBSD, NetBSD, OpenBSD

Total amount of swap space in bytes

System.Memory.Swap.Used

Data type: UInt64

Supported Platforms: Linux, AIX, HP-UX, FreeBSD, NetBSD, OpenBSD

Used swap space in bytes

System.Memory.Swap.UsedPerc

Data type: Float

Supported Platforms: Linux, AIX, HP-UX, FreeBSD

Percentage of used swap space

System.Memory.Virtual.Active

Data type: UInt64

Supported Platforms: AIX

Active virtual memory

System.Memory.Virtual.ActivePerc

Data type: Float

Supported Platforms: AIX

Percentage of active virtual memory

System.Memory.Virtual.Free

Data type: UInt64

Supported Platforms: Windows, Linux, AIX, HP-UX, FreeBSD, NetBSD, OpenBSD

Free virtual memory in bytes

System.Memory.Virtual.FreePerc

Data type: Float

Supported Platforms: Windows, Linux, AIX, HP-UX, FreeBSD

Percentage of free virtual memory

System.Memory.Virtual.Total

Data type: UInt64

Supported Platforms: Windows, Linux, AIX, HP-UX, FreeBSD, NetBSD, OpenBSD

Total amount of virtual memory in bytes

System.Memory.Virtual.Used

Data type: UInt64

Supported Platforms: Windows, Linux, AIX, HP-UX, FreeBSD, NetBSD, OpenBSD

Used virtual memory in bytes

System.Memory.Virtual.UsedPerc

Data type: Float

Supported Platforms: Windows, Linux, AIX, HP-UX, FreeBSD

Percentage of used virtual memory

System.MsgQueue.Bytes(*)

Data type: UInt64

Supported Platforms: Linux, Solaris, AIX, HP-UX

Parameters:

1. Queue ID or key

Bytes in given message queue.

System.MsgQueue.BytesMax(*)

Data type: UInt64

Supported Platforms: Linux, Solaris, AIX, HP-UX

Parameters:

1. Queue ID or key

Maximum allowed bytes in given message queue.

System.MsgQueue.ChangeTime(*)

Data type: UInt64

Supported Platforms: Linux, Solaris, AIX, HP-UX

Parameters:

1. Queue ID or key

Time of the last change for given message queue.

System.MsgQueue.Messages(*)

Data type: UInt

Supported Platforms: Linux, Solaris, AIX, HP-UX

Parameters:

1. Queue ID or key

Number of messages in given message queue.

System.MsgQueue.RecvTime(*)

Data type: UInt64

Supported Platforms: Linux, Solaris, AIX, HP-UX

Parameters:

1. Queue ID or key

Last recieved message time in given message queue.

System.MsgQueue.SendTime(*)

Data type: UInt64

Supported Platforms: Linux, Solaris, AIX, HP-UX

Parameters:

1. Queue ID or key

Last sent message time in given message queue.

System.OS.Build

Data type: String

Supported Platforms: Windows, Linux, FreeBSD

Operating system build.

Note

Might be not available on some Linux family platforms.

System.OS.LicenseKey

Data type: String

Supported Platforms: Windows

Operating system license key.

System.OS.ProductId

Data type: String

Supported Platforms: Windows

Operating system ID.

System.OS.ProductName

Data type: String

Supported Platforms: Windows, Linux, AIX, FreeBSD, Solaris

Operating system name.

System.OS.ProductType

Data type: String

Supported Platforms: Windows, Linux, FreeBSD

Operating system type.

Note

Might be not available on some Linux family platforms.

System.OS.ServicePack

Data type: String

Supported Platforms: Windows, AIX

Operating system service pack.

System.OS.Version

Data type: String

Supported Platforms: Windows, Linux, AIX, FreeBSD, Solaris

Operating system version.

System.PlatformName

Data type: String

Supported Platforms: Windows, Linux, Solaris, AIX, HP-UX, FreeBSD, NetBSD, OpenBSD

Unified platform name (used by agent upgrade component)

System.ProcessCount

Data type: UInt32

Supported Platforms: Windows, Linux, Solaris, AIX, HP-UX, FreeBSD, NetBSD, OpenBSD

Total number of processes in system

System.ServiceState(*)

Data type: Int32

Supported Platforms: Windows

Parameters:

1. Windows service name

State of system service. Possible values:

