# RAYVENTORY®

The most comprehensive Solution for Discovery and Inventory of Software and Hardware

User Guide
12.4

•rayNET

User Guide



Raynet GmbH Germany
See our website for locations.

www.raynet.de

# Contents

# Introduction

RayVentory Scan Engine is a component of the RayVentory HAM / SAM solutions. A RayVentory implementation may use RayVentory Scan Engine as a source for RayVentory data where this data is processed by a RayVentory server installation.

## About this Guide

This guide is an operation manual for RayVentory Scan Engine intended for end-users. The manual also covers certain aspects of RayVentory Scan Engine which are relevant to software architects and for the implementation of RayVentory Scan Engine.

## Documentation Requests

We welcome suggestions and input on the various documentation resources available for RayVentory Scan Engine and its components. Feedback and other concerns can be forwarded through local Raynet support representative.

# Installation

The installation sources for RayVentory Scan Engine are provided as an MSI package.

## Prerequisites

RayVentory Scan Engine can be installed on Microsoft Windows 7 or later. It does not require a Microsoft Windows Server Edition, but can be installed on those as well.

Furthermore, RayVentory Scan Engine requires the .NET Runtime Environment and Visual C++ Redistributable for Visual Studio 2015-2022 to run.

In order to use the Oracle Inventory feature of RayVentory Scan Engine, at least a Java SE 6 compatible runtime environment is required. RayVentory Scan Engine automatically detects the installed Java Runtime Environments and prompts the user to pick one if necessary.

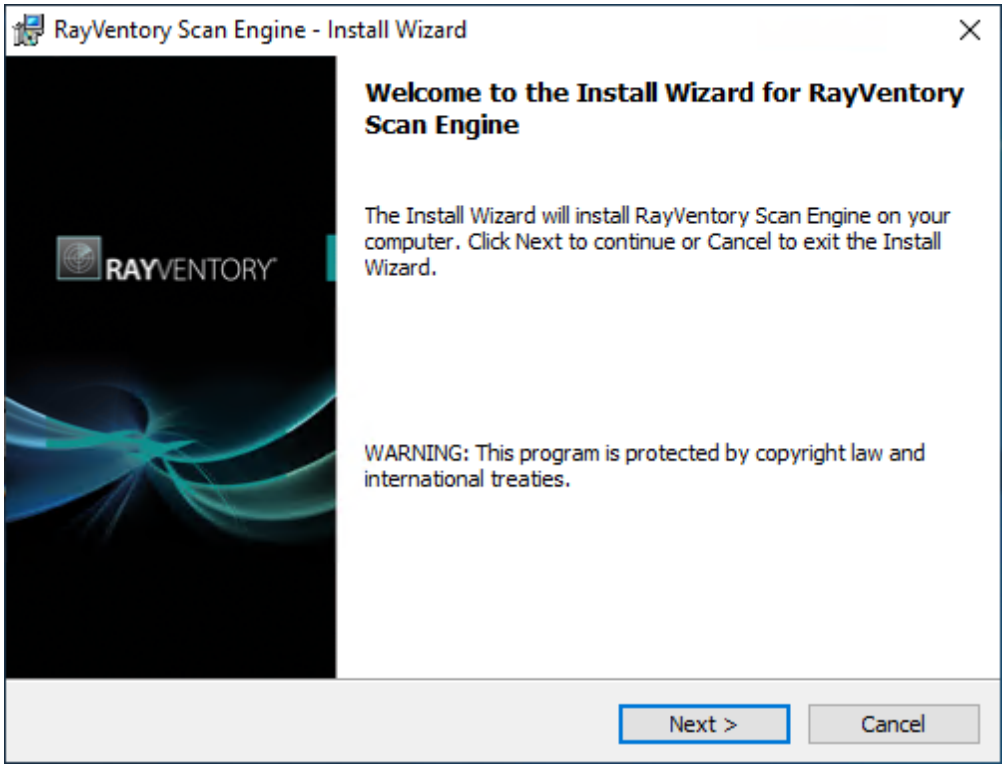
The Oracle Audit feature (support for running the Oracle Review Lite Script) and the Oracle Database Feature Usage Statistics feature (support for running the Oracle Database Feature Usage Statistics script) require SQLplus to be installed.

## Installation Process

In order to start the **Install** wizard, double-click the `.msi` file.

## Welcome Screen

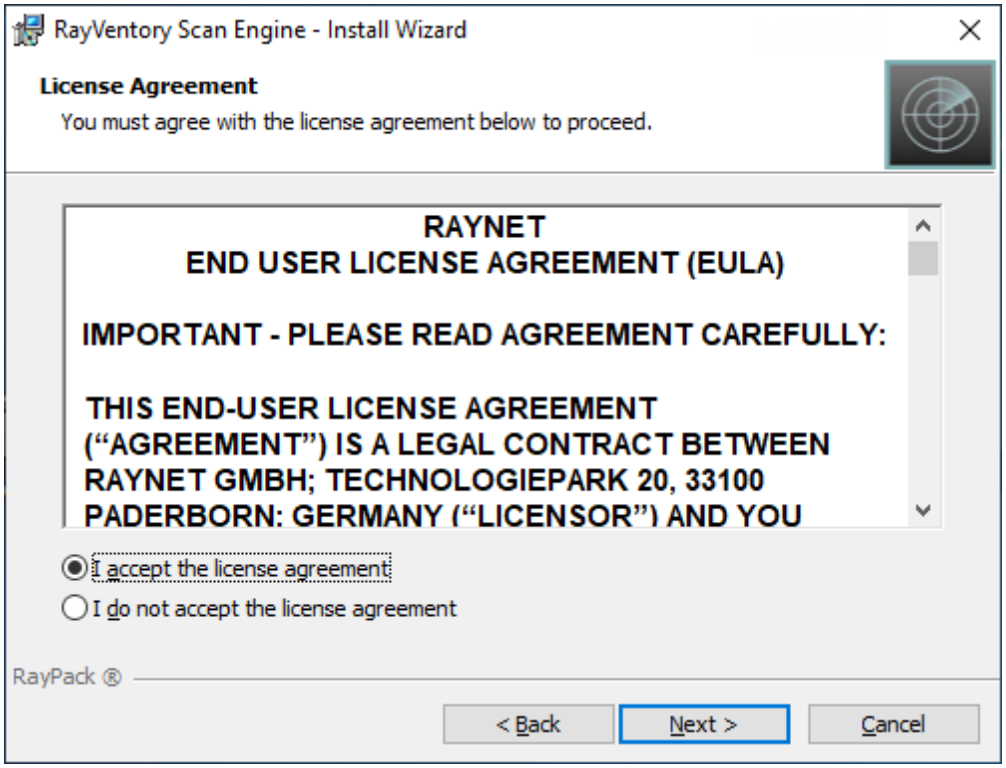
This is the first screen in the **Install** wizard.




Click on the **Next >** button to continue the installation.

## License Agreement Screen

This screen contains the license agreement. Read the license agreement and select either **I accept the license agreement** or **I do not accept the license agreement**.

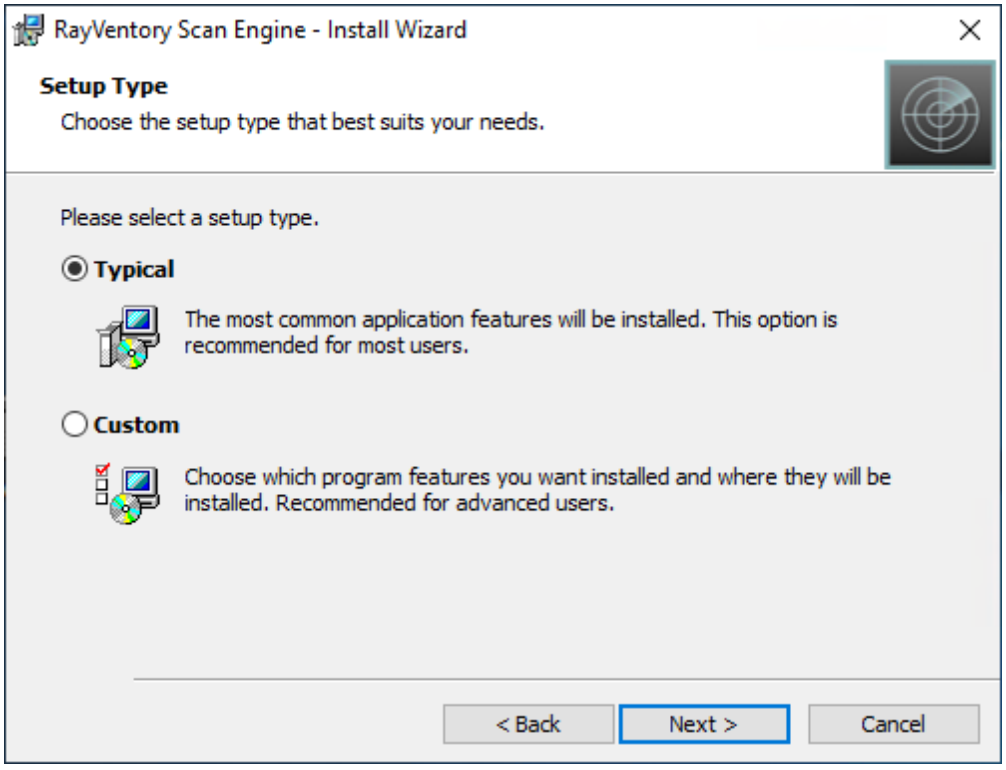



If **I do not accept the license agreement** has been selected, it will only be possible to either go back to the **Welcome** screen by clicking on the **< Back** button or to abort the installation by clicking on the **Cancel** button. If clicking on the **Cancel** button a confirmation prompt will appear to ensure that the installation should really be aborted.

After selecting **I accept the license agreement** click on the **Next >** button to continue with the installation.

## Setup Type Screen

In this screen the setup type of the installation can be chosen.




There following options are available:

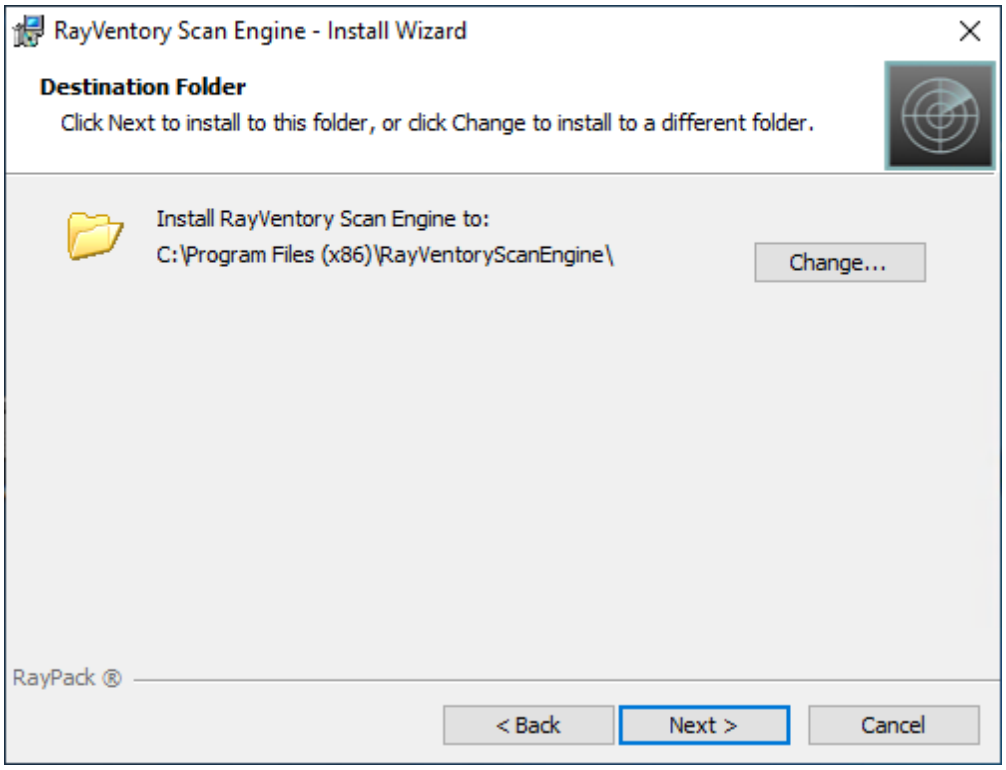
- **Typical**: This setup type will install the Scan Engine using the default setup. This setup type is recommended if there are no special requirements.
- **Custom**: This setup type needs further configuration. When there are special requirements regarding the features needed or the location of the installation choose this setup type.

Click on the **Next >** button to continue the installation.

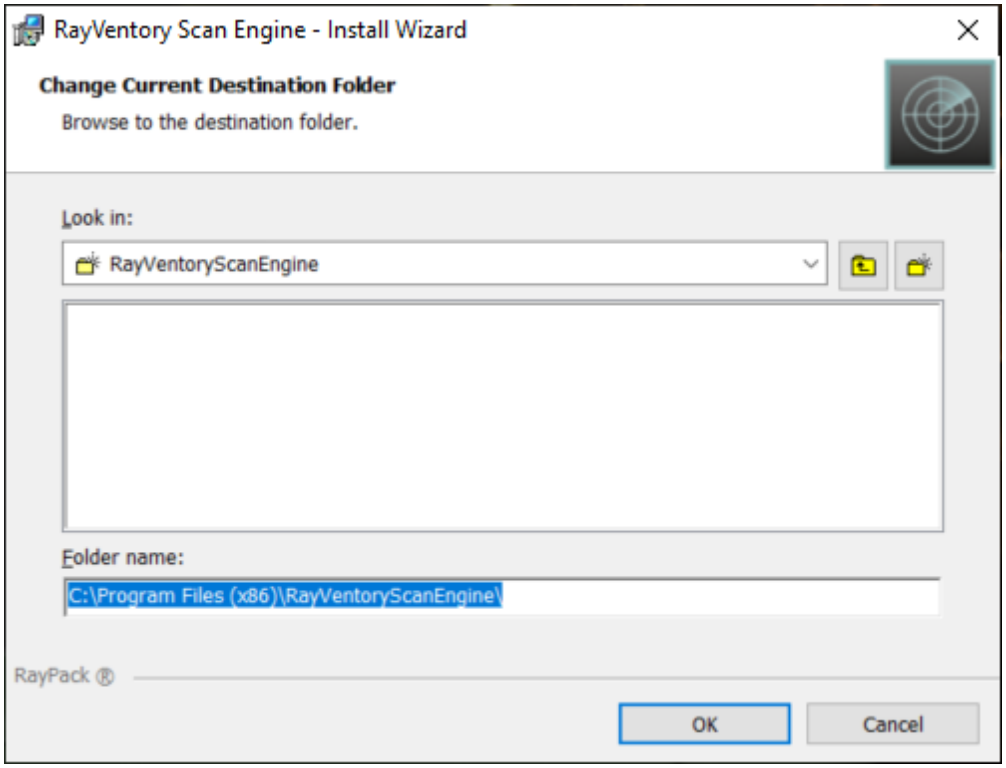
## Destination Folder Screen

(Only if **Custom** has been chosen in the **Setup Type** screen)

In this screen the install location of the Scan Engine can be configured. It will only be available if **Custom** has been chosen as setup type.




