Nexthink V6.9

Glossary and References

Generated: 5/02/2020 1:18 pm

Table of Contents

Table of Contents

Table of Contents

Glossary

Activity

An *activity* is a specific action detected in your IT infrastructure in which one or more of the objects monitored by Nexthink are involved. Nexthink supervises the following types of activities:

- Installation
- Execution
- Connection
- Web request
- Print job
- System boot
- User logon

Related concepts

- Object
- Event

Alert

Alerts warn you of particular situations in your IT infrastructure. An alert usually indicates the occurrence of an issue that needs to be addressed. The idea behind alerts is that you do not need to look constantly for problematic cases in the system, but it is the system that can detect them and notify you when they happen. There are three types of alerts:

System alerts
: Predefined alerts that reveal special circumstances in the operation of the system.

Investigation-based alerts
: Triggered by conditions that you can specify in the Finder using investigations.

Service-based alerts
: Triggered by warning or error conditions on the real-time services that you add to your Portal.

Related tasks

- Receiving alerts
- Creating a service-based alert
- Creating an investigation-based alert

Related references

- System alerts

Application

An *application* is an object that represents a set of executable programs bundled by a software manufacturer as a single software product.

Applies to platforms:

Related concepts

- Executable
- Binary
- Package
- Object

Binary

A *binary* is an object that represents the physical image on disk of a particular version of an executable file. The sequence of bytes in the executable file completely characterizes the binary. To reduce the quantity of information needed, Nexthink does not identify a binary by its complete sequence of bytes, but by a hash number computed from this sequence.

Applies to platforms:

Thus, executable files that share the same name are only represented by the same binary in Nexthink if they share exactly the same sequence of bytes in disk as well. Two executable files with the same name may have a different sequence of bytes because of two reasons:

Versioning
: If the versions of the executable files are different, the files have a different binary image. Therefore, a binary is created for each version of the executable file.

Modified executable file
: If one of the executable files has been altered by any means, the binary image is necessarily different even when the versions of the executable files are the same. A modified file is an indication of malware.

Nexthink detects the existence of a binary in your IT infrastructure the first time that the binary is run on one of the monitored devices.

Related references

- Binary paths
- Maximum number of Binaries

Related concepts

- Application
- Executable
- Object

Campaign

A *campaign* is a set of questions addressed to a group of users with the goal of getting their opinion on a particular matter. Usually, the subject matter of a campaign relates to the users' perception of the IT services and equipment that your organization provides to them.

Create your campaigns in the Finder and assign different target audiences for each campaign by using investigations on users. The targeted end-users receive notifications in their Collector equipped devices for answering the questions of your campaigns.

Track the results of your campaigns using investigations and metrics, as well as dashboards to follow their evolution.

Related tasks

- Creating a campaign

Category

A *category* is a way to group objects of the same type into user-defined classes. Each class is identified by a keyword or tag. Only users with the right privilege level can create categories and keywords.

Related tasks

- Creating categories and keywords

Related concepts

- Keyword
- Object
- Hierarchy

Connection

A *connection* is a link between a device and a destination through the use of network resources. There are two types of connections depending on the transport protocol used for communication:

TCP
: The connection has a status.

UDP
: The connection is stateless.

For TCP connections, the link between device and destination does not need to be fully established for Nexthink to record the connection. For UDP connections, it is not possible to know if the connection was established, due to the own nature of the protocol. Thus, for any protocol, every connection attempt that Nexthink detects is recorded on the Nexthink database no matter whether the connection is successful or not.

Some repetitive short-lived connections are automatically grouped into one single aggregated connection when some sort of scanning is detected:

Network scan
: A repeated attempt to connect to the same port on several destinations.

Port scan
: A repeated attempt to connect to different ports on the same destination.

Network and port scans can be launched by legitimate processes, but they may also indicate the existence of malicious activity in your network. Scanning communication ports in one or several machines is a widely used method to detect vulnerabilities in computer networks.

Only connections using the same transport protocol may be grouped into a single scan connection. Thus, there are four types of scan connections:

Protocol \ Type	Network	Port
TCP	TCP network scan	TCP port scan
UDP	UDP network scan	UDP port scan

See Network and port scan conditions to find out when Nexthink regards a set TCP and UDP connections as a scanning operation.

Related concepts

- Activity
- Port
- Device
- Destination

Related references

- Status of TCP connections
- Status of UDP connections
- Network and port scan conditions

Dashboard

A *dashboard* is a panel with selected information elements about your IT infrastructure that is displayed as a web page in the Portal.

Related tasks

- Creating a dashboard

Related concepts

- Module
- Widget

Destination

A *destination* is a computer that accepts incoming network connections. Therefore, destinations play the role of the server in the client-server model of communication. Devices, on the other hand, usually play the role of the client. However, devices can also function as servers in some cases. Devices that accept connections are thus listed both as devices and as destinations.

Applies to platforms:

Destinations are identified by their IP address. There are two special kinds of destinations:

external
: Destination that falls outside the networks supervised by the Engine.

multiple
: A virtual destination that represents a network scanning.

Related concepts

- Connection
- Device

Device

A *device* is an object that represents any personal computer or computerized system from which Nexthink can extract information to produce end-user analytics.

Nexthink classifies devices into four types:

Desktop
: A conventional or virtualized personal computer running a client version of either Windows or Mac operating systems.

Laptop
: A portable personal computer running a client version of either Windows or Mac operating systems.

Server
: A computer that provides functionality (e.g. email, web, or directory services) to client computers (desktops and laptops) and runs a version of the Windows Server operating system.

Mobile
A phone or tablet equipped with Exchange ActiveSync.

See the list of operating systems supported by the Collector.

Licensing

The licensing model of Nexthink is based on the number of devices of the different types in your setup. For licensing purposes, desktops and laptops are both considered **endpoints**; that is, regular devices that support the installation of the Collector.

Thus, when licensing your product, provide the number of:

- Endpoints (desktops plus laptops)
- Servers
- Mobile devices

Applies to platforms:
Related concepts

- Object

Related references

- Operating systems suported by the Collector
- Setting up a software license

Domain

A *domain* is an object that represents a realm of administrative authority on the Internet and is identified by a name. Domain names are formed using the rules and procedures described in the Domain Name System (DNS). Domain names are organized into levels (subdomains), being the top-level domains the country codes, and the well-known *com*, *org*, *net* or *edu*, among others. Organizations typically register second or third-level domain names through accredited registrar companies to publicly offer their services on the Internet; for example `www.nexthink.com`. Note that subdomain levels in a domain name are inversed with respect to their hierarchy (top-level subdomains are placed last). The main purpose of the DNS is to identify areas of the Internet with easy to memorize names and to translate those names into numerical IP addresses that

routing devices understand. A complete domain name with all levels specified is known as a Fully Qualified Domain Name (FQDN).

Applies to platforms:

Domain names are a central part of Uniform Resource Locators (URLs), which reference individual resources in the Internet; for example `http://www.nexthink.com/what-is-nexthink/`. A URL can refer to a web page, a text file, an image, a video stream or any other kind of resource in the Internet. The first part of a URL is called the *scheme*. The scheme usually designates the protocol used in the connection, such as ftp or http. In web browsers, users typically type URLs in the navigation bar to retrieve a particular web page, but other applications may use URLs internally to get information from the Internet without necessarily displaying them.

Nexthink records the domains of all the web requests initiated from a monitored device, regardless or the application that made the request. Nexthink considers a connection to be a web request when the scheme of the URL is **http** or **https**; that is, when the connection uses the Hypertext Transfer Protocol (HTTP) or the Hypertext Transfer Protocol Secure (HTTPS), which is an encrypted version of HTTP with Transport Layer Security (TLS).

Domain compaction

To avoid storing too many names for web domains, Nexthink has a strategy for compacting the names of those domains that share a common root, grouping them under a single name. On the other hand, internal domains are never compacted. In turn, domains that match the pattern defined in a web-based service are compacted only up to the point that the specified filters allow it. For instance:

- If a web-based service has a filter on domains ***.example.com**:
  A web request to **mail.example.com** is compacted to ***.example.com**.
- However, if an additional web-based service specifies the filter on domains **mail.*.com**:
  The domain **mail.example.com** is not compacted, as it must match both filters.

By default, Nexthink compacts domains when the domain name consists of more than five subdomain levels, or when the third or lower levels are repetitive (names with indexes) or automatically generated (random letters and digits). In those cases, the lower subdomains are replaced by the asterisk sign *. See the

table below for a few examples of compacted domains and a last example of a domain that is not compacted:

Domain name	Stored domain
`exceed.just.five.domain.levels.com` `again.exceed.just.five.domain.levels.com`	`*.just.five.domain.levels.com`

`dev01.cloud.example.com`
`dev02.cloud.example.com`
`8d271d.cloud.example.com`
	`*.cloud.example.com`

`svn.cloud.example.com`	`svn.cloud.example.com`

Note that compacted domains and FQDNs belonging to a same higher level domain can coexist in Nexthink. For instance, in the table above, both the compacted `*.cloud.example.com` and the FQDN `svn.cloud.example.com` are subdomains of `cloud.example.com`, but they are stored as separate domains in the Nexthink database. Thus, the asterisk does not refer to all the subdomains inside `cloud.example.com`, but only to those which are repetitive or randomly generated.

You can also specify a more aggressive compaction method that applies to all domains and not only to those complying with the pre-requisites above. The compaction in this case is made according to a vendor independent public list of domain suffixes, used to determine the highest level at which a domain can be registered. In this way, all domains are compacted up to the level that includes the name of the organization that registered the domain. From the Web Console, configure the compaction policy for domain names in the Engine.

Domain category

The following categories exist:

General:

- Business application
- Search engine and portals
- Information technology
- Social
- News and information
- Advertisement and marketing
- Internal
- Other

Communication:

- VoIP [beware of special capitalization]
- Instant messaging
- Email

High bandwidth:

- Network storage
- Peer-to-peer
- Video, image and sound

Potentially unwanted:

- Games
- Proxy avoidance and hacking
- Spam
- Freeware and software download
- Malicious

Related tasks

- Specifying your internal networks and domains
- Establishing a data retention policy in the Engine

Related concepts

- Service

Related references

- Public suffix list

Entity

An *entity* is a logical grouping of devices which are all reporting to the same Engine.

Entities are the basic building blocks of hierarchies. In a hierarchy tree, the entities are the leaf nodes.

The Service view of the Finder uses entities to provide a breakdown of the devices and the Engine computes some service-related errors and warnings at the entity level.

Related tasks

- Hierarchizing your infrastructure

Related concepts

- Hierarchy

Related references

- Service errors and warnings

Event

An *event* is the basic unit of information that the Engine stores about meaningful occurrences within your IT infrastructure. All objects and activities in Nexthink are linked to one or more events. When an object or activity is no longer related to any event, it is eventually removed from the system. Events are timestamped and ordered sequentially using both the clock of the Engine and the local time reported by the Collectors.

In the Finder, within the context of investigations, *events* refer only to those basic events that represent errors or warnings from devices or applications. The Device view of the Finder also displays the occurrences of these events in the **Errors** and **Warnings** timelines.

Related concepts

- Object
- Activity

Related references

- Errors and warnings for devices and executions
- Warnings tooltips
- Errors tooltips

Executable

An *executable* is an object that encompasses all the binaries that refer to the same program. For instance, the executable `nxfinder.exe` comprises all the binary files with that name that refer to the different versions of the Nexthink Finder.

Applies to platforms:

Related concepts

- Binary
- Application
- Object

Execution

An *execution* is an activity that indicates the loading of a binary into the memory of a computer to run it as a separate process.

If two (or more) executions of the same binary are separated by less than a few minutes, the Engine aggregates them into a single execution. The duration of the aggregated execution spans from the start of the first execution to the end of the last execution. The *cardinality* of the aggregated execution equals to the number of different independent executions.

This mechanism naturally groups repeated or parallel executions of the same program, helping the Engine save space for data retention.

Related concepts

- Activity
- Binary
- Executable

Hierarchy

A *hierarchy* is a way to organize your devices into a structured set of levels.

Related tasks

- Hierarchizing your infrastructure

Related concepts

- Category
- Entity

Installation

An *installation* is an activity that makes a software package available for use in a device. Nexthink detects both the installation and removal of packages in the devices of end-users.

Related concepts

- Activity
- Package

Investigation

An *investigation* is a construct to query the Nexthink database. You can run investigations on objects, activities, or events.

Related tasks

- Creating an investigation

Related concepts

- Object
- Activity
- Event

Keyword

A *keyword* or *tag* is a label to differentiate objects of the same type according to a category. For example, the keywords of the predefined category **Application**

type help you distinguish among different types of applications by identifying them as **browser**, **mail client** or **antivirus**.

The process of assigning a keyword to an object is called *tagging*. An object can be tagged with at most one keyword per suitable category. Tagging can be either manual or automatic.

Related concepts

- Category

Metric

A *metric* is a quantifiable measure of a part or a feature of your IT infrastructure in which you are particularly interested. Metrics offer you a way to define key indicators out of selected IT objects, their properties and their activities.

Define metrics in the Finder in a similar way to how you define investigations and follow their evolution over time from the Portal. Metrics are computed daily during the nightly data collection of the Portal.

Depending on what you want to measure, choose among three types of metrics:

Count metric
: A count metric measures the number of objects that fulfil a custom set of conditons. Count metrics optionally include a ratio between the number of counted objects (the value of the metric itself) and the number of objects that fulfil a second set of conditions, usually more relaxed than the conditions defining the metric.

Quantity metric
: A quantity metric groups the individual values of a particular numeric quantity (aggregate, score, etc.) of devices or users and computes a single value out of them. Depending on the quantity, choose among getting the average, the overall sum, the maximum, or the minimum of all the individual values of the aggregate. Note that quantity metrics are only available for objects of types *device* and *user.*

Top metric
: A top metric holds an ordered list of objects that have the highest or lowest quantity of an aggregated value. Specify the number of elements in the list and the way to compute the aggregated value. Compute either the average, the sum, the maximum, or the minimum of the individual aggregate values for each object, depending on the type of aggregate.

Related tasks

- Creating a metric

Module

Depending on the context, a *module* can refer to:

- One of the parts of Nexthink that can be purchased independently from the core product, adding extra capabilities to it.
- A container of Portal dashboards.

Related concepts

- Dashboard
- Widget

Object

An *object* is a representation of an identifiable element in your IT infrastructure whose properties, activities and generated events are monitored by Nexthink. An object can refer either to a hardware or to a software element.

There are ten types of objects in Nexthink. The availability of each type of object and the information that they carry depend on the platform of the end-user devices that originated them. Object types that are available for multiple platforms can be divided into *specific* to a platform and *shared* by platforms. Objects which are specific to a platform belong to one platform only. For instance, devices are platform specific because one device may be either a Windows device, a Mac OS device, or a Mobile device. On the other hand, users are platform shared, because the same user may have been seen in Windows, Mac OS, or Mobile.

Available for platforms	Object type	Platform specificity
	User	Shared
	Device	Specific
	Package	Specific
	Application	Specific

	Executable	Specific
	Binary	Specific
	Port	Shared
	Destination	Shared
	Domain	Shared
	Printer	Shared

Related concepts

- Activity
- Event

Package

A *package* is an object that represents a software product in its distributable form. Packages can be installed on a device or uninstalled from a device.

Applies to platforms:

Nexthink classifies packages into *program* packages and *update* packages:

Program
: A program package holds a complete software product ready for installation.

Update
: An update package, also called a *patch*, holds bug fixes, security fixes and any other kind of improvements for a program package.

In the Windows operating system, the dialog **Add** / **Remove Programs** in the Control Panel shows the list of packages installed in the computer.

Packages should not be confused with applications. Whereas packages represent a software product in its installable form, applications represent a software product in its executable form. Accordingly, Nexthink detects the presence of a package when it is installed and acknowledges the presence of an application when it is executed; that is, when one of the executable files that make up the application is run.

The link between an application and a corresponding package cannot be always established, so you cannot navigate from one to the other in the Finder. Still, the link is sometimes known for individual executables of the application. If this is the

case, you can find the name of the package in the field **packages** of the executable. One executable can in fact be linked to several packages, yet only the names of the packages are given and drilling-down from executables to packages is not allowed either.

Related concepts

- Installation
- Application
- Executable
- Object

Platform

The *platform* of a device is linked to the way Nexthink extracts information from it, either using the Windows Collector, the Mac OS Collector or the Mobile Bridge. The platform relates thus to the underlying hardware or operating system family of the device. Nexthink supports three types of platforms:

- Windows platform
- Mac OS platform
- Mobile platform

Investigations may apply to one or more platforms. When multiple platforms are selected in an investigation, only the fields that are common to all of the selected platforms are available in the results of the investigation.

Besides devices, other objects may depend on the platform of the device that generated them.

Related concepts

- Object
- Device

Port

A *port* is a communication resource identified by a transport protocol, TCP or UDP, and a number between 1 and 65 535. Ports are associated to connections. A device connects to a destination through a particular port.

Applies to platforms:

In reality, two ports are needed to establish a connection: one port in the device and another port in the destination. However, only the port of the destination is important, because it is the one that usually determines the network service. Server applications on destinations listen to connection requests on well-known ports that are associated to standardized network services. For example, the SMTP mail service uses TCP port 25 for incoming requests. For this reason, Nexthink stores information about the destination ports only.

The type of the port is determined by the transport protocol. There are two types of ports:

- TCP
- UDP

In addition to TCP and UDP ports, there are two other special types of ports in Nexthink which are associated to connections identified as port scans:

- TCP port scan
- UDP port scan

These special types of ports are not linked to just one but to multiple port numbers. To reflect this particularity, Nexthink assigns to these ports the port number 0. which is actually a reserved port number that is never used.

Related concepts

- Connection
- Device
- Destination
- Object

Related references

- Network and port scan conditions

Print job

A *print job* is an activity that represents a unit of work sent to a printer. A single print job may consist of one or several files to be printed. A print job puts in relationship a user and a device with a printer.

Related references

- Information on printers and printing

Related concepts

- Printer
- Activity

Printer

A *printer* is an object representing a computer peripheral that is capable of printing text or pictures on a physical support, typically paper.

Applies to platforms:

Related references

- Information on printers and printing

Related concepts

- Print job
- Object

Score

A *score* is a numerical value whose purpose is to offer you a high level view on the status of either a device or a user.

A score groups a coherent set of lower level analytics according to a particular concern and summarizes them into a single value.

Related tasks

- Computing scores

Related concepts

- Device
- User

Service

A *service* represents an IT service in your organization, such as the mail service or the directory service. Nexthink lets you measure the quality of your IT services as it is perceived by the end-users.

You are however not limited to monitor well-known IT services like mail. Rather, with Nexthink you can define the services that you want to monitor by specifying the resources that they need to operate. These resources characterize and identify each service. In this way, you can monitor any service that matches your own definitions. Because you see the connectivity of end-users, how they actually use the service and who are impacted when the service is malfunctioning, the distinctive user-centered approach of Nexthink provides an advantage over other server-centered solutions.

Services in Nexthink are divided into *connection-based services* and *web-based services*:

Connection-based services
: Monitor connections at the transport level (TCP). Connection-based services are simply known as services.
Applies to platforms:

Web-based services
: Monitor web requests (HTTP/TLS) and responses (HTTP) at the application level, letting you drill-down to their underlying connections as well.
Applies to platforms:

Note that the monitoring of web-based services is only available for Windows devices.

Connection-based services

Any IT service that requires TCP networking for its operation is suitable to be modelled as a service in Nexthink. With Nexthink services, you can supervise the state of your deployed IT services at a glance. If you wish to examine the data in depth, you can drill down through a service and get detailed information about:

- Network traffic associated to the Service.
- Load supported by each server.
- Connectivity of client computers to the Service.
- Crashes of applications related to the Service.
- Users impacted in case of Service failure.
- Performance of the Service in general.

An IT service can be characterized by the resources that are required to access it: the client applications that may be needed to access the service, the network ports that may be reserved to connect to the service or the servers in an organization that may be dedicated to provide the service. In Nexthink, you define a Service precisely by combining one or more of these resources:

- Device
- Executable
- TCP Port
- Destination

Nexthink associates every connection or connection attempt that matches the definition of a Service to it.

Web-based services

In addition to devices, executables, ports and destinations, web-based services are also characterized by domains.

Web based services detect request errors at the application (HTTP) level.

Related tasks

- Analyzing service quality
- Creating a service

Related references

- Service errors and warnings

Related concepts

- Device
- Executable
- Port
- Destination

- Connection
- Domain

System boot

A *system boot* is an activity of a device indicating that the device has been switched on.

Related concepts

- Activity

Related references

- Boot and logon duration

User

A *user* is an object that represents an individual account in a device (local user) or in a group of devices (domain user). The account may identify a physical user or a system user. Physical users, that is, persons who work in front of a device, are also called *end-users*.

Depending on the context, a user may also refer to an operator of the Finder or the Portal. In these cases, context information is usually enough to differentiate between Nexthink users and end-users.

Users across platforms

A user may have access to more than one kind of device. Those devices may, in turn, belong to different platforms. For a user that has access to multiple devices, possibly belonging to different platforms, Nexthink can detect that there is only one single user.

To be able to detect that a user accessing multiple devices is actually the same person, the accounts of the user must be unified in an Active Directory. For instance, to detect that a Mac user is the same as a Windows or Mobile user, the Mac device of the user must join the Active Directory.

Applies to platforms:

Related concepts

- Device
- Object

User logon

A *user logon* is an activity that takes place when a user authenticates in a device to open a session and gain control over it.

Related concepts

- User

Related references

- Boot and logon duration

Web request

A *web request* is a message sent from a client application to a server using the standard web protocol HTTP or its secure version over TLS.

Nexthink is able to record not only the web requests of web browsers, but of any application that runs on the device of the end-user.

Related concepts

- Activity
- Connection

Widget

A *widget* is a self-contained visual and logical software component to build dashboards in the Portal. Widgets display the results computed for metrics.

Related tasks

- Creating a dashboard

Related concepts

- Module
- Dashboard
- Metric

Related references

- Types of widgets

Search and information display

Search in Finder

Overview

The Finder divides the results of a search in the Start page into two columns:

1. The left-hand side column, entitled **Investigations**, shows **user's investigation** from *My investigations* and *Role-based investigations* and **suggested investigations**. This part is also known as the *smart search*. The display of results is as follows:
    - An icon that indicates the type of object or activity on which the investigation is based.
    - A label *suggested* if the investigation is a suggested investigation.
    - The name of the investigation.
    - The time frame that restrains the results to a particular interval of time.
2. The right-hand side column shows search results based on the name of **objects** (i.e. devices, executables, etc) and **categories**.

Applies to platforms:

Suggested investigations

The Finder will use the typed words to suggest investigations. It will lookup if the words match:

- An object type (e.g. *device*) or an activity type (e.g. *connection*)
- The name of a platform if you want to filter the results depending on the kind of devices (e.g. *windows*).
- A keyword (e.g. *crash*, *performance*).
- A condition on an object type.
- Names of objects.
- Names of services.
- Names of entities.
- The name of a category (e.g. *NXT - Server type*) or one of its keywords (e.g. *Proxy*).
- A timeframe

In order to iteratively reduce the scope of the search, we recommend that you

type the words following the previous order. After the first typed word, the Finder will provide you with search results that you can refine when typing more words. But this is not mandatory, as the Finder does not take words order into account.

Objects, activities and platforms

Find below the list of objects and activities that you can use:

Objects	Activities	Platforms
• users • devices • packages • applications • executables • binaries • ports • destinations • domains • printers	• installations • executions • connections • web requests • print jobs • system boots • user logons	• windows • mac • mobile

For example, search for packages.

Search	Finder suggestions
packages	All packages - full period

Keywords

As an example, you can look for errors and warnings in devices or applications using keywords. For instance, type *errors* in the Search box to get a list of any kind of error. You get the same results if you use synonyms of *error* such as *issue*, *problem* or *failure*.

If you want to be more specific in the kind of errors that you want to know about, you can use any of the following (or a valid synonym):

- system crash
- application crash
- application freeze (not responding)
- high cpu
- high memory

For example, to look for application crashes, just type in *application crash*:

Search	Finder suggestions
application crash	Application crashes - today

A condition on an object type

For example, you can type the name of an existing user and the Finder will show you suggested investigations that use the condition on the user name.

Search	Finder suggestions
user *UserName*	Devices used by user *UserName* - full period

Names of objects

Type in names of objects in your queries to look for a concrete instance of an object. As a Finder user, you may need to have the right privilege level to see the names of some objects (see step 3 of defining the profile of a user). Otherwise, they appear as anonymized in the search tool and you will be unable to search them by name.

As an example, type in the *name* of a device or a user in the Search box. You do not need to type in a full name. The Search fills the list of suggestions with investigations related to the objects with that *name* inside their properties. The Finder highlights the *name* in the list of results.

If the Finder detects that many objects match the *name*, it may infer that the word that you typed in is in fact a fragment of the actual name. In this case, the suggested investigations relate to groups of objects whose properties match the fragment. This is indicated by displaying the asterisk ***** wild card surrounding the *name*.

When you type names in the Search box, you can get a mix of suggested investigations that either match one object exactly or match a group of objects. For each investigation, the Finder may interpret the word as a full name or as a fragment. For example:

Search	Finder suggestions
nxtc	Application matching *nxtc*fg.exe - full period Applications used to access domain **nxtc**

Names of services

Similarly to names of objects, look for names of services in the Search box to get investigations related to a particular service. For instance, if you have a service called *Mail Service*, start typing *mail* and you will get the following results (among others):

Search	Finder suggestions
mail	Applications used for **Mail Service** - today Devices using **Mail Service** - today ...

Names of entities

If you have defined a set of entities for building up your hierarchies, type in the names of your entities in the Search box for the Finder to suggest investigations related to objects in those entities.

Suggested investigations based on categories

Use the **names of categories** to refine suggested investigations. For instance, given a category *RAM* that classifies devices according to the quantity of memory installed, the result of looking for devices with that category is the following:

Search	Finder suggestion
device *RAM*	Devices with *RAM* - full period

Where the name of the category is highlighted in the list of results and preceded by the label icon that identifies it as a category (not shown in the table).

Instead of the name of a category, you can directly use **the name of the keywords of the category**. For instance, let us assume that the keywords of the category RAM are:

- 2GB
- 3GB
- 4GB

You can directly look for devices using one of these keywords, or even combine several keywords, by typing:

Search	Finder suggestion
device *2GB*	Devices with *RAM* set to *2GB* - full period

device *3GB 4GB*	Devices with *RAM* set to *3GB* or *4GB* - full period

Alternatively, you can directly use the name of a category without specifying the type of object and optionally combine it with one of its keywords. In this case, the Finder deduces the type of object to which the category applies:

Search	Finder suggestion
RAM 1GB	Devices with *RAM* set to *1GB* - full period

Timeframe control

Limit the suggestions of the Finder to a particular time interval by specifying a timeframe. Find below the words that you can use to define a timeframe for the suggested investigations:

- Full period: The full time interval stored in the database of the Engine.
- Today: The current day (from 0 hours to the current time).
- Yesterday: The full day before today.
- Last hour: The last 60 minutes (including the current minute)
- Last week: The last seven days (including today).

Platform control for suggestions

If you use one of the platform names in your search, suggestions are adapted to match the available information for that platform. For instance, if you use the keyword **mobile** within a search for devices, the Finder suggests investigations about the access state, access rules and security policy of mobile devices.

Note that platform control in the smart search is only activated if devices of platforms other than Windows are detected inside your installation. If you only have Windows devices, the platform keywords (windows, mac os and mobile) are not recognized as such, but just as normal terms of your search.

Synonyms

To make its use more natural, the Search tool of the Finder has the ability to recognize the singular and plural forms of these words as well as some of their synonyms. In many cases you can use your own words to look for information in the Finder and still get the expected results. For instance, instead of looking for *devices*, you can search *computers*, *PCs* or *workstations*.

Once you get used to Nexthink terminology, however, you may find more practical, accurate or even easier to stick to the official terms to designate objects or activities.

Using quotes

When searching, you can use quotes to:

- Force the search on words with less than two letters. Normally, words with less than two letters are ignored by the Finder.

- Force the search to ignore spaces between words and consider the words together. For example, you can search for application with name that contains spaces. Let's say you search for *name of my application* (i.e. a name with spaces):

Search	Finder suggestion
Application *"name of my application"*	Application matching *name of my application*- full period

- Avoid name clashes with reserved words. The quotes instruct the Finder that the content inside is the value of an object name and not the name of a type of an object or activity. For instance, you get different results when you type the word *user* in the Search box with quotes and without quotes:

Search	Finder first suggestion
user	User logons - today
"user"	Devices with package *user* - full period

User's investigation

The Finder will search if the user's investigation contains all the words and if one of the words is the name of an object or an activity type. If this is the case, we will also check if a word matches the object of the conditions.

For example, let's say that the user have a saved investigation named *InvestigationABC* based on devices:

Search	Finder suggestion
device *InvestigationABC*	*InvestigationABC*

Time frame control

By default, the original timeframe is used. But it can be modified, using the "timeframe control" described for suggested investigations. It will apply if the underlying investigation is compatible with it.

Search	Finder suggestion
device *InvestigationABC* today	*InvestigationABC* - today

Platform control for investigations

Using platform keywords in the search makes the Finder suggest only those user investigations that are suitable for all the enumerated platforms.

Using synonyms and quotes

The use of "synonyms" and "quote" described above for suggested investigations is the same for user's investigations.

Show in investigations list

If you want to modify the user's investigation, you can do a right-click and select the option "show in investigations list". Then you can modify the original investigation with a right-click and selecting "edit".

Objects search

Up to now, we have discussed the results that the Search tool displays in the left column of the Start page under the title **Investigations**. This section covers the results of the Search tool that are displayed in the right column of the Start page.

The main use of the right column is to look for a single existing object in the database when you know its name, or at least part of it. In this case, the Finder does not have to deduce anything. It just performs a pure search by matching the terms that you type in with the names of objects or investigations in the database. Results are organized by type of object.

Using quotes will work in the same way as on the left panel. To increase the number of results, you can use wildcards:

*****
: To substitute for zero or more characters

?
: To substitute for zero or one character

The Finder runs the right and left panel search in parallel, so you do not have to choose between either one of them. Using wildcards, however, is not yet

supported by the investigation search, which is likely to show no suggestions at all if you type in an asterisk or a question mark in your search.

Related tasks

- Adding users

Related concepts

- Object
- Activity
- Category
- Entity
- Service

Campaign display compatibility

As new features are added to the end-user feedback module, the campaigns that you create with a particular version of the Finder may not be compatible with previous versions of the Collector.

In the table below, see the display capabilities of the different versions of the Collector for the features introduced in each version of the Finder:

Features / Collector version	Standard (Finder V6.7)	Workflows (Finder V6.8)	Internationalization (Finder V6.9)
Collector V6.9			
Collector V6.8			(Base)
Collector V6.7		(Seq)	(Base)
Collector V6.6 and lower			

(Base)
: The campaign is displayed in the base language only; that is, translations are not available.

(Seq)
: The campaign is displayed only if it follows a sequential workflow; that is, if the campaigns defines no *step to* options.

Related tasks

- Creating a campaign

Real-time and consolidated service data

How service data is computed

Nexthink starts to accumulate information about a service only after you have created it; thus, the service is empty right after its creation. As time passes, Nexthink collects data related to the service and stores the aggregated results in intervals of 10 minutes. The system keeps up to a maximum of six aggregated values that correspond to the six last 10 minutes intervals. After one hour, all the six aggregated values are filled with some data. At this point, the system erases the oldest aggregate and reuses it for a new 10 minutes interval. By rotating the stored results in this way, the system consistently displays the evolution of the service during the last hour with a granularity of 10 minutes.

In a similar way, for longer time intervals, the system stores an hourly aggregated result each time that it crosses an hour boundary, A total of 24 hourly aggregated results per service are stored and rotated, giving a dynamic full day view of the service.

Finally, the system also stores the daily aggregated results of a service when a day boundary is crossed. For each service, there are up to seven daily results stored to build up a full week of service data; although these latter are visible from the Portal only, not from the Finder.

The labels of the two rows that display the metrics in the Finder change thus with time according to the mechanisms for aggregating results described above. Both rows start with the label **Last 10 minutes**. After ten minutes have passed, both labels successively display **Last 20 minutes**, **Last 30 minutes**, etc until an hour boundary is crossed. At this point, the short-term row still increases every 10 minutes, while the long-term row label displays **Last 1 hour**, and increases to **Last 2 hours**, **Last 3 hours**, etc after every hour. When all results are available for the last 24 hours, the short-term row label displays **Last 60 minutes** and the long-term row label displays **Last 24 hours**. The labels no longer change after that point.