- 0 - service running
- 1 - service paused
- 2 - service starting (start pending)
- 3 - service pausing (pause pending)
- 4 - service starting after pause (continue pending)
- 5 - service stopping (stop pending)
- 6 - service stopped
- 255 - unable to get current service state

System.ThreadCount

Data type: UInt32

Supported Platforms: Windows, Linux, AIX, FreeBSD, NetBSD

Total number of threads in system

System.TimeZone

Data type: String

Supported Platforms: Windows, Linux

System time zone offset and name

System.TimeZoneOffset

Data type: Int32

Supported Platforms: Windows, Linux

System time zone offset from UTC time

System.Uname

Data type: String

Supported Platforms: Windows, Linux, Solaris, AIX, HP-UX, FreeBSD, NetBSD, OpenBSD

Output of uname command

System.Uptime

Data type: Int32

Supported Platforms: Windows, Linux, Solaris, AIX, HP-UX, FreeBSD, NetBSD, OpenBSD

Number of seconds since system boot

Note

On Windows this metric is provided by winperf subagent

X509.Certificate.ExpirationDate

Data type: String

Supported Platforms: Windows, Linux, Solaris, AIX, HP-UX, FreeBSD, NetBSD, OpenBSD

Parameters:

1. Path to the certificate file.

Expiration date (YYYY-MM-DD) of X.509 certificate from provided file.

X509.Certificate.ExpirationTime

Data type: UInt64

Supported Platforms: Windows, Linux, Solaris, AIX, HP-UX, FreeBSD, NetBSD, OpenBSD

Parameters:

1. Path to the certificate file.

Expiration date in UNIX timestamp format.

X509.Certificate.ExpiresIn

Data type: Int32

Supported Platforms: Windows, Linux, Solaris, AIX, HP-UX, FreeBSD, NetBSD, OpenBSD

Parameters:

1. Path to the certificate file.

Days until expiration of X.509 certificate from provided file.

X509.Certificate.Issuer

Data type: String

Supported Platforms: Windows, Linux, Solaris, AIX, HP-UX, FreeBSD, NetBSD, OpenBSD

Parameters:

1. Path to the certificate file.

Issuer of X.509 certificate from provided file.

X509.Certificate.Subject

Data type: String

Supported Platforms: Windows, Linux, Solaris, AIX, HP-UX, FreeBSD, NetBSD, OpenBSD

Parameters:

1. Path to the certificate file.

Subject of X.509 certificate from provided file.

X509.Certificate.TemplateID

Data type: String

Supported Platforms: Windows, Linux, Solaris, AIX, HP-UX, FreeBSD, NetBSD, OpenBSD

Parameters:

1. Path to the certificate file.

Template ID of X.509 certificate from provided file.

46.8.2 List metrics

DRBD.DeviceList

Data type: List of String

Supported Platforms: Linux

List of configured DRBD devices

FileSystem.MountPoints

Data type: List of String

Supported Platforms: Linux, Windows, Solaris, AIX, FreeBSD

Currently available mount points

Hardware.Batteries

Data type: List of String

Supported Platforms: Linux, Windows, Solaris

Information about batteries installed on the device

Hardware.MemoryDevices

Data type: List of String

Supported Platforms: Linux, Windows, Solaris

Information about available memory devices

Hardware.Processors

Data type: List of String

Supported Platforms: Windows

Information about available processors

Hardware.StorageDevices

Data type: List of String

Supported Platforms: Windows

Information about available storage devices

LVM.LogicalVolumes

Data type: List of String

Supported Platforms: AIX

Logical Volume Manager information - all logical volumes

LVM.LogicalVolumes(*)

Data type: List of String

Supported Platforms: AIX

Logical Volume Manager information - logical volumes of the specified volume group

LVM.PhysicalVolumes

Data type: List of String

Supported Platforms: AIX

Logical Volume Manager information - all physical volumes

LVM.PhysicalVolumes(*)

Data type: List of String

Supported Platforms: AIX

Parameters:

1. Volume group name.

Logical Volume Manager information - physical volumes of the specified volume group

LVM.VolumeGroups

Data type: List of String

Supported Platforms: AIX

Logical Volume Manager information - volume groups' names

Net.ArpCache

Data type: List of String

Supported Platforms: Linux, Windows, FreeBSD

Local ARP cache

Net.InterfaceList

Data type: List of String

Supported Platforms: Linux, Windows, Solaris, AIX, FreeBSD

Interface index, IP address, subnet mask, type, maximum transmission unit, MAC address and name