Click on the **Change...** button in order to open the file browser.




In the file browser, navigate to the target folder and select it. After selecting the folder, click on the **OK** button.

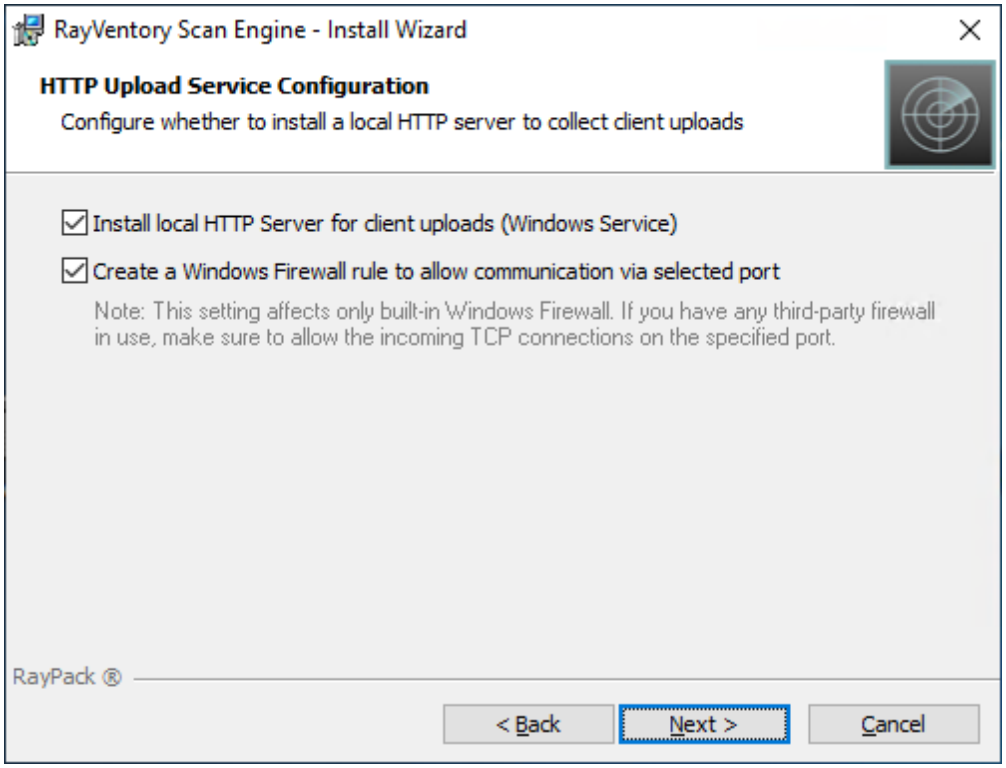
By default, the target folder for the installation is `C:\Program Files (x86)\RayVentoryScanEngine`.

Click on the **Next >** button to continue the installation.

## HTTP Upload Service Configuration Screen

(Only if **Custom** has been chosen in the **Setup Type** screen)

This screen can be used to define if a local HTTP server for client uploads and a Windows Firewall rule should be defined during the installation process.




Uncheck the **Install local HTTP Server for client uploads (Windows Service)** in order to not install the service or if the service is already installed on the device. By default, the service will be installed (Default Port: 591).

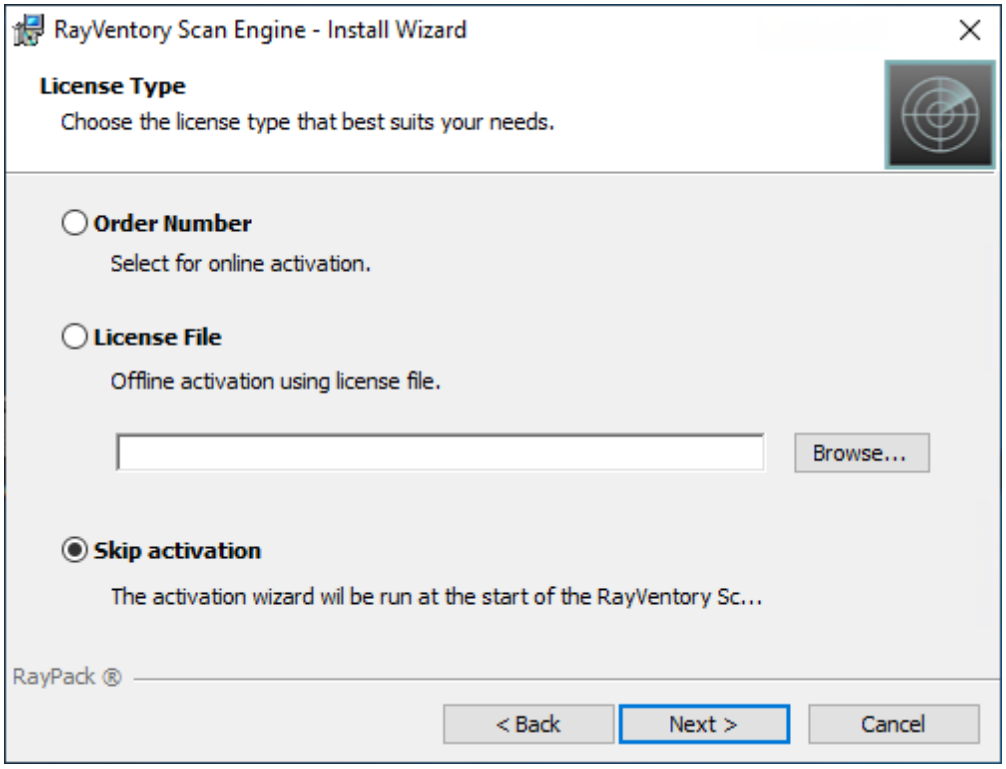
Uncheck the **Create a Windows Firewall rule to allow communication via selected port** if no rule for the Windows Firewall shall be created or if another firewall is being used. If a third-party firewall is in use, an appropriate rule should be created manually in order to allow the incoming TCP connections on the specified port. By default, the rule for the built-in Windows firewall will be created.

The default port used by RayVentory Scan Engine is `591`.

Click on the **Next >** button to continue the installation.

## License Type Screen

In this screen the type of licensing used to activate the product can be configured.




The different options available in this screen are:

- **Order Number**: This option can be used in order to activate RayVentory Scan Engine using the order number received from Raynet. The process on how to activate RayVentory Scan Engine using the order number is described in the Customer Information Screen step. In order to use this activation method an active internet connection is needed.
- **License File**: Select this option in order to activate RayVentory Scan Engine with an existing license file. The license file must be of the type `.rsl` and must be valid for the hardware id of the machine. In order to select the file, click on the **Browse...** button to open the file browser and select the file. After the file has been selected, click on the **Open** button of the file browser.
- **Skip activation**: Select this option in order to skip the activation during the setup and activate RayVentory Scan Engine at a later time. The activation wizard will be run automatically once RayVentory Scan Engine is started for the first time. It can also be executed manually using the `Raynet.LicenseActivation.exe`. This option should also be used if planning to use the Floating License Server.

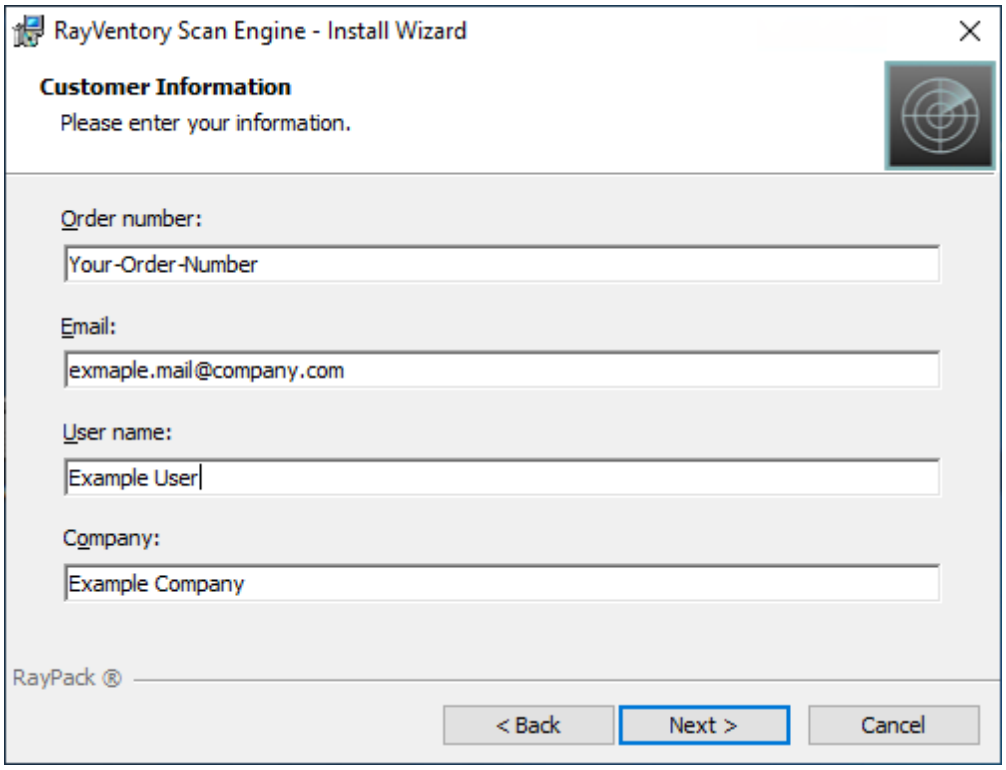
Detailed information about the different activation methods can be found in the License Wizard chapter.

Click on the **Next >** button to continue the installation.

## Customer Information Screen

(Only if **Order Number** has been chosen in the **License Type** screen)

This screen is only used if the **Order Number** option has been chosen for the activation. All information necessary for the activation needs to be entered.




The following information are necessary in order to use the activation method:

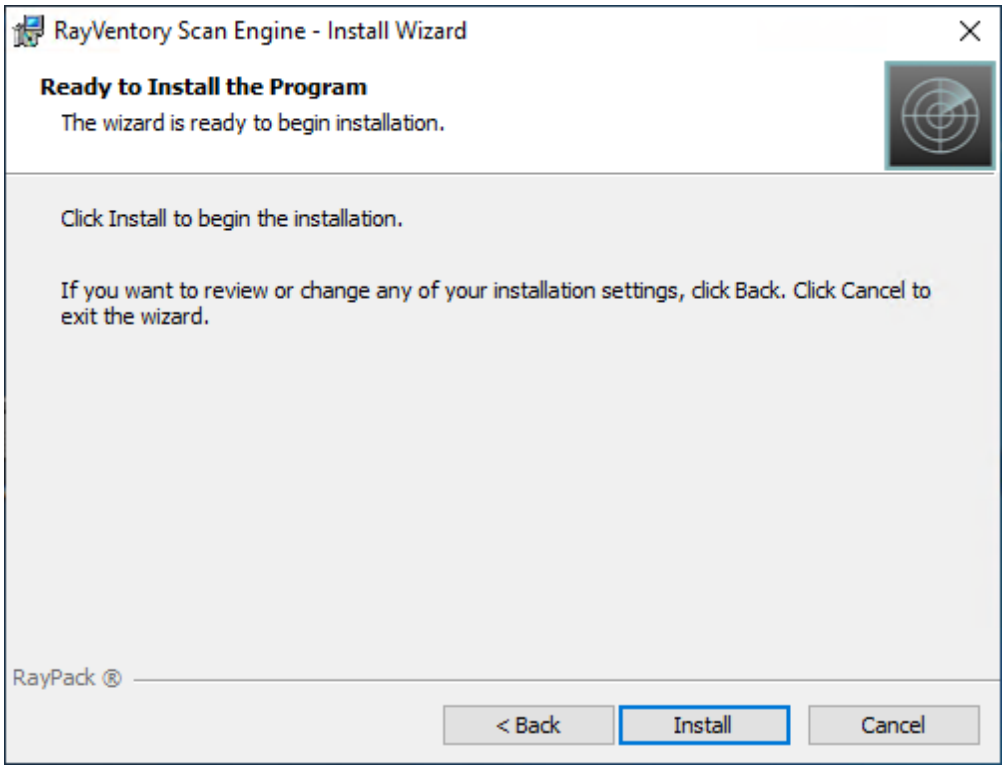
- **Order number**: The unique order number received when RayVentory Scan Engine was purchased. The order number can be recovered by contacting the Raynet sales team.
- **Email**: This is the email address of the user who performs the activation. The given email will only be used if there are any problems or important information regarding the license.
- **User name**: The name of the user performing the activation. It does not need to be the same name used for the order.
- **Company**: The name of the company for which RayVentory Scan Engine will be licensed. This name will appear in the License and Edition view of RayVentory Scan Engine.

When the order number has been confirmed by the RayVentory Scan Engine license server, the product has been activated successfully for the machine.

Click on the **Next >** button to continue the installation.