Aggregated results per service

Quantity	Interval
6 results	10 minutes
24 results	1 hour
7 results	1 day

Discrepancies between the real-time view and the consolidated view of services

In some situations, it is possible that you find small discrepancies between the real-time values of a service (the Service View in the Finder or a service widget in the Portal) and the values that you get when you drill-down from the real-time view or perform an equivalent investigation. For instance, it may happen that the Service View of the Finder reckons the number of devices using a service to be 10 devices, for a given service and a particular hour of the day; whereas if you drill-down from the Service View to obtain the detailed list of the devices involved, you may get a list with only 9 devices. These differences arise because of the two separate ways in which the Engine collects and organizes service-related events.

On one hand, for the consolidation of events in the long-term, the Engine processes events to extract their timing information. Since events represent end-user information, an event reaches the Engine some time after the Collector has generated it in the device of the end-user. Nevertheless, the Engine can precisely determine the moment in time at which an event really began. This value is the final timestamp of the event.

On the other hand, the real-time views of services require the Engine to react immediately. Thus, the Engine takes into account every new event that matches the definition of a service as soon as the Engine receives the event from a Collector. The Engine aggregates these events during intervals of 10 minutes, 1 hour and 1 day, according to the explantion given above. The Service View displays the values collected within these intervals.

Because of this difference in treatment, if the interval between the start time of a service-related event and the moment of its reception by the Engine crosses a 10 minutes interval boundary (Engine time), you may get discrepancies between the real-time view and the consolidated view of the event. Indeed, if an event is generated short before the end of a 10 minutes interval and the Engine receives it once the interval is over, the Engine properly consolidates the event according to its real start time, but it cannot aggregate the event to the appropriate 10 minutes interval, because that interval is already closed for aggregation. Therefore, the Engine has to report the event in the next 10 minutes interval.

Related tasks

- Analyzing service quality
- Observing service performance
- Following the evolution of a service

Related concepts

- Service

Related references

- Timestamping of events

Service errors and warnings

Nexthink constantly analyzes the state of services and provides information with respect to potential errors or warnings.

Remember that only Windows devices support web-based services.

Connection-based services

Applies to platforms:

	Type	Description
Failed connections	Device-level error	A device is marked in error state if it fails to connect to the destination for 60 seconds.
Application crashes	Device-level error	A device is marked in error state if the binary used to connect to the service experiences an application crash.
Network response time	Entity-level warning	All active devices in an entity are marked in warning state if the average network response time for the entity is 3 times greater than the automatically computed baseline for the previous 7 days.

Web-based services

Applies to platforms:

	Type	Description
Application crashes	Device-level error	A device is marked in error state if the binary used to connect to the service experiences an application crash.
Failed HTTP request (5xx)	Entity-level error	All active devices in an entity are marked in error state if the total number of failed HTTP requests with status 5xx is 3 times greater than the automatically computed baseline for the previous 7 days.
Failed HTTP request (4xx)	Entity-level warning	All active devices in an entity are marked in warning state if the total number of failed HTTP requests with status 4xx is 3 times greater than the automatically computed baseline for the previous 7 days.
Redirected HTTP request (3xx)	Entity-level warning	All active devices in an entity are marked in warning state if the total number of redirected HTTP requests with status 3xx is 3 times greater than the automatically computed baseline for the previous 7 days.
Web request duration	Entity-level warning	All active devices in an entity are marked in warning state if the average web request duration for the entity is 3 times greater than the automatically computed baseline for the previous 7 days.

Computation of averages and detection of outliers

Metrics described in the table above as *Entity-level* errors and warnings are computed for a set of devices instead of individual devices. The goal is to reduce false positives on metrics which are subject to a high degree of variation. For instance, a device might experience a long network response time during a few connections, but this usually does not mean that the service is compromised for this device. By computing such metrics at the entity (or location) level, we can obtain a more accurate representation of the actual quality of service.

For every metric that is computed at the *Entity-level*, there are minimum limits defined for issuing warnings. These are absolute minimum values below which the service quality is guaranteed, even in the case of a baseline violation. For a given metric, if the baseline is very low because the service has been performing extremely well in the past, even in the case that the computed average for the period is 3 times higher than the baseline, a warning is not issued when the average does not exceed the minimum limit.

In addition, an algorithm is put in place to detect and eliminate outliers. If a limited number of devices cause the mean value to exceed the error or warning level, the algorithm removes them from the computation of the baseline. The maximum number of devices that the algorithm can consider as outliers depend on the total number of devices in the Entity:

- 10% of the devices, if the total number of devices in the Entity is less than 100.
- 10 devices, if the total number of devices in the Entity is greater than or equal to 100.

Related concepts

- Service

Errors and warnings for devices and executions

Find below the list of errors and warnings on devices and executions that you can use in your investigations. All warnings and errors apply only to devices and executions on the Windows platform, except for the Hard Reset error, which is also reported by Mac OS devices.

Subject	Warnings	Errors
Device	• High overall CPU usage • High thread CPU usage (deprecated) • High IO usage • High memory usage • High page faults	• System crash • Hard reset ( ) • SMART disk error
Execution	• High thread CPU usage • High memory usage	• Application not responding • Crash

Applies to platforms:
The Device view of the Finder also highlights this kind of events for a particular device in the **Errors** and **Warnings** timelines.

Find the definition of each one of these events in the articles that describe their corresponding tooltips. The Finder displays these tooltips when you hover your mouse over a particular occurrence of the event in the timelines of the Device View or the User View.

Note however that the **High CPU usage** warning has been renamed to **High thread CPU usage**, when applied to executions, and to **High thread CPU usage (deprecated)**, when applied to devices. Indeed, on devices, the warning has

been deprecated in favor of the new **High overall CPU usage** warning. See the explanation in the section below.

Device warnings on CPU usage

There are two warnings generated by the Engine to indicate a high CPU condition on a device:

- **High thread CPU usage (deprecated)**
- **High overall CPU usage**

The first warning, **High thread CPU usage (deprecated)**, is triggered when the CPU load is above 80% in a single logical processor (hardware *thread*) of a device for more than 30 seconds. This threshold has demonstrated to be somewhat low for the high capacity of modern CPUs with multiple cores. For instance, a quad-core device with hyper-threading technology has two hardware threads per core, making a total of 8 logical processors (threads) and a capacity of 800%. Thus, a load of 80% represents only the 10% of the total capacity of the CPU, which is certainly not that significant.

The second warning, **High overall CPU usage**, takes into account the total capacity of the CPU in a device. It is triggered when the CPU load is above 70% over all the logical processors combined for more than 30 seconds. That is, the threshold is 70% over a normalized CPU capacity of 100%. For the same quad-core device as in the example above, that would mean a 560% load over 800% of total capacity, which is indeed a high CPU load.

Aggregates on CPU usage warnings

For the previous warning events, two aggregates let you know the percentage of time that devices are under a high CPU condition. There is also a similar aggregate for applications:

Name	Applies to	Description
High device thread CPU time ratio (deprecated)	• device	Aggregates the duration of the warning **High thread CPU usage (deprecated)** and divides by the uptime of the device.
High device overall CPU time ratio	• device	Aggregates the duration of the warning **High overall CPU usage** and divides by the uptime of the device.

High application thread CPU time ratio	• application • executable • binary	Time that the execution of the application was in high CPU divided by the total execution time.

Typically, the aggregate **High device thread CPU time ratio (deprecated)** displays higher ratios of time spent by devices under a high CPU condition. Those ratios may be unrealistically high for devices with multiple-core CPUs for the same reasons as explained above for its associated warning. Preferably use the aggregate **High device overall CPU time ratio**.

The aggregate **High application thread CPU time ratio** lets you find applications with high CPU consumption or, when applied to binaries, compare the different versions of a same application and see which one stays longer in a high CPU condition.

Related tasks

- Changing the thresholds of High CPU warnings

Related concepts

- Event

Related references

- Warnings tooltips
- Errors tooltips
- Memory and CPU usage

Types of widgets

Overview

Learn here about the different types of widgets that you can use on your dashboards for visually displaying data related to the computation of metrics. Use these types of widgets on dashboards within Basic modules, where you can choose the visualizations of your metrics individually.

You find similar widgets on dashboards within Service Monitoring modules, but the types of widgets and the layout are fixed on these dashboards. On their side, Software Metering modules display only one kind of widget on their dashboards, which is completely different from the widgets on the dashboards of both Basic

and Service Monitoring modules.

In this article, we focus on the types of widgets for Basic modules, whose purpose is to display the values of metrics.

KPI

A *KPI* (Key Performance Indicator) widget is mainly a single numerical figure that reflects the inner value of the represented metric (or metrics).

The figure in a KPI widget is either an absolute or a relative number (a percentage). It can be a percentage only if the definition of the associated metric includes a ratio computation; that is, a comparison between the value of the metric itself and the value returned by an additionally supplied investigation (usually a less constrained version of the investigation that computes the value of the metric).

In addition to the main figure, a KPI widget may also include a secondary figure that displays the variation in the value of the metric with respect to its previous value; where the meaning of *previous* depends on the time frame selected in the dashboard. Moreover, when specified in the definition the associated metric, a KPI widget indicates whether the increase or a decrease in the value of the metric is good or bad by coloring the arrow next to the variation figure in green or in red, respectively. If you decided that a variation in the value of the metric is not necessarily good or bad, the arrow is painted blue.

If the definition of a metric includes thresholds that limit the ranges of values that can be considered normal, worrying (optional), or bad, you can configure the KPI widget to display threshold information in the form of a colored dot that precedes the main KPI figure. The dot is green if the value lies within the normal range, yellow if in the worrying range (which exists only if you define two thresholds), and red if in the bad range.

The following example shows a KPI widget related to a metric that counts the number of devices with CPU issues. The definition of the associated metric includes a ratio (devices with CPU issues compared to the total number of devices) which is the main figure in the widget. An increase in the value of the metric is obviously bad (it implies more devices with CPU issues), so the increase is displayed with a red arrow. Two thresholds are defined: when up to 10% of the total number of devices have CPU issues, the situation is considered normal; from 10% to 20% the situation becomes worrying; more than 20% devices with CPU issues is considered bad.

A single KPI widget can display the values of more than one metric. Arrange the individual visualizations of each metric vertically or horizontally within the KPI widget.

The KPI widget is compatible with count and quantity metrics. Top metrics are not suitable for being displayed as a single figure.

Table

A *table* widget arranges the value of a metric (or metrics) in a grid. Add a table widget to your dashboards to display either a top metric or a group of count and quantity metrics.

Displaying a top metric

When displaying a top metric in a table widget, the table shows the list of top objects as rows, while the columns are the display fields chosen in the configuration of the top metric. You can limit the number of fields displayed in the configuration of the table widget: you are allowed to turn off only those which are not essential to the definition of the metric.

A single table widget may display at most one top metric, which takes the whole of the widget. Therefore, it is neither possible to combine a top metric with another top metric nor with any other kind of metric in the same table widget.

Count and quantity metrics

The display of count and quantity metrics in a table widget is very flexible. Add up to 50 metrics of these types to a single table widget. For each metric, choose to

display either the value of the metric itself or a computed ratio value (when specified in the definition of the metric). If the metric defines thresholds, choose among displaying the status (a green, yellow, or red dot) next to the value, the status alone, or not to display threshold information at all and display only the value. You can also choose to display the variation of the metric with respect to its previous value. These choices are intendedly very similar to those that you can find in the KPI widget.

Arrange the values into the rows and columns of the table according to the hierarchy, the metric names, or the grouping criteria of the metrics that you have added to the widget (if specified in the definition of the metrics). Beware that, depending on the particular metrics that you choose, not all combinations are allowed.

Line chart

A *line chart* graphically displays the historical evolution of the value of a metric (or metrics) with time. Line charts let you visually find out significant events in the history of metrics, compare values of different metrics along time and discover trends.

Add up to 5 metrics to a single line chart. All the metrics in the same line chart must be expressed in the same units, where the word *unit* must be understood in a broad sense. Note, for instance, that you can add count metrics of different objects, such as devices and binaries, to the same line chart, because the units are compatible (they are always a number of objects). You can even mix count metrics with those quantity metrics that measure a number of events. However, you cannot mix a metric that counts devices with a metric that measures the average boot time of devices, since a number of devices is not compatible with time units.

Make each line in the chart represent either the value of a metric or a computed ratio (when specified in the definition of the metric). When a metric defines thresholds, optionally display them in the line chart as horizontal lines. If you defined just one threshold, the chart displays a horizontal red line. When two thresholds are defined, the chart displays both a yellow and a red horizontal lines at the level specified by the thresholds. Points of the line above the thresholds are painted with the corresponding color. Even when you decide to show the thresholds, they may not be visible as horizontal lines in the chart if all the values of the metric are under the specified limits. Besides, displaying thresholds is only available when you display just one metric in the chart, since having thresholds in the same line chart for more than one metric would be too confusing.

Line charts do not span to the time frame selected in a dashboard, but further to the past. When you select a time frame for a dashboard (day, week, month, or quarter), each point in the line of a line chart represents the value of the metric aggregated for that period. The last point in the line, corresponds to the selected date in the dashboard, while the values to the left are past values. How far the line chart goes into the past depends on the selected time frame and on the amount of available historical data. The minimum span gradually grows into the maximum span as the Portal computes more and more data:

Time frame	Line chart min span	Line chart max span
Day	30 days (1 month)	60 days (2 months)
Week	12 weeks (3 months)	52 weeks (1 year)
Month	12 months (1 year)	24 months (2 years)
Quarter	8 quarters (2 years)	16 quarters (4 years)

The line chart widget is compatible with count and quantity metrics, but it is not adapted to display top metrics.

Bar chart

A *bar chart* graphically displays the values of count or quantity metrics in horizontal bars, letting you compare results visually with just a glimpse of the eye.

Depending on the metrics displayed on a bar chart, the bars that represent the measured values may be arranged differently:

- If the bar chart displays just one metric, arrange the bars either:
    - By the grouping criterion selected in the definition of the count or quantity metric.
    - By hierarchy. If you chose two grouping criteria (or none at all) when creating the metric, this is the only method available for arranging the bars.
- If the bar chart displays more than one metric, bars are arranged necessarily by metric.

Sort the results in the bar chart either by value or by name (the names of the metrics, the nodes in the hierarchy or the labels of the *group by* option). To see the most important values first, it is usually recommended to sort the bar chart by value in descending order as best practice.

If the bar chart holds a count metric, choose between displaying its value or one of the two following ratios:

Ratio defined in the metric

Because the ratio defined in the metric compares arbitrary groups (those determined by the conditions in both the metric and ratio definitions), the value of each bar is independently calculated and the sum of all the bars in the widget does not add up to 100% in general. This option is only valid if the count metric actually defines a ratio.

Ratio of total

In bar charts, you are seldom interested in displaying the ratio defined in the metric (if any), but rather the distribution of each grouping option with respect to the total number of objects seen in the widget. The bars in a widget that displays the ratio of total do add up to 100% and the related count metric does not need to define a ratio for this option to be valid.

If the bar chart displays a metric that defines thresholds, tick the box **Threshold** for the chart to paint the bars in a specific color depending on the value of each bar crossing the threshold or not: green, yellow (only for metrics with two thresholds), or red. If the metric does not define any threshold or you do not tick the **Threshold** box, bars are depicted in blue. If multiple metrics are displayed in the same bar chart, each one will be colored independently.

Choose the minimum number of bars that you wish to see in the chart. More bars may be visible if space in the dashboard allows it. Otherwise, when more bars than those visible are available, a small down arrow appears at the bottom right corner of the chart. Hover the mouse cursor over the bar chart and the small arrow turns into a slider. Use the slider to scroll through all the values in the bar chart. If you leave the bar chart after scrolling, small arrows appear again to

indicate the availability of more results either to the top or to the bottom of the chart, or both, when applicable.

The width of the bar chart is automatically scaled to accommodate enough space for the highest value to display. For bar charts with only one metric, it is possible to fix the scale to a maximum value that is meaningful for you. in the configuration of the bar chart, choose between **Automatic** or **Fixed**. In the latter case, specify the maximum value that the chart should show. If this maximum fixed value is exceeded by a bar in the chart, the widget is automatically redimensioned to fit the new maximum.

Title widget

A *title widget* lets you group several other widgets under the same title. The title widget does not hold metrics by itself, but acts as an umbrella for other widgets that do display metric results.

Title widgets are particularly useful for organizing widgets of different types into logical sets, offering you more flexibility in the design of layouts for your dashboards.

Compatibility matrix

Not every type of widget is suitable for displaying any kind of metric. In the table below, find the compatibility matrix among the types of metrics and the widgets to display them.

Metric Type	KPI	Table	Line chart	Bar chart
Count metric	Ok	Ok	Same units	Same units
Quantity metric	Ok	Ok	Same units	Same units
Top metric	-	Only one	-	-

Same units
: If you add more than one metric to the chart, the metrics must share the same units.

Only one
: The widget accepts at most one metric.

Related tasks

- Creating a metric

Widget compute state in charts

Overview

For line chart widgets, the state of the computation of the associated metric for a particular date is directly available from the chart. When the metric is successfully computed, the Portal displays a solid line in the line chart.

When the computation of the metric is consistently successful for several days, the solid line joins the points that represent the value of the metric along the whole selected period. Hovering the mouse cursor over the chart displays, the value of the computation on the selected date:

If a metric defines threshold values, make them visible in the line chart by ticking the option **Thresholds** when configuring the widget. The lower threshold is depicted as a horizontal orange line and the higher threshold as a horizontal red line. Likewise, value points in the chart are highlighted in orange when they exceed the first threshold value or in red when they exceed the second threshold value:

However, the Portal is not always able to compute the metric of a widget on a particular date. When this happens, the chart of the widget replaces the solid blue line by a dashed line or by empty space to indicate that the computation could not be carried out. Hovering over a non computed area of a chart with the mouse displays a message that helps you identify the cause for the lack of data.

We list below the reasons why there might be a lack of data in the chart of a widget and we illustrate them with example figures from the Portal. We finally discuss the

Line charts

First computation of the widget

Before the first computation of a widget, there is no data available for it. Therefore, the chart does not display any line before the first computation of the widget.

If you hover the mouse over the empty area, the chart displays the date, the name of the metric and the message **no data computed**:

Not computed

A widget can miss some days of computation because the Portal or the Engine were stopped or because the connection between the Portal and the Engine was lost for some time.

If the computation was not run on one or more days, the chart displays a dashed straight red line between the two dates where there is actually some data. Additionally, when you hover the mouse over it, the chart displays the message **no data computed**.

No activity

The widget was not computed because there was not data available for it. The connection with the Engine is fine in this case, but the Engine just does not provide proper data for the widget.

This occurs with activity widgets and service widgets when the type of activity that they monitor did not take place over the requested period of time and providing a zero value does not make sense. For example, a widget that computes the average response time of an application will show no activity if the application was not executed over the specified period. Showing zero response time for the application is not a valid option in this case.

This situation can also happen with inventory and issue widgets that refer to objects grouped by a specific keyword. If you change the group by property of the widget and you do not recompute it for the past dates, the data for those dates will be lost showing no activity.

The chart displays a dashed blue line at the level of the axis and the message **no activity** when hovering with the mouse.

Other widgets

KPI, table, and bar char widgets represent the absence of computed metric data by a dash sign (**-**) in the place of the value.

All widgets

One day only

Because of an undefined aggregation strategy, the computation of some metrics does not make sense for periods longer than one day. When selecting a period longer than one day in the Portal, the widgets related to these metrics show the following message:

Configuration issues

If a widget has a configuration error, e.g. because the associated metric has been deleted, the Portal displays the following message in the middle of the dashboard area assigned to the widget:

Related references

- Types of widgets
- Widget

Tooltips in the user and device views

Alerts tooltips

Single alert occurrence

The tooltip displays the name of the alert and its duration. The time range indicates the exact period when the alert was active.

Multiple alert occurrences

When an alert is triggered multiple times within the selected period, this tooltip shows the start time of each one of the occurrences and their duration. The time range above shows the interval when the two alerts were active, with the granularity of the units of the timeline. Thus, the interval displayed depends on the zoom of the device view.

Warnings tooltips

Applies to platforms:

High CPU usage

High device CPU usage

The tooltip displays the CPU usage on the device during a time interval, when the overall CPU usage is above 70%. The tooltip appears in two forms:

- When multiple applications are the cause of the high CPU load in the device and none is specially responsible for it, then the tooltip only gives a figure of the total CPU usage.
- When a few particular applications have a significant impact in the CPU load, the tooltip details these applications and their CPU usage per thread (this is different from the overall CPU usage, see explanation in the tooltip below), in addition to the overall CPU usage in the device.

The Collector takes a CPU load sample every 30 seconds and reports them every 5 minutes. To report high CPU usage, two or more consecutive samples must exceed 70% of CPU usage. The reported value corresponds to the average value of the samples above 70%.

High application CPU usage

The tooltip displays a list of the processes (and their corresponding application) that consumed more than 50% of the CPU processing power per thread (40% for the system process) for at least 30 seconds during the selected time interval. Processes are ordered in the list by their contribution to the CPU load from the most to the least demanding.

A 100% usage corresponds to the capacity of a single-core single-threaded CPU. You can get more than 100% usage in CPUs with multi-core or hyper-threading technologies:

- For each additional physical core in the CPU you get 100% more processing power. That is, the capacity of a dual-core CPU is 200%, for a quad-core is 400%, for an hex-core is 600%, and so on.
- If the CPU uses hyper-threading technology, the processing power of each physical core is doubled. A hyper-threaded dual-core CPU yields thus 400% capacity, a quad-core 800%, a hex-core 1200%, and so forth.

Note that this way of measuring is different from that of high CPU on the device, where the overall capacity of a CPU is 100%, regardless of the number of cores and threads in the CPU. For this reason, the bottom of the tooltip displays the capacity of the CPU in terms of hardware threads, which can be (and it usually is) bigger than 100%.

High memory

The tooltip displays the usage of physical memory when more than 70% of the total physical memory is in use for at least 5 minutes.

In its lower part, the tooltip lists the top five processes that consume the most memory. Only processes that take more than 1% of the total memory are listed.

High IO throughput

The warning shows up when IO operations exchange information at a rate higher than 20 MB/s during at least 30 seconds.

The total IO throughput includes all the bytes transferred during read, write or any other kind of IO operation performed on either real or virtual peripherals, such as hard disks, flash drives, network controllers, keyboards, mice, etc.

High page faults

The warning appears when the device generates more than 5000 page faults per second in memory accesses during at least 5 minutes.

Related concepts

- Event

Related references

- Errors and warnings for devices and executions
- Errors tooltips
- Memory and CPU usage

Errors tooltips

Application crash

Applies to platforms:

The tooltip displays the name of the executable that crashed, along with its version and the name of the application to which it belongs. In the case of a single crash, the time in the header of the tooltip is the exact time when the crash was reported.

In the case of several almost simultaneous crashes grouped in the same tooltip, the time in the header of the tooltip displays the interval during which all crashes took place. Each application listed in the tooltip is preceded by its own precise time of crashing.

Application not responding

Applies to platforms:

The tooltip for non responding applications displays the same information as the tooltip for application crashes, but for applications that hang or freeze instead of for applications that exit unexpectedly.

System crash

Applies to platforms:

The tooltip displays the stop error code and a brief textual description of the error that caused the device to crash. These are serious hardware or software errors that make your computer halt unexpectedly. For that reason, the time in the header of the tooltip cannot indicate the moment when the error occurred. Instead, it is the time of the first boot after the system crash.

Hard reset

Applies to platforms:

The tooltip indicates that the device was abruptly stopped and then rebooted. Pressing the reset button, power failures or crashes may be the cause of a hard reset. As in the case of the blue screen tooltip, the time reported here is the time of the first boot of the device after the hard reset.

SMART disk error

Applies to platforms:

The tooltip signals a disk error detected in a disk drive equipped with the SMART technology. It indicates an increase in the number of disk writing errors or in the

count of reallocated sectors.

Related concepts

- Event

Related references

- Errors and warnings for devices and executions
- Warnings tooltips

Activity tooltips

Executions

Standard

The tooltip shows the number of executions during the selected time interval. All the applications executed by user accounts were run at the user privilege level. Note that executions carried out by system accounts, which usually operate at the administrator privilege level, may also contribute to the count.

Privilege warning

The tooltip displays the number of executions during the selected time interval with an additional privilege warning. The warning indicates that at least one of the executions was carried out by a user account with power user or administrator privilege levels.

Connections

The connections tooltip displays the overall amount of traffic measured during the

selected time interval. This includes the TCP and UDP traffic that the device sent out and the TCP traffic that the device received.

User interaction

The tooltip displays the name of the user that interacted with the device along with the total duration of the interaction (in parenthesis). The maximum duration is limited by the selected time interval, which is indicated in the header of the tooltip. User interaction is detected as mouse or keyboard activity of the user. The user is considered inactive if the device does not receive any mouse or keyboard event for 15 minutes.

The tooltip can only be displayed if the monitoring of user interaction is enabled.

Mobile synchronization

At least one successful synchronization with the Exchange ActiveSync server has been detected within a one-hour window.

If **push notifications** are set-up, the device will **automatically synchronize** when a new event arrives on the server (email, calendar update, etc?). However, the user can also **manually synchronize** their device at any time, or disable push notifications and **schedule the synchronization** at a regular interval.

Services tooltips

Connection-based services

The tooltip displays summary information on the performance of the connection-based service during the selected time interval. If any of the collected statistics exceeds its normal range, the tooltip may display a warning message as well.

Web-based services

The tooltip displays summary information on the performance of the web-based service during the selected time interval. If any of the collected statistics exceeds its normal range, the tooltip may display a warning message as well.

Database information and organization

Data model

This reference article contains the complete description of Nexthink's data model.

Objects

Objects represent items recognized by Nexthink.

User

Users of devices (domain, local or system)

Field	Group	Type			
Activity start time	Activity	Aggregate			
	Start time of investigated activity				
	NXQL ID:	activity_start_time			
Activity stop time	Activity	Aggregate			
	Stop time of investigated activity				
	NXQL ID:	activity_stop_time			
Application crash ratio	Errors	Aggregate			
	Indicates the number of application crashes per 100 executions.				
	NXQL ID:	application_crash_ratio			
Application not responding event ratio	Errors	Aggregate			
	Indicates the number of application not responding events per 100 executions.				
	NXQL ID:	application_not_responding_event_ratio			
Average incoming network bitrate	Availability	Aggregate			
	Average incoming network bitrate				
	NXQL ID:	average_incoming_bitrate			
Average incoming web bitrate	Availability	Aggregate			
	Average incoming bitrate of all underlying web requests, consolidated over time				

	NXQL ID:	average_incoming_bitrate			
Average memory usage per execution	Activity	Aggregate			
	Indicates the average memory usage of all underlying executions before aggregation. The value is the average memory usage of all executions (calculated with a 5-minute resolution) multiplied by their cardinalities and divided by the total cardinality. • Example: if two tabs of the Chrome browser are opened at the same time, two distinct processes of chrome.exe are launched and they are aggregated by the Engine (i.e., event cardinality = 2). The average memory usage will be the average of the two processes before aggregation: it represents the average memory usage of a Chrome tab.				
	NXQL ID:	average_memory_usage_per_execution			
Average network response time	Availability	Aggregate			
	Indicates the average TCP connection establishment time of all underlying connections. The value is the average TCP connection establishment time of all executions weighted by their cardinality.				
	NXQL ID:	average_network_response_time			
Average outgoing network bitrate	Availability	Aggregate			
	Average outgoing network bitrate				
	NXQL ID:	average_outgoing_bitrate			
Average outgoing web bitrate	Availability	Aggregate			
	Average outgoing bitrate of all underlying web requests, consolidated over time				
	NXQL ID:	average_outgoing_bitrate			
Average web request duration	Availability	Aggregate			
	Average time between request and last response byte				
	NXQL ID:	average_request_duration			
Average web request size	Traffic	Aggregate			
	Average size of web requests				
	NXQL ID:	average_request_size			
Average web response size	Traffic	Aggregate			

	Average size of web responses				
	NXQL ID:	average_response_size			
Binary paths	Activity	Aggregate			
	List of executed binary paths (max. 50 paths)				
CPU usage ratio	Activity	Aggregate			
	Indicates the sum of the CPU time of all executions on each device in scope over all logical processors divided by their total duration. • Example: if we consider two executions with the first one taking 50% of a logical processor during 30 minutes and the second one taking 100% of 2 logical processors during 60 minutes, the CPU usage ratio is 150% (= [50% * 30 min + 2 * 100% * 60 min] / [30 min + 60 min]).				
	NXQL ID:	cpu_usage_ratio			
Cumulated execution duration	Activity	Aggregate			
	Cumulated duration of executions				
	NXQL ID:	cumulated_execution_duration			
Cumulated network connection duration	Activity	Aggregate			
	Cumulated duration of TCP connections				
	NXQL ID:	cumulated_connection_duration			
Department	Properties	Field			
	User department as listed in Active Directory				
	NXQL ID:	department			
Distinguished name	Properties	Field			
	Active Directory distinguished name (DN)				
	NXQL ID:	distinguished_name			
First seen	Properties	Field			
	First time activity of the user was recorded on any device				
	NXQL ID:	first_seen			
Full name	Properties	Field			
	Full user name as listed in Active Directory				
	NXQL ID:	full_name			
Highest local privilege level reached	Activity	Aggregate			

	Highest local privilege level reached for executions (user, power user, administrator)				
	NXQL ID:	highest_local_privilege_reached			
ID	Properties	Field			
	Unique user identifier code				
	NXQL ID:	id			
Incoming network traffic	Traffic	Aggregate			
	Total network incoming traffic				
	NXQL ID:	incoming_traffic			
Incoming web traffic	Traffic	Aggregate			
	Total web incoming traffic				
	NXQL ID:	incoming_traffic			
Job title	Properties	Field			
	Job title as listed in Active Directory				
	NXQL ID:	job_title			
Last seen	Properties	Field			
	Last time activity of the user was recorded on any device				
	NXQL ID:	last_seen			
Lowest observed web protocol version	Activity	Aggregate			
	Lowest protocol version observed in web requests (excluding web requests with unknown protocol version)				
	NXQL ID:	lowest_protocol_version			
Name	Properties	Field			
	User logon name				
	NXQL ID:	name			
Network availability level	Availability	Aggregate			
	Indicates the ratio of successful TCP connections. The possible values are: • high: the ratio is greater or equal to 98% • medium: the ratio is greater or equal to 90% and less than 98% • low: the ratio is lower than 90%				
	NXQL ID:	network_availability_level			
Number of application crashes	Errors	Aggregate			
	Number of application crashes				

	NXQL ID:	number_of_application_crashes
Number of application not responding events	Errors	Aggregate
	Number of application not responding events	
	NXQL ID:	number_of_application_not_responding_events
Number of applications	Inventory	Aggregate
	Number of applications	
	NXQL ID:	number_of_applications
Number of binaries	Inventory	Aggregate
	Number of binaries	
	NXQL ID:	number_of_binaries
Number of connections	Activity	Aggregate
	Number of connections	
	NXQL ID:	number_of_connections
Number of days since last seen	Properties	Field
	Indicates the number of days since the last time the user was seen by Nexthink. The field is updated every hour.	
	NXQL ID:	number_of_days_since_last_seen
Number of destinations	Inventory	Aggregate
	Number of destinations	
	NXQL ID:	number_of_destinations
Number of devices	Inventory	Aggregate
	Number of devices	
	NXQL ID:	number_of_devices
Number of domains	Inventory	Aggregate
	Number of domains	
	NXQL ID:	number_of_domains
Number of executables	Inventory	Aggregate
	Number of executables	
	NXQL ID:	number_of_executables
Number of executions	Activity	Aggregate
	Number of executions	
	NXQL ID:	number_of_executions
Number of ports	Inventory	Aggregate
	Number of ports	
	NXQL ID:	number_of_ports

Number of print jobs	Activity	Aggregate			
	Number of print jobs				
	NXQL ID:	number_of_printouts			
Number of printed pages	Activity	Aggregate			
	Number of printed pages				
	NXQL ID:	number_of_printed_pages			
Number of printers	Inventory	Aggregate			
	Number of printers				
	NXQL ID:	number_of_printers			
Number of web requests	Activity	Aggregate			
	Number of web requests				
	NXQL ID:	number_of_web_requests			
Outgoing network traffic	Traffic	Aggregate			
	Total network outgoing traffic				
	NXQL ID:	outgoing_traffic			
Outgoing web traffic	Traffic	Aggregate			
	Total web outgoing traffic				
	NXQL ID:	outgoing_traffic			
Protocols used in web requests	Activity	Aggregate			
	Protocols used in web requests (HTTP, TLS, HTTP/TLS)				
	NXQL ID:	protocols_used_in_requests			
SID	Properties	Field			
	Indicates the Windows security identifier for the user. • For Mac 0S: the value is 'S-0-0' if the user is not in Active Directory.				
	NXQL ID:	sid			
Successful HTTP requests ratio	Availability	Aggregate			
	Percentage of successful HTTP requests (1xx, 2xx and 3xx)				
	NXQL ID:	successful_http_requests_ratio			
Successful network connections ratio	Availability	Aggregate			
	Percentage of successful TCP connections				
	NXQL ID:	successful_connections_ratio			
Total active days	Activity	Field			
	Total number of days the user was active				