The format is: *[index] [IP]/[mask] [type]([MTU]) [MAC] [name]*

Net.InterfaceNames

Data type: List of String

Supported Platforms: Linux, Windows, Solaris, AIX, FreeBSD

Names of available interfaces

Net.IP.RoutingTable

Data type: List of String

Supported Platforms: Linux, Windows, FreeBSD

IP routing table

System.ActiveUserSessions

Data type: List of String

Supported Platforms: Linux, Windows

Currently active user sessions

System.Desktops(*)

Data type: List of String

Supported Platforms: Windows

Currently active desktops

System.IO.Devices

Data type: List of String

Supported Platforms: Linux, Windows

Currently available input and output devices' names

System.ProcessList

Data type: List of String

Supported Platforms: Linux, Windows, Solaris, AIX, FreeBSD

Running processes' names

System.Services

Data type: List of String

Supported Platforms: Windows

Running services' names

System.WindowStations

Supported Platforms: Windows

Window stations' names

46.8.3 Table metrics

Note

Columns marked with * are instance columns (primary keys). Such columns (or combination of columns) are designated to uniquely identify each table record.

FileSystem.Volumes

Supported Platforms: Linux, Windows, Solaris, AIX

Column name	Data type
Mount Point *	String
Volume	String
Label	String
FS Type	String
Total	UInt64
Free	UInt64
Free %	Float
Available	UInt64
Available %	Float
Used	UInt64
Used %	Float

Available file system volumes

Hardware.Batteries

Supported Platforms: Linux, Windows, Solaris

Column name	Data type
Handle *	Int32
Name	String
Location	String
Capacity	UInt32
Voltage	UInt32
Chemistry	String
Manufacturer	String
Manufacture Date	String
Serial Number	String

Hardware information about batteries installed on the device

Hardware.MemoryDevices

Supported Platforms: Linux, Windows, Solaris

Column name	Data type
Handle *	Int32
Location	String
Bank	String
Form factor	String
Type	String
Size	UInt64
Max Speed	UInt64
Configured Speed	UInt64
Manufacturer	String
Part Number	String
Serial Number	String

Hardware information about available memory devices

Hardware.NetworkAdapters

Supported Platforms: Linux, Windows

Column name	Data type
Index *	UInt32
Product	String
Manufacturer	String
Description	String
Type	String
MAC address	String
Interface index	UInt32
Speed	UInt64
Availability	UInt32

Hardware information about available network adapters

Hardware.Processors

Supported Platforms: Linux, Windows, Solaris

Column name	Data type
Handle *	Int32
Type	String
Family	String
Version	String
Socket	String
Cores	UInt32
Threads	UInt32
Max Speed	UInt64
Current Speed	UInt64
Manufacturer	String
Part Number	String
Serial Number	String

Hardware information about available processors

Hardware.StorageDevices

Supported Platforms: Linux, Windows

Column name	Data type
Number *	UInt32
Type	UInt32
Type description	String
Bus type	String
Removable	Int32
Size	UInt64
Manufacturer	String
Product	String
Revision	String
Serial number	String

Hardware information about available storage devices

Net.Wireguard.Interfaces

Supported Platforms: Linux, BSD, Mac OS X

Column name	Data type
NAME *	String
PUBLIC_KEY	String
LISTEN_PORT	UInt32

Example output:

NAME	PUBLIC_KEY	LISTEN_PORT
gw	eWfYktu1DjurgOUfCiBOfbiduddfmLiS1D+smdBj+28=	51820

Net.Wireguard.Peers

Supported Platforms: Linux, BSD, Mac OS X

Column name	Data type
INTERFACE	String
PEER_PUBLIC_KEY *	String
ENDPOINT	String
ALLOWED_IPS	String
HANDSHAKE_TIMESTAMP	UInt64
RX	UInt64
TX	UInt64

Example output:

INTERFACE	PEER_PUBLIC_KEY	ENDPOINT	
↪	ALLOWED_IPS	HANDSHAKE_TIMESTAMP	RX TX
gw	BWEY+dXnkkhl836PVpkDaAwImnFeCQogfZrnVz1Svmo=		
↪	[fd42:5c39:7438:816b:216:3eff:fed3:fd0a]:10687	192.168.1.2/32	1722296581
↪	3676	1012	
gw	TN77lQm65yIJKWGJyWwFSfa8QCuLYasap5m0x+/CBM=	10.107.72.157:6802	
↪	192.168.2.2/32	1722296582	3676 1012