## Ready to Install the Program Screen

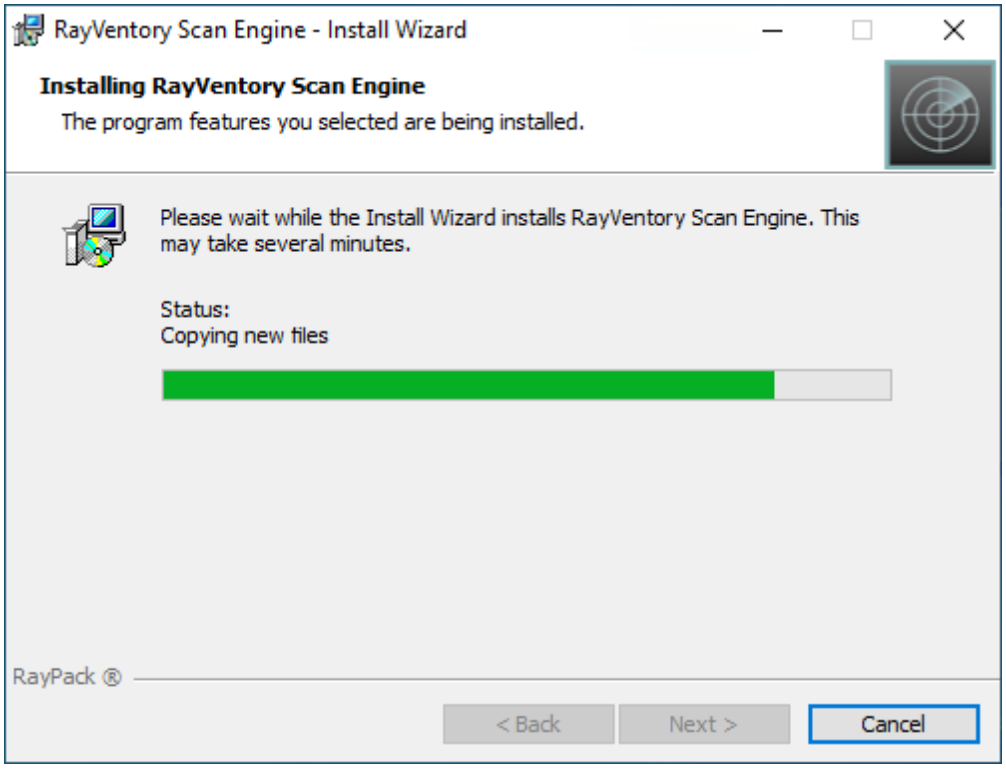
After everything has been configured successfully, the **Ready to Install the Program** screen will be shown.




Click on the **Install** button to start the install process.

## Installing RayVentory Scan Engine Screen

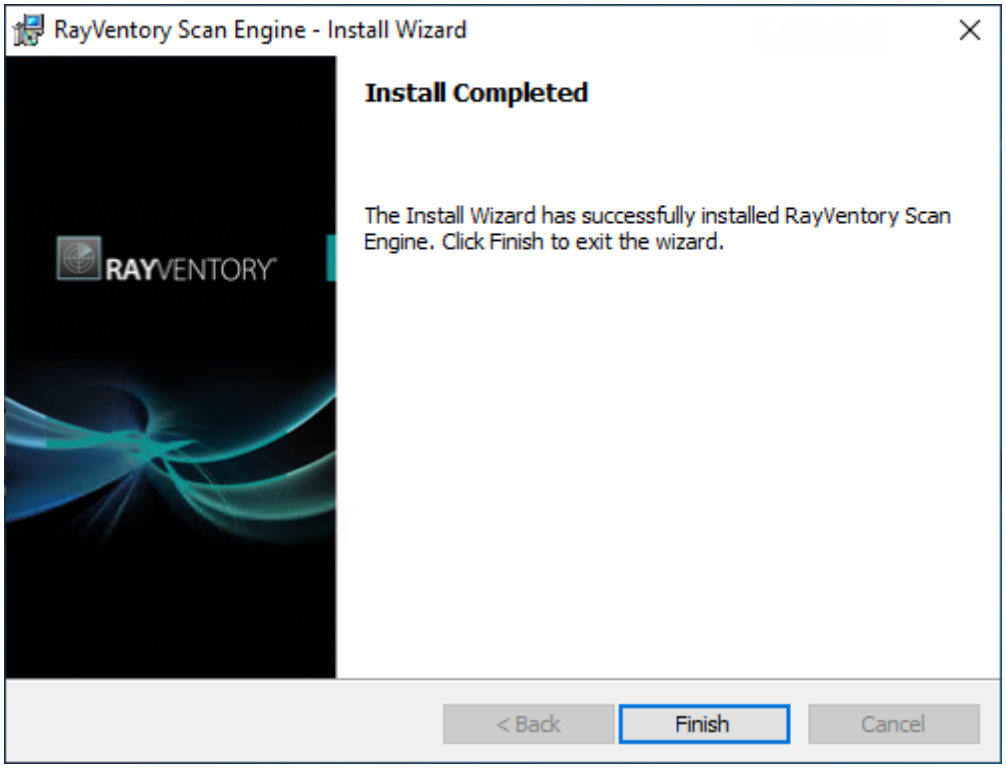
This screen is shown during the installation process.




It shows a progress bar and the specific action that is currently being executed by the installation wizard. While this screen is shown, it is possible to abort the installation by clicking on the **Cancel** button.

## Install Completed Screen

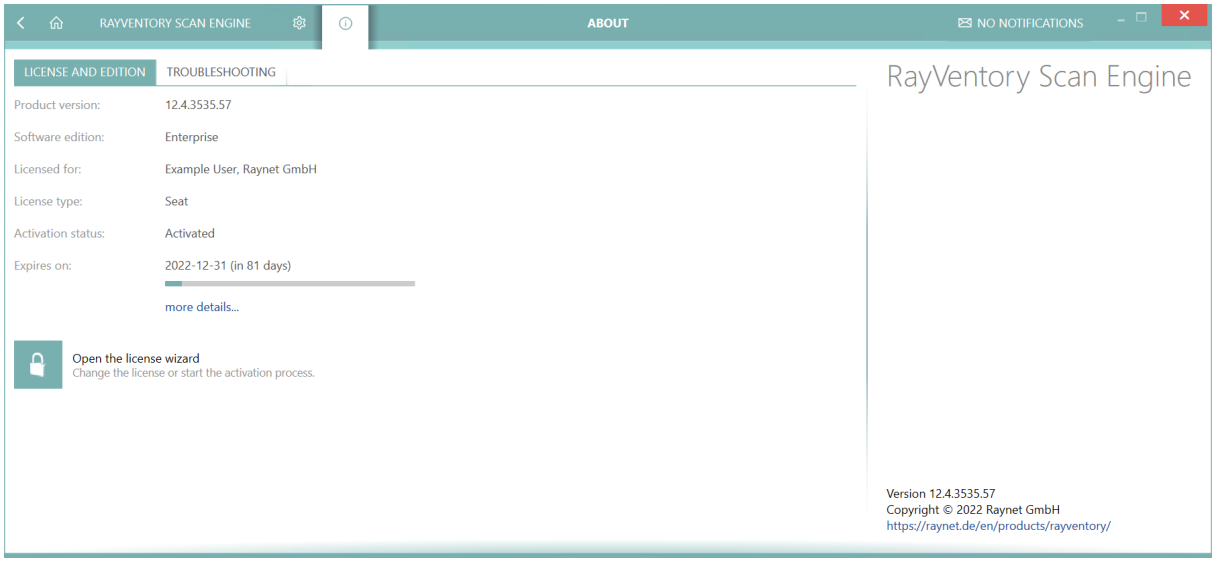
This screen will be shown after the installation has been successfully completed.




Click on **Finish** to **Close** the Install wizard and Finish the installation process.

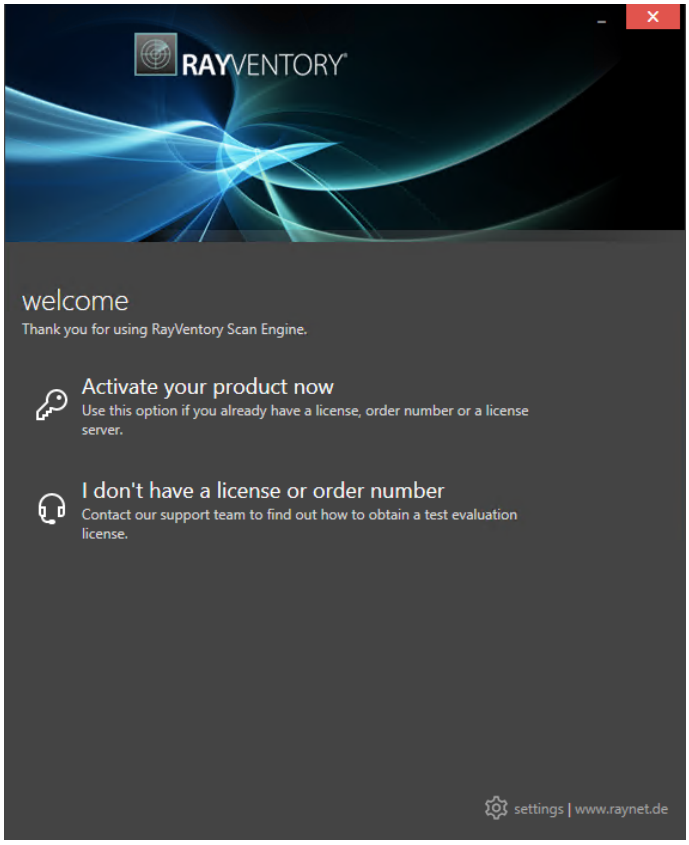
# License Manager

RayVentory Scan Engine is being controlled by a license. The product can be activated by using the built-in license manager with either a license file or an order number. The license controls the available features and it may have an expiration date.

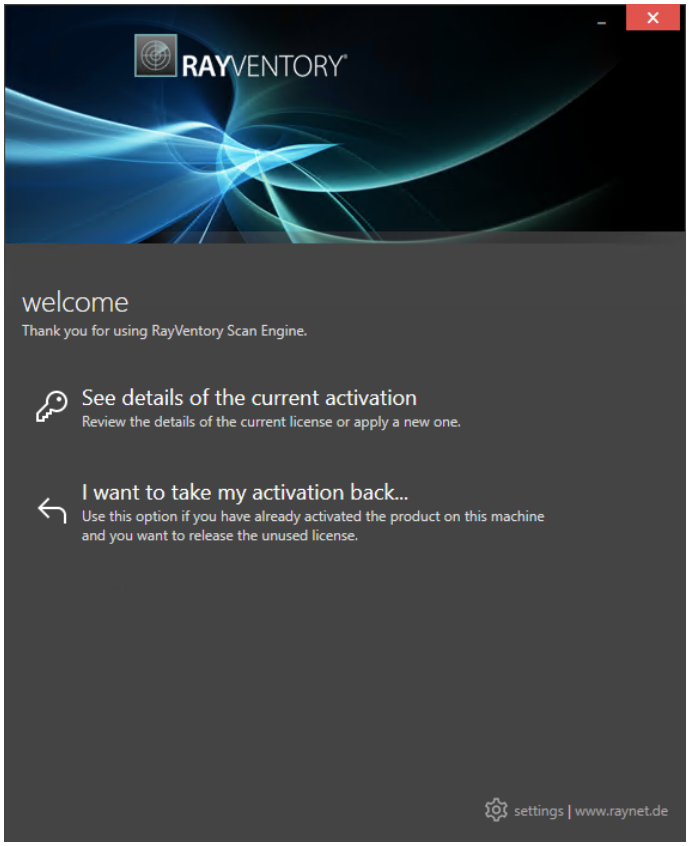


# License Wizard

On the initial start of RayVentory Scan Engine, the licensing wizard is shown. If the need to transfer an existing license arises, the license wizard can be started manually. There are a variety of ways in which a license can be activated and below they are described in detail.



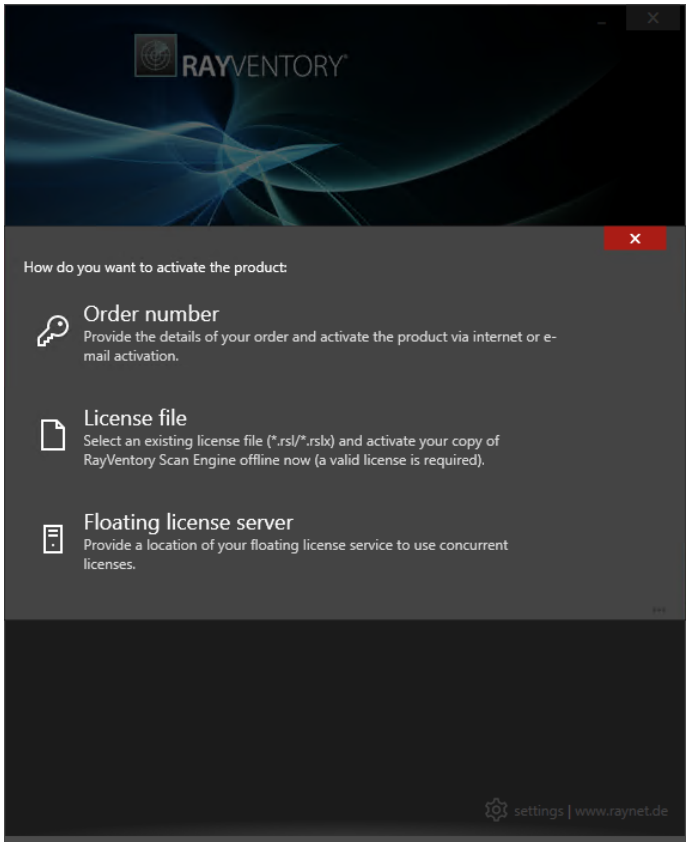

First time activation



The main screen when the product has already been activated

**Activate your product now**

This option should be used to activate the product using one of the following methods:




- **Order number**
  Online activation using a valid order number received from Raynet (recommended for most users)

- **License file**
  Offline activation using a license file (`.rsl`) received from Raynet

- **Floating license server**
  Activation using a local floating license server.

### See details of the current activation

This options shows the details of the current activation. This option is only visible if the product has already been activated or if a floating license server has been configured

This option also allows to reactivate the product using a different order number or a different floating license server connection details.