	NXQL ID:	total_active_days			
Total CPU time	Activity	Aggregate			
	Indicates the sum of the CPU time of all executions on each device in scope and over all logical processors. • Example: if we consider two executions with the first one taking 50% of a logical processor during 30 minutes and the second one taking 100% of 2 logical processors during 60 minutes, the total CPU time is 135 minutes (= 50% * 30 min + 2 * 100% * 60 min).				
	NXQL ID:	total_cpu_time			
Total network traffic	Traffic	Aggregate			
	Total network traffic (incoming and outgoing)				
	NXQL ID:	total_network_traffic			
Total web traffic	Traffic	Aggregate			
	Total web traffic (incoming and outgoing)				
	NXQL ID:	total_web_traffic			
Type	Properties	Field			
	Type of user (local/domain/system)				
	NXQL ID:	type			
UID	Properties	Field			
	Indicates the universally unique identifier (based on user SID).				
Web interaction time	Activity	Aggregate			
	Indicates the time during which at least one executable is doing HTTP or TLS traffic. This is counted with a 5-minute resolution.				
	NXQL ID:	cumulated_web_interaction_duration			

Device

Devices are Windows, Mac OS or mobile endpoints

Field	Group	Type			
Access state	Exchange	Field			
	Indicates whether the device can access the Exchange ActiveSync server. The possible states are:				

	• allowed: the device has access • blocked: the device is blocked • discovery: the device is temporarily quarantined while it is being identified by the Exchange ActiveSync server • quarantined: the device is waiting for Exchange ActiveSync administrator approval				
	NXQL ID:	eas_access_state			
Access state reason	Exchange	Field			
	Indicates the reason for the device access state. The possible values are: • global: caused by the global access settings • device rule: caused by a device access rule • individual: caused by an individual exemption • policy: caused by Exchange ActiveSync policy				
	NXQL ID:	eas_access_state_reason			
Activity start time	Activity	Aggregate			
	Start time of investigated activity				
	NXQL ID:	activity_start_time			
Activity stop time	Activity	Aggregate			
	Stop time of investigated activity				
	NXQL ID:	activity_stop_time			
Administrator account status	Policy	Field			
	Determines whether the local Administrator account is enabled or disabled				
	NXQL ID:	administrator_account_status			
All antispyware	Security	Field			
	Summary information about all the detected antispyware: • unknown: indicates that the information could not be retrieved • N/A: this field is not available on this operating system • '-' : no data, incompatible collector version or the data is not yet available				
	NXQL ID:	all_antispywares **Note**: this field is not available for Windows Server operating systems.			
All antiviruses	Security	Field			
	Summary information about all the detected antiviruses:				

	• unknown: indicates that the information could not be retrieved • N/A: this field is not available on this operating system • '-' : no data, incompatible collector version or the data is not yet available				
	NXQL ID:	all_antiviruses **Note**: this field is not available for Windows Server operating systems.			
All firewalls	Security	Field			
	Summary information about all the detected firewalls: • unknown: indicates that the information could not be retrieved • N/A: this field is not available on this operating system • '-' : no data, incompatible collector version or the data is not yet available				
	NXQL ID:	all_firewalls **Note**: this field is not available for Windows Server operating systems.			
Antispyware display name	Security	Field			
	Name of the main antispyware				
	NXQL ID:	antispyware_name **Note**: this field is not available for Windows Server operating systems.			
Antispyware RTP	Security	Field			
	Indicates whether the antispyware real time protection (RTP) is active: • on: indicates that RTP is active • off: indicates that either RTP is not active or no antivirus has been detected • unknown: indicates that the information could not be retrieved • N/A: this field is not available on this operating system • '-' : no data, incompatible collector version or the data is not yet available				
	NXQL ID:	antispyware_rtp			

		Note: this field is not available for Windows Server operating systems.			
Antispyware up-to-date	Security	Field			
	Indicates whether the antispyware is up-to-date: • yes: indicates that antispyware is up-to-date • no: indicates that either the antispyware is not up-to-date or no antispyware has been detected • unknown: indicates that the information could not be retrieved • N/A: this field is not available on this operating system • '-' : no data, incompatible collector version or the data is not yet available				
	NXQL ID:	antispyware_up_to_date **Note**: this field is not available for Windows Server operating systems.			
Antivirus display name	Security	Field			
	Name of the main antivirus				
	NXQL ID:	antivirus_name **Note**: this field is not available for Windows Server operating systems.			
Antivirus RTP	Security	Field			
	Indicates whether the antivirus real time protection (RTP) is active: • on: indicates that RTP is active • off: indicates that either RTP is not active or no antivirus has been detected • unknown: indicates that the information could not be retrieved • N/A: this field is not available on this operating system • '-' : no data, incompatible collector version or the data is not yet available				
	NXQL ID:	antivirus_rtp **Note**: this field is not available for Windows Server operating systems.			
Antivirus up-to-date	Security	Field			
	Indicates whether the antivirus is up-to-date:				

	• yes: indicates that antivirus is up-to-date • no: indicates that either the antivirus is not up-to-date or no antivirus has been detected • unknown: indicates that the information could not be retrieved • N/A: this field is not available on this operating system • '-' : no data, incompatible collector version or the data is not yet available				
	NXQL ID:	antivirus_up_to_date **Note**: this field is not available for Windows Server operating systems.			
Application crash ratio	Errors	Aggregate			
	Indicates the number of application crashes per 100 executions.				
	NXQL ID:	application_crash_ratio			
Application not responding event ratio	Errors	Aggregate			
	Indicates the number of application not responding events per 100 executions.				
	NXQL ID:	application_not_responding_event_ratio			
Audit account logon events	Policy	Field			
	Determines whether to audit each instance of a user logging on to or logging off from another computer in which this computer is used to validate the account				
	NXQL ID:	audit_account_logon_events			
Audit account management	Policy	Field			
	Determines whether to audit each event of account management on a computer				
	NXQL ID:	audit_account_management			
Audit directory service access	Policy	Field			
	Determines whether to audit the event of a user accessing an Active Directory object that has its own system access control list (SACL) specified				
	NXQL ID:	audit_directory_service_access			
Audit logon events	Policy	Field			
	Determines whether to audit each instance of a user logging on to or logging off from a computer				
	NXQL ID:	audit_logon_events			
Audit object access	Policy	Field			
	Determines whether to audit the event of a user accessing an object, e.g. a file, folder, registry key, printer, and so forth-that has its own system access				

	control list (SACL) specified				
	NXQL ID:	audit_object_access			
Audit policy change	Policy	Field			
	Determines whether to audit every incident of a change to user rights assignment policies, audit policies, or trust policies				
	NXQL ID:	audit_policy_change			
Audit privilege use	Policy	Field			
	Determines whether to audit each instance of a user exercising a user right				
	NXQL ID:	audit_privilege_use			
Audit process tracking	Policy	Field			
	Determines whether to audit detailed tracking information for events such as program activation, process exit, handle duplication, and indirect object access				
	NXQL ID:	audit_process_tracking			
Audit system events	Policy	Field			
	Determines whether to audit when a user restarts or shuts down the computer or when an event occurs that affects either the system security or the security log				
	NXQL ID:	audit_system_events			
Average extended logon duration	Startup	Aggregate			
	Indicates the average extended logon duration.				
	NXQL ID:	average_extended_logon_duration			
Average incoming network bitrate	Availability	Aggregate			
	Average incoming network bitrate				
	NXQL ID:	average_incoming_bitrate			
Average incoming web bitrate	Availability	Aggregate			
	Average incoming bitrate of all underlying web requests, consolidated over time				
	NXQL ID:	average_incoming_bitrate			
Average logon duration	Startup	Aggregate			
	Indicates the average logon duration.				
	NXQL ID:	average_logon_duration			
Average memory usage per execution	Activity	Aggregate			
	Indicates the average memory usage of all underlying executions before aggregation. The value is the average				

	memory usage of all executions (calculated with a 5-minute resolution) multiplied by their cardinalities and divided by the total cardinality. • Example: if two tabs of the Chrome browser are opened at the same time, two distinct processes of chrome.exe are launched and they are aggregated by the Engine (i.e., event cardinality = 2). The average memory usage will be the average of the two processes before aggregation: it represents the average memory usage of a Chrome tab.				
	NXQL ID:	average_memory_usage_per_execution			
Average network response time	Availability	Aggregate			
	Indicates the average TCP connection establishment time of all underlying connections. The value is the average TCP connection establishment time of all executions weighted by their cardinality.				
	NXQL ID:	average_network_response_time			
Average outgoing network bitrate	Availability	Aggregate			
	Average outgoing network bitrate				
	NXQL ID:	average_outgoing_bitrate			
Average outgoing web bitrate	Availability	Aggregate			
	Average outgoing bitrate of all underlying web requests, consolidated over time				
	NXQL ID:	average_outgoing_bitrate			
Average system boot duration	Startup	Aggregate			
	Indicates the average system boot duration.				
	NXQL ID:	average_boot_duration			
Average web request duration	Availability	Aggregate			
	Average time between request and last response byte				
	NXQL ID:	average_request_duration			
Average web request size	Traffic	Aggregate			
	Average size of web requests				
	NXQL ID:	average_request_size			
Average web response size	Traffic	Aggregate			
	Average size of web responses				
	NXQL ID:	average_response_size			

Binary paths	Activity	Aggregate			
	List of executed binary paths (max. 50 paths)				
BIOS serial number	Hardware	Field			
	BIOS serial number				
	NXQL ID:	bios_serial_number			
Chassis serial number	Hardware	Field			
	Chassis serial number				
	NXQL ID:	chassis_serial_number			
Collector installation log	Nexthink Collector	Field			
	Indicates the link to the last Nexthink Collector installation error log.				
	NXQL ID:	collector_installation_log			
Collector status	Nexthink Collector	Field			
	Indicates the status of the Nexthink Collector package installed on the device: • unmanaged: the Collector is not automatically updated • up-to-date: the Collector is up-to-date • outdated: a newer Collector version is available.				
	NXQL ID:	collector_status			
Collector tag	Nexthink Collector	Field			
	Indicates the Collector installation tag.				
	NXQL ID:	collector_tag			
Collector update group	Nexthink Collector	Field			
	Indicates the update group of Nexthink Collector: • manual: the Collector is manually updated • pilot: the Collector is updated as part of the pilot group • main: the Collector is updated as part of the main group.				
	NXQL ID:	upgrade_group			
Collector update status	Nexthink Collector	Field			
	Indicates the status of the Nexthink Collector updater.				
	NXQL ID:	collector_update_status			
Collector version	Nexthink Collector	Field			

	Indicates the version of the Nexthink Collector installed on the device.				
	NXQL ID:	collector_version			
CPU frequency	Hardware	Field			
	CPU frequency				
	NXQL ID:	cpu_frequency			
CPU model	Hardware	Field			
	CPU model				
	NXQL ID:	cpu_model			
CPU usage ratio	Activity	Aggregate			
	Indicates the sum of the CPU time of all executions on each device in scope over all logical processors divided by their total duration. • Example: if we consider two executions with the first one taking 50% of a logical processor during 30 minutes and the second one taking 100% of 2 logical processors during 60 minutes, the CPU usage ratio is 150% (= [50% * 30 min + 2 * 100% * 60 min] / [30 min + 60 min]).				
	NXQL ID:	cpu_usage_ratio			
Cumulated execution duration	Activity	Aggregate			
	Cumulated duration of executions				
	NXQL ID:	cumulated_execution_duration			
Cumulated interaction time	Activity	Aggregate			
	Cumulated time with user interaction (mouse or keyboard events)				
	NXQL ID:	cumulated_interaction_duration			
Cumulated network connection duration	Activity	Aggregate			
	Cumulated duration of TCP connections				
	NXQL ID:	cumulated_connection_duration			
Database usage	Properties	Field			
	Indicates the percentage of the Engine database used by the device.				
	NXQL ID:	database_usage			
Device access rule	Exchange	Field			
	Indicates the name of the Exchange ActiveSync device access rule and if the rule allows, blocks or quarantines the device.				
	NXQL ID:	eas_device_access_rule			
	Policy	Field			

Device encryption required	Indicates whether device encryption is required.				
	NXQL ID:	device_encryption_required			
Device identity	Exchange	Field			
	Indicates the identity of the device in Exchange ActiveSync server.				
	NXQL ID:	eas_device_identity			
Device manufacturer	Hardware	Field			
	Indicates the device manufacturer.				
	NXQL ID:	device_manufacturer			
Device model	Hardware	Field			
	Indicates the model of the device.				
	NXQL ID:	device_model			
Device password required	Policy	Field			
	Indicates whether a password is required on the device.				
	NXQL ID:	device_password_required			
Device product ID	Hardware	Field			
	Device product ID				
	NXQL ID:	device_product_id			
Device product version	Hardware	Field			
	Device product version				
	NXQL ID:	device_product_version			
Device serial number	Hardware	Field			
	Indicates the device serial number.				
	NXQL ID:	device_serial_number			
Device type	Hardware	Field			
	Indicates the device type: • desktop • laptop • server • mobile				
	NXQL ID:	device_type			
Device UUID	Properties	Field			
	Indicates the device universally unique identifier (UUID).				
	NXQL ID:	device_uuid			
	Hardware	Field			

Disks S.M.A.R.T. index	Lowest S.M.A.R.T. index of installed hard disks (index is based on S.M.A.R.T. attributes)				
	NXQL ID:	disks_smart_index			
Distinguished name	Properties	Field			
	Indicates the distinguished name (DN) as seen: • For Windows: in Active Directory (AD); if no connection with AD is set up, a '-' is displayed • For Mobile: in the Exchange ActiveSync server				
	NXQL ID:	distinguished_name			
Email attachment enabled	Policy	Field			
	Indicates whether attachments can be downloaded to the mobile device through the Exchange ActiveSync protocol.				
	NXQL ID:	email_attachment_enabled			
Enforce password history	Policy	Field			
	Indicates the number of unique password that have to be associated with a user account before an old password can be reused: • Windows: as set up in the group policy • Mobile: as set up in security policies				
	NXQL ID:	enforce_password_history			
Entity	Properties	Field			
	Entity to which the device belongs				
	NXQL ID:	entity			
Exemption	Exchange	Field			
	Indicates whether a personal exemption is set for the device and its user. Possible values are: • none • allow • block				
	NXQL ID:	eas_exemption			
Extended logon duration baseline	Startup	Field			
	Indicates the extended logon duration averaged over the last logons. In the calculation, recent logons weigh more than older logons (exponentially weighted moving average).				
	NXQL ID:	extended_logon_duration_baseline			
	Security	Field			

Firewall display name	Name of the main firewall				
	NXQL ID:	firewall_name **Note**: this field is not available for Windows Server operating systems.			
Firewall RTP	Security	Field			
	Indicates whether the firewall real time protection (RTP) is active: • on: indicates that RTP is active • off: indicates that either RTP is not active or no antivirus has been detected • unknown: indicates that the information could not be retrieved • N/A: this field is not available on this operating system • '-' : no data, incompatible collector version or the data is not yet available				
	NXQL ID:	firewall_rtp **Note**: this field is not available for Windows Server operating systems.			
First seen	Properties	Field			
	Indicates the first time when the activity of the device was recorded: • For Windows and Mac OS: the first time Collector reported activity • For Mobile: the first time the device was reported with a successful synchronization				
	NXQL ID:	first_seen			
Graphical card RAM	Hardware	Field			
	Amount of RAM of the graphical card with most RAM				
	NXQL ID:	graphical_card_ram			
Graphical cards	Hardware	Field			
	Installed graphical cards				
	NXQL ID:	graphical_cards			
Group name	Network	Field			
	Name of computer domain or workgroup				
	NXQL ID:	group_name			
Guest account status	Policy	Field			

	Determines if the Guest account is enabled or disabled				
	NXQL ID:	guest_account_status			
Hard disks	Hardware	Field			
	List of all hard disks				
	NXQL ID:	hard_disks			
Hard disks manufacturers	Hardware	Field			
	Indicates the list of hard disk manufacturers				
	NXQL ID:	disks_manufacturers			
High device IO throughput time ratio	Warnings	Aggregate			
	Indicates the ratio between the time the device is in high IO throughput and its uptime.				
	NXQL ID:	high_device_io_throughput_time_ratio			
High device memory time ratio	Warnings	Aggregate			
	Indicates the ratio between the time the device is in high memory usage and its uptime.				
	NXQL ID:	high_device_memory_time_ratio			
High device overall CPU time ratio	Warnings	Aggregate			
	Indicates the ratio between the time the device is in high overall CPU usage and its uptime.				
	NXQL ID:	high_device_overall_cpu_time_ratio			
High device page faults time ratio	Warnings	Aggregate			
	Indicates the ratio between the time the device is in high page faults and its uptime.				
	NXQL ID:	high_device_page_faults_time_ratio			
High device thread CPU time ratio (deprecated)	Warnings	Aggregate			
	Indicates the ratio between the time that the device is in high thread CPU usage and its uptime.				
Highest local privilege level reached	Activity	Aggregate			
	Highest local privilege level reached for executions (user, power user, administrator)				
	NXQL ID:	highest_local_privilege_reached			
ID	Properties	Field			
	Unique device identifier code				
	NXQL ID:	id			
Incoming network traffic	Traffic	Aggregate			
	Total network incoming traffic				

	NXQL ID:	incoming_traffic			
Incoming web traffic	Traffic	Aggregate			
	Total web incoming traffic				
	NXQL ID:	incoming_traffic			
Interaction time ratio	Activity	Aggregate			
	Percentage of time with user interaction (mouse or keyboard events)				
Internet security settings	Security	Field			
	Internet security settings (ok, at risk or unknown)				
	NXQL ID:	internet_security_settings			
IP addresses	Network	Field			
	List of IP addresses for the device				
	NXQL ID:	ip_addresses			
Last extended logon duration	Startup	Field			
	Indicates the last recorded value for the time between the user logging on and the device is ready.				
	NXQL ID:	last_extended_logon_duration			
Last IP address	Network	Field			
	Last IP address assigned to the device				
	NXQL ID:	last_ip_address			
Last logged on user	Startup	Field			
	Last logged on user				
	NXQL ID:	last_logged_on_user			
Last logged on user's privileges	Startup	Field			
	Privileges of the last logged on user (user, power user, administrator)				
	NXQL ID:	privileges_of_last_logged_on_users			
Last logon duration	Startup	Field			
	Indicates the last recorded value for the time between the user logging on and the desktop is displayed.				
	NXQL ID:	last_logon_duration			
Last logon time	Startup	Field			
	Indicates the time of the last logon.				
	NXQL ID:	last_logon_time			
Last policy update	Exchange	Field			
	Indicates the last time the Exchange ActiveSync policy was updated on the device.				

	NXQL ID:	eas_policy_update			
Last seen	Properties	Field			
	Indicates the last time when the activity of the device was recorded: • For Windows and Mac OS: the last time Collector reported activity • For Mobile: the last time the device was reported with a successful synchronization				
	NXQL ID:	last_seen			
Last system boot duration	Startup	Field			
	Duration of last system boot				
	NXQL ID:	last_boot_duration			
Last system boot time	Startup	Field			
	Indicates the time of the last system boot.				
	NXQL ID:	last_system_boot			
Last system update	Operating system	Field			
	Time of last system update				
	NXQL ID:	last_windows_update			
Last update	Nexthink Collector	Field			
	Indicates the last Collector update time.				
	NXQL ID:	last_update			
Last update status	Nexthink Collector	Field			
	Indicates the status of the last Collector update: • '-': the Collector was never updated • successful installation: the last Collector installation was successful • package download error: the Collector was not able to download the Collector package from Nexthink Appliance • package digital signature error: the Collector was not able to check the Collector package digital signature • device reboot required: the device needs to be rebooted to complete the Collector installation • package error: the Collector package installation has failed				

	• internal error: the Collector package installation has failed for an unexpected reason.				
	NXQL ID:	last_update_status			
Last Updater request	Nexthink Collector	Field			
	Indicates the last time the Nexthink Updater has checked for updates.				
	NXQL ID:	last_updater_request			
Local Administrators	Operating system	Field			
	Users and groups which are members of the Local Administrators group on the device				
	NXQL ID:	local_administrators			
Local Power Users	Operating system	Field			
	Users and groups which are members of the Local Powers Users group on the device				
	NXQL ID:	local_power_users			
Logical drives	Local drives	Field			
	List of all logical drives				
	NXQL ID:	logical_drives			
Logon duration baseline	Startup	Field			
	Indicates the logon duration averaged over the last logons. In the calculation, recent logons weigh more than older logons (exponentially weighted moving average).				
	NXQL ID:	average_logon_duration			
Lowest observed web protocol version	Activity	Aggregate			
	Lowest protocol version observed in web requests (excluding web requests with unknown protocol version)				
	NXQL ID:	lowest_protocol_version			
MAC addresses	Network	Field			
	List of MAC addresses for the device				
	NXQL ID:	mac_addresses			
Maximum password age	Policy	Field			
	Indicates the period in time (in days) during which the password can be used before the system requires the user to change it: • Windows: as set up in the group policy				

	• Mobile: as set up in security policies				
	NXQL ID:	maximum_password_age			
Membership type	Network	Field			
	Type of computer membership (domain/workgroup)				
	NXQL ID:	membership_type			
Minimum password age	Policy	Field			
	Period of time (in days) that a password must be used before the user can change it				
	NXQL ID:	minimum_password_age			
Minimum password length	Policy	Field			
	Least number of characters that a password for a user account may contain				
	NXQL ID:	minimum_password_length			
Monitor models	Hardware	Field			
	Models of connected monitors				
	NXQL ID:	monitor_models			
Monitor resolutions	Hardware	Field			
	Screen resolutions of connected monitors				
	NXQL ID:	monitor_resolutions			
Monitors	Hardware	Field			
	Connected monitors				
	NXQL ID:	monitors			
Monitors serial numbers	Hardware	Field			
	Serial numbers of connected monitors (ordered as in 'Monitors')				
	NXQL ID:	monitors_serial_numbers			
Name	Properties	Field			
	Indicates the name of the device: • For Windows: NetBios Name • For Mac OS: computer name used on the network • For Mobile: composed by mailbox name and device friendly name				
	NXQL ID:	name			
Network availability level	Availability	Aggregate			
	Indicates the ratio of successful TCP connections. The possible values are: • high: the ratio is greater or equal to 98%				

	• medium: the ratio is greater or equal to 90% and less than 98% • low: the ratio is lower than 90%				
	NXQL ID:	network_availability_level			
Number of antispyware	Security	Field			
	Number of antispyware detected: • unknown: indicates that the information could not be retrieved • N/A: this field is not available on this operating system • '-' : no data, incompatible collector version or the data is not yet available				
	NXQL ID:	number_of_antispyware **Note**: this field is not available for Windows Server operating systems.			
Number of antiviruses	Security	Field			
	Number of antiviruses detected: • unknown: indicates that the information could not be retrieved • N/A: this field is not available on this operating system • '-' : no data, incompatible collector version or the data is not yet available				
	NXQL ID:	number_of_antiviruses **Note**: this field is not available for Windows Server operating systems.			
Number of application crashes	Errors	Aggregate			
	Number of application crashes				
	NXQL ID:	number_of_application_crashes			
Number of application not responding events	Errors	Aggregate			
	Number of application not responding events				
	NXQL ID:	number_of_application_not_responding_events			
Number of applications	Inventory	Aggregate			
	Number of applications				
	NXQL ID:	number_of_applications			
Number of binaries	Inventory	Aggregate			

	Number of binaries				
	NXQL ID:	number_of_binaries			
Number of connections	Activity	Aggregate			
	Number of connections				
	NXQL ID:	number_of_connections			
Number of cores	Hardware	Field			
	Indicates the number of CPUs multiplied by the number of cores that are available on each CPU.				
	NXQL ID:	number_of_cores			
Number of CPUs	Hardware	Field			
	Indicates the number of central processing units (CPUs), also known as the number of sockets.				
	NXQL ID:	number_of_cpus			
Number of days since first seen	Properties	Field			
	Indicates the number of complete days since the device was first seen. The value is updated every hour.				
	NXQL ID:	number_of_days_since_first_seen			
Number of days since last logon	Startup	Field			
	Number of days since last logon				
	NXQL ID:	number_of_days_since_last_logon			
Number of days since last policy update	Exchange	Field			
	Indicates the number of days since the last Exchange ActiveSync policy update.				
	NXQL ID:	number_of_days_since_last_eas_policy_update			
Number of days since last seen	Properties	Field			
	Indicates the number of days since the last time the device was seen by Nexthink. The field is updated every hour.				
	NXQL ID:	number_of_days_since_last_seen			
Number of days since last system boot	Startup	Field			
	Number of days since last system boot				
	NXQL ID:	number_of_days_since_last_boot			
Number of days since last system update	Operating system	Field			
	Number of days since last system update				
	NXQL ID:	number_of_days_since_last_windows_update			
Number of destinations	Inventory	Aggregate			

	Number of destinations				
	NXQL ID:	number_of_destinations			
Number of domains	Inventory	Aggregate			
	Number of domains				
	NXQL ID:	number_of_domains			
Number of executables	Inventory	Aggregate			
	Number of executables				
	NXQL ID:	number_of_executables			
Number of executions	Activity	Aggregate			
	Number of executions				
	NXQL ID:	number_of_executions			
Number of firewalls	Security	Field			
	Number of firewalls detected: • unknown: indicates that the information could not be retrieved • N/A: this field is not available on this operating system • '-' : no data, incompatible collector version or the data is not yet available				
	NXQL ID:	number_of_firewalls **Note**: this field is not available for Windows Server operating systems.			
Number of graphical cards	Hardware	Field			
	Number of installed graphical cards				
	NXQL ID:	number_of_graphical_cards			
Number of hard resets	Errors	Aggregate			
	Number of hard resets				
Number of installations	Activity	Aggregate			
	Number of installations				
	NXQL ID:	number_of_installations			
Number of logical processors	Hardware	Field			
	Indicates the number of cores multiplied by the number of threads that can run on each core through the use of hyperthreading.				
	NXQL ID:	logical_cpu_number			
Number of logons	Startup	Aggregate			

	Number of logons				
	NXQL ID:	number_of_logons			
Number of monitors	Hardware	Field			
	Number of connected monitors				
	NXQL ID:	number_of_monitors			
Number of packages	Inventory	Aggregate			
	Number of packages				
	NXQL ID:	number_of_packages			
Number of ports	Inventory	Aggregate			
	Number of ports				
	NXQL ID:	number_of_ports			
Number of print jobs	Activity	Aggregate			
	Number of print jobs				
	NXQL ID:	number_of_printouts			
Number of printed pages	Activity	Aggregate			
	Number of printed pages				
	NXQL ID:	number_of_printed_pages			
Number of printers	Inventory	Aggregate			
	Number of printers				
	NXQL ID:	number_of_printers			
Number of system boots	Startup	Aggregate			
	Number of system boots				
	NXQL ID:	number_of_boots			
Number of system crashes	Errors	Aggregate			
	Indicates the number of Windows bluescreens.				
Number of users	Inventory	Aggregate			
	Number of users				
	NXQL ID:	number_of_users			
Number of web requests	Activity	Aggregate			
	Number of web requests				
	NXQL ID:	number_of_web_requests			
OS architecture	Operating system	Field			
	Architecture of device operating system (x86/x64)				
	NXQL ID:	os_architecture			

OS version	Operating system (deprecated)	Field			
	Version of device operating system				
OS version and architecture	Operating system	Field			
	Indicates name, version and architecture (when applicable) of the operating system: • Unknown: the OS version could not be retrieved or it could not be mapped to a recognized value				
	NXQL ID:	os_version_and_architecture			
Outgoing network traffic	Traffic	Aggregate			
	Total network outgoing traffic				
	NXQL ID:	outgoing_traffic			
Outgoing web traffic	Traffic	Aggregate			
	Total web outgoing traffic				
	NXQL ID:	outgoing_traffic			
Password complexity requirements enabled	Policy	Field			
	Indicates whether password complexity is required: • Windows: the password must meet complexity requirements as defined in the group policy • Mobile: no simple passwords are allowed or a minimum password length is set, as defined in the security policy				
	NXQL ID:	password_complexity_requirements			
Platform	Properties	Field			
	Indicates the platform of the device. A platform is a set of operating system families on which the same objects, activities, events and properties can be retrieved. The possible values are: • Windows • Mac OS • Mobile				
	NXQL ID:	platform			
Policy allows non provisionable devices	Exchange	Field			
	Indicates whether a device which does not fully support the policy is still allowed to connect to the Exchange Exchange ActiveSync server.				

	• If 'yes', the security policy is not guaranteed to be applied, even if the field 'ActiveSync policy application status' value is 'applied in full'				
	NXQL ID:	allow_non_provisionable_devices			
Policy application status	Exchange	Field			
	Indicates whether the Exchange ActiveSync policy is applied or not. Possible values are: • not applied • applied in full: the policy is applied (unless the field 'Allow non provisionable devices' value is 'yes') • partially applied				
	NXQL ID:	eas_policy_application_status			
Policy name	Exchange	Field			
	Indicates the name of the Exchange ActiveSync policy applied to the user's mailbox.				
	NXQL ID:	eas_policy_name			
Protocols used in web requests	Activity	Aggregate			
	Protocols used in web requests (HTTP, TLS, HTTP/TLS)				
	NXQL ID:	protocols_used_in_requests			
SD card encryption required	Policy	Field			
	Indicates whether SD card encryption is required.				
	NXQL ID:	sd_card_encryption_required			
SID	Properties	Field			
	Windows security identifier for the device				
	NXQL ID:	sid			
Storage policy	Properties	Field			
	Indicates the event storage policy for the device. Possible values are: • all: web requests, connections and executions are stored • connections and executions • executions • none: no activity is recorded • remove: the device will be removed from Engine during the next cleanup, as long as it is no longer sending data Note that available events depend on the device platform				
	NXQL ID:	storage_policy			

Successful HTTP requests ratio	Availability	Aggregate			
	Percentage of successful HTTP requests (1xx, 2xx and 3xx)				
	NXQL ID:	successful_http_requests_ratio			
Successful network connections ratio	Availability	Aggregate			
	Percentage of successful TCP connections				
	NXQL ID:	successful_connections_ratio			
System boot duration baseline	Startup	Field			
	Indicates the system boot duration averaged over the last boots. In the calculation, recent boots weigh more than older boots (exponentially weighted moving average).				
	NXQL ID:	average_boot_duration			
System drive capacity	Local drives	Field			
	Total capacity of system drive				
	NXQL ID:	system_drive_capacity			
System drive free space	Local drives	Field			
	Total available free space on system drive				
	NXQL ID:	system_drive_free_space			
System drive usage	Local drives	Field			
	Use percentage of system drive				
	NXQL ID:	system_drive_usage			
Target version	Nexthink Collector	Field			
	Indicates the Collector package version that is targeted.				
	NXQL ID:	collector_package_target_version			
Total active days	Activity	Field			
	Indicates the total number of days the device has been active. The value is updated every night.				
	NXQL ID:	total_active_days			
Total CPU time	Activity	Aggregate			
	Indicates the sum of the CPU time of all executions on each device in scope and over all logical processors. • Example: if we consider two executions with the first one taking 50% of a logical processor during 30 minutes and the second one taking 100% of 2 logical processors during 60 minutes, the total CPU time is 135 minutes (= 50% * 30 min + 2 * 100% * 60 min).				