System.ActiveUserSessions

Supported Platforms: Windows

Column name	Data type
ID *	UInt32
User name	String
Terminal	String
State	String
Client name	String
Client address	String
Client display	String
Connect time	UInt64
Logon time	UInt64
Idle for	UInt32

Currently active user sessions

System.InstalledProducts

Supported Platforms: Linux, Windows, Solaris, AIX, FreeBSD

Column name	Data type
Name *	String
Version	String
Vendor	String
Install Date	String
URL	String
Description	String

Products installed on the system

System.OpenFiles

Supported Platforms: Linux

Column name	Data type
PID *	UInt32
Process	String
Handle *	UInt32
Name	String

Files opened by processes

System.Processes

Supported Platforms: Linux, Windows, Solaris, AIX, FreeBSD

Column name	Data type
PID *	UInt32
Name	String
User	String
Threads	UInt32
Handles	UInt32
Kernel Time	UInt64
User Time	UInt64
VM Size	UInt64
RSS	UInt64
Page Faults	UInt64
Command Line	String

Running processes information

System.Services

Supported Platforms: Windows

Column name	Data type
Name *	String
Display name	String
Type	String
State	String
Startup	String
Run As	String
PID	UInt32
Binary	String
Dependencies	String

Running services information

GLOSSARY

802.1x

IEEE 802.1X (also known as Dot1x) is an IEEE Standard for Port-based Network Access Control (PNAC). It is part of the IEEE 802.1 group of networking protocols. It provides an authentication mechanism to devices wishing to attach to a *LAN* or WLAN. More details in [Wikipedia](#)

Action

Configurable operation which can be executed by the system when *Event* is passing thru *Event Processing Policy*. Multiple action types are supported, including email or notifications (SMS, instant messages), executing OS commands and forwarding events to another instance of NetXMS server.

Alarm

Outstanding issue which require operator attention. Alarms are created by the system as a result of *Event* passing thru *Event Processing Policy*.

Alarm Browser

View in user interface, which shows all active alarms in the system and allow user to interact with them.

ARP

The Address Resolution Protocol (ARP) is a telecommunication protocol used for resolution of network layer addresses into link layer addresses, a critical function in multiple-access networks. More details in [Wikipedia](#)

Business Service

An IT Service that directly supports a Business Process, as opposed to an Infrastructure Service which is used internally by the IT Service Provider and is not usually visible to the Business.

CA

Certification authority is an entity that issues digital certificates. More details in [Wikipedia](#)

CDP

Cisco Discovery Protocol is a Cisco proprietary protocol that runs between direct connected network entities (routers, switches, remote access devices, IP telephones etc.). The purpose of the protocol is to supply a network entity with information about its direct connected neighbors. More details in [Wikipedia](#).

Condition

(Create condition in infrastructure services)

Container

Object that can store other containers and *nodes*.

CSR

Certificate signing request is a message sent from an applicant to a certificate authority in order to apply for a digital identity certificate. More details in [Wikipedia](#)

Dashboard

Manually generated *Object* that can combine any available visualization components with data from multiple sources in order to create high-level views to see network or parts of it, and it's health.

Data Collection Item

Configuration entity of a single *Metric*.

DCI

Abbreviation for *Data Collection Item*

DNS

Domain Name System. More details in [Wikipedia](#)

Entire Network

Automatically generated object hierarchy that contains all nodes and IP subnets known to NetXMS.

EPP

Abbreviation for *Event Processing Policy*

Event

TBD A change of state which has significance for the management of IT Service.

Event Processing Policy

List of rules which defines system reaction on *events*. All events are matched against list of rules in Event Processing Policy, if match is found - configured actions are executed.

Event Template

TBD

GPL

GNU General Public License. *Full text of the License, version 2* <<http://www.gnu.org/licenses/gpl-2.0.html>>

GUID

A Globally Unique Identifier is a unique reference number used as an identifier in computer software. More details in [Wikipedia](#)

ICMP

The Internet Control Message Protocol (ICMP) is one of the main protocols of the Internet Protocol Suite. It is used by network devices, like routers, to send error messages indicating, for example, that a requested service is not available or that a host or router could not be reached. More details in [Wikipedia](#).

Infrastructure services

System *container* which can be used by Administrator to define logical structure of the network.

LAN

A local area network (LAN) is a computer network that interconnects computers within a limited area such as a home, school, computer laboratory, or office building, using network media. The defining characteristics of LANs, in contrast to wide area networks (WANs), include their smaller geographic area, and non-inclusion of leased telecommunication lines. More details in [Wikipedia](#).