### I don't have a license or order number

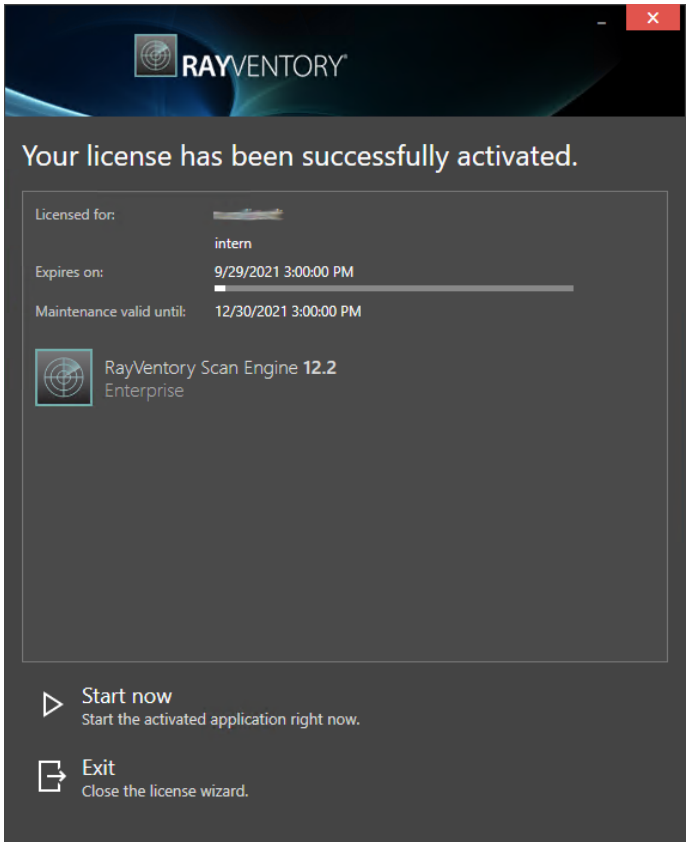
Choose this option if there is neither a license nor an order number available. For in-depth information please read this section. This option is only visible if the product has not been activated yet.

### I want to take my activation back...

Use this option to deactivate a currently licensed version of RayVentory Scan Engine. For in-

depth information please read this section. This options is only visible if the product has been already activated.

Once the license file has been generated or copied to the correct location the following will be shown...




Then the option of starting RayVentory Scan Engine or just closing the activation wizard is made available.

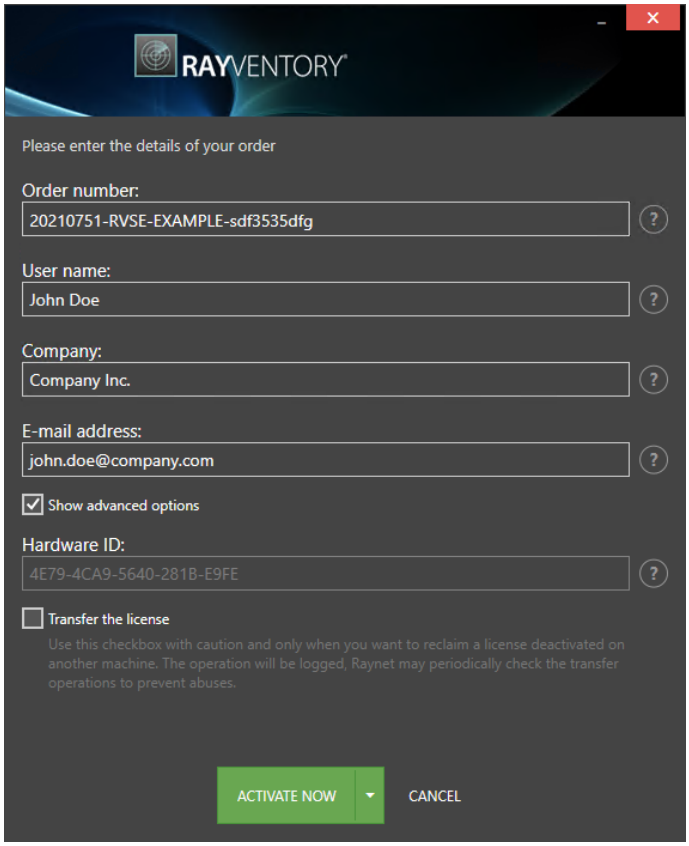
# Troubleshooting

If any issues arise during the activation process, please contact our help desk to receive assistance in activating RayVentory Scan Engine.

## Order Number

RayVentory Scan Engine can be activated either directly online or via email once the order number has been delivered. The activation process generates a license file (`*.rsl`) that is created (or must be copied) to the installation directory of RayVentory Scan Engine (in the same location as the `RayVentory Scan Engine.exe`). When performing an online activation, sufficient permissions must be readily available to allow the creation of the license file in the installation directory. The activation **binds** the license to the machine on which it was activated

on. This is the only time that an active connection to the internet is required (if activating online).




Choosing the **ACTIVATE NOW** button, connects to the Raynet license server using the information provided and will dynamically generate a license file. Choosing the **ACTIVATE MANUALLY** button will open a dialog as shown here. Choosing the **CANCEL** button will abort the activation process.

# Order Details

**Order number:**
This is the unique order number received when RayVentory Scan Engine has been purchased. If it is necessary to recover the order number, please contact our sales team.

**User name:**
This is the name of the user that is activating RayVentory Scan Engine. It does not need to be the same name used to order RayVentory Scan Engine.

**Company:**
This is the name of the company for which RayVentory Scan Engine will be licensed. This name will appear in the **License and Edition** view of RayVentory Scan Engine.

**E-mail address:**

This is the email address of the person that performs the activation. We respect the privacy of our customers, this email address will only be used by Raynet and only when there are any problems or important information regarding the license.

# Advanced Options

On choosing the advanced options check box, extended information and possibilities of the licensing and activation of RayVentory Scan Engine are shown.

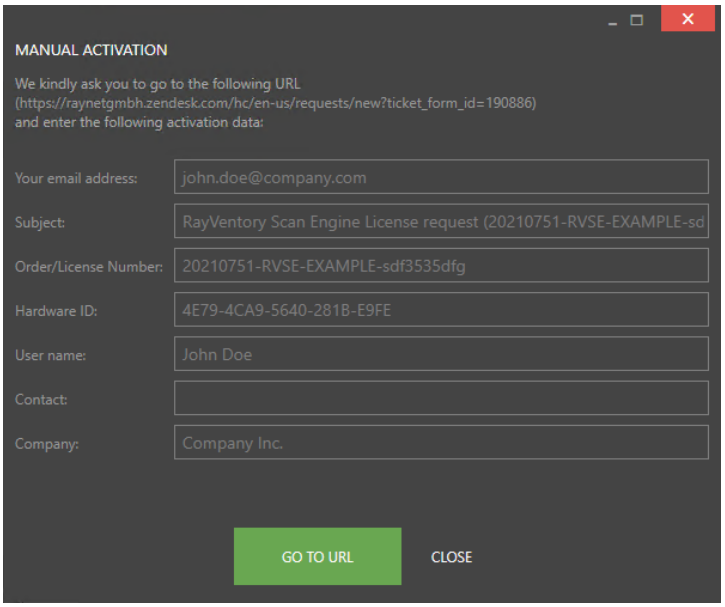
**Hardware ID:**
This is a ID calculated based on the hardware on which the activation is taking place on. The ID is unique, but cannot be used to personally identify a user. It is used to generate the license for the machine on which the activation process is carried out on.

**Transfer the license**
If this option is selected, the order number and details may be used to activate RayVentory Scan Engine on a second machine, that has differing hardware (which obviously has a different Hardware ID). This assumes that RayVentory Scan Engine has been deinstalledfrom the machine on which it was previously activated on. The transfer license functionality is logged on our license servers and is periodically checked to ensure that no abuse is made of this functionality.

If the license transfer is part of a regular maintenance and can therefore be prepared and scheduled, it is highly recommended to use the deactivation function first, to disconnect license and packaging machine. This is the standard way for transferring licenses. The option offered here is intended for unscheduled transfers, required if a machine, for whatever reason, cannot be accessed or used operational any longer.

# Manual Activation

On choosing the manual activation, the dialog shown below is displayed.

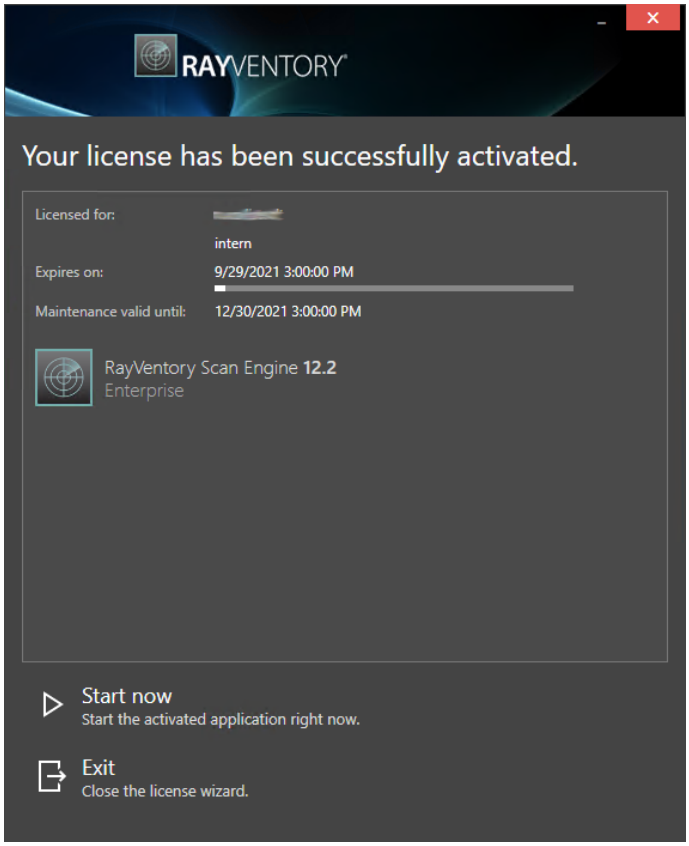



This basically shows the contents of the ticket form that will be opened at Raynet. If there is an internet connection available on the machine, click on the **GO TO URL** button to open the URL shown in the top of the window in the default browser of the system. After a File Order has been opened in the Raynet Support Panel, a license file will be delivered. Information of how to use this file are available here.

If no internet connection is present on the machine on which the activation process is taking place, copy the contents of the dialog onto a machine which has an internet connection and use the URL on that machine. On receiving the ticket, a license file will be generated and sent back. Information on how to use the license file can be found here.

**Tip:**
Please ensure that when copying the information from the **MANUAL ACTIVATION** dialog everything is added as shown above.

Once the license file has been generated the following will be shown:




**Note:**
Depending on the license, more available products may be shown. As an example, see the image above.

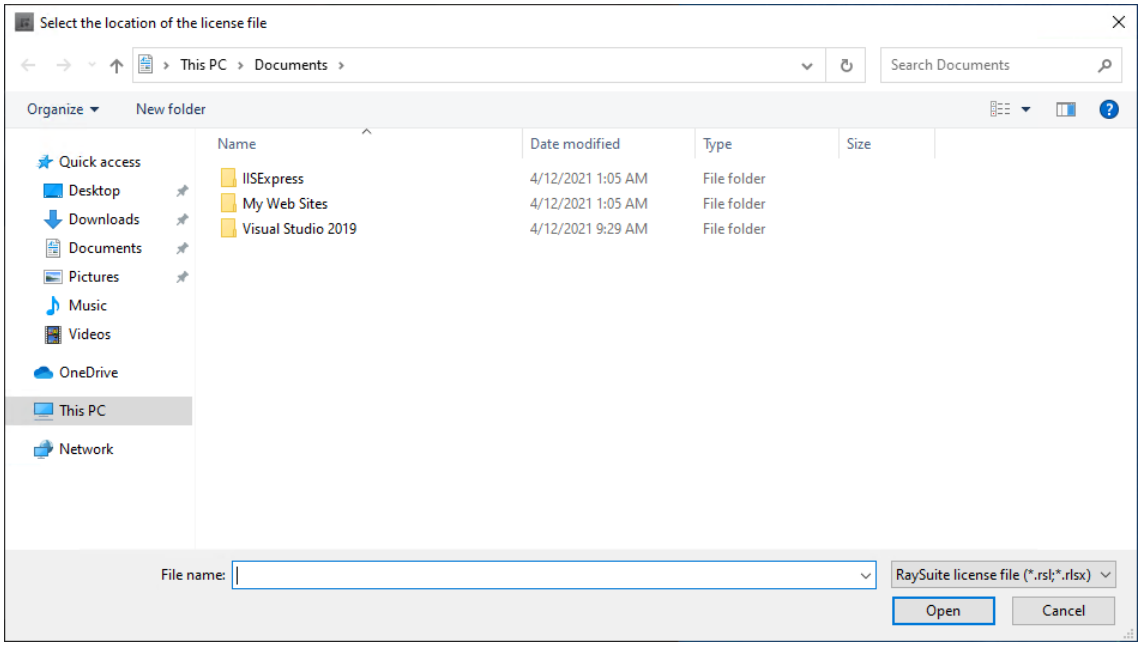
The option of starting RayVentory Scan Engine or just closing the activation wizard are available now.

# Troubleshooting

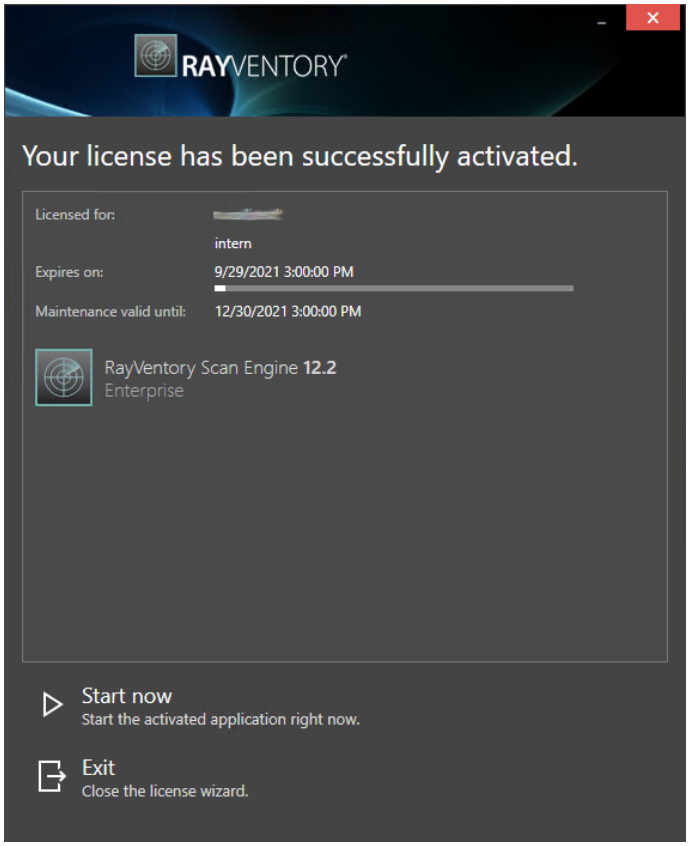
If there are any problems during the activation process, please contact our help desk for receiving assistance in activating RayVentory Scan Engine.