	NXQL ID:	total_cpu_time			
Total drive capacity	Local drives	Field			
	Total capacity of all drives				
	NXQL ID:	total_drive_capacity			
Total drive free space	Local drives	Field			
	Total free space on all drives				
	NXQL ID:	total_drive_free_space			
Total drive usage	Local drives	Field			
	Total use percentage of all drives				
	NXQL ID:	total_drive_usage			
Total network traffic	Traffic	Aggregate			
	Total network traffic (incoming and outgoing)				
	NXQL ID:	total_network_traffic			
Total non-system drive capacity	Local drives	Field			
	Total capacity of all non-system drives				
	NXQL ID:	total_nonsystem_drive_capacity			
Total non-system drive free space	Local drives	Field			
	Total free space on all non-system drives				
	NXQL ID:	total_nonsystem_drive_free_space			
Total non-system drive usage	Local drives	Field			
	Total use percentage of all non-system drives				
	NXQL ID:	total_nonsystem_drive_usage			
Total RAM	Hardware	Field			
	Total amount of RAM				
	NXQL ID:	total_ram			
Total web traffic	Traffic	Aggregate			
	Total web traffic (incoming and outgoing)				
	NXQL ID:	total_web_traffic			
UID	Properties	Field			
	Indicates the universally unique identifier (based on Engine name and device ID).				
Updater error	Nexthink Collector	Field			
	Indicates the last Nexthink Collector Updater error.				
	NXQL ID:	updater_error			

Updater version	Nexthink Collector	Field			
	Indicates the Nexthink Collector Updater version.				
	NXQL ID:	updater_version			
Uptime	Activity	Aggregate			
	Amount of time the machine has been running				
	NXQL ID:	uptime			
User account control status	Security	Field			
	User account control status (ok, at risk or unknown)				
	NXQL ID:	user_account_control_status			
Web interaction time	Activity	Aggregate			
	Indicates the time during which at least one executable is doing HTTP or TLS traffic. This is counted with a 5-minute resolution.				
	NXQL ID:	cumulated_web_interaction_duration			
Windows license key	Operating system	Field			
	Windows license key				
	NXQL ID:	windows_license_key			
Windows Update status	Operating system	Field			
	Windows Update status (ok, at risk or unknown)				
	NXQL ID:	windows_updates_status			
WMI status	Operating system	Field			
	Windows WMI service status (ok, failure)				
	NXQL ID:	wmi_status			

Package

Software packages (programs or updates)

Field	Group	Type			
First installation	Properties	Field			
	Date of the first package installation on any device. This field is based on data reported by the operating system and requires devices date and time to be properly set				
	NXQL ID:	first_installation			

ID	Properties	Field			
	Unique package identifier code				
	NXQL ID:	id			
Name	Properties	Field			
	Package name				
	NXQL ID:	name			
Number of devices	Inventory	Aggregate			
	Number of devices				
	NXQL ID:	number_of_devices			
Number of updates	Properties	Field			
	Number of updates (for programs)				
	NXQL ID:	number_of_updates			
Package status	Inventory	Aggregate			
	Package status (installed/removed)				
Platform	Properties	Field			
	The platform (operating system family) on which the package is installed				
	NXQL ID:	platform			
Program	Properties	Field			
	Package program				
	NXQL ID:	program			
Publisher	Properties	Field			
	Package publisher				
	NXQL ID:	publisher			
Status	Properties	Field			
	Package status (installed/removed)				
	NXQL ID:	status			
Type	Properties	Field			
	Package type: • program • update (Windows only)				
	NXQL ID:	type			
UID	Properties	Field			
	Indicates the universally unique identifier (based on package name and package publisher).				

Version	Properties	Field			
	Package version				
	NXQL ID:	version			
Windows 7 (32-bit) compatibility	Nexthink Library (deprecated)	Field			
	Indicates the Windows 7 (32-bit) compatibility of the package: • '-' : not yet tagged • No information available: not known by Nexthink Library • Compatible: compatible with Windows 7				
	NXQL ID:	windows_7_32bit_compatibility			
Windows 7 (64-bit) compatibility	Nexthink Library (deprecated)	Field			
	Indicates the Windows 7 (64-bit) compatibility of the package: • '-' : not yet tagged • No information available: not known by Nexthink Library • Compatible: compatible with Windows 7				
	NXQL ID:	windows_7_64bit_compatibility			

Application

Sets of executables (e.g. 'Microsoft Office')

Field	Group	Type			
Activity start time	Activity	Aggregate			
	Start time of investigated activity				
	NXQL ID:	activity_start_time			
Activity stop time	Activity	Aggregate			
	Stop time of investigated activity				
	NXQL ID:	activity_stop_time			
Application crash ratio	Errors	Aggregate			
	Indicates the number of application crashes per 100 executions.				
	NXQL ID:	application_crash_ratio			
	Errors	Aggregate			

Application not responding event ratio	Indicates the number of application not responding events per 100 executions.				
	NXQL ID:	application_not_responding_event_ratio			
Average incoming network bitrate	Availability	Aggregate			
	Average incoming network bitrate				
	NXQL ID:	average_incoming_bitrate			
Average incoming web bitrate	Availability	Aggregate			
	Average incoming bitrate of all underlying web requests, consolidated over time				
	NXQL ID:	average_incoming_bitrate			
Average memory usage per execution	Activity	Aggregate			
	Indicates the average memory usage of all underlying executions before aggregation. The value is the average memory usage of all executions (calculated with a 5-minute resolution) multiplied by their cardinalities and divided by the total cardinality. • Example: if two tabs of the Chrome browser are opened at the same time, two distinct processes of chrome.exe are launched and they are aggregated by the Engine (i.e., event cardinality = 2). The average memory usage will be the average of the two processes before aggregation: it represents the average memory usage of a Chrome tab.				
	NXQL ID:	average_memory_usage_per_execution			
Average network response time	Availability	Aggregate			
	Indicates the average TCP connection establishment time of all underlying connections. The value is the average TCP connection establishment time of all executions weighted by their cardinality.				
	NXQL ID:	average_network_response_time			
Average outgoing network bitrate	Availability	Aggregate			
	Average outgoing network bitrate				
	NXQL ID:	average_outgoing_bitrate			
Average outgoing web bitrate	Availability	Aggregate			
	Average outgoing bitrate of all underlying web requests, consolidated over time				

	NXQL ID:	average_outgoing_bitrate			
Average web request duration	Availability	Aggregate			
	Average time between request and last response byte				
	NXQL ID:	average_request_duration			
Average web request size	Traffic	Aggregate			
	Average size of web requests				
	NXQL ID:	average_request_size			
Average web response size	Traffic	Aggregate			
	Average size of web responses				
	NXQL ID:	average_response_size			
Binary paths	Activity	Aggregate			
	List of executed binary paths (max. 50 paths)				
Company	Properties	Field			
	Company producing the application				
	NXQL ID:	company			
CPU usage ratio	Activity	Aggregate			
	Indicates the sum of the CPU time of all executions on each device in scope over all logical processors divided by their total duration. • Example: if we consider two executions with the first one taking 50% of a logical processor during 30 minutes and the second one taking 100% of 2 logical processors during 60 minutes, the CPU usage ratio is 150% (= [50% * 30 min + 2 * 100% * 60 min] / [30 min + 60 min]).				
	NXQL ID:	cpu_usage_ratio			
Cumulated execution duration	Activity	Aggregate			
	Cumulated duration of executions				
	NXQL ID:	cumulated_execution_duration			
Cumulated network connection duration	Activity	Aggregate			
	Cumulated duration of TCP connections				
	NXQL ID:	cumulated_connection_duration			
Database usage	Properties	Field			
	Indicates the percentage of the Engine database used by the application.				

	NXQL ID:	database_usage			
Description	Properties	Field			
	Application description				
	NXQL ID:	description			
First seen	Properties	Field			
	First time activity of the application was recorded on any device				
	NXQL ID:	first_seen			
High application thread CPU time ratio	Warnings	Aggregate			
	Indicates the ratio between the time that the underlying executions are in high thread CPU usage and their execution duration.				
	NXQL ID:	high_application_thread_cpu_time_ratio			
Highest local privilege level reached	Activity	Aggregate			
	Highest local privilege level reached for executions (user, power user, administrator)				
	NXQL ID:	highest_local_privilege_reached			
ID	Properties	Field			
	Unique application identifier code				
	NXQL ID:	id			
Incoming network traffic	Traffic	Aggregate			
	Total network incoming traffic				
	NXQL ID:	incoming_traffic			
Incoming network traffic per device	Traffic	Aggregate			
	Indicates the incoming network traffic divided by the number of devices.				
	NXQL ID:	incoming_network_traffic_per_device			
Incoming web traffic	Traffic	Aggregate			
	Total web incoming traffic				
	NXQL ID:	incoming_traffic			
Incoming web traffic per device	Traffic	Aggregate			
	Indicates the incoming web traffic divided by the number of devices.				
	NXQL ID:	incoming_web_traffic_per_device			
Known packages	Properties	Field			
	List of packages known to contain the application. This list is not exhaustive: the presence of a package does not necessarily imply that on a given device the application was installed through that package				
	NXQL ID:	known_packages			
Last seen	Properties	Field			

	Last time activity of the application was recorded on any device				
	NXQL ID:	last_seen			
Lowest observed web protocol version	Activity	Aggregate			
	Lowest protocol version observed in web requests (excluding web requests with unknown protocol version)				
	NXQL ID:	lowest_protocol_version			
Name	Properties	Field			
	Application name				
	NXQL ID:	name			
Network availability level	Availability	Aggregate			
	Indicates the ratio of successful TCP connections. The possible values are: • high: the ratio is greater or equal to 98% • medium: the ratio is greater or equal to 90% and less than 98% • low: the ratio is lower than 90%				
	NXQL ID:	network_availability_level			
Number of application crashes	Errors	Aggregate			
	Number of application crashes				
	NXQL ID:	number_of_application_crashes			
Number of application not responding events	Errors	Aggregate			
	Number of application not responding events				
	NXQL ID:	number_of_application_not_responding_events			
Number of binaries	Inventory	Aggregate			
	Number of binaries				
	NXQL ID:	number_of_binaries			
Number of connections	Activity	Aggregate			
	Number of connections				
	NXQL ID:	number_of_connections			
Number of destinations	Inventory	Aggregate			
	Number of destinations				
	NXQL ID:	number_of_destinations			
Number of devices	Inventory	Aggregate			
	Number of devices				
	NXQL ID:	number_of_devices			

Number of domains	Inventory	Aggregate			
	Number of domains				
	NXQL ID:	number_of_domains			
Number of executables	Inventory	Aggregate			
	Number of executables				
	NXQL ID:	number_of_executables			
Number of executions	Activity	Aggregate			
	Number of executions				
	NXQL ID:	number_of_executions			
Number of ports	Inventory	Aggregate			
	Number of ports				
	NXQL ID:	number_of_ports			
Number of users	Inventory	Aggregate			
	Number of users				
	NXQL ID:	number_of_users			
Number of web requests	Activity	Aggregate			
	Number of web requests				
	NXQL ID:	number_of_web_requests			
Outgoing network traffic	Traffic	Aggregate			
	Total network outgoing traffic				
	NXQL ID:	outgoing_traffic			
Outgoing network traffic per device	Traffic	Aggregate			
	Indicates the outgoing network traffic divided by the number of devices.				
	NXQL ID:	outgoing_network_traffic_per_device			
Outgoing web traffic	Traffic	Aggregate			
	Total web outgoing traffic				
	NXQL ID:	outgoing_traffic			
Outgoing web traffic per device	Traffic	Aggregate			
	Indicates the outgoing web traffic divided by the number of devices.				
	NXQL ID:	outgoing_web_traffic_per_device			
Platform	Properties	Field			
	The platform (operating system family) on which the application is running				
	NXQL ID:	platform			
	Activity	Aggregate			

Protocols used in web requests	Protocols used in web requests (HTTP, TLS, HTTP/TLS)				
	NXQL ID:	protocols_used_in_requests			
Storage policy	Properties	Field			
	Indicates the event storage policy for the application. Possible values are: • all: web requests, connections and executions are stored • connections and executions • executions • none: no activity is recorded				
	NXQL ID:	storage_policy			
Successful HTTP requests ratio	Availability	Aggregate			
	Percentage of successful HTTP requests (1xx, 2xx and 3xx)				
	NXQL ID:	successful_http_requests_ratio			
Successful network connections ratio	Availability	Aggregate			
	Percentage of successful TCP connections				
	NXQL ID:	successful_connections_ratio			
Total active days	Activity	Field			
	Total number of days the application was active				
	NXQL ID:	total_active_days			
Total CPU time	Activity	Aggregate			
	Indicates the sum of the CPU time of all executions on each device in scope and over all logical processors. • Example: if we consider two executions with the first one taking 50% of a logical processor during 30 minutes and the second one taking 100% of 2 logical processors during 60 minutes, the total CPU time is 135 minutes (= 50% * 30 min + 2 * 100% * 60 min).				
	NXQL ID:	total_cpu_time			
Total network traffic	Traffic	Aggregate			
	Total network traffic (incoming and outgoing)				
	NXQL ID:	total_network_traffic			
Total web traffic	Traffic	Aggregate			
	Total web traffic (incoming and outgoing)				
	NXQL ID:	total_web_traffic			

UID	Properties	Field			
	Indicates the universally unique identifier (based on package name and application company).				
Web interaction time	Activity	Aggregate			
	Indicates the time during which at least one executable is doing HTTP or TLS traffic. This is counted with a 5-minute resolution.				
	NXQL ID:	cumulated_web_interaction_duration			

Executable

Executable programs (e.g. 'winword.exe')

Field	Group	Type			
Activity start time	Activity	Aggregate			
	Start time of investigated activity				
	NXQL ID:	activity_start_time			
Activity stop time	Activity	Aggregate			
	Stop time of investigated activity				
	NXQL ID:	activity_stop_time			
Application company	Properties	Field			
	Application company				
	NXQL ID:	application_company			
Application crash ratio	Errors	Aggregate			
	Indicates the number of application crashes per 100 executions.				
	NXQL ID:	application_crash_ratio			
Application name	Properties	Field			
	Application name				
	NXQL ID:	application_name			
Application not responding event ratio	Errors	Aggregate			
	Indicates the number of application not responding events per 100 executions.				
	NXQL ID:	application_not_responding_event_ratio			
Average incoming network bitrate	Availability	Aggregate			
	Average incoming network bitrate				
	NXQL ID:	average_incoming_bitrate			

Average incoming web bitrate	Availability	Aggregate			
	Average incoming bitrate of all underlying web requests, consolidated over time				
	NXQL ID:	average_incoming_bitrate			
Average memory usage per execution	Activity	Aggregate			
	Indicates the average memory usage of all underlying executions before aggregation. The value is the average memory usage of all executions (calculated with a 5-minute resolution) multiplied by their cardinalities and divided by the total cardinality. • Example: if two tabs of the Chrome browser are opened at the same time, two distinct processes of chrome.exe are launched and they are aggregated by the Engine (i.e., event cardinality = 2). The average memory usage will be the average of the two processes before aggregation: it represents the average memory usage of a Chrome tab.				
	NXQL ID:	average_memory_usage_per_execution			
Average network response time	Availability	Aggregate			
	Indicates the average TCP connection establishment time of all underlying connections. The value is the average TCP connection establishment time of all executions weighted by their cardinality.				
	NXQL ID:	average_network_response_time			
Average outgoing network bitrate	Availability	Aggregate			
	Average outgoing network bitrate				
	NXQL ID:	average_outgoing_bitrate			
Average outgoing web bitrate	Availability	Aggregate			
	Average outgoing bitrate of all underlying web requests, consolidated over time				
	NXQL ID:	average_outgoing_bitrate			
Average web request duration	Availability	Aggregate			
	Average time between request and last response byte				
	NXQL ID:	average_request_duration			
Average web request size	Traffic	Aggregate			
	Average size of web requests				

	NXQL ID:	average_request_size			
Average web response size	Traffic	Aggregate			
	Average size of web responses				
	NXQL ID:	average_response_size			
Binary paths	Activity	Aggregate			
	List of executed binary paths (max. 50 paths)				
CPU usage ratio	Activity	Aggregate			
	Indicates the sum of the CPU time of all executions on each device in scope over all logical processors divided by their total duration. • Example: if we consider two executions with the first one taking 50% of a logical processor during 30 minutes and the second one taking 100% of 2 logical processors during 60 minutes, the CPU usage ratio is 150% (= [50% * 30 min + 2 * 100% * 60 min] / [30 min + 60 min]).				
	NXQL ID:	cpu_usage_ratio			
Cumulated execution duration	Activity	Aggregate			
	Cumulated duration of executions				
	NXQL ID:	cumulated_execution_duration			
Cumulated network connection duration	Activity	Aggregate			
	Cumulated duration of TCP connections				
	NXQL ID:	cumulated_connection_duration			
Database usage	Properties	Field			
	Indicates the percentage of the Engine database used by the executable.				
	NXQL ID:	database_usage			
Description	Properties	Field			
	Executable description				
	NXQL ID:	description			
First seen	Properties	Field			
	First time activity of the executable was recorded on any device				
	NXQL ID:	first_seen			
High application thread CPU time ratio	Warnings	Aggregate			

	Indicates the ratio between the time that the underlying executions are in high thread CPU usage and their execution duration.				
	NXQL ID:	high_application_thread_cpu_time_ratio			
Highest local privilege level reached	Activity	Aggregate			
	Highest local privilege level reached for executions (user, power user, administrator)				
	NXQL ID:	highest_local_privilege_reached			
ID	Properties	Field			
	Unique executable identifier code				
	NXQL ID:	id			
Incoming network traffic	Traffic	Aggregate			
	Total network incoming traffic				
	NXQL ID:	incoming_traffic			
Incoming network traffic per device	Traffic	Aggregate			
	Indicates the incoming network traffic divided by the number of devices.				
	NXQL ID:	incoming_network_traffic_per_device			
Incoming web traffic	Traffic	Aggregate			
	Total web incoming traffic				
	NXQL ID:	incoming_traffic			
Incoming web traffic per device	Traffic	Aggregate			
	Indicates the incoming web traffic divided by the number of devices.				
	NXQL ID:	incoming_web_traffic_per_device			
Known packages	Properties	Field			
	List of packages known to contain the executable. This list is not exhaustive: the presence of a package does not necessarily imply that on a given device the executable was installed through that package				
	NXQL ID:	known_packages			
Last seen	Properties	Field			
	Last time activity of the executable was recorded on any device				
	NXQL ID:	last_seen			
Lowest observed web protocol version	Activity	Aggregate			
	Lowest protocol version observed in web requests (excluding web requests with unknown protocol version)				
	NXQL ID:	lowest_protocol_version			
Name	Properties	Field			
	Executable name				

	NXQL ID:	name			
Network availability level	Availability	Aggregate			
	Indicates the ratio of successful TCP connections. The possible values are: • high: the ratio is greater or equal to 98% • medium: the ratio is greater or equal to 90% and less than 98% • low: the ratio is lower than 90%				
	NXQL ID:	network_availability_level			
Number of application crashes	Errors	Aggregate			
	Number of application crashes				
	NXQL ID:	number_of_application_crashes			
Number of application not responding events	Errors	Aggregate			
	Number of application not responding events				
	NXQL ID:	number_of_application_not_responding_events			
Number of binaries	Inventory	Aggregate			
	Number of binaries				
	NXQL ID:	number_of_binaries			
Number of connections	Activity	Aggregate			
	Number of connections				
	NXQL ID:	number_of_connections			
Number of destinations	Inventory	Aggregate			
	Number of destinations				
	NXQL ID:	number_of_destinations			
Number of devices	Inventory	Aggregate			
	Number of devices				
	NXQL ID:	number_of_devices			
Number of domains	Inventory	Aggregate			
	Number of domains				
	NXQL ID:	number_of_domains			
Number of executions	Activity	Aggregate			
	Number of executions				
	NXQL ID:	number_of_executions			
Number of ports	Inventory	Aggregate			
	Number of ports				

	NXQL ID:	number_of_ports			
Number of users	Inventory	Aggregate			
	Number of users				
	NXQL ID:	number_of_users			
Number of web requests	Activity	Aggregate			
	Number of web requests				
	NXQL ID:	number_of_web_requests			
Outgoing network traffic	Traffic	Aggregate			
	Total network outgoing traffic				
	NXQL ID:	outgoing_traffic			
Outgoing network traffic per device	Traffic	Aggregate			
	Indicates the outgoing network traffic divided by the number of devices.				
	NXQL ID:	outgoing_network_traffic_per_device			
Outgoing web traffic	Traffic	Aggregate			
	Total web outgoing traffic				
	NXQL ID:	outgoing_traffic			
Outgoing web traffic per device	Traffic	Aggregate			
	Indicates the outgoing web traffic divided by the number of devices.				
	NXQL ID:	outgoing_web_traffic_per_device			
Platform	Properties	Field			
	The platform (operating system family) on which the executable is running				
	NXQL ID:	platform			
Protocols used in web requests	Activity	Aggregate			
	Protocols used in web requests (HTTP, TLS, HTTP/TLS)				
	NXQL ID:	protocols_used_in_requests			
Storage policy	Properties	Field			
	Indicates the event storage policy for the executable. Possible values are: • all: web requests, connections and executions are stored • connections and executions • executions • none: no activity is recorded				
	NXQL ID:	storage_policy			

Successful HTTP requests ratio	Availability	Aggregate			
	Percentage of successful HTTP requests (1xx, 2xx and 3xx)				
	NXQL ID:	successful_http_requests_ratio			
Successful network connections ratio	Availability	Aggregate			
	Percentage of successful TCP connections				
	NXQL ID:	successful_connections_ratio			
Total active days	Activity	Field			
	Total number of days the executable was active				
	NXQL ID:	total_active_days			
Total CPU time	Activity	Aggregate			
	Indicates the sum of the CPU time of all executions on each device in scope and over all logical processors. • Example: if we consider two executions with the first one taking 50% of a logical processor during 30 minutes and the second one taking 100% of 2 logical processors during 60 minutes, the total CPU time is 135 minutes (= 50% * 30 min + 2 * 100% * 60 min).				
	NXQL ID:	total_cpu_time			
Total network traffic	Traffic	Aggregate			
	Total network traffic (incoming and outgoing)				
	NXQL ID:	total_network_traffic			
Total web traffic	Traffic	Aggregate			
	Total web traffic (incoming and outgoing)				
	NXQL ID:	total_web_traffic			
UID	Properties	Field			
	Indicates the universally unique identifier (based on application name, application company and executable name).				
Web interaction time	Activity	Aggregate			
	Indicates the time during which at least one executable is doing HTTP or TLS traffic. This is counted with a 5-minute resolution.				
	NXQL ID:	cumulated_web_interaction_duration			

Binary

Executable binary files (e.g. 'winword.exe - 10.0.6843')

Field	Group	Type			
Activity start time	Activity	Aggregate			
	Start time of investigated activity				
	NXQL ID:	activity_start_time			
Activity stop time	Activity	Aggregate			
	Stop time of investigated activity				
	NXQL ID:	activity_stop_time			
Application category	Properties	Field			
	Indicates the category of the application: • '-' : not yet tagged • Unknown: not categorized by Nexthink Library				
	NXQL ID:	application_category			
Application company	Properties	Field			
	Application company				
	NXQL ID:	application_company			
Application crash ratio	Errors	Aggregate			
	Indicates the number of application crashes per 100 executions.				
	NXQL ID:	application_crash_ratio			
Application name	Properties	Field			
	Application name				
	NXQL ID:	application_name			
Application not responding event ratio	Errors	Aggregate			
	Indicates the number of application not responding events per 100 executions.				
	NXQL ID:	application_not_responding_event_ratio			
Average CPU usage (deprecated)	Activity	Field			
	Indicates the average CPU usage over all logical processors since the first time the binary was seen. The value is the average CPU usage sampled every 5 minutes for each execution divided by the number of samples.				
	NXQL ID:	average_cpu_usage			
Average incoming network bitrate	Availability	Aggregate			
	Average incoming network bitrate				
	NXQL ID:	average_incoming_bitrate			

Average incoming web bitrate	Availability	Aggregate			
	Average incoming bitrate of all underlying web requests, consolidated over time				
	NXQL ID:	average_incoming_bitrate			
Average memory usage (deprecated)	Activity	Field			
	Indicates the average memory usage since the first time the binary was seen. The value is the sum of the memory usage sampled every 5 minutes for each execution divided by the number of samples.				
	NXQL ID:	average_memory_usage			
Average memory usage per execution	Activity	Aggregate			
	Indicates the average memory usage of all underlying executions before aggregation. The value is the average memory usage of all executions (calculated with a 5-minute resolution) multiplied by their cardinalities and divided by the total cardinality. • Example: if two tabs of the Chrome browser are opened at the same time, two distinct processes of chrome.exe are launched and they are aggregated by the Engine (i.e., event cardinality = 2). The average memory usage will be the average of the two processes before aggregation: it represents the average memory usage of a Chrome tab.				
	NXQL ID:	average_memory_usage_per_execution			
Average network response time	Availability	Aggregate			
	Indicates the average TCP connection establishment time of all underlying connections. The value is the average TCP connection establishment time of all executions weighted by their cardinality.				
	NXQL ID:	average_network_response_time			
Average number of graphical handles	Activity	Field			
	Average number of graphical handles (GDI)				
	NXQL ID:	average_number_of_graphical_handles			
Average outgoing network bitrate	Availability	Aggregate			
	Average outgoing network bitrate				

	NXQL ID:	average_outgoing_bitrate			
Average outgoing web bitrate	Availability	Aggregate			
	Average outgoing bitrate of all underlying web requests, consolidated over time				
	NXQL ID:	average_outgoing_bitrate			
Average web request duration	Availability	Aggregate			
	Average time between request and last response byte				
	NXQL ID:	average_request_duration			
Average web request size	Traffic	Aggregate			
	Average size of web requests				
	NXQL ID:	average_request_size			
Average web response size	Traffic	Aggregate			
	Average size of web responses				
	NXQL ID:	average_response_size			
Binary paths	Activity	Aggregate			
	List of executed binary paths (max. 50 paths)				
CPU usage ratio	Activity	Aggregate			
	Indicates the sum of the CPU time of all executions on each device in scope over all logical processors divided by their total duration. • Example: if we consider two executions with the first one taking 50% of a logical processor during 30 minutes and the second one taking 100% of 2 logical processors during 60 minutes, the CPU usage ratio is 150% (= [50% * 30 min + 2 * 100% * 60 min] / [30 min + 60 min]).				
	NXQL ID:	cpu_usage_ratio			
Cumulated execution duration	Activity	Aggregate			
	Cumulated duration of executions				
	NXQL ID:	cumulated_execution_duration			
Cumulated network connection duration	Activity	Aggregate			
	Cumulated duration of TCP connections				
	NXQL ID:	cumulated_connection_duration			
Database usage	Properties	Field			
	Indicates the percentage of the Engine database used by the binary.				

	NXQL ID:	database_usage			
Description	Properties	Field			
	Description as it appears in the binary file				
	NXQL ID:	description			
Executable name	Properties	Field			
	Executable name				
	NXQL ID:	executable_name			
File size	Properties	Field			
	Binary file size				
	NXQL ID:	file_size			
First seen	Properties	Field			
	First time activity of the binary was recorded on any device				
	NXQL ID:	first_seen			
High application thread CPU time ratio	Warnings	Aggregate			
	Indicates the ratio between the time that the underlying executions are in high thread CPU usage and their execution duration.				
	NXQL ID:	high_application_thread_cpu_time_ratio			
Highest local privilege level reached	Activity	Aggregate			
	Highest local privilege level reached for executions (user, power user, administrator)				
	NXQL ID:	highest_local_privilege_reached			
ID	Properties	Field			
	Unique binary identifier code				
	NXQL ID:	id			
Incoming network traffic	Traffic	Aggregate			
	Total network incoming traffic				
	NXQL ID:	incoming_traffic			
Incoming network traffic per device	Traffic	Aggregate			
	Indicates the incoming network traffic divided by the number of devices.				
	NXQL ID:	incoming_network_traffic_per_device			
Incoming web traffic	Traffic	Aggregate			
	Total web incoming traffic				
	NXQL ID:	incoming_traffic			
Incoming web traffic per device	Traffic	Aggregate			
	Indicates the incoming web traffic divided by the number of devices.				

	NXQL ID:	incoming_web_traffic_per_device			
Last seen	Properties	Field			
	Last time activity of the binary was recorded on any device				
	NXQL ID:	last_seen			
Lowest observed web protocol version	Activity	Aggregate			
	Lowest protocol version observed in web requests (excluding web requests with unknown protocol version)				
	NXQL ID:	lowest_protocol_version			
MD5 hash	Properties	Field			
	Indicates the MD5 hash of the binary.				
	NXQL ID:	hash			
Network availability level	Availability	Aggregate			
	Indicates the ratio of successful TCP connections. The possible values are: • high: the ratio is greater or equal to 98% • medium: the ratio is greater or equal to 90% and less than 98% • low: the ratio is lower than 90%				
	NXQL ID:	network_availability_level			
Number of application crashes	Errors	Aggregate			
	Number of application crashes				
	NXQL ID:	number_of_application_crashes			
Number of application not responding events	Errors	Aggregate			
	Number of application not responding events				
	NXQL ID:	number_of_application_not_responding_events			
Number of connections	Activity	Aggregate			
	Number of connections				
	NXQL ID:	number_of_connections			
Number of destinations	Inventory	Aggregate			
	Number of destinations				
	NXQL ID:	number_of_destinations			
Number of devices	Inventory	Aggregate			
	Number of devices				
	NXQL ID:	number_of_devices			
Number of domains	Inventory	Aggregate			

	Number of domains				
	NXQL ID:	number_of_domains			
Number of executions	Activity	Aggregate			
	Number of executions				
	NXQL ID:	number_of_executions			
Number of ports	Inventory	Aggregate			
	Number of ports				
	NXQL ID:	number_of_ports			
Number of users	Inventory	Aggregate			
	Number of users				
	NXQL ID:	number_of_users			
Number of web requests	Activity	Aggregate			
	Number of web requests				
	NXQL ID:	number_of_web_requests			
Outgoing network traffic	Traffic	Aggregate			
	Total network outgoing traffic				
	NXQL ID:	outgoing_traffic			
Outgoing network traffic per device	Traffic	Aggregate			
	Indicates the outgoing network traffic divided by the number of devices.				
	NXQL ID:	outgoing_network_traffic_per_device			
Outgoing web traffic	Traffic	Aggregate			
	Total web outgoing traffic				
	NXQL ID:	outgoing_traffic			
Outgoing web traffic per device	Traffic	Aggregate			
	Indicates the outgoing web traffic divided by the number of devices.				
	NXQL ID:	outgoing_web_traffic_per_device			
Paths	Properties	Field			
	List of paths of the binary				
	NXQL ID:	paths			
Platform	Properties	Field			
	The platform (operating system family) on which the binary is running				
	NXQL ID:	platform			
Protocols used in web requests	Activity	Aggregate			
	Protocols used in web requests (HTTP, TLS, HTTP/TLS)				

	NXQL ID:	protocols_used_in_requests	
SHA-1 hash	Properties	Field	
	Indicates the SHA-1 hash of the binary.		
	NXQL ID:	sha1	
Storage policy	Properties	Field	
	Indicates the event storage policy for the binary. Possible values are: • all: web requests, connections and executions are stored • connections and executions • executions • none: no activity is recorded		
	NXQL ID:	storage_policy	
Successful HTTP requests ratio	Availability	Aggregate	
	Percentage of successful HTTP requests (1xx, 2xx and 3xx)		
	NXQL ID:	successful_http_requests_ratio	
Successful network connections ratio	Availability	Aggregate	
	Percentage of successful TCP connections		
	NXQL ID:	successful_connections_ratio	
Threat level	Properties	Field	
	Indicates the threat level of the binary: • '-' : not yet tagged • none detected: no known threat • low: low threat • intermediate: intermediate threat • high: high threat		
	NXQL ID:	threat_level	
Total active days	Activity	Field	
	Total number of days the binary was active		
	NXQL ID:	total_active_days	
Total CPU time	Activity	Aggregate	
	Indicates the sum of the CPU time of all executions on each device in scope and over all logical processors. • Example: if we consider two executions with the first one taking 50% of a logical processor during 30 minutes and the second one taking 100% of 2		

	logical processors during 60 minutes, the total CPU time is 135 minutes (= 50% * 30 min + 2 * 100% * 60 min).				
	NXQL ID:	total_cpu_time			
Total network traffic	Traffic	Aggregate			
	Total network traffic (incoming and outgoing)				
	NXQL ID:	total_network_traffic			
Total web traffic	Traffic	Aggregate			
	Total web traffic (incoming and outgoing)				
	NXQL ID:	total_web_traffic			
UID	Properties	Field			
	Indicates the universally unique identifier (based on binary hash).				
User interface	Properties	Field			
	Application has interactive user interface				
	NXQL ID:	user_interface			
Version	Properties	Field			
	Version of the binary				
	NXQL ID:	version			
Web interaction time	Activity	Aggregate			
	Indicates the time during which at least one executable is doing HTTP or TLS traffic. This is counted with a 5-minute resolution.				
	NXQL ID:	cumulated_web_interaction_duration			

Port

Connection ports (TCP or UDP)

Field	Group	Type			
Activity start time	Activity	Aggregate			
	Start time of investigated activity				
	NXQL ID:	activity_start_time			
Activity stop time	Activity	Aggregate			
	Stop time of investigated activity				
	NXQL ID:	activity_stop_time			
Average incoming network bitrate	Availability	Aggregate			