LDAP

The Lightweight Directory Access Protocol (LDAP) is an open, vendor-neutral, industry standard application protocol for accessing and maintaining distributed directory information services over an Internet Protocol (IP) network. More details in [Wikipedia](#)

LLDP

The Link Layer Discovery Protocol (LLDP) is a vendor-neutral link layer protocol in the Internet Protocol Suite used by network devices for advertising their identity, capabilities, and neighbors on an IEEE 802 local area network, principally wired Ethernet. The protocol is formally referred to by the IEEE as Station and Media Access Control Connectivity Discovery specified in standards document IEEE 802.1AB. More details in [Wikipedia](#)

MAC address

A media access control address (MAC address) is a unique identifier assigned to network interfaces for communications on the physical network segment. MAC addresses are used as a network address for most IEEE 802

network technologies, including Ethernet and WiFi. Logically, MAC addresses are used in the media access control protocol sublayer of the OSI reference model. More details in [Wikipedia](#).

Management Client

NetXMS user interface. Available in form of [rich client](#) for both desktop and mobile or as web user interface.

Metric

One entity of collected data

MIB Explorer

[View](#) in user interface, which allows to navigate SNMP MIB tree and run SNMP walk on [nodes](#).

Mobile Device Object

Special type of [Node](#) that represents monitored mobile device.

Monitoring Agent

NetXMS or SNMP agent that provides information to NetXMS Server.

NDP

The Neighbor Discovery Protocol (NDP) is a protocol in the Internet protocol suite used with Internet Protocol Version 6 (IPv6). More details in [Wikipedia](#)

Network Discovery

Network investigation in order to find new [nodes](#). There are 2 types of discovery: active and passive. In passive mode, information about new hosts and devices obtained from [ARP](#) tables and routing tables of already known devices. In active discovery mode, NetXMS server will send an [ICMP](#) echo requests to all IP addresses in given range, and consider each responding address for adding to database.

Network Map

Visual representation of network topology.

NetXMS Agent

NetXMS daemon that is installed on monitored [Node](#) to provide additional monitoring options.

Node

[Object](#) that represents server or device.

NXSL

NetXMS Scripting Language.

Object

Representation of logical or physical entity.

Object tool

Configurable operation that can be executed on [Node](#).

Package Manager

[View](#) that manages update packages for NetXMS agents.

Perspective

A perspective defines the initial set and layout of views in the Eclipse Workbench window.

Policy

Configuration parameter set that can be applied on a [Node](#).

Polling

Polling is process of gathering information by server from nodes. This is usually done automatically at specified intervals of time, but can be triggered manually also. There are different types of polling: Status, Configuration, Topology, Discovery and Routing.

Proxy Agent

NetXMS Agent capable of forwarding requests to [nodes](#) which are not directly accessible to NetXMS server. Agent support proxying of native agent protocol as well as SNMP.

Push parameter

Type of *DCI*, where collected data is pushed into the server by the agent.

RADIUS

Remote Authentication Dial In User Service (RADIUS) is a networking protocol that provides centralized Authentication, Authorization, and Accounting (AAA) management for users who connect and use a network service. More details in [Wikipedia](#)

SMCLP

Server Management Command Line Protocol

SNMP

Simple Network Management Protocol (SNMP) is an “Internet-standard protocol for managing devices on IP networks”. Devices that typically support SNMP include routers, switches, servers, workstations, printers, modem racks and more. SNMP is used mostly in network management systems to monitor network-attached devices for conditions that warrant administrative attention. SNMP is a component of the Internet Protocol Suite as defined by the Internet Engineering Task Force (IETF). It consists of a set of standards for network management, including an application layer protocol, a database schema, and a set of data objects. More details in [Wikipedia](#).

SNMP Trap

Asynchronous notification from *SNMP* agent to *SNMP* manager. SNMP traps enable an agent to notify the management station of significant events by way of an unsolicited SNMP message. More details in [Wikipedia](#).

STP

The Spanning Tree Protocol (STP) is a network protocol that ensures a loop-free topology for any bridged Ethernet local area network. The basic function of STP is to prevent bridge loops and the broadcast radiation that results from them. Spanning tree also allows a network design to include spare (redundant) links to provide automatic backup paths if an active link fails, without the danger of bridge loops, or the need for manual enabling/disabling of these backup links. More details in [Wikipedia](#)

Subagent

Extension module (shared library) which can be loaded into NetXMS agent to provide additional functionality.