## License File

If a license is already available, or a license file has been received as a result of activating RayVentory Scan Engine via email, then all that is required is to copy the license file into the installation directory of RayVentory Scan Engine (the directory in which the `RayVentory Scan Engine.exe` resides). Clicking on the **I have a license** button on the **License wizard** dialog opens a dialog box which allows to choose the license file. Once chosen, the file will be copied automatically to the RayVentory Scan Engine installation directory. Please ensure that sufficient permissions to allow the creation / copying of a file to the installation directory of RayVentory Scan Engine are available.




Once the license file has been copied to the correct location the following will be shown:




The option of starting RayVentory Scan Engine or just closing the activation wizard are available

now.

# Troubleshooting

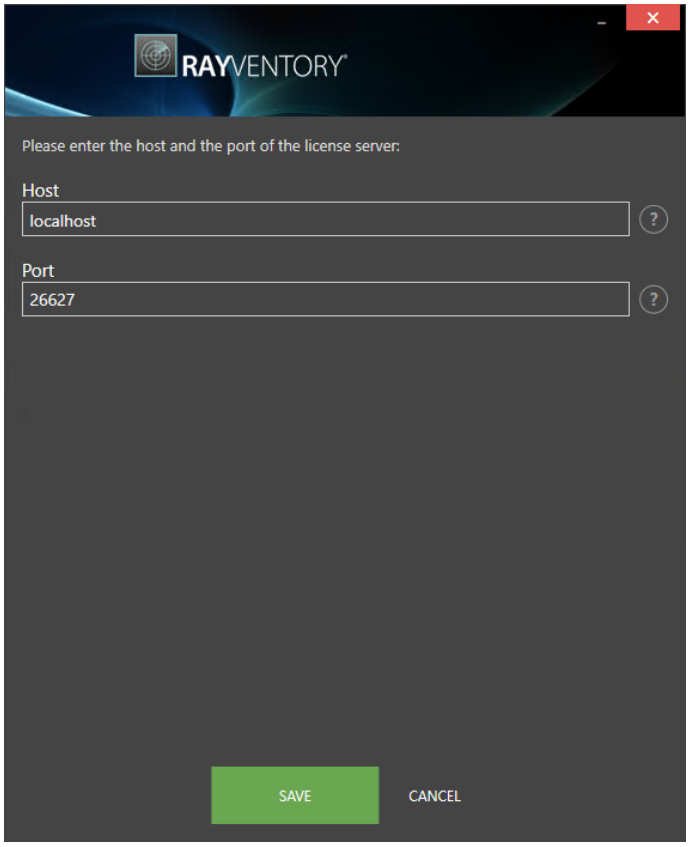
If there are any problems during the activation process, please contact our help desk for receiving assistance in activating RayVentory Scan Engine.

## Floating License Server

RayVentory Scan Engine can be activated using a local floating license server. This requires that the server component is installed (the installation is available separately from the product installer).

Once the server is configured, the following details are required from the server administrator:

- Server name or IP address
- Configured port (by default `26627`)




Enter required values and confirm them by clicking on the **SAVE** button. The server will be contacted once to verify the correctness of the data. If the server is not available at that time, an option will be presented to write the data anyway.

Once the connection details are saved, please restart the product to activate it using the floating license server.

### I Do Not Have a License or Order Number

If neither a license or order number is available, then just simply register with Raynet and get into contact with Raynet. Choosing **I don't have a license or order number** opens the Raynet website in the default browser, allowing potential customers to send a request to Raynet.

### I Want to Take My Activation Back

Deactivating an existing license for RayVentory Scan Engine may be required if the packaging machine used has to be switched. Whenever there is a scheduled migration, e. g. when a virtual machine is transferred in a way that affects the **Hardware ID**, or when a physical machine is no longer used for packaging purposes, deactivating the license is the right thing to do.

## To Deactivate a Licensed RayVentory Scan Engine Installation

1. Launch RayVentory Scan Engine and open the license and edition tab of the **about** area.

2. Click on the **Open the license wizard** button on the lower left hand side of the application window.

3. Use the option **I want to take my activation back...**

4. Enter the **order number** that was originally used to activate RayVentory Scan Engine on the current machine. It was part of the resources and information material delivered during product purchase.

5. If required, adjust the user name already entered into the input field **User name**. The users who activate and deactivate an installation do not necessarily have to be the same.

6. Click on **DEACTIVATE NOW**.

   The license wizard will connect to the Raynet licensing server and send the deactivation information. On success, the number of licenses available for activation, which are bound to the used order number, is incremented by one. With this new free license it is possible to activate any RayVentory Scan Engine installation, on the current machine or any other.

## Troubleshooting

If any problems during this process occur, please contact our help desk for receiving assistance in deactivating RayVentory Scan Engine.

# Getting Started

RayVentory Scan Engine is intended to easily gather inventory data on computers in the network. The many components available for implementation in RayVentory Scan Engine allow to suit a wide range of different requirements.

This chapter describes how RayVentory Scan Engine ideally is operated. For simplicity, one of the simpler solutions is discussed, the one that only uses the components RayVentory Scan Engine and RayVentory Server.

## Software Architecture

The simplest approach to work with RayVentory Scan Engine is to use it in conjunction with the RayVentory server. In this case, RayVentory Scan Engine discovers devices / services and gathers inventory data, which are afterwards uploaded to the Server. In turn, the RayVentory Server processes the discovery and inventory data to provide reports for the hard- and software asset management. The intelligence from these reports can be used to support management decisions.

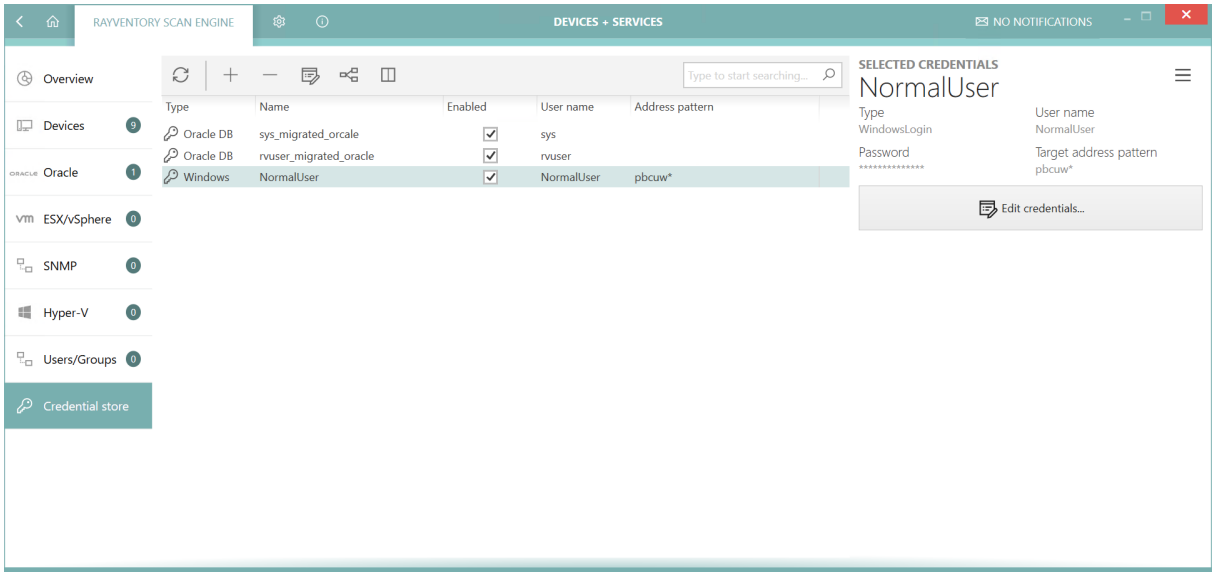
A typical RayVentory implementation consists of a RayVentory Server and one RayVentory Scan Engine instance for each site. All RayVentory Scan Engine instances must be able to directly upload to the RayVentory Server or to indirectly upload to the RayVentory Server by uploading to another RayVentory Scan Engine instance which, in this case, acts as a relay.


Unless a VPN for cross-site connectivity is present, the upload to RayVentory Server would have to use a WAN connection. For an upload via WAN, it is recommended to use RayVentory Server with HTTPS instead of HTTP. For using HTTPS a certificate is needed to be installed on the RayVentory Server host and IIS must be configured to use it. If the certificate has not been created / published by a certificate authority known to the RayVentory Scan Engine hosts then the public part of the certificate must be installed there too.

# Credential Store

The account information that are needed for scanning are stored in the Credential Store.

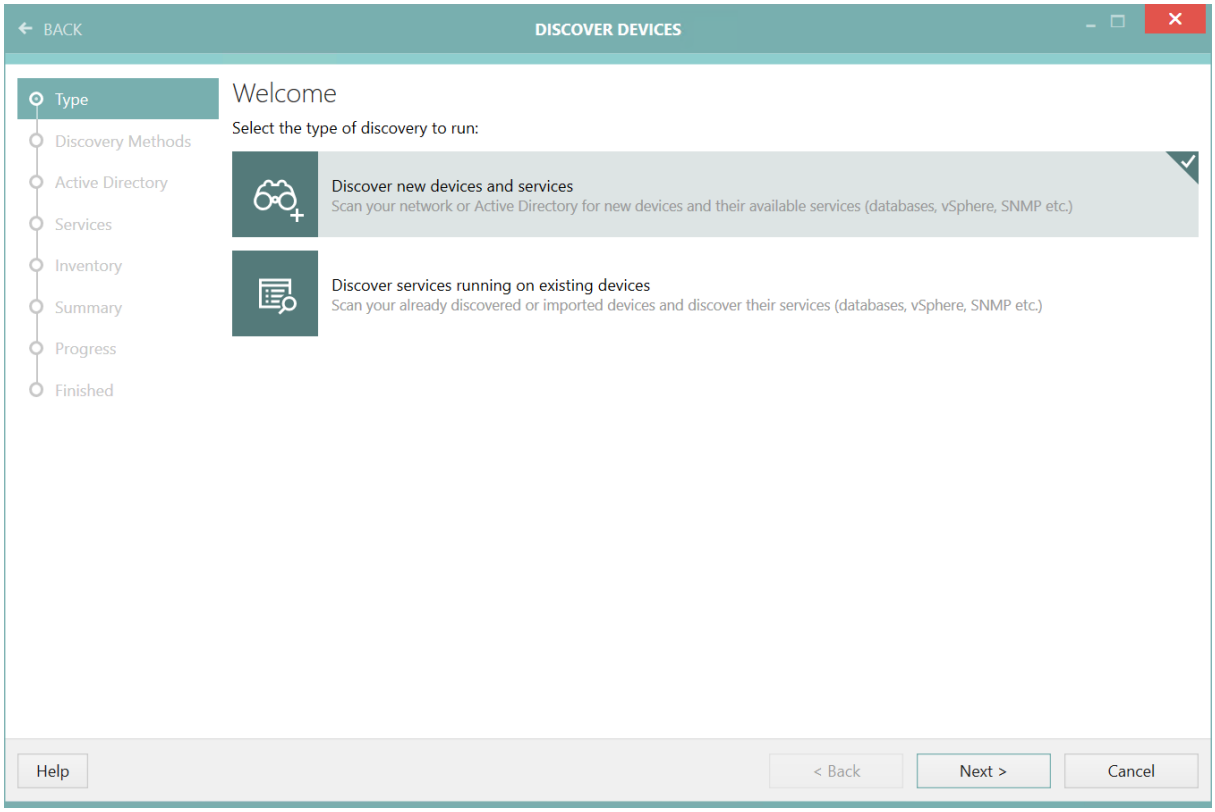
The Credential Store can be opened by clicking on the **Devices + Services** tile on the Dashboard, and then selecting **Credential Store** tab. Clicking on **+** icon will open the credential wizard. Choose the login type of the new account and click on **Accept** to proceed. Fill in the necessary information into the fields that define the credentials.




**Note:**
Windows credentials are needed for OS inventories on Windows based hosts. SSH credentials are needed for OS inventories on hosts running Linux and Unix-like platforms. DB credentials are needed for inventories of Oracle databases and vSphere / ESX credentials are needed for inventories on VMware vSphere / ESX virtual infrastructures.

# Discovering the Network

Before discovering devices and services the inventory scope needs to be defined: For what devices and services is data needed and what devices and services are expected.




Currently, RayVentory Scan Engine gathers data on computers and network devices from Active Directory, network scans, and VMware ESX / vSphere infrastructures. RayVentory Scan Engine gathers device specific data on OS, platform, hardware, and installed software for Windows, a range of Linux distributions, and certain unix-like OSs like HP-UX (11i v1-v3), AIX (7.1, 7.2), OSX / MacOS, and later versions of Solaris / SunOS. It gathers service specific data on Oracle Databases from version 9 to 12 and MS SQL Server. Furthermore, hardware and configuration data is collected for a range of SNMP enabled devices.

**Hint:**
More options like custom inventory of WSMAN enabled devices or the support of virtual infrastructures managed by OpenStack are available for RayVentory Server.

The Discovery Wizard allows to configure and to perform device and service discovery. The discovery offers an overview on what computers, Oracle Databases, ESX / vSphere hosts, and SNMP devices exist in the infrastructure. Later, the inventory collects and offers details.

Successive discovery runs are cumulative – later results are merged into existing discovery results and may update hostnames and IP addresses by DNS query and target OS type by port probing or Active Directory attributes.

More information:

- Discovery Wizard

# Discovering Oracle Databases

Discovering Oracle Databases during a discovery run can be achieved by two different methods. The default method requires knowledge of the ports that TNSListeners is listening to, for probing hosts for indication of a potential Oracle Database and credentials of type Windows resp. SSH, for authentication as privileged users to the Databases' host systems. The credentials are used for reading the configuration by remote execution, resp. WMI query. Such users need at least permissions to read `/etc/oratab` on a Linux / unix-like host systems and to query `Win32_Process` by WMI on a Windows host system.