	Average incoming network bitrate				
	NXQL ID:	average_incoming_bitrate			
Average incoming web bitrate	Availability	Aggregate			
	Average incoming bitrate of all underlying web requests, consolidated over time				
	NXQL ID:	average_incoming_bitrate			
Average network response time	Availability	Aggregate			
	Indicates the average TCP connection establishment time of all underlying connections. The value is the average TCP connection establishment time of all executions weighted by their cardinality.				
	NXQL ID:	average_network_response_time			
Average outgoing network bitrate	Availability	Aggregate			
	Average outgoing network bitrate				
	NXQL ID:	average_outgoing_bitrate			
Average outgoing web bitrate	Availability	Aggregate			
	Average outgoing bitrate of all underlying web requests, consolidated over time				
	NXQL ID:	average_outgoing_bitrate			
Average web request duration	Availability	Aggregate			
	Average time between request and last response byte				
	NXQL ID:	average_request_duration			
Average web request size	Traffic	Aggregate			
	Average size of web requests				
	NXQL ID:	average_request_size			
Average web response size	Traffic	Aggregate			
	Average size of web responses				
	NXQL ID:	average_response_size			
Cumulated network connection duration	Activity	Aggregate			
	Cumulated duration of TCP connections				
	NXQL ID:	cumulated_connection_duration			
First seen	Properties	Field			
	First time activity of the port was recorded on any device				
	NXQL ID:	first_seen			
	Activity	Aggregate			

Highest local privilege level reached	Highest local privilege level reached for executions (user, power user, administrator)				
	NXQL ID:	highest_local_privilege_reached			
ID	Properties	Field			
	Unique port identifier code				
	NXQL ID:	id			
Incoming network traffic	Traffic	Aggregate			
	Total network incoming traffic				
	NXQL ID:	incoming_traffic			
Incoming network traffic per device	Traffic	Aggregate			
	Indicates the incoming network traffic divided by the number of devices.				
	NXQL ID:	incoming_network_traffic_per_device			
Incoming web traffic	Traffic	Aggregate			
	Total web incoming traffic				
	NXQL ID:	incoming_traffic			
Incoming web traffic per device	Traffic	Aggregate			
	Indicates the incoming web traffic divided by the number of devices.				
	NXQL ID:	incoming_web_traffic_per_device			
Last seen	Properties	Field			
	Last time activity of the port was recorded on any device				
	NXQL ID:	last_seen			
Lowest observed web protocol version	Activity	Aggregate			
	Lowest protocol version observed in web requests (excluding web requests with unknown protocol version)				
	NXQL ID:	lowest_protocol_version			
Network availability level	Availability	Aggregate			
	Indicates the ratio of successful TCP connections. The possible values are: • high: the ratio is greater or equal to 98% • medium: the ratio is greater or equal to 90% and less than 98% • low: the ratio is lower than 90%				
	NXQL ID:	network_availability_level			
Number of applications	Inventory	Aggregate			

	Number of applications				
	NXQL ID:	number_of_applications			
Number of binaries	Inventory	Aggregate			
	Number of binaries				
	NXQL ID:	number_of_binaries			
Number of connections	Activity	Aggregate			
	Number of connections				
	NXQL ID:	number_of_connections			
Number of destinations	Inventory	Aggregate			
	Number of destinations				
	NXQL ID:	number_of_destinations			
Number of devices	Inventory	Aggregate			
	Number of devices				
	NXQL ID:	number_of_devices			
Number of domains	Inventory	Aggregate			
	Number of domains				
	NXQL ID:	number_of_domains			
Number of executables	Inventory	Aggregate			
	Number of executables				
	NXQL ID:	number_of_executables			
Number of users	Inventory	Aggregate			
	Number of users				
	NXQL ID:	number_of_users			
Number of web requests	Activity	Aggregate			
	Number of web requests				
	NXQL ID:	number_of_web_requests			
Outgoing network traffic	Traffic	Aggregate			
	Total network outgoing traffic				
	NXQL ID:	outgoing_traffic			
Outgoing network traffic per device	Traffic	Aggregate			
	Indicates the outgoing network traffic divided by the number of devices.				
	NXQL ID:	outgoing_network_traffic_per_device			
Outgoing web traffic	Traffic	Aggregate			
	Total web outgoing traffic				

	NXQL ID:	outgoing_traffic			
Outgoing web traffic per device	Traffic	Aggregate			
	Indicates the outgoing web traffic divided by the number of devices.				
	NXQL ID:	outgoing_web_traffic_per_device			
Port number	Properties	Field			
	Port number				
	NXQL ID:	port_number			
Port type	Properties	Field			
	Port type (tcp, udp, tcp port scan, udp port scan)				
	NXQL ID:	port_type			
Port type/Port number	Properties	Field			
	Port value for tagging				
	NXQL ID:	port_value			
Protocols used in web requests	Activity	Aggregate			
	Protocols used in web requests (HTTP, TLS, HTTP/TLS)				
	NXQL ID:	protocols_used_in_requests			
Successful HTTP requests ratio	Availability	Aggregate			
	Percentage of successful HTTP requests (1xx, 2xx and 3xx)				
	NXQL ID:	successful_http_requests_ratio			
Successful network connections ratio	Availability	Aggregate			
	Percentage of successful TCP connections				
	NXQL ID:	successful_connections_ratio			
Total network traffic	Traffic	Aggregate			
	Total network traffic (incoming and outgoing)				
	NXQL ID:	total_network_traffic			
Total web traffic	Traffic	Aggregate			
	Total web traffic (incoming and outgoing)				
	NXQL ID:	total_web_traffic			
UID	Properties	Field			
	Indicates the universally unique identifier (based on port number).				
Web interaction time	Activity	Aggregate			
	Indicates the time during which at least one executable is doing HTTP or TLS traffic. This is counted with a 5-minute resolution.				

	NXQL ID:	cumulated_web_interaction_duration

Destination

Devices receiving connections

Field	Group	Type			
Activity start time	Activity	Aggregate			
	Start time of investigated activity				
	NXQL ID:	activity_start_time			
Activity stop time	Activity	Aggregate			
	Stop time of investigated activity				
	NXQL ID:	activity_stop_time			
Average incoming network bitrate	Availability	Aggregate			
	Average incoming network bitrate				
	NXQL ID:	average_incoming_bitrate			
Average incoming web bitrate	Availability	Aggregate			
	Average incoming bitrate of all underlying web requests, consolidated over time				
	NXQL ID:	average_incoming_bitrate			
Average network response time	Availability	Aggregate			
	Indicates the average TCP connection establishment time of all underlying connections. The value is the average TCP connection establishment time of all executions weighted by their cardinality.				
	NXQL ID:	average_network_response_time			
Average outgoing network bitrate	Availability	Aggregate			
	Average outgoing network bitrate				
	NXQL ID:	average_outgoing_bitrate			
Average outgoing web bitrate	Availability	Aggregate			
	Average outgoing bitrate of all underlying web requests, consolidated over time				
	NXQL ID:	average_outgoing_bitrate			
Average web request duration	Availability	Aggregate			
	Average time between request and last response byte				

	NXQL ID:	average_request_duration			
Average web request size	Traffic	Aggregate			
	Average size of web requests				
	NXQL ID:	average_request_size			
Average web response size	Traffic	Aggregate			
	Average size of web responses				
	NXQL ID:	average_response_size			
Cumulated network connection duration	Activity	Aggregate			
	Cumulated duration of TCP connections				
	NXQL ID:	cumulated_connection_duration			
Database usage	Properties	Field			
	Indicates the percentage of the Engine database used by the destination.				
	NXQL ID:	database_usage			
First seen	Properties	Field			
	First time activity to the destination was recorded on any device				
	NXQL ID:	first_seen			
Highest local privilege level reached	Activity	Aggregate			
	Highest local privilege level reached for executions (user, power user, administrator)				
	NXQL ID:	highest_local_privilege_reached			
ID	Properties	Field			
	Unique destination identifier code				
	NXQL ID:	id			
Incoming network traffic	Traffic	Aggregate			
	Total network incoming traffic				
	NXQL ID:	incoming_traffic			
Incoming network traffic per device	Traffic	Aggregate			
	Indicates the incoming network traffic divided by the number of devices.				
	NXQL ID:	incoming_network_traffic_per_device			
Incoming web traffic	Traffic	Aggregate			
	Total web incoming traffic				
	NXQL ID:	incoming_traffic			
Incoming web traffic per device	Traffic	Aggregate			

	Indicates the incoming web traffic divided by the number of devices.				
	NXQL ID:	incoming_web_traffic_per_device			
IP address	Properties	Field			
	IP address for the destination				
	NXQL ID:	ip_address			
Last seen	Properties	Field			
	Last time activity to the destination was recorded on any device				
	NXQL ID:	last_seen			
Lowest observed web protocol version	Activity	Aggregate			
	Lowest protocol version observed in web requests (excluding web requests with unknown protocol version)				
	NXQL ID:	lowest_protocol_version			
Name	Properties	Field			
	Reverse lookup name				
	NXQL ID:	name			
Network availability level	Availability	Aggregate			
	Indicates the ratio of successful TCP connections. The possible values are: • high: the ratio is greater or equal to 98% • medium: the ratio is greater or equal to 90% and less than 98% • low: the ratio is lower than 90%				
	NXQL ID:	network_availability_level			
Number of applications	Inventory	Aggregate			
	Number of applications				
	NXQL ID:	number_of_applications			
Number of binaries	Inventory	Aggregate			
	Number of binaries				
	NXQL ID:	number_of_binaries			
Number of connections	Activity	Aggregate			
	Number of connections				
	NXQL ID:	number_of_connections			
Number of devices	Inventory	Aggregate			
	Number of devices				

	NXQL ID:	number_of_devices			
Number of domains	Inventory	Aggregate			
	Number of domains				
	NXQL ID:	number_of_domains			
Number of executables	Inventory	Aggregate			
	Number of executables				
	NXQL ID:	number_of_executables			
Number of ports	Inventory	Aggregate			
	Number of ports				
	NXQL ID:	number_of_ports			
Number of users	Inventory	Aggregate			
	Number of users				
	NXQL ID:	number_of_users			
Number of web requests	Activity	Aggregate			
	Number of web requests				
	NXQL ID:	number_of_web_requests			
Outgoing network traffic	Traffic	Aggregate			
	Total network outgoing traffic				
	NXQL ID:	outgoing_traffic			
Outgoing network traffic per device	Traffic	Aggregate			
	Indicates the outgoing network traffic divided by the number of devices.				
	NXQL ID:	outgoing_network_traffic_per_device			
Outgoing web traffic	Traffic	Aggregate			
	Total web outgoing traffic				
	NXQL ID:	outgoing_traffic			
Outgoing web traffic per device	Traffic	Aggregate			
	Indicates the outgoing web traffic divided by the number of devices.				
	NXQL ID:	outgoing_web_traffic_per_device			
Protocols used in web requests	Activity	Aggregate			
	Protocols used in web requests (HTTP, TLS, HTTP/TLS)				
	NXQL ID:	protocols_used_in_requests			
Successful HTTP requests ratio	Availability	Aggregate			
	Percentage of successful HTTP requests (1xx, 2xx and 3xx)				

	NXQL ID:	successful_http_requests_ratio			
Successful network connections ratio	Availability	Aggregate			
	Percentage of successful TCP connections				
	NXQL ID:	successful_connections_ratio			
Total network traffic	Traffic	Aggregate			
	Total network traffic (incoming and outgoing)				
	NXQL ID:	total_network_traffic			
Total web traffic	Traffic	Aggregate			
	Total web traffic (incoming and outgoing)				
	NXQL ID:	total_web_traffic			
UID	Properties	Field			
	Indicates the universally unique identifier (based on destination ip address).				
Web interaction time	Activity	Aggregate			
	Indicates the time during which at least one executable is doing HTTP or TLS traffic. This is counted with a 5-minute resolution.				
	NXQL ID:	cumulated_web_interaction_duration			

Domain

Domain names

Field	Group	Type			
Activity start time	Activity	Aggregate			
	Start time of investigated activity				
	NXQL ID:	activity_start_time			
Activity stop time	Activity	Aggregate			
	Stop time of investigated activity				
	NXQL ID:	activity_stop_time			
Average incoming web bitrate	Availability	Aggregate			
	Average incoming bitrate of all underlying web requests, consolidated over time				
	NXQL ID:	average_incoming_bitrate			
Average outgoing web bitrate	Availability	Aggregate			

	Average outgoing bitrate of all underlying web requests, consolidated over time				
	NXQL ID:	average_outgoing_bitrate			
Average web request duration	Availability	Aggregate			
	Average time between request and last response byte				
	NXQL ID:	average_request_duration			
Average web request size	Traffic	Aggregate			
	Average size of web requests				
	NXQL ID:	average_request_size			
Average web response size	Traffic	Aggregate			
	Average size of web responses				
	NXQL ID:	average_response_size			
Database usage	Properties	Field			
	Indicates the percentage of the Engine database used by the domain.				
	NXQL ID:	database_usage			
Domain category	Properties	Field			
	Indicates the category of the domain: • '-' : not yet tagged or internal domain				
	NXQL ID:	domain_category			
First seen	Properties	Field			
	The first time the domain has been seen				
	NXQL ID:	first_seen			
Highest local privilege level reached	Activity	Aggregate			
	Highest local privilege level reached for executions (user, power user, administrator)				
	NXQL ID:	highest_local_privilege_reached			
Hosting country	Properties	Field			
	Indicates in which country the domain is hosted: • '-' : not yet tagged, internal domain or not known by Nexthink Library				
	NXQL ID:	hosting_country			
Hostname	Properties	Field			
	The hostname of the fully qualified domain name				

	NXQL ID:	hostname			
ID	Properties	Field			
	Unique domain identifier code				
	NXQL ID:	id			
Incoming web traffic	Traffic	Aggregate			
	Total web incoming traffic				
	NXQL ID:	incoming_traffic			
Incoming web traffic per device	Traffic	Aggregate			
	Indicates the incoming web traffic divided by the number of devices.				
	NXQL ID:	incoming_web_traffic_per_device			
Internal domain	Properties	Field			
	Indicates whether the domain is considered internal: • yes: the domain is not reported to Nexthink Library and subdomains are not compressed using the '*' pattern • no: the domain is reported to the Nexthink Library (if the license includes the Security module); complex subdomains are compressed using the '*' pattern				
	NXQL ID:	internal_domain			
Last seen	Properties	Field			
	The last time the domain has been seen				
	NXQL ID:	last_seen			
Lowest observed web protocol version	Activity	Aggregate			
	Lowest protocol version observed in web requests (excluding web requests with unknown protocol version)				
	NXQL ID:	lowest_protocol_version			
Name	Properties	Field			
	The fully qualified domain name				
	NXQL ID:	name			
Number of applications	Inventory	Aggregate			
	Number of applications				
	NXQL ID:	number_of_applications			
Number of binaries	Inventory	Aggregate			
	Number of binaries				

	NXQL ID:	number_of_binaries			
Number of destinations	Inventory	Aggregate			
	Number of destinations				
	NXQL ID:	number_of_destinations			
Number of devices	Inventory	Aggregate			
	Number of devices				
	NXQL ID:	number_of_devices			
Number of executables	Inventory	Aggregate			
	Number of executables				
	NXQL ID:	number_of_executables			
Number of ports	Inventory	Aggregate			
	Number of ports				
	NXQL ID:	number_of_ports			
Number of users	Inventory	Aggregate			
	Number of users				
	NXQL ID:	number_of_users			
Number of web requests	Activity	Aggregate			
	Number of web requests				
	NXQL ID:	number_of_web_requests			
Outgoing web traffic	Traffic	Aggregate			
	Total web outgoing traffic				
	NXQL ID:	outgoing_traffic			
Outgoing web traffic per device	Traffic	Aggregate			
	Indicates the outgoing web traffic divided by the number of devices.				
	NXQL ID:	outgoing_web_traffic_per_device			
Protocols used in web requests	Activity	Aggregate			
	Protocols used in web requests (HTTP, TLS, HTTP/TLS)				
	NXQL ID:	protocols_used_in_requests			
Storage policy	Properties	Field			
	Event storage policy for the domain (web request or none)				
	NXQL ID:	storage			
Successful HTTP requests ratio	Availability	Aggregate			
	Percentage of successful HTTP requests (1xx, 2xx and 3xx)				
	NXQL ID:	successful_http_requests_ratio			

Threat level	Properties	Field			
	Indicates the threat level of the domain: • '-' : not yet tagged or internal domain • none detected: no known threat • low: low threat • intermediate: intermediate threat • high: high threat				
	NXQL ID:	threat_level			
Total web traffic	Traffic	Aggregate			
	Total web traffic (incoming and outgoing)				
	NXQL ID:	total_web_traffic			
UID	Properties	Field			
	Indicates the universally unique identifier (based on domain name).				
Web interaction time	Activity	Aggregate			
	Indicates the time during which at least one executable is doing HTTP or TLS traffic. This is counted with a 5-minute resolution.				
	NXQL ID:	cumulated_web_interaction_duration			

Printer

Installed printers (local, network, shared or virtual)

Field	Group	Type			
Display name	Properties	Field			
	Most frequently seen display name				
	NXQL ID:	real_name			
First seen	Properties	Field			
	First time activity of the printer was recorded on any device				
	NXQL ID:	first_seen			
Hostname	Properties	Field			
	Indicates where the printer is hosted: • for local and smb printers: the hostname of the device the printer				

	is connected to • for tcp/ip and wsd printers: usually the hostname of the printer itself				
	NXQL ID:	host_name			
ID	Properties	Field			
	Unique print identifier code				
	NXQL ID:	id			
Last seen	Properties	Field			
	Last time activity of the printer was recorded on any device				
	NXQL ID:	last_seen			
Location	Properties	Field			
	Printer location				
	NXQL ID:	location			
Model	Properties	Field			
	Printer model				
	NXQL ID:	model			
Name	Properties	Field			
	Unique printer name				
	NXQL ID:	name			
Number of devices	Inventory	Aggregate			
	Number of devices				
	NXQL ID:	number_of_devices			
Number of print jobs	Activity	Aggregate			
	Number of print jobs				
	NXQL ID:	number_of_printouts			
Number of printed pages	Activity	Aggregate			
	Number of printed pages				
	NXQL ID:	number_of_printed_pages			
Number of users	Inventory	Aggregate			
	Number of users				
	NXQL ID:	number_of_users			
Type	Properties	Field			
	The type of the printer:				

	• local: a locally connected or virtual printer • tcp/ip: a printer connected through a TCP/IP port • smb: a printer connected through a SMB (Server Message Block) port • wsd: a printer connected through a WSD (Web Services for Devices) port	
	NXQL ID:	type
UID	Properties	Field
	Indicates the universally unique identifier (based on printer name and model).	

Activities

Activities represent actions performed by Objects.

Installation

Installations or uninstallations of software packages

Field	Group	Type
Device ID	Device	Field
	Unique identifier code of the installation target device	
Device name	Device	Field
	Indicates the name of the device: • For Windows: NetBios Name • For Mac OS: computer name used on the network	

	• For Mobile: composed by mailbox name and device friendly name					
Device SID	Device	Field				
	Windows security identifier of the installation target device					
ID	Properties	Field				
	Unique installation identifier code					
	NXQL ID:	id				
Operation type	Properties	Field				
	Type of operation (installation, uninstallation)					
	NXQL ID:	type				
Package ID	Package	Field				
	Unique identifier code of the installed package					
Package name	Package	Field				
	Name of the installed package					
Package program	Package	Field				
	Program of the installed package					
Package publisher	Package	Field				
	Name of the installed package publisher					
Package type	Package	Field				
	Package type: • program • update (Windows only)					
Package version	Package	Field				
	Version of the installed package					
Time of installation	Properties	Field				
	Installation start time					

	NXQL ID:	time

Execution

Executing processes (merged when in close succession)

Field	Group	Type			
Application name	Application	Field			
	Executed application name				
Average memory usage	Activity	Field			
	Indicates the average memory usage of the underlying executions before aggregation with a sampling resolution of 5 minutes. • Example: if two tabs of the Chrome browser are opened at the same time, two distinct processes of chrome.exe are launched and they are aggregated by the Engine (i.e., event cardinality = 2). The average memory usage will be the average of the two processes before aggregation: it represents the average memory usage of a single Chrome tab.				
	NXQL ID:	average_memory_usage			
Binary path	Application	Field			
	Executed binary path				
	NXQL ID:	binary_path			
Binary version	Application	Field			
	Executed binary version				
Cardinality	Properties	Field			
	Number of underlying processes, consolidated over time				
	NXQL ID:	cardinality			
Device ID	Device	Field			
	Unique identifier code of the executing device				

Device IP addresses	Device	Field			
	List of IP addresses of the executing device				
Device name	Device	Field			
	Indicates the name of the device: • For Windows: NetBios Name • For Mac OS: computer name used on the network • For Mobile: composed by mailbox name and device friendly name				
Device SID	Device	Field			
	Windows security identifier of the executing device				
Duration	Properties	Field			
	Total execution duration				
	NXQL ID:	duration			
End time	Properties	Field			
	Execution end time				
	NXQL ID:	end_time			
Executable name	Application	Field			
	Executed executable name				
ID	Properties	Field			
	Unique execution identifier code				
	NXQL ID:	id			
Incoming TCP traffic	Traffic	Field			
	Incoming TCP traffic				
	NXQL ID:	incoming_tcp_traffic			
Lifespan	Properties	Field			
	Execution lifespan in relation to investigation time frame				
Outgoing TCP traffic	Traffic	Field			
	Outgoing TCP traffic				
	NXQL ID:	outgoing_tcp_traffic			
Outgoing UDP traffic	Traffic	Field			
	Outgoing UDP traffic				
	NXQL ID:	outgoing_udp_traffic			

Privilege level	Properties	Field			
	Privilege level of the execution (user, power user, administrator)				
	NXQL ID:	privilege_level			
Signature ID	Properties	Field			
	ID of the related execution signature, i.e. a user executing a certain process on a particular device				
	NXQL ID:	usage			
Start time	Properties	Field			
	Execution start time				
	NXQL ID:	start_time			
Status	Properties	Field			
	Status of the execution (started, stopped)				
	NXQL ID:	status			
Total CPU time	Properties	Field			
	Indicates the sum of the CPU time of all executions (before aggregation by the Engine) over all logical processors. • Example: if we consider two executions that are launched at the same time (hence aggregated by the Engine), with the first one taking 50% of a logical processor during 30 minutes and the second one taking 100% of 2 logical processors during 60 minutes, the total CPU time is 135 minutes (= 50% * 30 min + 2 * 100% * 60 min).				
	NXQL ID:	total_cpu_time			
User ID	User	Field			
	Unique identifier code of the executing user				
User name	User	Field			
	Name of the executing user				
User SID	User	Field			
	Indicates the Windows security identifier for the user.				

	• For Mac 0S: the value is 'S-0-0' if the user is not in Active Directory

Connection

TCP or UDP connections (merged when in close succession)

Field	Group	Type			
Application name	Application	Field			
	Name of the connecting application				
Average network response time	Availability	Field			
	Indicates the average TCP connection establishment time. The value is the average over all underlying connections before aggregation.				
	NXQL ID:	network_response_time			
Binary paths	Application	Field			
	Paths of the connecting binary				
Binary version	Application	Field			
	Version of the connecting binary				
Cardinality	Properties	Field			
	Number of underlying connections, consolidated over time				
	NXQL ID:	cardinality			
Connection type	Properties	Field			
	Type of the connection (tcp, udp, tcp network scan, tcp port scan, udp network scan, udp port scan)				
	NXQL ID:	type			
Destination IP address	Destination	Field			
	IP address of the connection destination				
	NXQL ID:	destination_ip_address			
Destination name	Destination	Field			
	Name of the connection destination				
Device ID	Device	Field			

	Unique identifier code of the connecting device					
Device IP address	Device	Field				
	IP address of the connecting device					
	NXQL ID:	device_ip_address				
Device name	Device	Field				
	Indicates the name of the device: • For Windows: NetBios Name • For Mac OS: computer name used on the network • For Mobile: composed by mailbox name and device friendly name					
Device SID	Device	Field				
	Windows security identifier for the connecting device					
Duration	Properties	Field				
	The time between the start of the first connection and end of the last underlying connection					
	NXQL ID:	duration				
End time	Properties	Field				
	Connection end time, corresponding to the moment when the last underlying TCP connection was closed					
	NXQL ID:	end_time				
Executable name	Application	Field				
	Name of the connecting executable					
ID	Properties	Field				
	Unique connection identifier code					
	NXQL ID:	id				
Incoming bitrate	Availability	Field				
	Average incoming bitrate of all underlying connections, consolidated over time					
	NXQL ID:	incoming_bitrate				
Incoming TCP traffic	Traffic	Field				
	Incoming TCP traffic					
Lifespan	Properties	Field				

	Connection lifespan in relation to investigation time frame				
Outgoing bitrate	Availability	Field			
	Average outgoing bitrate of all underlying connections, consolidated over time				
	NXQL ID:	outgoing_bitrate			
Outgoing TCP traffic	Traffic	Field			
	Outgoing TCP traffic				
Outgoing UDP traffic	Traffic	Field			
	Outgoing UDP traffic				
Port number	Port	Field			
	Port number of the connection				
Signature ID	Properties	Field			
	ID of the related connection signature, i.e. a user executing a certain process on a particular device which connects to a certain destination/port				
	NXQL ID:	signature_id			
Start time	Properties	Field			
	Connection start time				
	NXQL ID:	start_time			
Status	Properties	Field			
	Status of the connection (established, rejected, no service, no host, closed)				
	NXQL ID:	status			
User ID	User	Field			
	Unique identifier code of the connecting user				
User name	User	Field			
	Name of connecting user				
User SID	User	Field			
	Indicates the Windows security identifier for the user. • For Mac 0S: the value is 'S-0-0' if the user is not in Active Directory				

Web request

HTTP or TLS requests

Field	Group	Type			
Application name	Application	Field			
	Name of the application which made the web request				
Binary paths	Application	Field			
	Paths of the binary which made the web request				
Binary version	Application	Field			
	Version of the binary which made the web request				
Cardinality	Properties	Field			
	Number of underlying web requests, consolidated over time				
	NXQL ID:	cardinality			
Connections duration	Properties	Field			
	The time between start of the first connection and end of the last underlying connection				
	NXQL ID:	connections_duration			
Device ID	Device	Field			
	Unique identifier code of the web request source				
Device name	Device	Field			
	Indicates the name of the device: • For Windows: NetBios Name • For Mac OS: computer name used on the network • For Mobile: composed by mailbox name and device friendly name				
Device SID	Device	Field			
	Windows security identifier of the web request source				
Domain name	Domain	Field			

	Name of the web request destination domain				
End time	Properties	Field			
	Web request end time, corresponding to the moment when the last underlying TCP connection was closed				
	NXQL ID:	end_time			
Executable name	Application	Field			
	Name of the executable which made the web request				
HTTP status	Properties	Field			
	HTTP response status code				
	NXQL ID:	http_status			
ID	Properties	Field			
	Unique request identifier code				
	NXQL ID:	id			
Incoming web traffic	Traffic	Field			
	Incoming web traffic of all underlying web requests, consolidated over time				
	NXQL ID:	incoming_traffic			
Network response time	Availability	Field			
	Average TCP connection establishment time of all underlying connections, consolidated over time				
	NXQL ID:	network_response_time			
Outgoing web traffic	Properties	Field			
	Outgoing web traffic of all underlying web requests, consolidated over time				
	NXQL ID:	outgoing_traffic			
Port number	Port	Field			
	Port number of the web request				
Protocol	Properties	Field			
	Web request protocol (HTTP, TLS)				
	NXQL ID:	protocol			
Protocol version	Properties	Field			
	Web request protocol version				
	NXQL ID:	protocol_version			
Service related	Properties	Field			

	Indicates whether the web request is related to a configured service: • yes: these requests are always visible by all users • no: depending on the privacy settings, requests not related to a service might not be visible by everyone				
	NXQL ID:	service_related			
Signature ID	Properties	Field			
	ID of the related web request signature, i.e. a user executing a certain process on a particular device which emits requests to a specific domain				
	NXQL ID:	signature_id			
Start time	Properties	Field			
	Web request start time				
	NXQL ID:	start_time			
URL path	Properties	Field			
	Indicates the expression used to match the web request against web-based services with URL path: `'-': the web request did not match against any service with URL path`				
User ID	User	Field			
	Unique identifier code of the user who made the web request				
User name	User	Field			
	Name of the user who made the web request				
User SID	User	Field			
	Indicates the Windows security identifier for the user who made the web request. • For Mac 0S: the value is 'S-0-0' if the user is not in Active Directory				
Web request duration	Properties	Field			
	Average time between request and last response byte of all underlying requests, consolidated over time				

	NXQL ID:	web_request_duration

Print job

Print job submissions to printer drivers

Field	Group	Type			
Color enabled	Properties	Field			
	Indicates whether the print job has the capability to print in color. Color settings defined by the application performing the print job (usually through the application print dialog) are not taken into account.				
	NXQL ID:	color_print			
Device ID	Device	Field			
	Unique identifier of the print job source				
Device name	Device	Field			
	Indicates the name of the device: • For Windows: NetBios Name • For Mac OS: computer name used on the network • For Mobile: composed by mailbox name and device friendly name				
Device SID	Device	Field			
	Windows security identifier of the print job source				
Document type	Properties	Field			
	Type of printed document				
	NXQL ID:	document_type			
Duplex print	Properties	Field			
	Indicates whether the pages are printed on both sides of the sheet.				
	NXQL ID:	duplex			
ID	Properties	Field			
	Unique print job identifier				
	NXQL ID:	id			
Number of pages	Properties	Field			

	Number of printed pages				
	NXQL ID:	number_of_printed_pages			
Paper size	Properties	Field			
	Paper size for printed pages				
	NXQL ID:	page_size			
Print quality	Properties	Field			
	Print quality				
	NXQL ID:	print_quality			
Printer model	Printer	Field			
	Model of printer				
Printer name	Printer	Field			
	Name of printer				
Size	Properties	Field			
	Print job size in bytes				
	NXQL ID:	size			
Status	Properties	Field			
	Status of the print job: • success: the job has been successfully sent to the printer • error: an error was detected during the print; the job might have been partially printed • unknown: the status of the print job could not be detected				
	NXQL ID:	status			
Time	Properties	Field			
	Print job time				
	NXQL ID:	time			
User ID	User	Field			
	Unique identifier code of the printing user				
User name	User	Field			
	Name of the printing user				
User SID	User	Field			
	Indicates the Windows security identifier for the user.				

	• For Mac 0S: the value is 'S-0-0' if the user is not in Active Directory

System boot

System boots (timed between kernel start and launch of 'logonui.exe' process)

Field	Group	Type			
Device ID	Device	Field			
	Unique device identifier				
Device IP addresses	Device	Field			
	List of IP addresses for the device				
Device name	Device	Field			
	Indicates the name of the device: • For Windows: NetBios Name • For Mac OS: computer name used on the network • For Mobile: composed by mailbox name and device friendly name				
Device SID	Device	Field			
	Windows security identifier of the device				
Duration	Properties	Field			
	Indicates the time between the kernel start and the launch of the 'logonui.exe' process				
ID	Properties	Field			
	Boot event identifier				
Time	Properties	Field			

	Time of boot

User logon

User logons

Field	Group	Type			
Device ID	Device	Field			
	Unique device identifier code				
Device IP addresses	Device	Field			
	List of IP addresses for the device				
Device name	Device	Field			
	Indicates the name of the device: • For Windows: NetBios Name • For Mac OS: computer name used on the network • For Mobile: composed by mailbox name and device friendly name				
Device SID	Device	Field			
	Windows security identifier of the device				
Duration	Properties	Field			
	Indicates the time between the user logging on and the desktop being shown.				
Extended duration	Properties	Field			
	Indicates the time between the user logging on and the device being ready to use. Desktops and laptops are considered fully functional				

	once the CPU usage drops below 15% and the disk usage drops below 80%, and servers once the CPU usage of all processes belonging to the corresponding user drops below 15%.				
ID	Properties	Field			
	User logon event identifier code				
Time	Properties	Field			
	Time of user logon				
User ID	User	Field			
	Unique user identifier code				
User name	User	Field			
	Name of user				
User SID	User	Field			
	Indicates the Windows security identifier for the user. • For Mac 0S: the value is 'S-0-0' if the user is not in Active Directory				

Events

Events are warning or errors.