Syslog

Widely used standard for message logging. More details in [Wikipedia](#).

Template

A preset of one or more *DCIs* that can be applied on *Node*.

Threshold

Part of *DCI* configuration, which define events to be generated when collected value is outside of expected range.

TLS

Transport Layer Security is a cryptographic protocols that provide communications security over a computer network. More details in [Wikipedia](#).

Trim Stack

View Stack in minimized state, represented as a set of buttons, one for each *View* in the stack.

UPS

An uninterruptible power supply, also uninterruptible power source, UPS or battery/flywheel backup, is an electrical apparatus that provides emergency power to a load when the input power source, typically mains power, fails. More details in [Wikipedia](#)

URL

A uniform resource locator (URL) is a reference to a resource that specifies the location of the resource on a computer network and a mechanism for retrieving it. More details in [Wikipedia](#)

View

In the Eclipse Platform a view is typically used to navigate a hierarchy of information, open an editor, or display properties for the active editor.

View Stack

Multiple [views](#) combined into single one, with tab navigation on top of it.

VLAN

In computer networking, a single layer-2 network may be partitioned to create multiple distinct broadcast domains, which are mutually isolated so that packets can only pass between them via one or more routers; such a domain is referred to as a virtual local area network, virtual LAN or VLAN. More details in [Wikipedia](#).

VPN

A virtual private network (VPN) extends a private network across a public network, such as the Internet. It enables a computer or network-enabled device to send and receive data across shared or public networks as if it were directly connected to the private network, while benefiting from the functionality, security and management policies of the private network. A VPN is created by establishing a virtual point-to-point connection through the use of dedicated connections, virtual tunneling protocols, or traffic encryptions. Major implementations of VPNs include OpenVPN and IPsec. More details in [Wikipedia](#).

VRRP

The Virtual Router Redundancy Protocol (VRRP) is a computer networking protocol that provides for automatic assignment of available Internet Protocol (IP) routers to participating hosts. This increases the availability and reliability of routing paths via automatic default gateway selections on an IP subnetwork. More details in [Wikipedia](#)

Zone

Zone in NetXMS is a group of IP subnets which form non-overlapping IP address space. There is always zone 0 which contains subnets directly reachable by management server. For all other zones server assumes that subnets within that zones are not reachable directly, and proxy must be used. It is used to monitor subnets with overlapping IP address space.

Non-alphabetical

802.1x, [561](#)

A

Action, [561](#)

Alarm, [561](#)

Alarm Browser, [561](#)

ARP, [561](#)

B

Business Service, [561](#)

C

CA, [561](#)

CDP, [561](#)

Condition, [561](#)

Container, [561](#)

CSR, [561](#)

D

Dashboard, [561](#)

Data Collection Item, [562](#)

DCI, [562](#)

DNS, [562](#)

E

Entire Network, [562](#)

EPP, [562](#)

Event, [562](#)

Event Processing Policy, [562](#)

Event Template, [562](#)

G

GPL, [562](#)

GUID, [562](#)

I

ICMP, [562](#)

Infrastructure services, [562](#)

L

LAN, [562](#)

LDAP, [562](#)

LLDP, [562](#)

M

MAC address, [562](#)

Management Client, [563](#)

Metric, [563](#)

MIB Explorer, [563](#)

Mobile Device Object, [563](#)

Monitoring Agent, [563](#)

N

NDP, [563](#)

Network Discovery, [563](#)

Network Map, [563](#)

Node, [563](#)

NXSL, [563](#)

O

Object, [563](#)

Object tool, [563](#)

P

Package Manager, [563](#)

Perspective, [563](#)

Policy, [563](#)

Polling, [563](#)

product_name Agent, [563](#)

Proxy Agent, [563](#)

Push parameter, [564](#)

R

RADIUS, [564](#)

S

SMCLP, [564](#)

SNMP, [564](#)

SNMP Trap, [564](#)

STP, [564](#)

Subagent, [564](#)

Syslog, [564](#)

T

Template, [564](#)
Threshold, [564](#)
TLS, [564](#)
Trim Stack, [564](#)

U

UPS, [564](#)
URL, [564](#)

V

View, [564](#)
View Stack, [565](#)
VLAN, [565](#)
VPN, [565](#)
VRRP, [565](#)

Z

Zone, [565](#)