The advanced Oracle Database discovery option, named **Use Remote-Execution based Oracle discovery** in the wizard does not need knowledge on the TNSListeners' ports for discovery. This method also needs credentials of a privileged user to perform remote execution. The users for Linux / unix-like host systems should have permissions for reading environment variables, files in the Databases' home directories like `listener.ora`, `tnsnames.ora`, and `sqlnet.ora`, the files `/var/opt/oracle/oratab`, `/etc/oratab`, `.oratab` and to execute `lsnrctl`. The users for Windows like host systems should have permissions for reading environment variables, files in the Databases' home directories like `listener.ora`, `tnsnames.ora`, and `sqlnet.ora`, to execute `lsnrctl` and to query the registry HKLM.

**Be aware:**
In order to use the Oracle discovery on Windows the `java.exe` on the target machine needs to be located in a subfolder of the paths that are mentioned in the `PATH` variable!

**Tip:**
The advanced Oracle Database discovery option can also be run on specific devices from within the Devices list, by the context menu option **Oracle Discovery**.

The discovery is able to discover database connections on its own, but this feature is still very limited and requires a user with sufficient privileges to read certain configuration files on a unix-like host or to query running processes by WMI on a Windows host. Adding connections manually is also an option that is worth considering.

More information:

- Settings for Oracle Zero-Touch Scan
- Overview of Oracle Instances
- Discovery Wizard

# Manually Adding Devices

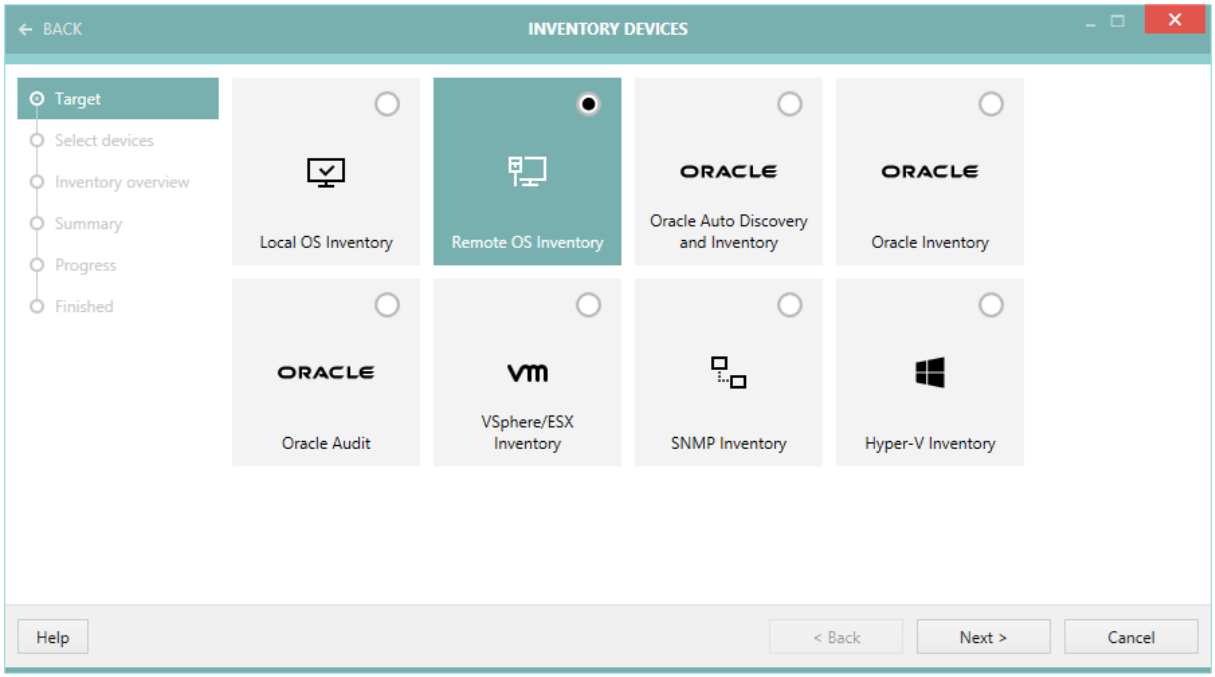
It is possible to manually add hosts and services using the related screens (devices and services like **vSphere**, **Oracle databases**, and **SNMP**).

To access the overview of the connections, open the **Devices + Services** screen directly from the dashboard. The overview is common for all base types and each has a dedicated tab on the left side that contains a dedicated data grid.

Press the **+** icon to add a host or click **Edit** to change the properties of an existing host. This opens the properties dialog which can be used to set the type of operating system, the hostname, or network address and optionally choose the credentials that are tried first when authenticating to this host for running the inventory. Finally, the dialog allows for the configuration of the device-specific capabilities which will later be used to determine how to access it when doing the inventory scan.

# Running Inventory

The discovery wizard offers an option to directly run an inventory on newly found devices and services. Later, an inventory can be run on all or specific targets by using the inventory wizard, which can be started from the home screen and from within the context menu and side-bar in the device and service specific tabs of the **Devices + Services** screen.




By default, if there are multiple options, RayVentory Scan Engine is configured to find a suitable inventory method on its own. Currently, there are multiple options for Device and Oracle inventories. RayVentory Scan Engine shows its reasons for considering or discarding certain inventory methods in the inventory wizards. See Application of Inventory Options for

requirements, prerequisites, and options.

RayVentory Scan Engine will save and show error messages for each inventory target and the failed inventory methods to assist troubleshooting. Additionally, RayVentory Scan Engine logs its activities and errors regarding discovery and inventory.

More information

- Inventory Wizard
- Inventory Methods Overview
- Configuration of Inventory

# Uploading and Reporting

The intention of RayVentory Scan Engine is to gather data. It offers a basic reporting on inventory health and status to assist in managing an inventory campaign. The report **Summary of inventory health** integrates the inventory wizard to support workflows regarding inventory updates and troubleshooting. The tiles on the report, which show number that represent certain sets of devices, are clickable. When clicked then the inventory wizard will be opened with the respective devices selected.

The devices and service lists offer views on the gathered inventory data including summaries of certain devices or service specific facts.
More in-depth reports that will show discovery and inventory data are available on the RayVentory Server. There are facts shown, that RayVentory Scan Engine will not show in its summaries, for example MS SQL Server instances or data on Oracle Database options.

The data gathered by RayVentory Scan Engine must be uploaded to RayVentory Server in order to fill the discovery and inventory database for reporting.
In the RayVentory Scan Engine settings, the upload endpoints for inventory and (optionally) discovery must be set. Uploading discovery data is needed to see the gap between devices known without inventory data and devices known with inventory data in the reports that the RayVentory Server offers. See **SETTINGS** > **HTTP Services** > **UPLOAD LOCATION URL to upload inventory files** and **URL to upload legacy discovery data**.

Usually, the URL is of the following form: `http://yourRVServerAddress/ManageSoftRL/inventories` or `http://yourRVServerAddress/ManageSoftRL/discovery`.

More information:

- Uploading Results to Parent Servers
- Receiving Uploads from Remote Scans
- Settings for Server and Upload Location

# Automation

For a continuous inventory, it is recommended to frequently run discovery, inventory, and upload to keep the inventory data up to date.

If a network discovery is needed instead of or in addition to the Active Directory import then such a discovery should be run at different times during the day, to cover devices which are rarely active or not active during the whole day like workstations and desktop computers. For the same reason, device inventories should be run at different times too.

All inventory operations can be limited to a certain set of targets either based on an explicit list of targets or implicitly, by a regular expression which is matched against the target hostname / address / URL. The Oracle inventory operation also allows to set regular expressions for matching SIDs / service names and ports. If no regular expression is set for a **Target definition by filter,** then all available targets will be addressed by the inventory operation.
Uploads may be run on a weekly or monthly basis.

RayVentory Scan Engine's built-in task scheduler enables automation of these operations. Each task may consist of an operation like upload, devices import, discovery, inventory, oracle discovery, or a sequence of these operations.

More information:

- PowerShell Automation
- Scheduled Operations
- Command-line Tools

# Maintenance

Once RayVentory Scan Engine and RayVentory Server are configured to frequently run all necessary operations, the system health can be monitored by the inventory health related reports. Then RayVentory Scan Engine inventory health summary, the saved target status / error messages, the logs of RayVentory Scan Engine, and RayVentory Server can be used for troubleshooting.

From time to time updates and fixes / patches for RayVentory Scan Engine and RayVentory Server are released.

A knowledge base for self-service regarding troubleshooting is also available.

# Inventory Agent

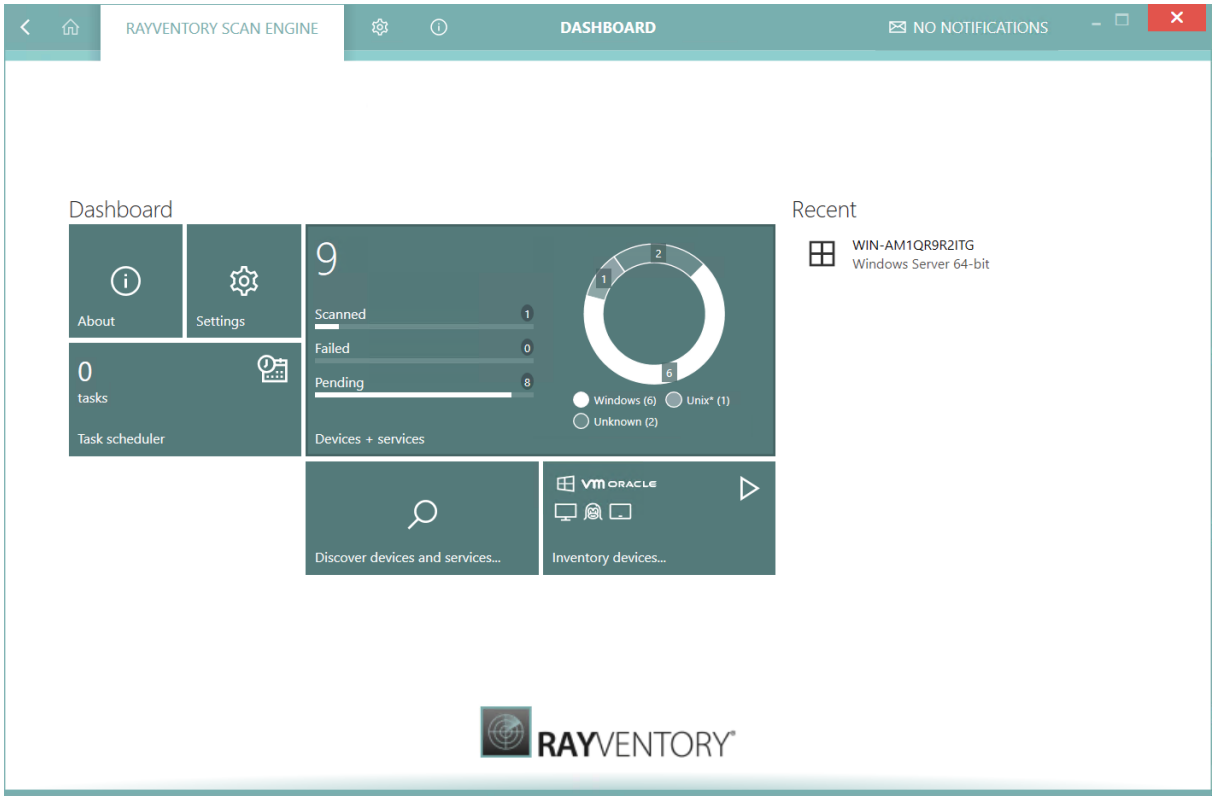
RayVentory Scan Engine includes a bundle for RayVentory Inventory Agent. It is designed to continuously deliver hard- and software inventory and usage data from computer systems running Windows, Linux or Unix.

The agent must be installed or deployed separately, as it runs remotely from the RayVentory Scan Engine.

For more information about how to install and set-up the Inventory Agent, refer to the dedicated chapter: Inventory Agent.

# Home Screen

The home screen, also called **Dashboard**, grants access to the different functions and screens which are part of RayVentory Scan Engine. The tiles on the **Dashboard** can be used to start certain operations or to navigate to other screens where more operations and information will be available. Some of the tiles on the **Dashboard** also show information. An example for this is the tile showing the number of hosts which are currently known to RayVentory Scan Engine. Furthermore, the recent inventory results can be shown in the **Recent** panel.




There is a handful of buttons that lead to various screens containing data and settings of RayVentory Scan Engine.

## About

Opens the **About** screen with license and troubleshooting data.

## Settings

Opens the **Settings** screen with various options for inventory, execution, scanning, uploads, etc.

## Task scheduler

Opens the view which shows scheduled tasks and operations

## Devices + Services

The main view of RayVentory Scan Engine, which provides both, insights and reports on the already discovered and scanned assets, as well as, the ability to perform various operations like the import of devices, the execution of new discoveries and inventories, database maintenance, etc.

### Discover devices and services...

Opens the **Discovery** wizard, which allows for the import of new devices and services from Active Directory or from a network scan.

### Inventory devices

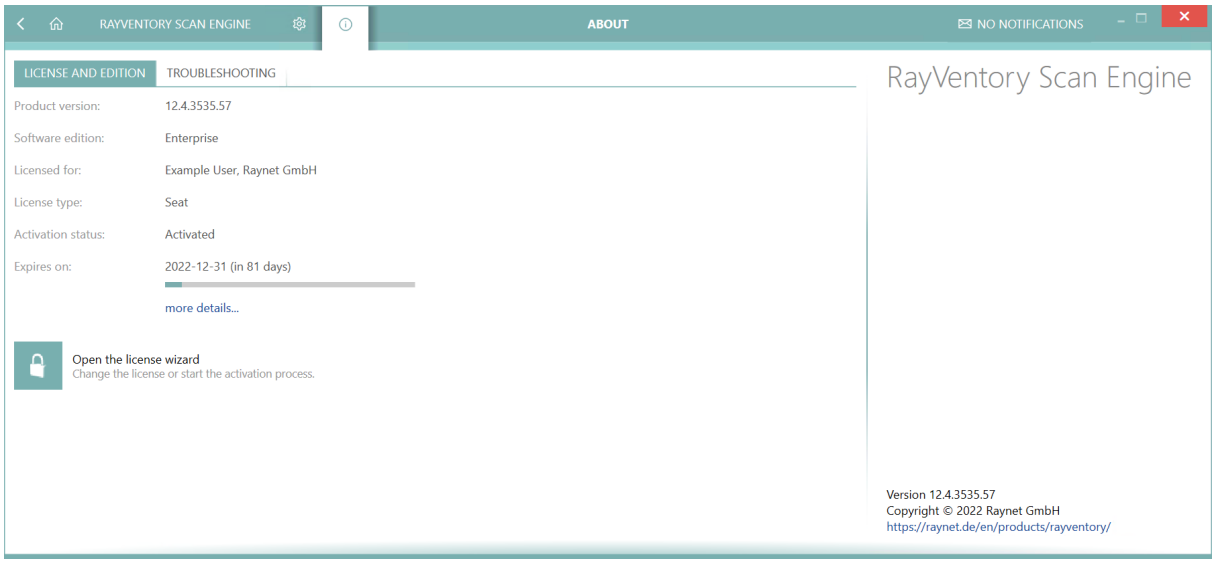
Opens the **Inventory** wizard, which allows to scan the already imported computers and services for a new software and hardware asset.