Device warning

Peaks in system resource usage (CPU, memory or I/O)

Field	Group	Type			
Device ID	Device	Field			

	Unique device identifier				
	NXQL ID:	device			
Device IP addresses	Device	Field			
	List of IP addresses for the device				
Device name	Device	Field			
	Indicates the name of the device: • For Windows: NetBios Name • For Mac OS: computer name used on the network • For Mobile: composed by mailbox name and device friendly name				
Device SID	Device	Field			
	Windows security identifier of the device				
Duration	Properties	Field			
	Indicates the duration of the event.				
	NXQL ID:	duration			
End time	Properties	Field			
	Performance event end time				
	NXQL ID:	end_time			
Event info	Properties	Field			
	Performance event information				
	NXQL ID:	info			
High io usage	Warnings	Field			
	High io usage				
High memory usage	Warnings	Field			
	High memory usage				
High overall CPU usage	Warnings	Field			
	High overall CPU usage.				
High page faults	Warnings	Field			
	High number of page faults				
High thread CPU usage (deprecated)	Warnings	Field			
	High thread CPU usage (deprecated).				
ID	Properties	Field			

	Unique performance event identifier				
	NXQL ID:	id			
Start time	Properties	Field			
	Performance event start time				
	NXQL ID:	start_time			
Warning duration	Properties	Field			
	Indicates the effective duration of the warning; it can be shorter than the event duration when the high CPU usage is not continuous. • Example: a high CPU usage warning started at 08:00 and lasted until 08:15 (event duration is 15 min). During this 15min, the device was effectively in high CPU usage once during 60s, twice during 120s and once during 30s; the warning duration is therefore 5min 30s.				
	NXQL ID:	warning_duration			

Device error

Critical system errors (system crash, hard reset, or disk failure)

Field	Group	Type			
Device ID	Device	Field			
	Unique device identifier code				
	NXQL ID:	device			
Device IP addresses	Device	Field			
	List of IP addresses for the device				
Device name	Device	Field			
	Indicates the name of the device:				

	• For Windows: NetBios Name • For Mac OS: computer name used on the network • For Mobile: composed by mailbox name and device friendly name				
Device SID	Device	Field			
	Windows security identifier of the device				
Error code	Properties	Field			
	Indicates the error code for system crashes (Windows bluescreens).				
	NXQL ID:	error_code			
Error label	Properties	Field			
	Indicates the error label for system crashes (Windows bluescreens).				
	NXQL ID:	error_label			
ID	Properties	Field			
	Problem identifier code				
	NXQL ID:	id			
Time	Properties	Field			
	Time of error				
	NXQL ID:	time			
Type	Properties	Field			
	Indicates the device error type, with the following possible values: • system crash: a Windows bluescreen • hard reset: the device was abruptly stopped and then rebooted. It might be caused by pressing the reset button, a power				

	failure or a crash • SMART disk failure: a disk error was detected on a disk with SMART technology	
	NXQL ID:	type

Execution warning

Peaks in application resource usage (CPU or memory)

Field	Group	Type			
Application name	Application	Field			
	Name of application				
Binary version	Application	Field			
	Version of binary				
Device ID	Device	Field			
	Unique device identifier				
Device IP addresses	Device	Field			
	List of IP addresses for the device				
Device name	Device	Field			
	Indicates the name of the device: • For Windows: NetBios Name • For Mac OS: computer name used on the network • For Mobile: composed by mailbox name and device friendly name				
Device SID	Device	Field			
	Windows security identifier of the device				
Duration	Properties	Field			
	Indicates the duration of the event.				
	NXQL ID:	duration			
End time	Properties	Field			

	Performance event end time				
	NXQL ID:	end_time			
Event info	Properties	Field			
	Performance event information				
Executable name	Application	Field			
	Name of executable				
High memory usage	Warnings	Field			
	High memory usage				
High thread CPU usage	Warnings	Field			
	High thread CPU usage				
ID	Properties	Field			
	Unique performance event identifier code				
	NXQL ID:	id			
Signature ID	Properties	Field			
	ID of the related execution event signature, i.e. a user executing a certain process on a particular device				
	NXQL ID:	signature_id			
Start time	Properties	Field			
	Performance event start time				
	NXQL ID:	start_time			
User ID	User	Field			
	Unique user identifier code				
User name	User	Field			
	Name of user				
User SID	User	Field			
	Indicates the Windows security identifier for the user. • For Mac 0S: the value is 'S-0-0' if the user is not in Active Directory				
Warning duration	Properties	Field			
	Indicates the effective duration of the warning; it can be shorter than the event duration when the high				

	CPU usage is not continuous. • Example: a high CPU usage warning started at 08:00 and lasted until 08:15 (event duration is 15 min). During this 15min, the device was effectively in high CPU usage once during 60s, twice during 120s and once during 30s; the warning duration is therefore 5min 30s.
	NXQL ID: warning_duration

Execution error

Application errors (crash or not responding)

Field	Group	Type			
Application name	Application	Field			
	Name of application				
Binary version	Application	Field			
	Version of binary				
Device ID	Device	Field			
	Unique device identifier code				
Device IP addresses	Device	Field			
	List of IP addresses for the device				
Device name	Device	Field			
	Indicates the name of the device: • For Windows: NetBios Name • For Mac OS: computer name used on the network • For Mobile: composed by mailbox name and				

	device friendly name				
Device SID	Device	Field			
	Windows security identifier of the device				
Executable name	Application	Field			
	Name of executable				
ID	Properties	Field			
	Error identifier code				
	NXQL ID:	id			
Signature ID	Properties	Field			
	ID of the related execution error signature, i.e. a user executing a certain process on a particular device				
	NXQL ID:	signature_id			
Time	Properties	Field			
	Time of error				
	NXQL ID:	time			
Type	Properties	Field			
	Error type				
	NXQL ID:	type			
User ID	User	Field			
	Unique user identifier code				
User name	User	Field			
	Name of user				
User SID	User	Field			
	Indicates the Windows security identifier for the user. • For Mac 0S: the value is 'S-0-0' if the user is not in Active Directory				

Local and shared content

Overview

When a user of the Finder adds new content to an Engine, some types of items are created locally in the Engine. These items belong exclusively to the user that created them, meaning that only that particular user can see the items, use them or modify them when connecting to the same Engine.

Other types of items are directly added to a centralized content manager instead. The content manager lives in the same Appliance as the Portal and has access to all the Engines managed by the Portal. Items added to the content manager are instantly shared by all Engines after their creation in the Finder. For that reason, content shared in this fashion is also called *centralized* content. As a result, any user can see the items, use them or, if the user has enough privileges, modify them from their own instance of the Finder, as long as they are connected to the same Portal.

Finally, a few types of items are neither stored in the Engine nor in the content manager of the Portal, but in the computer that runs the Finder. You can think of these items as a kind of Finder preferences.

Classification

The following table shows the lists of items that are created locally and those which are shared:

Local to the Finder	Local to the Engine	Shared (centralized in the Portal)
• Sessions • Custom actions	• Investigations • Investigation-based alerts • One-click investigations • Web API investigations (deprecated) • Tags	• Categories and keywords • Metrics • Services • Campaigns • Scores

Sharable content

Local and centralized content

Even if some content is created locally in the Engine, you can still share it manually by exporting items to the clipboard or to a file.

Manually export and import your investigations, one-clicks, alerts, categories, metrics, scores, and services from the Finder.

Custom actions

Although local to the Finder, custom actions are also exportable. Share your custom actions with Finders installed in other computers by exporting them to XML files:

1. Click the sprocket icon in the top right part of the Finder window.
2. Select **Custom actions...** to display the list of available custom actions.
3. Select one or more entries in the list by clicking on them. Use **Ctrl+click** to select more than one entry, **Shift+click** to select a group of consecutive entries, or **Ctrl+A** to select them all.
4. Right-click your selection and choose **Export...** from the menu.
5. Type in a name for the XML file.
6. Click **Save**.

To import custom actions into a Finder, click the **Import...** button at the bottom of the list of custom actions and select the file to import. If an imported custom action exists already in the list, it is duplicated.

Tags

As a final remark, note that tags are neither centralized nor shareable. Tags are the result of applying a keyword to an object either manually, automatically, or with the help of text files. Objects (devices, destinations, etc) live in the database of each Engine; therefore, objects are local to an Engine, and so are the tags applied to an object. The locality of tags has some important implications that we explain with the following example.

Suppose that you have a setup with two Engines and one Portal, and in both Engines there is a destination with the same IP address. In fact, even if it is logically the same destination, there are two destination objects: one per Engine. If you create a category and a keyword to automatically tag the destinations with that IP address as, for instance, a *Mail server*, both Engines will tag the destination identically, though separately. Indeed, categories and keywords are centralized, so the auto-tagging condition on the IP address applies to all Engines.

Let us imagine that you connect to one of the Engines and that you manually tag the mentioned destination as a *Proxy server*, overriding the auto-tagging rule. Now you find that the same destination is tagged as *Mail server* in one Engine

and as *Proxy server* in the other, which is probably not what you want. Therefore, be careful when you apply tags manually to objects, because tags are not shared among Engines.

Centralizing local content via roles

Administrators can assign investigations, one-click investigations, and investigation-based alerts to other users by making them role-based. Once linked to a role, the items are seen by all the users playing that role. To add investigations, one-clicks, or alerts to a role, administrators use the method seen above for exporting their content to the clipboard.

Related tasks

- Manually sharing Finder content
- Tagging objects manually
- Tagging objects automatically
- Importing tags from text files

Related references

- Adding users (defining user roles)

Device Identification

Overview

To update their database with consistent information, Engines must correctly identify the different devices from which they receive Collector data. The Engine is able to distinguish devices from one another thanks precisely to the hardware information and operating system-level data sent by the Collectors.

Because device hardware may get upgraded and the device data stored at the operating system-level may change with time, the Engine uses an algorithm to either recognize a device to be the same as a device seen before, despite possible minor changes, or decide that a new device joined the network. Failing to correctly identify a device may result in a single device being split into two or in two different devices being merged into one in the database of the Engine.

To prevent Engines from misidentifying special groups of devices, such as those in virtualized environments, replace the default identification algorithm by an

algorithm exclusively based on the name of the device, as seen by the operating system. Apply this name-based recognition method to groups of devices selected by name patterns.

The methods to identify devices described in this article do not apply to mobile devices.

Applies to platforms:

Default algorithm to identify a device

To identify a device, the default algorithm considers the following pieces of information:

- The name of the device, as reported by the operating system.
- A hardware identifier that is derived from:
    - The BIOS serial number.
    - The chassis serial number.
    - The motherboard serial number.
- The MAC addresses of the network adapters that are enabled on the device.
- The Machine SID of the device.

Considerations about the data that identifies a device

Devices that have not joined a domain may share the same name. For devices in a domain, the name of the device is unique at a given time within a given domain. Name uniqueness is ensured by the domain controller, but two different devices may have the same name at different points in time.

The list of MAC addresses that are enabled by the operating system change whenever a network adapter is added or removed.

The derived hardware identifier is usually unique for branded PCs but it may not be unique for no name or self-assembled PC. In the case of devices being virtual machines, VMWare defines a BIOS serial number that is unique and thus yields a valid hardware id.

The Machine SID of a Windows device is the Security Identifier of the Windows operating system. The SID is generated during the Windows installation process and is supposed to be globally unique. However if Windows is installed using a cloned image which has not been carefully crafted using sysprep, the SID may not be unique. Experience shows that SIDs are rarely unique within corporate network and they appear in bunches of 10 to 50 machines.

How the device identification algorithm works

The exact identification algorithm is quite intricate; therefore, it is not described here in detail, but only sketched out. Basically, when the Collector sends to the Engine all the pieces of information about a device mentioned above, the device identification algorithm compares them with the corresponding data of each device that is already present in the database of the Engine:

- If the received information precisely matches that of an existing device, the algorithm concludes that the information belongs to the same device that is already in the database.
- If most of the information at least partially matches that of an existing device, in a majority of cases the algorithm still concludes that the information belongs to the same device. The Engine updates therefore the existing device with the received information. For instance, if the received hardware id, MAC addresses and SID all match those of an existing device, but the received name is different from the name of the device as recorded in the database, the algorithm determines that it is the same device and updates its name in the database.
- If the received information differs significantly from that of any of the existing devices, the Engine adds a new device to its database.

Identifying devices solely by their name

Starting from the Engine release V5.3.3, it is possible to override the default algorithm to identify devices and instruct the Engine to exclusively identify Windows devices with domain membership by their name. From release V6.8 on, the feature has been extended to support all devices regardless of their platform (Windows or Mac) and membership type.

Note that the default device identification algorithm should be preferred in most cases. Use this alternative method only in setups where the default algorithm fails to reliably identify a specific group of devices. A misconfiguration may lead to devices being artificially merged or split, so use the identification of devices by name carefully.

This feature is particularly useful in virtualized environments, where devices are virtual machines (VMs) recreated at every user session. By applying the default algorithm for identifying devices, the Engine regards every new instance of a VM as a new device and ends up with multiple devices that share the same name and that succeed each other over time. By identifying devices on the basis of their name only, the Engine consistently maps a particular VM to a single device time after time, even when its hardware properties change.

The Engine provides a mechanism to apply the name algorithm to a reduced set of devices only. Specify name patterns in the configuration file of the Engine to select those groups of devices that must be identified by their name. For instance, if the name of all your virtual machines begins with **vm1-ws** or **vm2-ws**, add the following lines to the configuration file of the Engine (**/var/nexthink/engine/01/etc/nxengine.xml**):

```
<config>
  <engine>
    <device_identification>
       <netbios_pattern>vm1-ws*</netbios_pattern>
       <netbios_pattern>vm2-ws*</netbios_pattern>
    </device_identification>
  </engine>
</config>
```

Valid substitution characters in the name patterns (called `netbios_pattern` for historical reasons) are:

- The asterisk ***** to substitute for zero or more characters.
- The question mark **?** to substitute for one single character.

Timestamping of events

Overview

Learn how the Engine computes the timestamps of the events in its database by combining the time of reception of the packets sent by the Collectors with the individual time information of each event stored inside every packet.

Timestamping in the Collector

The Collector reacts to events of interest by recording them into memory. Later, it sends the collected events to the Engine either periodically or when it has accumulated a sufficient quantity of events. To detect activity in the system where it is installed, the Collector employs different techniques, such as intercepting system calls, that allow it to precisely determine the moment at which the event takes place.

This timestamping of the events in the device of the end-user is done according to the time elapsed since the last system boot. Therefore, the Collector is using

relative time to timestamp events in the machine of the end-user. The fact that the time used by the Collector is relative is not important for computing a precise timestamp in the Engine, as you will see below.

Once the Collector has gathered enough events, along with their corresponding timestamps, it builds a network packet and sends it to the Engine. Right before sending it, the Collector sets a timestamp in the packet using again the local time relative to the system boot. Therefore, in every packet sent by the Collector we have:

- The timestamps of each individual event sent in the packet.
- One general timestamp for the packet itself.

The difference between the time of the packet and the time of each individual event is used by the Engine to compute the global timestamp of the events.

Timestamping in the Engine

Once the Engine receives a packet from the Collector, it records the time of reception using the system time. For the recorded time to be correct, the local time of the Engine must be synchronized to an accurate clock. That is one of the reasons why it is recommended to set up NTP in the appliance that hosts the Engine.

For computing the global time of events, the Engine assumes that the transmission time of the Collector packets from the computers of the end-users to the Engine is negligible. In this way, the absolute time at which the Collector sent the packet is considered to be the same absolute time of reception of the packet in the Engine. Therefore, the time that the Collector set in the packet just before sending it is the local time of the end-user machine relative to system boot that is equivalent to the Engine time of reception of the packet. To get the occurrence time of each event, the Engine just has to subtract the difference between the local times of the packet and of each event from the reception time, as shown in the figure below:

Note that the events received in the Engine may not follow a sequential order. The most common case is when you receive two packets in a short interval of time and each packet is coming from a different Collector. Most probably, the two packets have events that overlap in time, but the Engine processes all the events of the second packet after those of the first. The Engine deals well with this situation. Note also that the Engine always inserts events in the past with respect to its current time. This is obvious, for the Engine cannot receive events that have not happened yet. However, for events that lie too far in the past, the Engine will not be able to update the in-memory database, since it would be a too costly operation:

- The Engine rejects events that lie more than **30 minutes** in the past with respect to its present time.

For Collectors in a local network, however, this is a very unlikely case and it often indicates a problem in the device that hosts the Collector.

The case of TCP connections

The Collector treats TCP connections differently from all other events regarding the setting of their timestamp. All other events have their timestamp set as soon as they begin to do some kind of activity. On the other hand, when the end-user device opens a TCP connection to a server, the Collector waits for the connection to be established to set the timestamp of the TCP connection event.

In versions of the Engine previous to 4.4.3, the Engine does not take into account the time for establishing the connection to compute the start time of the event. From version 4.4.3 on, the Engine substracts the connection establishment time from the timestamp of TCP connections to get the actual initial moment of the event.

Boot and logon duration

Overview

The startup time of a device has a direct impact on the productivity and the experience of end-users. Since the first activities that a user performs on a device are to power it on and to log on, users typically have a very negative perception of devices that take too long to start. Indeed, a long boot or logon process are often a symptom of other underlying problems in a device, such as disk failures, network issues, low memory, or general obsolescence. Nexthink

provides the following measurements of the boot and logon duration of a device:

Boot duration
: After powering on the device, the boot duration is the time between the start of the OS kernel and the launch of the logon screen.

Logon duration
: The time between user authentication and the desktop being shown.

Extended logon duration
: The time between user authentication and the device being ready.

Because of the techniques employed in the measurement of boot and logon duration, these values apply to Windows devices only.

Applies to platforms:

Measurement of the boot duration

The measurement of the boot duration begins when the kernel of the operating system loads the Collector driver during its initialization. Once up and running, the Collector notifies the boot of the device and then continuously reports the time elapsed since the kernel started (the system boot, as recorded by the operating system) to the Engine. Any steps in the boot sequence previous to the start of the kernel, such as the BIOS hardware checks and the loading of the kernel itself, are therefore not included in the boot duration. The Engine establishes the absolute boot time of the device according to this information.

In addition to the boot time, the Engine needs to know when the operating system launches the logon screen to compute the boot duration. The launch of the logon screen corresponds to the execution of the system process *logonUI.exe*. Since the Collector successively informs the Engine of the processes being executed in the device, the Engine just needs to wait for the Collector to detect the launch of *logonUI.exe*. The Engine records the interval between the boot time and the start of *logonUI.exe* as the boot duration.

Note that Nexthink records boot events only for *full boot* sequences. Waking up the device after being in a standby (sleep) or a hibernation state is not considered a device boot. Moreover, the boot technique known as *Fast Startup* in Windows 8 (and higher) is not a full boot sequence either; therefore, it is not recorded as such.

Boot duration	
Start	**Stop**

• System boot (as recorded by the OS)	• Start of logon screen (launch of *logonUI.exe*)

Inspecting boot duration through Finder and NXQL

System boot is the activity that holds boot duration information in Nexthink. Look for system boots in the Finder by creating a new investigation:

1. In the bottom of the **Start** page, click **New Investigation**.
2. Under the **Activities** tab, select **system boots**.
3. Fill the conditions and time frame as desired.
4. Ensure that the **Columns** in the **DISPLAY** section include the **Duration** of the system boot.
5. Click **Run**

In NXQL, system boots are a type of the more general **device_activity** events. For instance, to get a list of all boot events, open the NXQL editor and type in:

```
(select * (from device_activity (where device_activity (eq type
(enum boot)))))
```

To get boot duration information for a particular device in the Finder, create an investigation on devices and look for the following columns to display:

- **Last system boot duration**:
  The duration of the last boot measured on the device.
- **System boot duration baseline**:
  The exponentially weighted moving average of the duration of all system boots of the device stored in the Engine database. For a measured boot duration *B(n)* on the *n*th logon, the weighted average *S(n)* is recursively computed with the following formula:
  - *S(n) = B(1)*, for $n = 1$
  - *S(n) = B(n) + (1 -) S(n - 1)*, for $n > 1$
    Where coefficient = 0.1 and the contribution to the duration of old boot events is removed as soon as they are dropped from the history of the database.

  Note that this field is precomputed from all boot events available in the database, so its value does not depend on the time frame of the investigation.

In NXQL, the equivalent for **Last system boot duration** and **System boot duration baseline** are called **last_boot_duration** and **average_boot_duration**, respectively. For instance, to get the last and the baseline boot duration for every device, type in the query:

```
(select (last_boot_duration average_boot_duration) (from device))
```

We have seen that the **System boot duration baseline** is precomputed for a device and does not depend on the time frame. To actually compute the average boot duration of a device over a time frame, add the **Average system boot duration** aggregate to your displayed columns. The average is calculated as the sum of the duration of the boot events divided by the number of boot events within the time frame.

In NXQL, the equivalent for the **Average system boot duration** is the **average_boot_duration** aggregate. Note that it has the same name as the boot duration baseline, but it must be placed inside a **compute** clause in the NXQL query, so there is no confusion possible. For example, to query both for the boot duration baseline and average (for the last day) in an NXQL query, type in:

```
(select average_boot_duration (from device (with device_activity
(compute average_boot_duration) (between now-1d now))))
```

Measurement of the logon and extended logon durations

The moment when the user finishes authenticating, either by typing in their credentials or by any other identification means, marks the start of the logon process. The Collector has two ways to detect the start of the logon process:

- Look in the Security log for an audit logon event.
- Wait for a session creation event.

The preferred method for the Collector to detect a user logon is to look for audit logon events in the Security log of Windows. For the Security log to include logon information, it is necessary that the system administrator activates the corresponding audit policy option. The logon time detected by the Collector in this case matches thus the time recorded by Windows.

Nevertheless, if the audit policy on the device does not include the audit of logon events, the Collector defaults to detecting user logons by listening to session

creation events. Capturing the moment of creation of a session is usually a precise method to determine the time of a user logon. However, in setups with *roaming user profiles*, using this method could yield logon durations that are much shorter than the actual logon duration experienced by users. To avoid sending inaccurate information, if the audit of logon events is not enabled, the Collector does not report the logon duration of users with roaming profiles. For more information on roaming user profiles, see the next section.

Both the logon and the extended logon durations take the start of the logon process as the beginning of their measurement, but they differ from each other in their ending point:

- The appearance of the desktop marks the end of the logon duration.
- After the desktop is shown, the readiness of the device to being used marks the end of the extended logon duration. The device is considered to be *ready to use* when the operating system frees enough resources so that the device becomes responsive again to the commands of the user. Depending on the type of device, the resource consumption for considering the device to be ready is as follows:
    - Desktops and laptops: the CPU usage drops below 15% and the disk usage below 80%.
    - Servers: the CPU usage of all the processes that belong to the logged on user drops below 15%.

If the consumption of resources in the device is still higher than required 25 minutes after user authentication, the Collector stops waiting and reports the logon duration as 25 minutes.

Logon duration		
Start	**Stop**	
• User authentication	• Desktop is shown	• Device is ready to use
Start	**Continue**	**Stop**
Extended logon duration		

Logon duration in devices with roaming user profiles

A roaming user profile is a concept in Windows that allows users to have a consistent desktop experience across different computers within the same network. Independently of the computer that they choose to work with, the users have access to their personal documents, the applications remember their

preferences and the desktop appearance remains the same. In this section, learn how roaming user profiles may impact the measurement of the logon duration.

When roaming users log on a device, the loading of their profile can take a substantial part of the logon time. However, the new session starts only after the profile is completely loaded. If the Collector just waited for the session creation event to compute the logon duration, it would ignore the time spent to load the user profile as part of the logon duration. Because of this omission, the Collector would report much smaller logon durations than the actual values for the logon duration of roaming users. Therefore, the Collector never uses this method for computing the logon duration of roaming users.

The alternative is to get logon information from the Security log of Windows. Logon events in the Security log always report the correct logon time. For this reason, auditing logon events is the preferred method for the Collector to compute the logon duration of all kinds of users and it is mandatory for roaming users. For devices with roaming user profiles, remember to always activate the audit of logon events. Failing to do so results in the Collector not reporting the logon duration of users with roaming profiles.

Inspecting logon duration through Finder and NXQL

User logon is the activity that holds logon duration information in Nexthink. Look for user logons in the Finder by creating a new investigation:

1. In the bottom of the **Start** page, click **New Investigation**.
2. Under the **Activities** tab, select **user logons**.
3. Fill the conditions and time frame as desired.
4. Ensure that the **Columns** in the **DISPLAY** section include the **Duration** and the **Extended duration** of the user logon.
5. Click **Run**

In NXQL, user logons are a type of the more general **user_activity** events. For instance, to get a list of all logon events, open the NXQL editor and type in:

```
(select * (from user_activity (where user_activity (eq type (enum
logon)))))
```

Note that the extended logon duration is called **real_duration** in NXQL.

To get logon duration information for a particular device in the Finder, create an investigation on devices and look for the following columns to display (note that there is a normal and an extended duration version of each field):

- **Last [extended] logon duration**:
  The (extended) duration of the last user logon measured on the device.
- **[Extended] Logon duration baseline**:
  The exponentially weighted moving average of the (extended) duration of all user logons of the device stored in the Engine database. For a measured logon duration *L(n)* on the *n*th logon, the weighted average *S(n)* is recursively computed with the following formula:
  - *S(n) = L(1)*, for *n* = 1
  - *S(n) = L(n) + (1 -) S(n - 1)*, for *n* > 1
    Where coefficient = 0.1 and the contribution to the duration of old user logons is removed as soon as they are dropped from the history of the database.

  Note that this field is precomputed from all user logons available in the database, so its value does not depend on the time frame of the investigation.

In NXQL, the equivalent fields are called as follows:

Finder	NXQL
Last logon duration	**last_logon_duration**
Logon duration baseline	**average_logon_duration**
Last extended logon duration	**last_extended_logon_duration**
Extended logon duration baseline	**extended_logon_duration_baseline**

For instance, to get all types of logon duration for every device, type in the query:

```
(select (last_logon_duration average_logon_duration
last_extended_logon_duration extended_logon_duration_baseline)
(from device))
```

We have seen that the logon duration baselines (normal and extended) are precomputed for a device and do not depend on the time frame. To actually compute the average logon duration of a user on a device over a time frame, add the **Average [extended] logon duration** aggregate to your displayed columns. The average is calculated as the sum of the duration of the logon events divided by the number of logon events within the time frame.

In NXQL, the equivalent for the **Average logon duration** is the **average_logon_duration** aggregate. Note that it has the same name as the

normal logon duration baseline, but it must be placed inside a **compute** clause in the NXQL query, so there is no confusion possible.

Finder	NXQL
Average logon duration	**average_logon_duration**
Average extended logon duration	**average_extended_logon_duration**

For example, to query both for both the normal and the extended logon duration baselines and averages (for the last day) in an NXQL query, type in:

```
(select (average_logon_duration extended_logon_duration_baseline)
(from device (with user_activity (compute average_logon_duration
average_extended_logon_duration) (between now-1d now))))
```

Related tasks

- Auditing logon events

Memory and CPU usage

Applies to platforms:

Overview

Measuring the utilization of the hardware resources within each device in your organization is key to evaluate both the efficiency of devices and the impact of resource consumption on end-user experience. Users that perceive their devices as slow usually suffer from scarcity or misuse of two basic system resources: main memory and CPU processing power.

In this article, learn about the fields and aggregate values that measure the usage of memory and processing power in Nexthink. Based on these figures, assess the amount of resources given to a particular device and find out those applications that are most eager for resources.

Memory usage

The Collector takes samples of the amount of memory used by each running process with a period of 30 seconds. Every 5 minutes, interval, the Collector calculates the average value of these samples over the past 5 minutes for

sending it later to the Engine. Note that if a process allocates and frees memory very quickly, the Collector may miss some peaks of memory consumption when taking its samples every 30 seconds. Therefore, there is always some uncertainty in the values offered by the Collector, but it is usually negligible for well-behaved applications. Moreover, memory issues typically arise because of a sustained high consumption of memory and not because of short-lived allocations.

Based on the data collected, the following fields and aggregates are available for measuring memory usage in Nexthink:

Name	Type	Applies to	Description
Average memory usage	field	• execution	The average memory usage of the execution before being aggregated.
Average memory usage per execution	aggregate	• user • device • application • executable • binary	The average memory usage of all the underlying executions divided by their cardinality.
Average memory usage (deprecated)	field	• binary	The average memory usage of the underlying execution with a sampling rate of 5 minutes

Note that the memory usage is calculated per process before they may be aggregated by into a single execution. A single binary may spawn several identical processes in memory, resulting in a total memory consumption higher than that of the individual processes.

For instance, we can look at the behaviour of two well-known web browsers: Chrome and Firefox. Chrome creates a new process for every tab that the browser opens, while Firefox uses a single process for all tabs. Therefore, Firefox will typically report a higher average memory usage than Chrome for similar use cases when multiple tabs are open, because the memory utilization is reported per process, before processes are aggregated in the Engine.

CPU usage

In the case of CPU usage, the Collector takes samples of the CPU load of all running processes every 30 seconds. The CPU load is measured as a percentage value from 0 to 100 for each logical processor that is present in the device. Therefore, the CPU load can be higher than 100% for devices with multiple logical processors. For instance, a device with 12 logical processors has

a maximum CPU load capacity of 1200%.

Contrary to memory usage samples, CPU samples are not averaged before sending them to the Engine, which lets the Engine know about peaks of CPU utilization. Still, note that the maximum instantaneous load of CPU may not occur simultaneously with the moment when the Collector takes a sample. The Collector sends the CPU samples to the Engine every 5 minutes. For every sample, the Engine calculates the effective CPU utilization of each process during its execution. Retrieve it using the following fields and aggregates:

Name	Type	Applies to	Description
Total CPU time	field	• execution	The effective utilization time of the CPU during the aggregated execution. Note that the total CPU time can exceed the total duration of the execution if the average CPU load was over 100%.
Total CPU time	aggregate	• user • device • application • executable • binary	The sum of the total CPU time of all executions within the scope of the selected object.
CPU usage ratio	aggregate	• user • device • application • executable • binary	The sum of the total CPU time of all executions divided by their duration within the scope of the selected object.
Average CPU usage (deprecated)	field	• binary	Average CPU load of a binary over all logical processors, taking into account all its executions since the binary was first seen. Note therefore that this value does not depend on the selected time frame.

Note that these figures are based on the samples taken directly from running processes. The Collector also takes samples of the total CPU load reported by a device (not broken down by processes), but these are just used to signal high CPU conditions in the device.

Related references

- Errors and warnings for devices and executions
- Warnings tooltips

Status of TCP connections

The status of a TCP connection can be one of the following:

established
: The TCP connection has been accepted by the remote party and is still active.

closed
: The TCP connection has been closed after being successfully established.

no service
: The remote party acknowledged the initial SYN message by a RST message; i.e. the remote party does exist, but no service is bound to the request port. Note: Most personal computers are protected by a firewall that discards RST messages to prevent effective port scanning.

no host
: The remote party does not acknowledge the SYN message; i.e. the remote party does not exist or it is protected by a firewall.

rejected
: The TCP connection was rejected by the operating system itself; for instance due to security settings.

Related concepts

- Connection

Status of UDP connections

UDP is a stateless protocol; therefore, a UDP connection has no status in a strict sense. Nevertheless, for keeping them similar to TCP connections, UDP connections also include a *status* field in Nexthink.

Because of the very nature of the UDP protocol, the status of a UDP connection does not indicate success or failure in the delivery. However, the system can deduce whether the connection is still ongoing or if it has expired. Thus, the two possible statuses of a UDP connection are:

established
: While events are being aggregated to the same connection (increasing its cardinality), the system considers that the connection is still open; that is, the device keeps sending datagrams to the same destination via the same UDP port.

closed
: When the aggregation time has passed and there are no new events to add to the connection, the UDP connection is considered closed.

The aggregation time depends on your settings for aggregating events (minimum 5 minutes).

Related tasks

- Establishing a data retention policy in the Engine

Related references

- Status of TCP connections

Related concepts

- Connection

Network and port scan conditions

Nexthink considers a set of connections to be a network or port scan when the following conditions are met:

- A single process is starting all the connections.
- Each connection in the set is separated from the previous connection by less than 90 seconds; that is, one minute and a half.
- The set of connections contains at least 50 connections.
- The set of connections only contains *failed* connections.

The reason to include the last condition is that a scan operation does not usually complete the vast majority of its connection attempts. Since a scan blindly tests every port or destination, most of the connections are rejected. The way to express this last condition depends on the transport protocol of the connection. In the case of TCP, the status of the connection directly shows whether the connection failed or not. In the case of UDP, however, there is no clear status of the connection. Therefore, Nexthink suspects a UDP scan when many small UDP packets are sent in a short period of time:

TCP
: All connections in the set are unsuccessful.

UDP

The size of each packet sent is less than 10 KB.
The total duration of the whole scan is less than 15 minutes.

To summarize, this is the list of all the types of network and port scan that you can find:

TCP network scan
: A process launches a burst of unsuccessful TCP connections to the same port of at least 50 destinations.

UDP network scan
: A process sends a burst of small UDP datagrams to the same port of at least 50 destinations within 15 minutes.

TCP port scan
: A process launches a burst of unsuccessful TCP connections to at least 50 ports on the same destination.

UDP port scan
: A process sends a burst of small UDP datagrams to at least 50 ports on the same destination within 15 minutes.

Related concepts

- Connection
- Port

Incoming traffic measurement

Overview

Nexthink measures the network traffic that enters a device in two different ways:

Per connection
: The traffic received as response to an outgoing connection.

Per execution
: The amount of traffic received during the execution of a program.

These two ways of measuring the incoming traffic may produce different results if some devices in your network behave as servers.

Incoming traffic per connection

Since Nexthink records the connections of a device only when the device acts as client, that is, when they are initiated by the device itself, the incoming traffic per connection is exclusively due to the responses received from the corresponding servers through these outgoing connections.

Because the UDP protocol requires a device to act as a server to receive any data, only TCP connections may report incoming traffic.

Rembember that Nexthink does not record the incoming connections to a device; that is, the connections that enter a device when it acts as a server.

Incoming traffic per execution

Contrary to the measurement of incoming traffic for individual connections, the measurement of incoming traffic during the execution of a program does take into account the incoming connections to the device.

Therefore, if a program accepts connections on a particular port, making the device act like a server, the received data is visible in the amount of traffic associated to the execution of the program, but not as an individual connection.

Network and Local activity views

Both the Network and the Local activity views in the Finder let you visually examine the incoming network traffic of devices. While the Network activity view aggregates incoming traffic values per connection, the Local activity view collects measurements of the incoming traffic per execution.

Therefore, the Local activity view may report more incoming traffic data than the Network activity view if any of the devices included in the visualization is acting as a server.

Related tasks

- Viewing network connections
- Viewing executions

Related references

- Server support

Binary paths

Overview

Nexthink stores the paths from where end-users execute each binary file of their applications, up to a maximum of 20 paths per binary. Binary paths are stored in lowercase letters (converting from uppercase when needed), and they use the forward slash (/) to separate the names of folders in the hierarchy, independently of the convention used by the underlying operating system of the devices.

Typical applications usually install their executable binary files in the same standard locations in the filesystem, independently of the device on which they are run. For example, most software applications are installed under the **Program Files** directory of a Windows device. The execution of binaries from multiple or unusual locations usually indicate a strange behaviour of users or even the presence of malware.

To avoid reporting too many paths for every single binary, Nexthink uses some techniques that are detailed below. Paths that do not fall into any of the special categories shown below are stored in their full form.

Path aliases

Path aliases replace well-known directories by keywords, using a format similar to that of environment variables in Windows. In this way, binary paths of well-known locations become language neutral and independent of the drive in which the binary is located. For instance, the paths **D:\Program Files** (English version) and **C:\Programme** (German version) become both **%ProgramFiles%** when stored in Nexthink as a binary path.

Contrary to the general rule for binary paths, path aliases may contain uppercase characters. Find below a table with the list of all path aliases, their description, and a few exaples of the folders that they replace:

Path alias	Description	Example
%Windows%	Windows directory	DRIVE:\Windows
%System%	Windows system directory	DRIVE:\Windows\System32
%ProgramFiles%	Software installation directory	DRIVE:\Program Files DRIVE:\Program Files (x86)

%UserProfile%	Directory holding user-specific data	DRIVE:\Documents and Settings\USERNAME DRIVE:\Users\USERNAME
%AllProfile%	Directory holding data accessible by all users	DRIVE:\Document and Settings\All users `DRIVE:\Users\Public (Windows Vista and higher)`
%ProfileTemp%	Directory holding user-specific temporary files.	DRIVE:\Documents and Settings\USERNAME\Local Settings `DRIVE:\Users\USERNAME\AppData\Local`
%WindowsTemp%	Temporary folders in hexadecimal format under the root directory	DRIVE:\c7fa349ced49048e8941a819b264eb8d
%NetDrive%	Network shared folder	\\SERVER\shared-dir
%RemovableDrive%	Non-permanent storage devices	MEDIA_DRIVE:\ (USB stick, CD / DVD, etc.)
%RecycleBin%	Directory holding deleted files	DRIVE:\$RECYCLE.BIN

Ellipsis in binary paths

Ellipsis in aliased paths

For privacy reasons and to avoid path explosion, the complete binary path is not recorded for binaries whose working path lies inside some of the aliased locations. Binaries executed from these locations do not have their full path stored:

- %RecylceBin%
- %UserProfile%
- %AllProfile%
- %ProfileTemp%
- %WindowsTemp%
- %RemovableDrive%

Instead, a three dot ellipsis (`/.../`) replaces the part of the path after the alias. For example, the path of a typical binary installer `setup.exe` executed from a temporary Windows folder is recorded as:

```
%WindowsTemp%/.../setup.exe
```

Ellipsis for automatically generated folders

Nexthink is also capable of detecting folders whose names are automatically generated identifiers. These are usually very long alphanumerical names that are meaningless to a human reader. Therefore, the name of those folders is not stored *as is* in binary paths, but replaced by an ellipsis (`/...`).

The following table contains the types of identifiers recognized by Nexthink and some examples of how each one of them looks like in the filesystem:

Type of ID	Examples
GUID	4AQlP4lP0xGaDAMF6CwzAQ 3F2504E0-4F89-11D3-9A0C-0305E82C3301
MD5	79054025255fb1a26e4bc422aef54eb4
SHA1	2fd4e1c67a2d28fced849ee1bb76e7391b93eb12
Long Hexadecimal strings	Most hexadecimal strings containing 10 or more characters
Long numbers	Most strings containing at least 10 digits in a row, except if the digits are all the same.

Related concepts

- Binary

Maximum number of Binaries

By default, the Engine database supports a maximum of 40 000 simultaneous binaries. When this limit is reached, the Engine sends a system alert to the administrator account. No new binaries are incorporated into the Engine until any of the existing binaries is removed from the system.

The Engine automatically removes a binary from its database when the maximum inactivity period of the binary has elapsed and there is no event related to the binary in the database. By default, the maximum inactivity period of a binary is one month (30 days to be precise).

The Engine does not keep track of any information related to a non-recorded binary, such as executions or connections. Therefore, it is important for the administrator to keep the total number of binaries under control. If necessary, an administrator can increase the maximum number of binaries in an Engine and

modify their maximum inactivity period. Beware that modifying the default values may have an impact on the performance of the whole system.

Modifying the limits on binaries

To modify the maximum number of binaries and their maximum inactivity period:

1. Log in to the CLI of the appliance hosting the Engine.
2. Edit the configuration file of the Engine:
   ```
   sudo vi /var/nexthink/engine/01/etc/nxengine.xml
   ```
3. Inside the **<limit>** tag, add the following lines to increase the maximum number of binaries to 60 000 and decrease the maximum inactivity period of binaries to 20 days, for instance:
   ```
   <max_binaries>60000</max_binaries>
   <max_binary_inactivity_period>1728000</max_binary_inactivity_period>
   ```
4. To save the file and exit, type in:
   ```
   :wq
   ```
5. Restart the Engine:
   ```
   sudo systemctl restart nxengine@1
   ```

Note that the maximum number of binaries has a hard limit of **100 000** binaries and that the maximum inactivity period of binaries is expressed in seconds:

```
20 days * 24 hours/day * 60 min/hour * 60 s/min = 1 728 000 s
```

The operations described in this article should only be performed by a Nexthink Engineer or a Nexthink Certified Partner.

If you need help or assistance, please contact your Nexthink Certified Partner.
Related concepts

- Binary

Information on printers and printing

Getting accurate information on the utilization of printers is essential to ensure compliance with the printing policies established inside your organization and to optimize print costs. The Engine records every printing activity of the end-users that is initiated from any device in which the Collector is running.

Because of the technologies involved in the detection of print jobs, only Windows devices are able to send printing information for the moment.

Applies to platforms:

Printer information

The Engine knows about a printer in your organization once a device that is equipped with the Collector tries to print a document on it. Depending on how the printer is connected to the device, Nexthink distinguishes four types of printers:

local
: The printer is directly connected to the device via a serial or parallel port (USB, COM or LPT) and it is visible to all the users of the device under the same name. Virtual printers, that is, software drivers that behave like a printer driver but lack the physical apparatus and typically redirect their output to a file, also fall into the category of local printers.

tcp/ip
: The printer is connected to the network and it is made available to the device via a standard TCP/IP port. All users of the same device see the printer with the same name or IP address.

wsd
: The printer is connected to the network and it is made available to the device by means of a Web Services for Devices port. All users of the same device see the printer with the same name.

smb
: The printer is made available to the device by sharing it from another device where the printer is locally connected via the SMB protocol. The printer is therefore considered as local in the hosting device, and as an SMB printer in the remote device. Each user in the remote device must individually import the SMB printer to be able to use it, so users may see the printer under different names. Note that SMB printer support is disabled by default in Nexthink.

Besides the type, Nexthink records the following information on printers:

Name
: The name of the printer as it appears in the properties dialog.

Hostname
: For local and smb printers, this is the name of the device to which the printer is directly connected.
: For tpc/ip and wsd printers, this is usually the DNS name or IP address of the printer itself.

Display name

Since different users may see the same of printer under different names, the *display name* shows the most frequent name assigned to the printer.

Location
: The place where the printer is found, according to its configuration properties.

Limit on the number of printers

The Collector supports up to 62 printers connected to a single device. A Collector that runs on a device with more than 62 connected printers fails to report any print job and, consequently, any printer.

Print job information

A print job is an activity that puts in relationship a user, a device and a printer. Thus, for a given print job, you can display in the Finder the name of the device that sent the print job, its ID, or its SID, without the need to drill-down to devices. Likewise, you can display the name of the user, its ID, its SID, or the name and model of the printer that took part in the print job.

At the end of the printing process, the print job is added to the Engine with one of the following status:

success
: The print job has been successfully completed.

error
: The print job was not completed because of an error.

unknown
: The Collector could not determine the final status of the print job.

The rest of the fields of the print job reflect the options selected to configure the printer: **Number of pages**, **Paper size**, **Duplex print**, **Color enabled**, **Print quality** and **Size**.

Beware that software may modify the actual printout despite of the settings sent to the printer. In particular, the fact that the **Color enabled** field of the print job is reported as **yes** does not necessarily mean that the output is in color. For example, a user of PowerPoint who decides to print a colorful presentation in black and white may select the **Grayscale** option in the printing dialog of PowerPoint:

In this case, even if the user sets up a color printer to print in color and the Finder reports the **Color enabled** field as **yes**, the actual output is obviously in black and white. As a side note, depending on the printer model, enabling color output can still cause consumption of color ink to get a grey or black tone.

On the other hand, if the user sets up the printer itself to print in black and white, the resulting printout can never be in color and the Finder reports **Color enabled** as **no**.

Something similar happens with applications that are able to group several document pages into one before sending the document to the printer. If an external application does the grouping, the printer knows nothing about it. Therefore, the printer reports the **Number of pages** that it actually printed. When it is the printer that makes the grouping, the **Number of pages** depend on the

model of the printer. Some printers give the number of pages actually printed and some others take into account the grouping and give the number of pages of the original document.

For some print jobs, the Collector is also able to determine the **Document type**, which reflects the file format of the printed document or the application that produced it. Find the list of supported document types below:

• Crystal	• Outlook	• Publisher
• Excel	• PDF	• Text
• Illustrator	• Photoshop	• Visio
• Image	• Postscript	• Word
• InDesign	• Powerpoint	• Web
• InfoPath	• Project	• unknown

Print information in virtual environments

The basic principles of print support remain the same for virtual environments. For both VDI and streamed applications, the same information detailed above on printers and print jobs applies. In addition to the standard printing techniques, however, virtual environments introduce the concept of *redirected printers*. Consider for instance a user connecting to a virtual machine from a client device. If the client device is equipped with a local printer, the user can map the printer in the remote session. Thus, printing in the remote computer using the mapped printer effectively redirects the print jobs to the local printer.

In the example above, all the printers that display **(de NXT-PAT-W7)** at the end of their name are redirected to the local printer in the client device **NXT-PAT-W7**.

If the Collector is running on the client device and the user prints on one of these redirected printers, the print job is reported as originating from **NXT-PAT-W7** and not from the virtual machine. This indeed corresponds to reality, since the printer is merely redirected, and not connected to the VM.

On the other hand, if the Collector is also installed in the VM, any print job sent to the redirected printer is reported to the Engine as originating from the VM as well. In this case, even the redirected printer is reported as local to the VM too. By default, these virtual printers and print jobs are discarded by the Engine to avoid duplications that might alter overall statistics. You can nevertheless control what kinds of print jobs are discarded by editing the configuration file of the Engine.

Print notifications of SMB printers

Devices that do not belong to a domain may fail to receive print notifications from SMB printers in the case that an anonymous user initiated the print job. To receive print notifications of anonymous users from SMB printers, ensure that the registry key **NullSessionPipes** holds the value **spoolss**:

1. Open the registry editor by pressing the **Windows + R** keys and typing **regedit**.
2. Locate the registry key **NullSessionPipes** here:
    - HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services\LanmanServer\Para
3. Edit its value and set it to **spoolss**.

An alternative solution is again to join all your devices to a domain. In a domain, all users are authenticated.

Related tasks

- Ignoring specific print ports
- Enabling support for SMB printers

Related concepts

- Printer
- Print job

Metro apps

Windows 8 Metro apps behave in a different way than normal Windows applications. Nexthink abstracts those differences in order to provide a comprehensive view of the users' activity.

Executions of metro apps

There are mainly two kind of architectures for metro apps: compiled and JavaScript. Compiled applications (written in C#, VisualBasic or C++) behave similarly to normal Windows applications and have a .exe file associated with them. JavaScript applications however are not compiled and there is no associated .exe file. Instead the default windows executable WWAHost.exe is used to host them. When analysing the task manager of Windows 8 device we can usually see several executions of WWAHost, each one corresponding to a specific metro app.

Only reporting executions of WWAHost would be confusing as Nexthink users would no longer be able to distinguish between different metro apps. Instead Nexthink Collector abstracts this information and reconstructs name and properties of the metro app being hosted by WWAHost by reporting this as a **Binary**. Similar techniques are used to abstract the corresponding **Executables** and **Applications**.

Executions, connections, web requests, crashes, freezes, etc. are reported as usual. This means that for instance in Nexthink it is possible to distinguish the web requests performed by the *Weather* app with respect to the *Food & Drink* app. The **Description** field of Metro executables is also always set to *Windows 8 Metro Style App*.

Metro packages

Metro apps are different from a packaging perspective as well. In fact they are not installed like traditional applications and are not visible in Windows **Programs and Features**. As for binaries Nexthink abstracts these differences; all metro apps are reported as a **Package** and the corresponding installations and uninstallations are reported as well.

Related concepts

- Metro app architecture (MSDN)

Mobile data and ActiveSync

This article or section is in the process of an expansion or major restructuring.

Investigation with packages

Starting from Nexthink V6.3, those investigations retrieving packages or including a condition on packages have been simplified. The results take into account only those packages that are effectively installed, discarding uninstalled packages.

Nevertheless, it is still possible to find installations or uninstallations of packages by directly querying the installation events.

The following tables summarize the typical uses of investigations related to packages. Download all the investigations and import them into the Finder to try them out.

Retrieving devices

Investigation name	Condition	Aggregate condition
Devices *with* a specific package	Package Name is "abc"	-
Devices *without* a specific package	Package Name is "abc"	Number of packages = 0
Devices *with* two specific packages	Package Name is "abc", publisher is "ABC", version is "123", type "program" or Package Name is "xyz", publisher is "XYZ", version is "789", type "program" NOTE: A package is uniquely identified by its name, publisher, version and type.	Number of packages = 2

Retrieving installation events

Investigation name	Condition	Aggregate condition
Installation of a specific package	Package Name is "abc" and Operation Type is "installation"	-
Uninstallation of a specific package	Package Name is "abc" and Operation Type is "uninstallation"	-

Retrieving packages

Investigation name	Condition	Aggregate condition	Order by
Least used packages	-	-	Number of devices is ascending
Most used packages	-	-	Number of devices is descending
Packages installed on less than 5 devices (but at least installed on one device)	-	Number of devices is less or equal to 5	-

Portal aggregation and grouping

Overview

The Portal aggregates metric data along two dimensions: the defined hierarchy and time. To aggregate the data, the Portal also takes into account the *group by* and *aggregate by* options that you set when you create the metric in the Finder (depending on the kind of metric, not all options are available). Learn here how the possible combinations of aggregation and grouping options, as well as the hierarchy node and period selection, produce different results in the dashboards of the Portal.

The example hierarchy

For demonstration purposes, let us suppose that we have defined a very simple hierarchy based on locations that consists of two levels only:

- Country (top level)
- City (entity level)

Our imaginary organization has offices in two countries: Switzerland (CH) and Spain (ES). For each country, offices are located in two cities: Geneva (GVA) and Zurich (ZRH), in the case of Switzerland; Madrid (MAD) and Barcelone (BCN), in the case of Spain.

Finally, for simplicity, let us suppose that there are just two devices per city, whose names are composed by the initial letter of the city, followed by either the number 1 or the number 2. That makes a total of eight devices:

Hierarchy								
Country	CH				ES			
City (entity)	GVA		ZRH		MAD		BCN	
Devices	G1	G2	Z1	Z2	M1	M2	B1	B2

Count metrics

Depending on the type of object, count metrics may take into account, for a particular day, only those objects that were active during that day or all the objects regardless of their activity. For some types of objects, you can only count active instances. For packages, you always count all of them. For users and devices, the user may choose whether to count all of them or only those that were active. See the table below as reference:

Only active	Only all objects	User choice
• Application • Executable • Binary • Port • Destination • Domain • Printer	• Packages	• Users • Devices

Except in the case of packages, you are usually interested in counting only those objects which were active during the day. However, in cases such as transformation projects, it is important to know which objects already fulfil the transformation condition, no matter whether they were active during a particular day or not. For instance, if you have a migration project from Windows 8 to Windows 10, you probably want to count every day the devices which have already installed Windows 10, even when they were not active during the

measurement day. In this way, when observing the results of the metric in the Portal, you get a non-decreasing value, which may not be the case if you measure only active devices instead. For instance, in a typical company, the number of active devices decreases dramatically during the weekend, increasing again during weed days.

The value of a metric that counts all objects is thus valid for the particular day when it is computed and cannot be aggregated through time. Therefore, the Portal does not show widgets associated to these metrics when selecting time periods different from one day. In addition, note that the computation of these metrics requires to look through all the history available in the Engine for counting in all possible objects. The conditions that you can specify for these metrics are consequently the same as for full period investigations, namely they cannot depend on activities or events.

When counting active objects only, an extra condition (*count devices that meet conditions on ... in period*) influences how the Portal displays the results of the metrics for periods longer than one day. Use this extra condition to count objects, either:

- *The last day* that the objects were active within the selected period (for inventory purposes), or
- *At least one day* during the selected period (for detecting occurrences).

Grouping by properties

As a first example, let us consider a metric that counts the number of devices and groups them both by type and OS (in the **COMPUTE DAILY** section, select **Group by *device type* and *OS version and architecture***).

Since this metric is intended to make an inventory of devices and inventories are usually based on the latest state of the catalogued objets, select the extra condition of count metrics to **last active day** (in the **MATCHING** section, select **Count devices that meet conditions on *last active day* in period**). Leave the rest of the options for creating the metric to their defaults.

The devices in our example are of the following types and have the following operating systems installed:

Devices	Device Type	OS version and architecture
G1	server	Windows 2012 Server Standard (64 bits)
G2	desktop	Windows 8.1 Pro (64 bits)

Z1	laptop	Windows 7 Enterprise SP1 (64 bits)
Z2	desktop	Windows 8.1 Pro (64 bits)
M1	laptop	Windows 10 Pro (64 bits)
M2	desktop	Windows 7 Enterprise SP1 (64 bits)
B1	laptop	OS X Yosemite 10.10.5 (64 bits)
B2	desktop	Windows 7 Enterprise SP1 (64 bits)

If the devices and their operating system do not change, the Portal consistently displays the same global results for the metric, regardless of the period selected. For instance, when you display the count metric in a table widget, arranging the rows by operating system and the columns by device type, you always get the following global results:

	Device type		
OS version and architecture	**desktop**	**laptop**	**server**
Windows 8.1 Pro (64 bits)	2	0	0
Windows 7 Enterprise SP1 (64 bits)	2	1	0
Windows 2012 Server Standard (64 bits)	0	0	1
Windows 10 Pro (64 bits)	0	1	0
OS X Yosemite 10.10.5 (64 bits)	0	1	0

Now let us consider the case that B1 (the only Mac OS device in the list) gets its OS upgraded from version *Yosemite* to *El Capitan*. After this upgrade, the choice of the extra condition of count metrics influences the results displayed in the Portal. Differences appear only when viewing periods are longer than one day in the Portal; therefore, let us see the results of our widget when selecting a period of one week (the week when the device got its OS upgraded). If you chose **last active day** as extra condition, you get:

	Device type		
OS version and architecture	**desktop**	**laptop**	**server**
Windows 8.1 Pro (64 bits)	2	0	0
Windows 7 Enterprise SP1 (64 bits)	2	1	0
Windows 2012 Server Standard (64 bits)	0	0	1
Windows 10 Pro (64 bits)	0	1	0
OS X El Capitan 10.11.4 (64 bits)	0	1	0

That is, you get the same results as before, except for the last row in the list, where **OS X Yosemite 10.10.5 (64 bits)** is replaced by **OS X El Capitan 10.11.4 (64 bits)**. That implies that the last day when the Mac device was active during

the selected week, it already had the *El Capitan* version installed. However, if the metric was defined with the extra condition option set to **at least one day** instead of **the last active day**, the widget displays the following results:

	Device type		
OS version and architecture	**desktop**	**laptop**	**server**
Windows 8.1 Pro (64 bits)	2	0	0
Windows 7 Enterprise SP1 (64 bits)	2	1	0
Windows 2012 Server Standard (64 bits)	0	0	1
Windows 10 Pro (64 bits)	0	1	0
OS X Yosemite 10.10.5 (64 bits)	0	1	0
OS X El Capitan 10.11.4 (64 bits)	0	1	0

That is, during the week of the upgrade, the Mac computer was seen some days with the *Yosemite* version installed and some other days with the *El Capitan* version. Thus, it appears twice in the table widget. For that reason, the option **at least one day** is not well-adapted to inventorying, which was the original purpose of the metric.

On the other hand, the option **at least one day** is useful to detect occurrences of particular situations. For example, imagine that you want to know whether any device had the antivirus real-time protection turned off any day during the selected period. To that end, create a metric that counts devices and performs any of the following two functions:

- Group the devices by the status of their antivirus real-time protection
- Set a condition to get only those devices with the protection turned off.

Choose **at least one day** as the extra condition of the metric to count those devices that had the protection turned off at any day and not necessarily the last day that they were active.

Note that the Portal verifies the state of the antivirus real-time protection of the devices when it computes the value for the metric. If you switch the antivirus real-time protection of a device off and on in the same day before Portal computation, the situation will go undetected for the previously defined metric. In general, this applies to all metrics that set conditions on the state of objects. On the other hand, metrics with conditions based on events keep a history of occurrences. For instance, if a metric counts the number of devices executing high threat binaries, the Portal will see these executions during the computation of the metric in any case.

Coming back to our inventory example, let us consider now the result of counting all devices instead of counting the active devices only. Since the migration from Yosemite to El Capitan looks like a transformation project, counting all objects is probably more suitable than counting just the active objects. Indeed, imagine that the Mac laptop in the example is turned off for a whole day after being upgraded to El Capitan. While a metric that counts all objects would include the Mac laptop in the results of that day, a metric counting only active devices would leave it out of its results. Think carefully about the choice of counting only active objects or all objects when you create a count metric for devices or users.

Grouping by foreign category

Count metrics let you group results not only by the properties of the counted objects themselves, but also by categories of related objects (*foreign* categories). Let us illustrate this kind of grouping by creating a metric that counts the number of users. The metric groups the users by a foreign category called *Ownership*. We define Ownership as a category of devices that has two keywords:

- **Corporate**, which indicates that a device belongs to the company.
- **BYOD**, which indicates that a device belongs to the user himself.

From the example hierarchy, let us focus on the CH node, that is, the devices in Switzerland, and imagine that we had the following usage pattern during the last day:

- User 1: used his own computer G1 in Geneva and then travelled to Zurich and used the corporate computer Z1.
- User 2: used his own computer G2 in Geneva.
- User 3: used the corporate computer Z2 in Zurich.

When retrieving the data during the nightly computation, the Portal stores the following data internally:

Users	Entity	Ownership
User 1	GVA, ZRH	BYOD, Corporate
User 2	GVA	BYOD
User 3	ZRH	Corporate

Note that the Portal is unable to deduce from this table whether the Corporate device of User 1 is located in Geneva or Zurich (same for his BYOD device). To be on the safe side, the convention followed by the Portal is to count the user in for all possible combinations, although that may lead to situations where the actual combination did not occur. In our example for User 1, only the

combinations GVA-BYOD (because of the use of G1) and ZRH-Corporate (because of the use of Z1) actually occur. However, the Portal counts as well User 1 for the combinations GVA-Corporate and ZRH-BYOD.

Thus, displaying the metric in a table widget, with the rows organized by hierarchy and the columns by Device-Ownership, yields the following results for the last day (when CH is selected as the Country of the hierarchy):

	Ownnership		
City	**Overall**	**Corporate**	**BYOD**
Overall	3	2	2
GVA	2	1	2
ZRH	2	2	1

Note that because of User 1 using computers in both Geneva and Zurich, and because of the additional combinations added by convention, none of the partial results add up to the overall values. The values over all the hierarchies are shown in the Table widget after ticking the option **Display overall value** in the configuration fof the widget for dividing by hierarchy.

Considerations on ratios and thresholds

When creating a count metric that includes a ratio computation and thresholds based on the ratio, threshold violations occur more frequently when exploring the lower levels of the hierarchy. The reason is that ratios tend to get more extreme when there are less objects to count. To avoid triggering threshold violations with only a few objects involved, tick the checkbox **Ignore unless at least x objects are impacted in a hierarchy node** and specify the minimum number of objects that must be impacted in order to consider a threshold violation effective.

On the other hand, if you base the thresholds on absolute values instead of ratios, threshold violations occur more frequently in higher levels of the hierarchy, because they involve more objects.

Quantity metrics

There are two types of quantity metrics: those that measure a countable number of actions (such as the number of executions, the number of printed pages, etc.) and those that measure a continous quantity related to the activity of a device (the memory usage, the boot or logon duration, etc.). In both types of quantity metrics, the measured quantity changes over time. It is therefore necessary to aggregate all the collected values to display a final result in the Portal for each

available time frame. For quantity metrics, there are four ways of aggregating the results:

- **sum over all devices and the whole timeframe**
- **average value per device per day**
- **maximum value per device per day**
- **minimum value per device per day**

Not all of the aggregation strategies are available for all quantity metrics. Only the options that make sense for a particular metric can be selected. Typically, the computation of the average and the maximum values per device per day are available to any quantity metric, whereas the sum or the minimum values only make sense for some kinds of quantity metrics.

Let us examine the different aggregation options for quantity metrics through some examples. For instance, consider a metric that counts the number of executions of the Finder -your favorite real-time analysis application from Nexthink- on each device; that is, a quantity metric that computes the **number of executions** with a condition on the executable name *nxfinder.exe*. Suppose that, for the current week, the devices of our original example located in Spain have the following usage pattern:

		Finder executions	
City	**Device**	**Monday**	**Tuesday**
MAD	**M1**	4	2
	M2	1	1
BCN	**B1**	0	1
	B2	1	0

Consider first the results of the metric for the different aggregation strategies when looking at the last day:

Maximum value per device per day: 2
: The device that executed the Finder the most on Tuesday is M1, which did it twice.

Sum over all devices and the whole timeframe: 4
: Results from adding all the executions on Tuesday 2 (M1) + 1 (M2) + 1 (B1) = 4.

Average value per device per day: 1.3
: Results from dividing the sum of all executions (the previous value calculated) by the number of devices that participate in the metric 4 / 3 = 1.3. Note that B2 is not counted in for computing the average on Tuesday

because it does not satisfy the condition of executing *nxfinder.exe*.

Now let us consider the results when selecting the last week. Since the week is not over, the Portal has data only for Monday and Tuesday:

Maximum value per device per day: 4
Device M1 executed the Finder four times on Monday. That is the maximum for any device during the week.

Sum over all devices the whole timeframe: 10
That adds up for the four executions on Tuesday plus six on Monday.

Average value per device per day: 1.7
Results from dividing the sum of all executions over the whole week by the sum of devices participating in the metric each day. That is 10 / 6 = 1.7. Note again that neither B1 not counted in on Monday nor B2 is counted in on Tuesday because they do not satisfy the condition of executing *nxfinder.exe*, so we only have three devices each day, making a total of 6 devices for computing the average.

We have just seen an example of a quantity metric that counts individual occurrences. Let us now consider a metric based on continuous value; for instance, a metric that computes the **average memory usage per execution** of the Finder (as in the previous example, we use a condition on the executable *nxfinder.exe*). As you probably know, the memory used by the Finder increases with the number of tabs that are simultaneously open, so we can expect significant differences between the memory used by any two distinct executions of the Finder. Again, for the devices in Spain, suppose that we have the following usage pattern for the current week:

		Finder memory	
City	**Device**	**Monday**	**Tuesday**
MAD	**M1**	200 MB	100 MB
	M2	No execution	200 MB
BCN	**B1**	300 MB	500 MB
	B2	400 MB	400 MB

The metric yields the following results for Tuesday, according to the chosen aggregation strategy:

Maximum value per device per day: 500 MB
Device B1 had the highest memory usage on Tuesday. Note that this is not the absolute maximum value of memory used by the Finder in the device, since the metric is actually measuring the average along the day.

Minimum value per device per day: 100 MB

Device M1 had the lowest memory usage on Tuesday. Again, note that this is an average along the day, so it is not the absolute minimum amount of memory that the Finder used.

Average value per device per day: 300 MB

Adding the values for all devices makes a total of 100 (M1) + 200 (M2) + 500 (B1) + 400 (B2) = 1200 MB, dividing by four devices gives 300 MB in average.

If you select to see the last week results in the Portal, you get the following results:

Maximum value per device per day: 500 MB

No device used more memory for the Finder than B1 on Tuesday, so 500 MB is again the maximum for the week.

Minimum value per device per day: 100 MB

The same for the minimum usage of mermory, which corresponds to M1 on Tuesday. Note that the memory usage of M2 on Monday is not counted as 0 because it does not meet the condition of executing the Finder. Thus, the metric does not take M2 into account.

Average value per device per day: 300 MB

We have 1200 MB from Tuesday plus a total of 200 (M1) + 300 (B1) + 400 (B2) = 900 MB on Monday, making a grand total of 2100 MB for the week. Dividing by three active devices on Monday (M2 had no executions) plus four devices on Tuesday, that is, a total of seven devices, gives 2100 / 7 = 300 MB in average.

In quantity metrics, as in count metrics, you can group the results by up to two criteria. Since quantity metrics are related to devices only, the criteria for grouping results can be based either on the attributes or on categories of devices (no foreign categories are available in this case). Setting **group by** options lets you break down the results of the metric when they are displayed in a table widget within a dashboard, in the same way as shown for count metrics.

Top metrics

Top metrics return a list of objects ordered by their contribution to a particular activity. When defining a top metric, you choose:

- The type of object to show in the list.
- The total number of objects in the list (from the top 10 to the top 100).
- An activity that is linked to that type of object.

- The criterion for including the objects in the list: include those with either the highest or the lowest contribution to the activity.
- The *aggregate by* option for the activity, which determines how to compute the contribution of each object.

The available aggregation options are similar to those available in quantity metrics. The difference is that they are not necessarily based on devices, so they can cross device boundaries. For instance, a metric that computes the *top 10 users with the highest number of executions* aggregates into a single value the number of executions carried out by a same user in different devices. Thus, these are the **Aggregate by** options of top metrics:

- **sum over the whole timeframe**
- **average value per day**
- **maximum value per day**
- **minimum value per day**

As in the case of quantity metrics, not all the aggregation possibilities are available for all top metrics, but only those that make sense.

Grouping by hierarchy

In addition to the **group by** options that you specify for count and quantity metrics, the Portal always aggregates the results of all metrics (including top metrics) by hierarchy.