### Recent

The view of recently inventoried devices and services. Clicking these items will jump to the corresponding place where more details can be seen.

# About Screen

The **About** screen shows the version of the RayVentory Scan Engine installation. It also allows for access to the License Manager and the directory with the log files.




The **About** view shows basic information about the license, its expiration date, and the hardware ID. It is possible to view the details of the current Raynet license and / or apply a new license by pressing the **Open the license wizard** button.

## Troubleshooting

The **TROUBHLESHOOTING** tab contains various performance, diagnostics, and logging information.

### Logs and diagnostic information

RayVentory Scan Engine writes its log files (by default) into a text file, which path is displayed in the UI. You can press the button **Open logs folder** to jump to this location.

## Versioning

This section shows various sub-components belonging to RayVentory Scan Engine and their current version. The table can be easily copied to the clipboard.

**Note:**
Both information about the product version and the current log file may be required when opening a Support ticket for RayVentory Scan Engine.

# Devices + Services

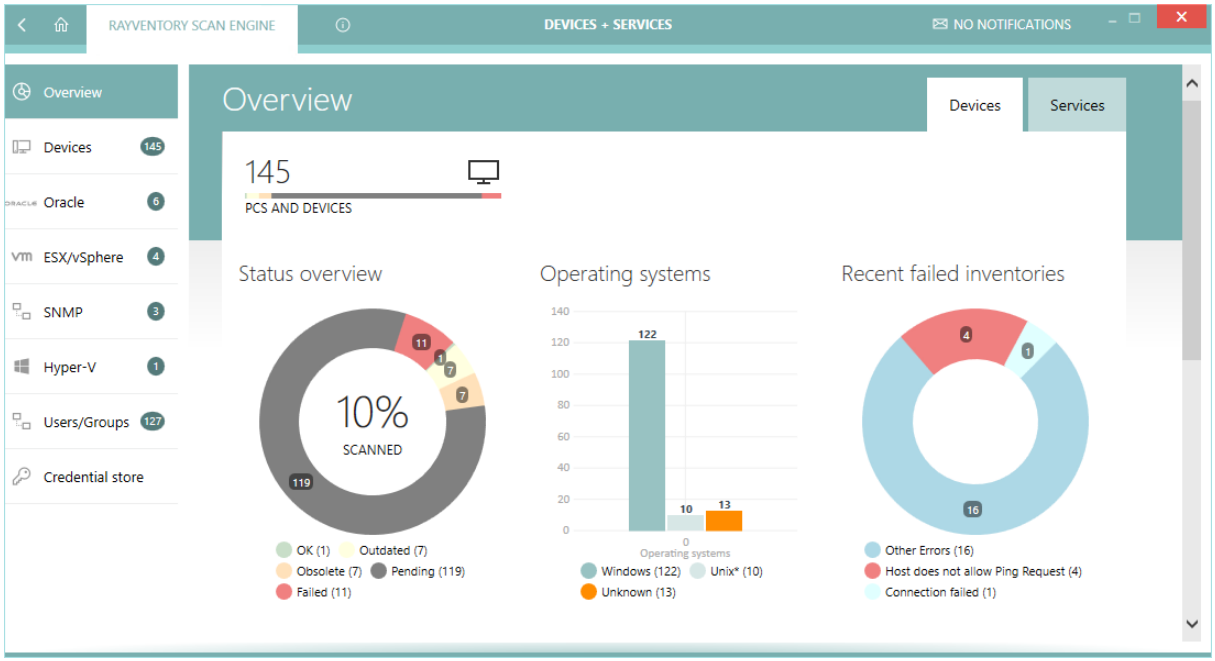
This is the core view of RayVentory Scan Engine. In this view the following options are available:

- Adding, importing, and managing connections (physical devices, virtual devices, databases, and services)
- Triggering discovery and inventory on your asset
- Identifying old and failed scans
- Managing credentials
- Browsing inventory results of scanned assets

## Overview

The **Overview** section is the dashboard which aggregates data from various sources and shows them in a simplistic style for an easier identification of:

- Successful, pending, and failed inventories
- Share of operating systems
- Top reasons of failed inventories
- A summary of inventory health




This view has two tabs on the top, which toggle between **Devices** and **Services** data sources. Operating systems (hardware and software scans) are contained within the first tab while services (Oracle, SNMP, vSphere / ESX) are part of the second tab.

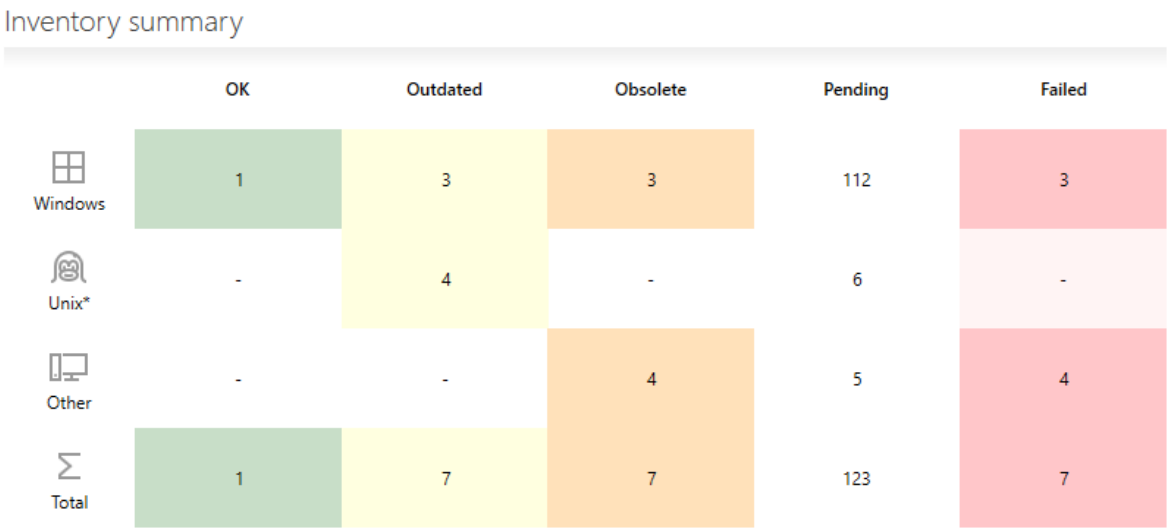
## Working with Inventory Health Overview

Each section (**Devices** and **Services** respectively) has its own dedicated table that identifies the "health" of recent inventories. An inventory file is considered "healthy" if (both must be true):

- The inventory finished successfully and
- The inventory was executed not more than 30 days ago

Inventory jobs that failed are presented in the **Failed** columns. Devices that have not been scanned yet are grouped under the **Pending** column. The three remaining columns (**0-30 days, 30-90 days, 90+ days**) show inventory jobs which succeeded sorted by their time frame.

As a rule of thumb, inventory results older than 90 days should be treated as already outdated and requiring attention.




This view is interactive. It is possible to click on the cells with numbers inside to open the **Inventory Wizard** and automatically preselect the devices or services which match the corresponding criteria. For example (see the figures in the above picture):

- Pressing "5" in the row **Other** and the column **Pending** opens the Inventory Wizard with 5 unscanned devices with an unknown system preselected.
- Pressing "3" in the row **Windows** and the column **Failed** opens the Inventory Wizard with the Windows devices that failed preselected. The one device that succeeded will not be selected.

**Note:**
The rules for colouring and assessing the status are customizable. Refer to the chapter Health Assessment for more information.

## Executing Common Tasks

The lower part of the overview contains shortcuts to some common operations:

### Devices Tab

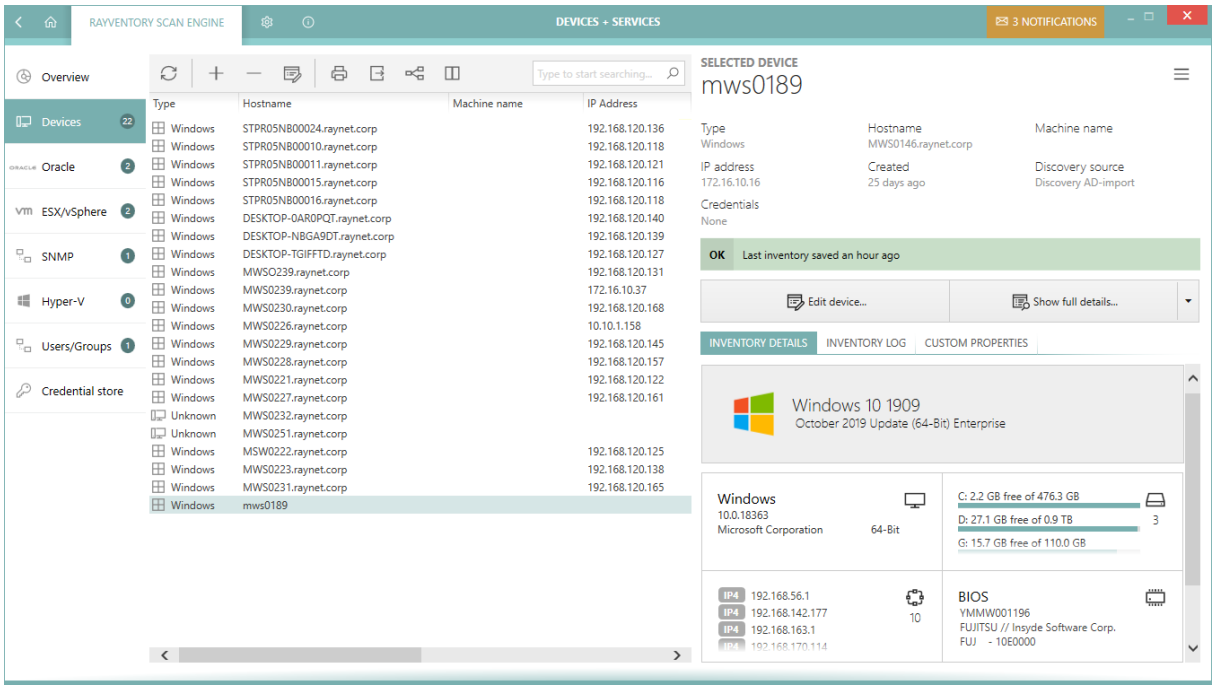
- **Scan your infrastructure for new or changed devices...** - opens the **Discovery** wizard.
- **Import devices connections from CSV file...** - opens the **Import Devices** wizard that can be used to import data from `.csv` file.
- **Export device connections to XML file...** - opens a file open dialog prompting where to save the exported `.xml` file.

### Services Tab

- **Export Oracle connections to XML file...** - opens a file open dialog prompting where to save the exported `.xml` file with all Oracle connections.
- **Export vSphere connections to XML file...** - opens a file open dialog prompting where to save the exported `.xml` file with all vSphere connections.
- **Export SNMP connections to XML file...** - opens a file open dialog prompting where to save the exported `.xml` file with all SNMP connections.

# Devices

This view aggregates discovery and inventory data of your devices.



The view is divided into three parts:

- **Toolbar** - showing buttons to perform quick operations like adding, editing, and removing entries.
- **Main grid** - showing the list of all saved devices (physical and virtual).
- **Sidebar** - showing the details of the current selection, including inventory data if available.

## Columns

The grid supports a predefined set of columns, only some of which are visible by default. It is possible to select more columns by pressing the **Column chooser** button in the grid toolbar. The following columns are available:

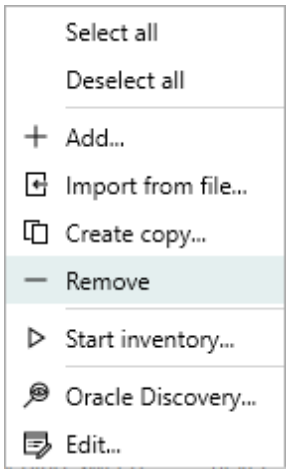
- **Type** - The device family (Windows / Unix / Unknown). This determines the inventory methods available for each device.
- **Hostname** - Either the DNS hostname of a device or it is empty if no name has been configured (in that case the IP address will be non-empty).
- **IP Address** - Either the IP address of a device or it is empty if the IP address is resolved automatically from the DNS name.
- **Status** - The inventory status. There are three possible values:
  - n/a - The device has not been inventoried yet
  - OK - The device has been inventoried and returned some results
  - In any other case, the column **Status** contains a short description of the most recent issue
- **Discovery source** - A text value determining the import source (for example *Discovery AD-import* or *Discovery ping-sweep*). Manually added devices have an empty value in this column.
- **Last inventory attempt** - The last time the inventory has been started (whether successful or not).
- **Credentials** - The logical names of credentials used when scanning this device.
- **Capabilities** - The short version of enumeration of allowed inventory capabilities for this device. The string consists of two-letter tokens, with the following meaning:
  - *ZT* = Zero-Touch
  - *RE* = Remote-Execution
  - *FS* = Access to File System
  - *SMB* = Upload to SMB Shares
  - *WMI* = Windows Management Instrumentation (WMI) Queries
  - *WSM* = Windows Service Manager
- **Last inventory attempt (time)** - The last time the inventory has been started (whether successful or not) (contains full date and time).
- **Last inventory** - The last time the inventory has been successfully started (only date part).
- **Last inventory (time)** - The last time the inventory has been successfully started (contains full date and time).
- **Last successful inventory method** - The name of the last inventory method that worked for this device. This method will be preferred for future scans.
- **Last failed inventory methods** - The names of the inventory methods that failed the last

time the device was scanned.