When selecting one **group by** option in the definition of a count or quantity metric, it is always possible to break down the results by hierarchy and by grouping option, arranging them as the rows and columns (or columns and rows) of a Table widget. If you define two **group by** options and you select one of them to break down the results by rows or columns in a Table widget, you must select the other option for the columns or rows, respectively. In this case, the option to break down by hierarchy is not available. Nevertheless, choosing to display the total value of the metric or metrics added to the widget (i.e. the **Metrics** option), instead one of the *group by* criteria, lets you again choose the hierarchical criterion to arrange the results.

In top metrics, however, there is no **group by** option and it is not possible to arrange the hierarchy in a Table widget. You can still add display fields to the definition of a top metric to see them arranged in a Table widget.

Hierarchy navigation in the Portal

For those widgets that do not explicitly break down the results of a metric by hierarchy (KPI, Line charts, and those Table widgets not arranged by hierarchy), use the hierarchy navigation tool of the Portal that is located in the top blue ribbon to explore the results at different levels in the hierarchy. All the widgets in the dashboard adapt their results to the node selected in the hierarchy navigation tool.

The Table widgets that do break down the results by hierarchy divide in fact their results by the nodes placed at the immediately lower level of the node selected in the hierarchy navigation tool (that is, its child nodes).

For instance, consider a dashboard built around the hierarchy of our original exaple. While viewing the dashboard in the Portal, if you switch from the Global view to the node CH in the hierarchy navigation tool, all the widgets in the dashboard will display the results limited to the values that they got for Switzerland. In addition, Table widgets arranged by hierarchy will divide their results by the nodes GVA and ZRH, which are the children of the CH node.

Considerations on aggregation along time

As time passes by, the Portal accumulates data day by day. Every night, the Portal collects and computes the values of the past day and aggregates the results to the current week, month, and quarter.

However, when switching from a view of the current week to the current month in the Portal, a counterintuitive situation may occur: the amount of data available for the current month might be less than the amount of data for the current week. This situation may occur when a month boundary has recently been crossed, because a week can overlap over two different months.

Let us explain it with one of our previous examples on quantity metrics. Consider again the metric that counts the number of executions of the Finder and suppose that we chose to aggregate by the **sum over all devices and the whole timeframe**. If you look at the values that we obtained, we had 6 executions of the Finder on Monday and 4 executions on Tuesday. Assume now that Monday was April 30, Tuesday was May 1, and it is May 2. Therefore, a month boundary was crossed yesterday.

Date	April 30	May 1	May 2
Week day	Monday	Tuesday	Wednesday

Executions	6	4	N/A

If we navigate to the most recent date available in the Portal, the result depends on the period selected:

- **Day:** 4 executions, corresponding to Tuesday, May 1.
- **Week:** 10 executions, corresponding to the 6 on Monday plus the 4 on Tuesday.
- **Month:** 4 executions, corresponding to Tuesday, May 1.

That is, the week started before the new month and it is still ongoing, so there are more days available for the current week than for the current month. The value for the week may therefore be bigger than the value for the month.

Related tasks

- Creating a metric

Security

Access rights and permissions

Overview

Nexthink users have the right to see and manage content depending on their profile and assigned roles. The definition of a profile includes the account type, administration and view domains, mandatory roles, and other settings that determine the permissions of the users for managing content and performing system administration tasks.

The following tables display the access rights of the different types of users to the features of the product, including all the additional requirements to their profile or roles -when needed.

System management

Feature	Main central administrator	Central administrator	Administrator	User
Manage accounts	Ok	Ok	Profile	No
Manage profiles	Ok	Ok	No	No
Manage roles	Ok (Domain)	Ok (Domain)	Domain	No
Manage hierarchies	Ok	Ok	No	No
Manage entities	Ok	Ok	No	No
Manage engines	Ok	Ok	No	No
Manage appliance	Ok	Ok	No	No
Manage license	Ok	Ok	No	No

Profile
: Administrators can create accounts if the option **Allow creation of user accounts** is checked in the definition of their profile.

OK (Domain)

Central administrators (including the main central administrator) can manage roles in the highest domain. For roles created in lower administration domains, central administrators have the power to delete them, but not to edit them.

Domain
: Administrators can create and edit roles that fall under their administration domain.

Portal content

Feature	Main central administrator	Central administrator	Administrator	User
Create modules and dashboards	Ok	Ok	Ok	Profile
View published modules	Ok	Ok	Domain + Roles	Roles
Manage published modules	Ok (Domain)	Ok (Domain)	Domain	No
Manage service alerts	Ok	Ok	Ok	No

Profile
: Normal users can create modules if the option **Allow creation of personal dashboards** is checked in the definition of their profile.

Domain + Roles
: Administrators can view those published modules that fall under their administration domain, in addition to those included in their roles.

Roles
: Normal users can only view those published modules included in their roles.

Ok (Domain)
: Central administrators (including the main central administrator) can manage published modules in the highest domain. For modules created in lower administration domains, central administrators have the power to delete them, but not to edit them.

Domain
: Administrators can manage and publish only those modules that fall under their administration domain.

Finder and Engine content

Feature	Main central administrator	Central administrator	Administrator	User
Access to the Finder	Ok	Profile1	Profile1	Profile1
Manage categories, services, metrics, campaigns, global alerts, import and export content	Ok	Profile2	Profile2	Profile2
Manually tag objects	Ok	Profile3	Profile3	Profile3
Web API V1	Ok	Profile4	Profile4	Profile4
Web API V2 (NXQL)	Ok	Profile5	Profile5	Profile5
Management of Collector	Ok	Profile6	Profile6	Profile6

Profile1
: The main central administrator has the access to the Finder granted by default. Other users must have the option **Finder access** checked in the definition of their profile.

Profile2
: Users with data privacy disabled (**Data privacy** settings in the profile set to **none (full access)**) are able to manage categories, services, metrics, scores, campaigns, global alerts, as well as import and export content and manually synchronize users and devices with AD, if they have the suboption **Allow system configuration** checked, in addition to the **Finder access** option, in the definition of their profile.

Profile3
: Users other than the main central administrator can tag objects and edit applications if they have the suboption **Allow editing of applications and object tags** checked, in addition to the **Finder access** option, in the definition of their profile.

Profile4
: Users with data privacy disabled (**Data privacy** settings in the profile set to **none (full access)**) are able to manage Web API V1 investigations, if they have the suboption **Allow management of Engine web API V1 (deprecated)** checked, in addition to the **Finder access** option, in the definition of their profile.

Profile5
: Users other than the main central administrator can access the Web API V2 (make requests to the Engine written in the NXQL language) if they have their **Data privacy** set to **none (full access)** and the option **Finder**

access enabled in the definition of their profile.

Profile6
: Users other than the main central administrator are able to supervise the installation of the Collector with the Updater from the Finder if they have the suboption **Allow management of Collectors** checked in their profile.

Related tasks

- Adding users

Active Directory authentication

Overview

Nexthink supports the authentication of users via Active Directory services. Microsoft Active Directory (AD) is currently used as the authoritative user directory in a vast number of organizations, controlling the authentication and the access rights of users.

The benefits of integrating Nexthink with the AD services include the following:

- Users need only one login and password (no need for a dedicated Nexthink account).
- Administrators can take advantage of the password policy defined in AD.

For a better user experience, Nexthink recommends to combine AD authentication with Windows authentication, so that users can log in to Nexthink without having to retype their Windows credentials.

How authentication via AD works

To enable AD authentication in Nexthink, provide the user logon in the form `someone@example.com` when creating their account in Nexthink. Make sure that the AD account exists before adding it to Nexthink.

The user logon must be composed of the *sAMAccountName* of the user, followed by the *domain* or *realm*; both separated by the @ character. Note that the previous Windows logon format `DOMAIN\username` is not supported. Note as well that if a user got a *User Principal Name (UPN)* whose user logon name is different from the sAMAccountName, you still need to use the sAMAccountName when manually configuring the user's Nexthink account; otherwise, AD

authentication will not work.

For example, if the sAMAccountName of user *John Wick* is `jwick` and he got assigned the user logon name (UPN prefix) `john.wick`, configure his Nexthink account with the first logon only:

- `jwick@example.com` - UPN prefix equal to the *sAMAccountName* - **Use this one**.
- `john.wick@example.com` - UPN prefix different from *sAMAccountName*.

To avoid the error prone manual creation of users and let users log in with their UPN (even if the UPN prefix does not match the sAMAccountName), Nexthink recommends the automatic provisioning of user accounts from Active Directory.

Beware that the account name part of the UPN is case sensitive. Thus, specify exactly the same name in Nexthink as it is registered in the AD, respecting the case. Nexthink uses the suffix part to resolve the name of the AD server (**example.com** in the example above), also known as the *domain controller*.

Once added to Nexthink, users can log in to the Finder or the Portal using their AD accounts (or Windows authentication, if enabled). During Finder login, the AD credentials provided by the user are forwarded to the Portal back-end using an encrypted channel. In the case of Portal login, the browser itself sends the AD credentials provided by the user to the Portal back-end. The Portal back-end is then responsible for contacting the AD server to authenticate the user.

Requirements for AD authentication

Allowed characters in user names

Use only printable characters in user names. The space and the following symbols are not allowed inside a user name:

```
/\[]:;|=,+*?<>@"&
```

Alternate UPN suffixes

Since Nexthink uses the UPN suffix to resolve the name of the AD server, you must use fully qualified domain names in the UPN of users. Alternate UPN suffixes defined by administrators do not work.

For instance, if the fully qualified domain name of a department inside a company is **department.example.com**, an administrator may create an alternate UPN suffix **dep.com** that is easier to memorize and quicker to type. A user in that department may thus log in to Windows using either:

- username@department.example.com
- username@dep.com

However, the user cannot log in to the Finder or the Portal using the shorter UPN suffix. Neither the Engine nor the Portal know about alternate UPN suffixes. You must use the fully qualified domain name version as user account in Nexthink.

To know the full name of a user that is logged in to Windows, open the Finder and tick the box **Use Windows authentication** in the login dialog. The retrieved user name is the actual full name of the user:

Connectivity with AD server

For the Portal to be able to connect with the AD server, the appropriate ports must be open in the appliances on which they run.

- UDP 53 for DNS.
- TCP 389 and TCP 636 for non-secure and secure LDAP connections to the AD server.

Time synchronization

Because of the technique used for authenticating users, the Portal must be synchronized with the clock of the AD server. The configuration of the AD server may nevertheless specify a tolerance regarding clock discrepancy. A difference of at most 5 minutes is generally accepted by default.

Encryption methods

Nexthink supports the following encryption methods:

- AES (128 bits)
- RC4-HMAC

On the other hand, DES encryption (legacy for Windows 98) is not supported.

Finder session saving

The Finder allows to save sessions and user credentials. This applies to AD credentials as well. If the user chooses to additionally save the password, then the Finder stores only a hash of the password for security reasons.

Related tasks

- Adding users
- Enabling Windows authentication of users
- Provisioning user accounts from Active Directory
- Preventing password saving in the Finder

Related references

- Connectivity requirements

Canonical domain names for Windows authentication

Overview

Thanks to Windows authentication (SSO), Nexthink users can conveniently log in either to the Portal or to the Finder without the need to type in their credentials each time.

When configuring Windows authentication for Nexthink, ensure that you set the canonical name of your domain (and not an alias) as the Service Principal Name which associates the service instance to the service logon account of Nexthink (*nxtPortalSSO*). Failing to do so results in users not able to log in through Windows authentication.

DNS zones and resource records

DNS zones map domain names to IP addresses and other resources. Each resource record in a DNS zone defines a single mapping. We focus our attention on two types of records:

- Type **A** records, which map a domain name to an IP address.
- Type **CNAME** records, which define a domain name that is an alias for another domain name (the *canonical* name).

Let us consider an example of a DNS zone with two resource records. It is a Forward Lookup Zone whose name is `example.com`, which is the suffix of all the hosts in the zone. The DNS snap-in of the **Administrative Tools** of Windows Server shows the resource records as follows:

Name	Type	Data
portal	Host (A)	192.168.1.100
myportal	Alias (CNAME)	portal.example.com.

The first resource record in the zone, **portal**, is a record of type A. The record maps the name `portal.example.com` to the IP address 192.168.1.100. If this host provides the authentication service, `portal.example.com` is the canonical name that you must set as Service Principal Name to properly configure Windows authentication for Nexthink.

The second resource record in the zone, **myportal**, is a record of type CNAME. The record defines an alias for `portal.example.com` that is called `myportal.example.com`. While the alias `myportal` can replace the canonical name `portal` in many contexts, it is **not** suitable for configuring Windows authentication in Nexthink.

Related tasks

- Enabling Windows authentication of users

System alerts

System alerts inform you of a special circumstance during system operation. There are four types of system alerts. For each type of system alert, find below its associated warning messages along with the description of the situation that is at the origin of the alert.

License alerts (Central License Manager)

The Central License Manager sends notifications related to the status of the product license.

The recipients of the notification are:

- To: Customer contact
- Cc: Partner contact
- Cc: Nexthink sales contact

And the following types of notification exist:

Activation key

When a new license is created on the Central License Manager, an activation key is sent. This activation key is used to activate the product.

Modification of the license

When the license is modified in the Central License Management, an automatic notification is sent.

For **online license**, the modification will be automatically applied after a maximum of 6 hours. If necessary, you can force the refresh of the license. For that purpose, go to the Portal, open the view "License Management" and click the refresh button on the top right corner of the page.

For **offline license**, the modified license file will be sent attached to the notification. This file has to be uploaded to the Portal, on the view "License

Management" such that changes are applied.

Limitation: notification are sent only if the license was activated and is not revoked.

Connectivity issues

After three days without connectivity between the Portal and the Central License Manager, an email is sent. The notification is repeated every 7 days.

Limitation: only for commercial license.

License alerts (Engine)

Besides the Central License Manager, the Engine sends some kinds of license notifications as well.

The recipient of the notifications is the administrator of the Engine

Maximum number of licensed devices reached

Triggered when the Engine reaches the maximum number of devices specified in the license and a new source appears in the network.

Limit alerts

Limit alerts warn you about a possible loss of information related to a technical limitation of the Engine.

Too many processes started on [device IP address]/[device name]
: Triggered when more than 10 000 processes have been started by a single user on a device within 15 minutes and the processes are running simultaneously. The Engine does not store information about any other process for that user beyond that limit.

Too many connections generated by executable [executable name] on [device IP address]/[device name]
: Triggered when more than 10 000 connections are established by a process on a device within 15 minutes. No more concurrent connections are stored in the Engine for that process beyond this limit.

Engine is about to reach or has reached the limit for the maximum number of ...
: Triggered when more than the 95% of the maximum number of objects of a particular type are already stored in the Engine. When the limit is reached, the Engine stores no more objects of the given type. The Engine

generates this alert for the following types of objects:

- ◊ Binaries, the maximum allowed are 40 000 binaries.
- ◊ Domains, the maximum allowed are 150 000 web domains.

Engine has detected a large amount of the following objects, which might casuse performance issues ...

Triggered when the number of objects of a particular type reaches an amount that may degrade the performance of the Engine. The Engine generates this alert for the following type of objects:

- ◊ Destinations, when the Engine has recorded more than 50 000 destinations.

The limit alerts for binaries, domains and destinations include additional information on the Engine that generated the alert (Source), and the user concerned (User), which is *admin* for system alerts.

Internal alerts

Internal alerts provide you with general information on the status of the Engine.

Server started

Triggered when the Nexthink Engine reboots.

Unable to connect to Nexthink Application Library

Occurs when the Engine cannot connect to the Application Library to get information on binaries and packages.

Server Crash alert

The server crash alert is issued on the occurrence of an unrecoverable error in the Engine.

Server crash

Triggered when the Nexthink Engine finds a minidump file in the database directory while rebooting, meaning that the Engine crashed previously.

Related tasks

- Receiving alerts

Related concepts

- Alert

Audit trail

Overview

To trace relevant activities in your Appliances (accesses, configuration modifications, starts, stops, etc), Nexthink components write to the audit log file:

```
/var/log/nexthink/audit.log
```

Find below the complete list of audit events.

Appliance

See how to configure the system log for the Appliance to record the following events:

- Logon with the SSH Nexthink account
- Commands launched with super-user privileges

Web Console

Code	Description
50000	User logged in
50001	User login failed
50003	User logged out
51000	Web Console password updated
51010	Portal remote management account password updated
51011	Portal remote management account enabled
51012	Portal remote management account disabled
51020	SSH Nexthink account password updated
51021	SSH Nexthink account enabled
51022	SSH Nexthink account disabled
51100	Appliance hostname updated
51101	Appliance static route updated
51102	Appliance static route deleted
51103	Appliance DNS server updated
51104	Appliance default gateway updated
51106	Appliance NTP servers updated

51107	Appliance NTP service enabled
51108	Appliance NTP service disabled
51109	Appliance network insterface updated
51111	rsyslog service restarted
51112	crond service restarted
51603	Automatic updates enabled / disabled
51609	Updates email recipient updated
51610	Check for updates triggered
51611	Start updates triggered
51800	Appliance reboot triggered
52000	Portal parameters updated
52001	Engine name updated
52007	Maximum stored events updated
52010	Portal server address updated
52010	Portal admin account reset
52011	Aggregation policy updated
52012	Domain compression updated
52090	Engine stopped
52091	Engine started
52100	Internal network removed
52100	Internal network added
52105	Engine internal domains configuration updated
52200	Active directory added
52201	Active directory removed
52550	Engine Mobile Bridge parameters updated
53090	Portal stopped
53091	Portal started
53092	LLM started
53093	LLM stopped

Portal

Code	Description
20001	Portal is starting
20002	Portal is up and running

20004	Portal is stopped
20101	User logged in
20102	User logged out
20103	User login failed
20201	User created
20202	User removed
20203	User updated
20204	User profile updated
20205	User domain ownership updated
20206	Role added
20207	Role updated
20208	Role removed
20209	Profile added (with number of roles)
20210	Profile updated (with number of roles)
20211	Profile removed
20501	Hierarchy added
20502	Hierarchy removed
20503	Hierarchy updated
20504	Definition of entities updated
20701	Engine added
20702	Engine removed
20703	Engine connected
20704	Engine disconnected
20801	Finder user logged in
20803	Finder user login failed

Engine

Code	Description
10001	Engine is up and running
10002	Engine stopped with error
10003	Engine stopped gracefully
10004	Engine stopped forcefully
10005	Database created
10006	Finder user logged in

10007	Finder user logged out
10008	Finder user login attempt
10009	Finder account created
10010	Finder account deleted
10011	Finder account updated
10012	Finder account password changed
10017	Global alert created
10018	Global alert deleted
10019	Global alert updated
10026	LDAP synchronization request
10028	Object manually tagged
10029	Binary filtering rule updated
10030	Executable filtering rule updated
10031	Application filtering rule updated
10032	Device filtering rule updated
10034	Finder request execution
10035	Alert execution
10036	Web API request execution (deprecated)
10038	License updated
10039	NXQL request execution

The start and stop commands for the Engine that are executed from the CLI are logged in journalctl. Use the following command to retreive them:

```
sudo journalctl -u nxengine@*.service | grep systemd
```

Related tasks

- Configuring the system log

References

Components of the Collector

Overview

The Collector is mainly composed of a couple of kernel drivers, along with a small set of services and libraries, that gather information about the devices in your corporate network and their activity. The Collector periodically sends all the gathered information to an Engine, where it is processed and stored. Other tools that are delivered with the Collector help you with its installation and configuration.

Find in this document the description of all the different components of the Collector and the filesystem paths where to find them in the devices of the end-users after installation. This article details as well the registry keys and the additional files created or modified during the installation of the Collector.

Windows Collector

The Windows version of the Collector includes several features in addition to the gathering of user activity. These extra features require a comprehensive set of components.

Applies to platforms:
Windows Collector binaries

For all versions of Windows, the following components are installed:

- **Main driver**: A kernel mode driver that gathers valuable information from the device of the end-user.
- **Network specific driver**: A kernel mode driver that detects network connections.
- **Helper service**: A Windows service that complements the main driver by collecting additional information.
- **Printing info library**: A dynamic link library that is responsible for detecting printing activity.
- Optional **Command line configuration tool**: A tool to configure the Collector from the command line.
- Optional **Control Panel extension**: A tool to control the behaviour of the

Collector that is added to the Control Panel of Windows.
- **Automatic updates**: A component of the Collector that is responsible for downloading new versions and updating the installed components.
- **Coordinator**: Coordination of the Collector with the Appliance for detecting new updates and communicating end-user feedback.
- **End-user feedback**: Components for presenting the questions of campaigns and getting answers from the end-users.

Component	File	Path
Main driver	nxtrdrv.sys	C:\Windows\System32\drivers
Network specific driver	nxtrdrv5.sys	C:\Windows\System32\drivers
Helper service	nxtsvc.exe	C:\Windows\System32
Printing info helper library	nxtdll.dll	
Command line configuration tool	nxtcfg.exe	
Control Panel extension	nxtpanel.cpl	
Automatic updates	nxtupdater.exe	
Coordinator	nxtcoordinator.exe	
End-user feedback	• nxteufb.exe • nxtray.exe • nxtray.exe.config	

Starting from **Windows 8**, these additional binaries are also installed:

- **Metro apps helper library**: A dynamic link library that detects the execution of Metro apps.

Component	File	Path
Metro apps helper library	nxtwrt.dll	C:\Windows\System32

Registry keys of the Windows Collector

On installation, the Collector creates the following keys in the Registry of Windows:

- HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services\nxtrdrv
- HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services\nxtrdrv5

- HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services\Nexthink Service
- HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Nexthink Coordinator\Modules\Updater
- HKEY_LOCAL_MACHINE\SYSTEM\Nexthink\Updater
- HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Nexthink Coordinator
- HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Nexthink Coordinator\Modules\EndUserFeedback
- HKEY_USERS\S-1-5-21-2281471460-584676728-3927365163-1676\SOFTWARE\NEXThi
- HKEY_CLASSES_ROOT\nxtrayproto

On **Windows 10**, this additional key is created, used and maintained by the Action Center:

- HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Notifications\Cu

Additional files of the Windows Collector

Find the log files of the Collector here:

- C:\Windows\nxtsvc.log
- C:\Windows\nxtsvc.1.log
- C:\Windows\nxtsvc.2.log
- C:\Windows\nxtupdater.log
- C:\Windows\nxtupdater.log.bk
- C:\Windows\nxtcoordinator.log
- C:\Windows\nxtcoordinator.log.bk
- C:\Windows\nxteufb.log
- C:\Windows\nxteufb.log.bk
- %temp%\nxtray.log
- %temp%\nxtray.log.<timestamp>

Finally, Windows creates a cached copy of the kernel drivers in two folders whose names start with the name of the drivers (**nxtrdrv** and **nxtrdrv5**, respectively) followed by an unique identifier that depends on the version of the driver itself. Find the folders here:

- C:\Windows\System32\DRVSTORE

Mac Collector

The Mac version of the Collector has just the necessary components to report user activity.

Applies to platforms:
Mac Collector binaries

- **Driver**: A kernel mode driver that gathers valuable information from the device of the end-user.
- **Helper service**: A Mac Os daemon that complements the driver by collecting additional information.

Component	File	Path
Driver	nxtdrv.kext	/Library/Extensions
Helper service	nxtsvc	/Library/Application Support/Nexthink

Configuration files of the Mac Collector

Component	File	Path
Daemon registration file	com.nexthink.collector.driver.nxtsvc.plist	/Library/LaunchDaemons/
Daemon configuration file	config.plist	/Library/Application Support/Nexthink
CrashGuard file	crashguard	

Additional files of the Mac Collector

Find the log files of the Mac Collector here:

- /var/log/kernel.log
- /Library/Logs/nxtsvc.log
- /Library/Logs/nxtsvc.1.log
- /Library/Logs/nxtsvc.2.log
- /Library/Logs/CrashReporter

Related tasks

- Installing the Collector
- Updating the Collector

Related references

- Collector MSI parameters reference table
- Nxtcfg - Collector configuration tool
- Collector (Product Overview)

Operating systems supported by the Collector

Windows

The Collector V6 supports neither Windows XP nor Windows Vista, but it has better support for Windows 7 and Windows 8 / 8.1 than the Collector V5. Besides, the Collector V6 supports Windows 10.

Note that the Windows Collector V5 has reached its end-of-life and it is no longer supported by Nexthink.

Windows OS	V5 (EOL)	V6
Windows XP	( )	
Windows Vista	( )	
Windows 7	( )	
Windows 8 / 8.1	( )	
Windows 10		

Windows server

Whereas previous versions of the Collector V5 support the installation on Citrix / RDS servers only, the latest Collector V5 and the Collector V6 fully support the installation on Windows Server (see supported versions below).

Note that the Windows Collector V5 is no longer supported by Nexthink.

Windows Server OS	V5.3 (EOL)	V6
Windows Server 2003	( )	
Windows Server 2003 R2	( )	
Windows Server 2008	( )	
Windows Server 2008 R2		

Windows Server 2012		
Windows Server 2012 R2		

Mac

The Mac Collector for OS X and macOS is technically a V5, but it is officially supported by Nexthink V6. Nexthink supports the installation of the Mac Collector only on those operating systems which are currently supported by Apple.

Mac OS	V5.3	V5.4	V5.5	V5.6
OS X 10.11 El Capitan				
macOS 10.12 Sierra				
macOS 10.13 High Sierra				

Related references

- Server support

Server support

Overview

Although Nexthink is a solution designed for monitoring the devices of the end-users, the same monitoring techniques may be applied to servers to some extent. In Nexthink V5, it is possible to install the Collector in Citrix / RDS servers, where each server is roughly equivalent to having a group of end-user devices. Starting from V6, Nexthink also supports the installation of the Collector in other types of Windows servers (note that V5.3 offers server support for specific Windows Server versions as well). Therefore, a new type of device is available in Nexthink: the **server**.

The Collector reports basically the same analytics from a server as from the device of an end-user, except for some security-related information. Indeed, the technology used by the Collector to retrieve security information is not available on servers, so data on the following areas is missing:

- Antivirus
- Antispyware
- Firewall

Also keep in mind that, as for normal devices, the Collector does not report incoming connections for servers. Only outgoing connections are recorded.

Sizing your installation

Servers usually generate much more activity than end-user devices. As a rule of thumb for sizing your installation, consider that a server is equivalent to 20 end-user devices. Calculate the number of Engines that you need according to the following formula, where **S** is the number of servers and **D** is the number of end-user devices in your setup:

- Number of Engines = (20 * S + D) / 6000

The hardware requirements for the Engines apply.

Traffic reduction

The typically higher network activity of servers with respect to end-user devices often generates lots of connections and events that might saturate the Engine. Therefore, the Engine has set in place a strategy to reduce the traffic of servers, although it applies to the traffic of all types of devices. When a device connects to too many destinations or opens too many ports in a burst, the Engine can automatically decide to aggregate these connections into a single connection, setting its port or its destination value to **multiple**. In the case of a device launching a burst of web connections to many domains, the Engine aggregates the connections as one connection where the value of the domain is **multiple**.

In this manner, individual information about the connections is lost, but the amount of traffic information stored in the Engine is kept to a reasonable level. Otherwise, the explosion of connections could drastically reduce the history available in the Engine. Even with the traffic reduction policy in place, you should expect a slight reduction on the history available in the Engine if you install the Collector on servers.

The strategy for reducing traffic is configurable (see below). Choose between **normal** and **aggressive**, depending on whether you prefer to aggregate connections gently or almost right away. An aggressive policy lets you keep a longer history in the Engine at the expense of losing the individual information of more connections.

Taxonomy of servers

Find below a classification of servers according to their function. Depending on the function of the server, you should be aware of the chances that the Engine reduces server traffic.

Client-like (Citrix, RDS)
: Supported from V5. Traffic reduction rare.

Application (Mail, SQL Database)
: Traffic reduction depending on load.

Agent manager (SCCM)
: Traffic reduction probable.

UDP server (DNS)
: Traffic reduction certain.

Proxy (Web proxy)
: Expect to have web traffic reduction and bigger Collector usage of your network. Installation on a web proxy is therefore not recommended.

Bot (Scanner, Stress test)
: Not supported, since just one server may behave as thousands of end-user devices. Do not install the Collector on servers of the bot class.

Engine configuration

Set the traffic reduction policy by editing the configuration file of the Engine. In addition, configure whether you want the outgoing traffic of servers to be used in the computation of services. Usually, this only makes sense for client-like (Citrix / RDS) servers.

1. Log in to the CLI of the Engine.
2. Open the configuration file for editing:
   ```
   sudo vi /var/nexthink/engine/01/etc/nxengine.xml
   ```
3. Write the following settings in the correspoding section of the configuration file:
   - In the **aggregation** section, set **destination_reduction_policy** to **normal** or **aggressive**.
   - In the **service** section, set **compute_service_on_server** to **true** or **false**.
4. Save your changes and exit:
   ```
   :wq
   ```

Alternatively, use the nxinfo command to change the configuration settings of the Engine. For instance, to set the destination reduction policy to normal and turn on the computation of services on servers, type in:

- `sudo nxinfo config -s aggregation.destination_reduction_policy=normal`
- `sudo nxinfo config -s service.compute_service_on_server=true`

After modifying the settings, find the following lines with the provided values in the configuration file:

```
<aggregation>
  <destination_reduction_policy>normal<destination_reduction_policy>
</aggregation>
<service>
  <compute_service_on_server>true<compute_service_on_server>
</service>
```

Depending on the types of servers that you have, use the settings described below.

Only client-like (Citrix / RDS) servers

- Destination reduction policy: **normal**.
- Compute service on servers: **true**.

Whenever possible, assign the Citrix / RDS servers to the same Engines of the end-user devices that they serve.

Only non-client servers

- Destination reduction policy: **aggressive**.
- Compute service on servers: **false**.

If possible, assign non-client servers to an Engine separate from those used by end-user devices.

Mixed setup

- Destination reduction policy: **aggressive**.
- Compute service on servers: **true**.

In the case of a mixed setup with both client-like and non-client servers, you may want to compute a service for client-like servers and actual clients (end-user devices) only.

To compute services selectively, manually tag those devices that you want to include in the computation and set a condition on the services for taking into account only those tagged devices. For instance:

1. Create a category called, for example, **Compute services**.
2. Add two keywords to the categores without auto-tagging rules: **yes** and **no**.
3. Manually tag all your end-user devices and client-like servers with the keyword **yes** and the non-client servers with the keyword **no**.
4. Add a condition on devices in the definition of each service to include in the service only those devices tagged with the **yes** keyword:

Whenever possible, assign Citrix / RDS servers to the same Engines of the end-user devices that they serve and group non-client servers in separate Engines.

Incompatibility of Collector with Receive Segment Coalescing (RSC)

Currently, the Collector is incompatible with Receive Segment Coalescing, a technology present in Windows Server 2012 and later that improves the reception of network traffic by offloading network processing from the CPU to the network interface. Windows desktop operating systems also support RSC since Windows 8, but the use of RSC is usually limited to Windows servers with high input traffic.

The driver of a network interface that is compatible with RSC can coalesce multiple TCP segments received and present them as a single larger segment to the networking layer of the operating system.

The monitoring capabilities of the Windows Collector at the kernel level effectively disable RSC on supported network interfaces.

Related references

- Hardware requirements
- Operating systems supported by the Collector
- Receive Segment Coalescing (external)

Compatibility mode

Compatibility mode is a temporary state of Appliances V6.6 (or higher) that is previous to federation. In compatibility mode, Appliances work in much the same way as in V6.5 (or previous), hence its name.

Appliances enter compatibility mode right after their installation or after being updated from V6.5 (or previous). In this mode, there are no centralized configuration settings or coordinated updates. Unless you have a particular problem when federating your Appliances, it is recommended to switch from compatibility mode to federated mode as soon as possible.

An Appliance in master / slave configuration, that is, an Appliance with both the Portal and the Engine installed, is automatically federated. Therefore, it can never enter compatibility mode.

On the other hand, a slave Appliance which runs in compatibility mode shows two configuration settings in the Web Console that disappear when the Appliance is federated:

- The **Update** section
- The **Portal address** field

The **Update** section is found in the left-hand side menu of the Web Console, when the **Appliance** tab is selected. It allows for slave Appliances to be updated individually. After federation, the **Update** section is only available in the Web Console of the master Appliance.

When working in compatibility mode, the Engine is able to send real-time information to the Portal only if you specify the address of the Portal in the Web Console of the slave Appliance. The **Portal address** setting is found by selecting the **Engine** tab, the **General** section of the left-hand side menu, and looking under **Parameters**. After federation, the address of the Portal is known to the Engine and this parameter is no longer needed; thus, it is removed from the Web Console.

Related tasks

- Federating your Appliances