- **Last failed inventory method (details)** - Details about the methods that failed when the device was scanned for the last time.
- **Show inventory** - Shows the details of the inventory (only available if an inventory has been already performed).
- **Created** - The date when the device was created or imported.

## Context Menu

Pressing the Right-Mouse-Button after highlighting an item opens a context menu for it.




- **Select all** - Selects all visible entries in the grid.
- **Deselect all** - Deselects all visible entries in the grid.
- **Add...** - Opens the New Device Dialog.
- **Import from file...** - Opens the Import Wizard.
- **Create copy...** - Opens the New Device Dialog where the default values are automatically set to the values from the current selection.
- **Remove** - Removes the currently selected devices (see Removing Devices).
- **Start inventory...** - Opens the Inventory Wizard for the currently selected devices.
- **Oracle Discovery...** - Scans for the presence of Oracle instances on the currently selected devices.
- **Edit...** - Opens the Edit Device Dialog.

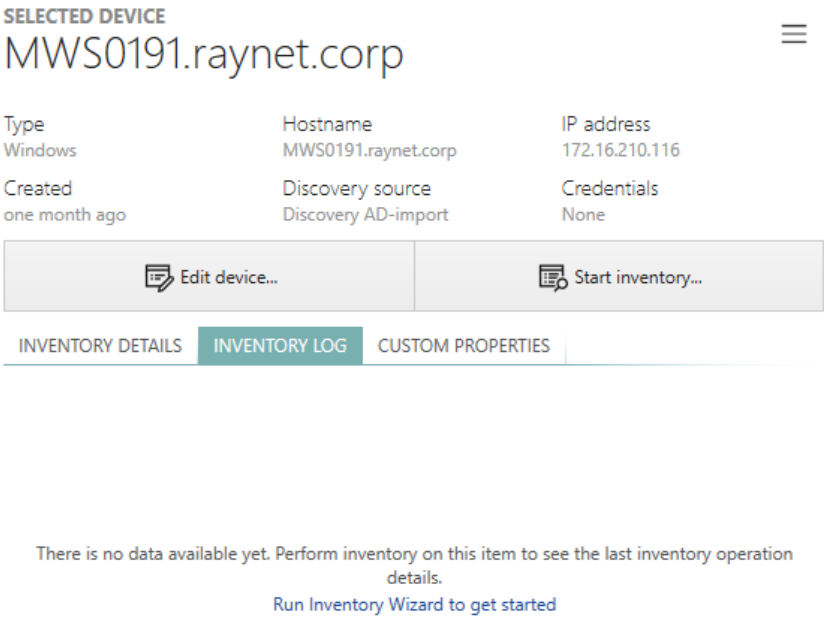
**Note:**
Some options may be conditionally hidden or disabled (for example the menu item **Edit...** is disabled if more than one device is selected).

## Recent Scan Details

The details of the last scan are represented in two tabs **INVENTORY DETAILS** and **INVENTORY LOG**.

### If the selected devices have not been scanned yet...

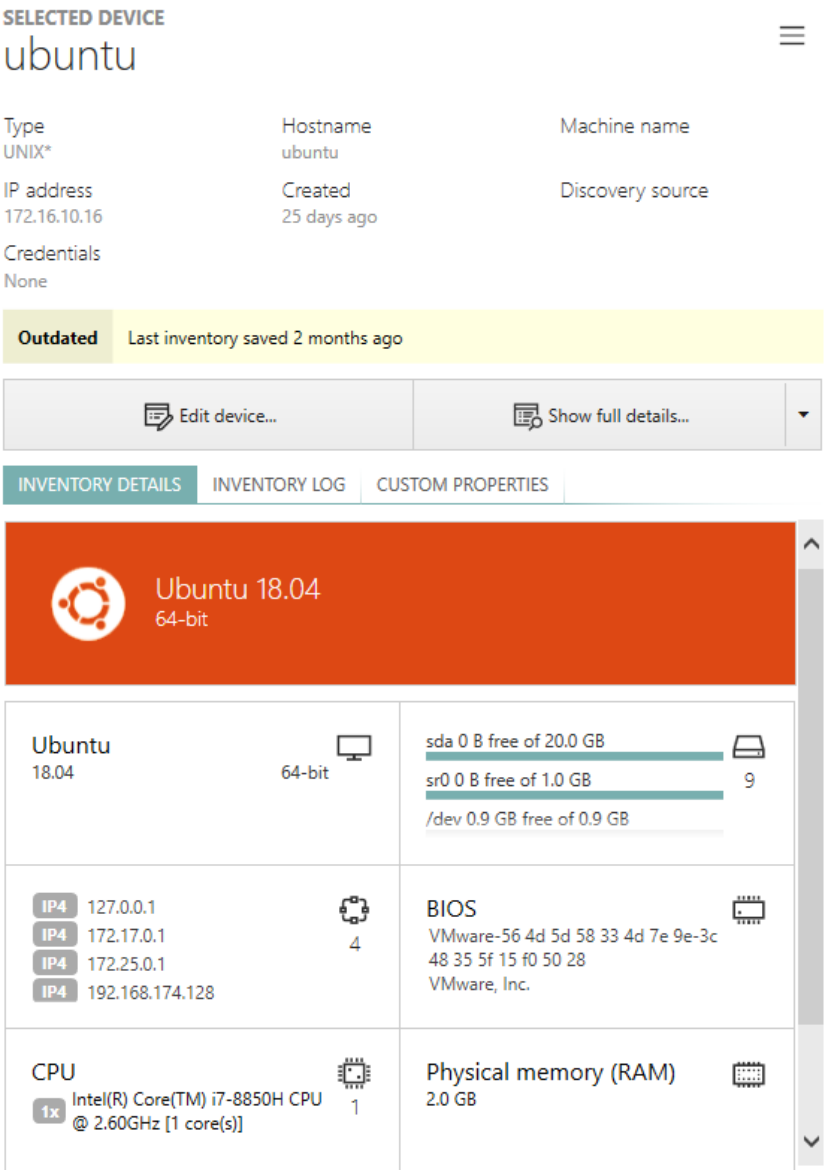
For devices that have not been scanned yet, both tabs show initially no data.




Press **Run Inventory Wizard to get started** to open the Inventory Wizard, which will guide you through the process of collecting the data.

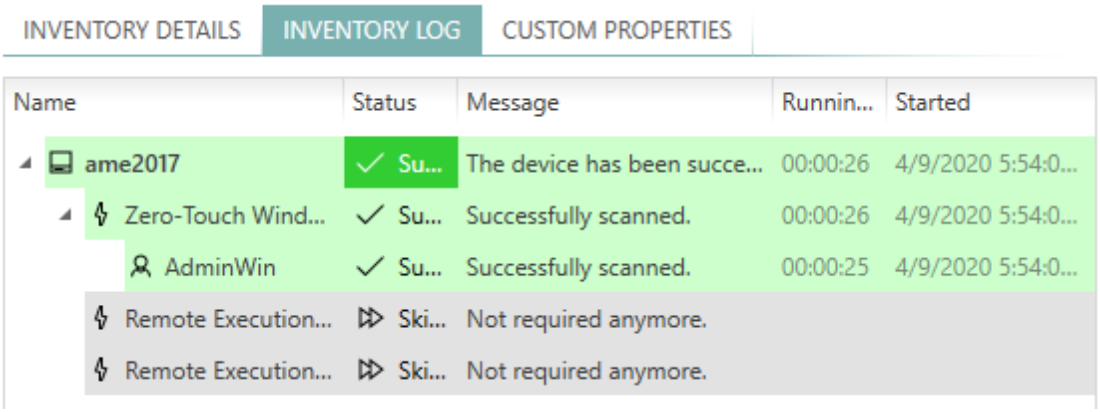
### If the selected devices have already scanned...

For devices that have been scanned, the **INVENTORY DETAILS** show the details of the last successful scan.




The exact content of the tab varies, depending on the target operating system. For many popular systems, RayVentory Scan Engine shows a logo and uses a dedicated key colour for an easy identification. Some basic details, including the list of IP addresses, CPU, BIOS, RAM are also displayed. The tiles will be stacked if the width of the sidebar is too small to fit them in two columns.

**INVENTORY LOG** tab is a summary of recent successful and failed scans. Typically, once the inventory results are available, the view would have the following content:

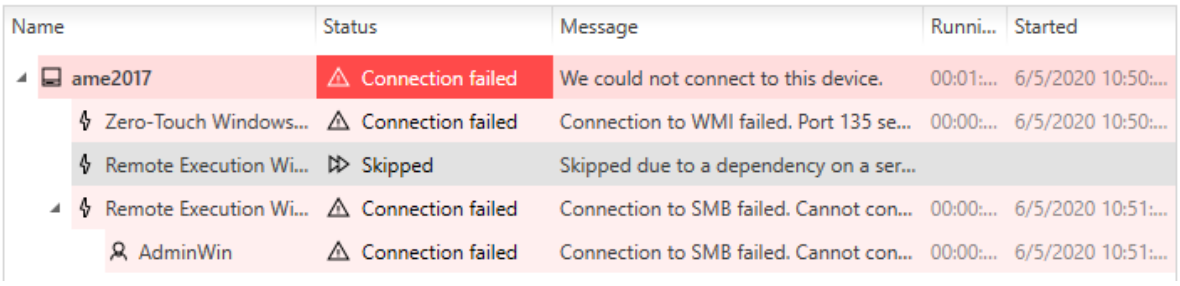



These details are specific to a concrete device from the current selection. The tree view uses several levels to distinguish between the following actions, that were executed during the inventory scan:

- Targeting a device (root entry),
- Used inventory method (for example Zero-Touch Windows),
- (Optionally) Sub-methods and variants of execution paths (for example using particular version of JRE),
- Used credentials (for example AdminWin). This is the display name from credential store, or a default string "Current user" if the default credentials were used.

For every entry, its date, duration time and status are shown. The items may have different statuses, and the logic that RayVentory Scan Engine applies when interpreting them varies. For example, if the first credentials from the list fail, the program tries the next ones available and so on until it finds matching ones. This is going to be represented as a series of sub-nodes from a particular method, with statuses indicating whether the credentials failed or succeeded. A similar logic is applied to the methods selection - RayVentory Scan Engine executes them from top to the bottom and stops on the first one that succeeded. Other methods are marked as **Not required anymore** if any of previous method returned the expected results.

If a method failed due to a non-critical error (for example closed port, missing credentials or access denied) the method result is also shown, which simplifies troubleshooting. For example, the following inventory log:




Can be interpreted as it follows:

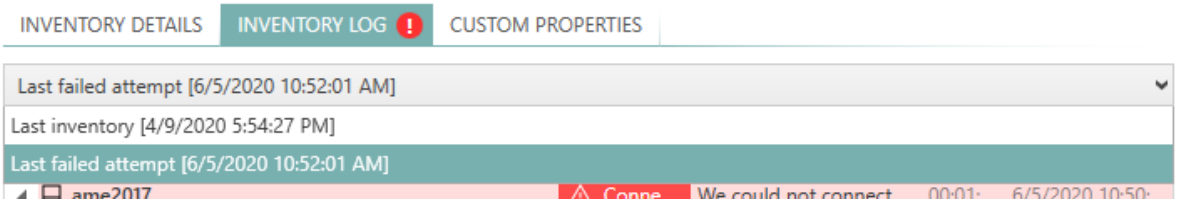
- RayVentory Scan Engine executed an inventory scan of machine **RVP10** and it failed using all available methods.
- The following methods were used: Zero-Touch Windows, Remote Execution Windows [WMI/SMB], Remote Execution Windows [ServiceManager/SMB].
- The Zero-Touch scan on Windows failed due to port 135 being closed. Since Zero-Touch on Windows relies on WMI, not being able to communicate with the machine on that port lead to an error for that particular method.
- The Remote-Execution Windows [WMI/SMB] was not executed at all. RayVentory Scan Engine was aware that a similar method using WMI failed, and it implied that since WMI was not possible in the previous step, there was no necessity to perform it again. Thus, the method never executed and reported the status **Skipped**, and the message indicates that the reason was indeed the WMI check which had failed previously.
- The last method executed, and it start probing the credentials store. It picked first the best-match **Windows**. After running for a few seconds, it also failed with the message that `ADMIN$` share could not be accessed, which indicates unavailability of the machine. Since no more credentials were available, it stopped and returned the error back to the caller.
- Overall, since no method returned a positive result, RayVentory Scan Engine summarized the results and picked the status **Connection failed** as the best one that describes the issue. IT Pro would be now able to address the issue by accessing the machine (ensuring it is up and running) and verifying whether the port 135 is open. After that, the scan should be repeated.

## Working with the last positive and failed result

If the last result of the inventory scan was positive, only its details are shown in the sidebar. However, consider the following scenario:

- The machine has been scanned successfully at least once,
- But due to configuration changes it is not available anymore, and all new inventory results report various connection issues.

In this case, RayVentory Scan Engine shows both the last failed and positive result of scan. You can determine if this is the case by observing the content of the **INVETORY LOG** tab:

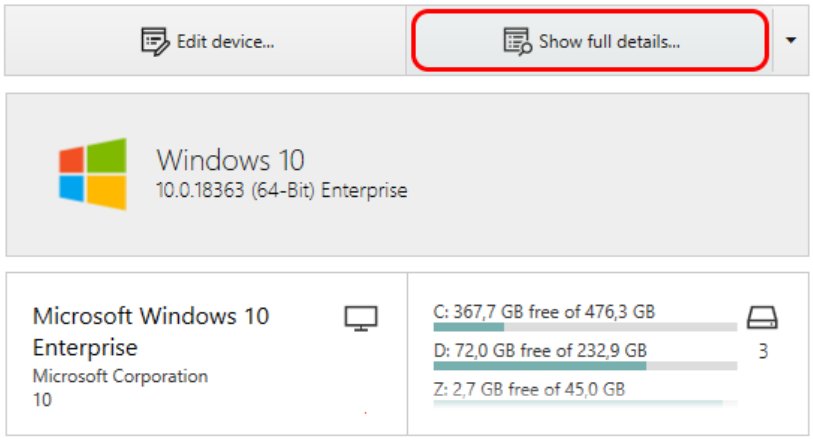



The exclamation mark drawn next to the tab indicates, that the last scan failed. You will then be able to use the drop-down menu below to select which results are to be analyzed. By default, the most recent log is shown (this provides on the other hand a quick overview of the relevant data, because usually it is the last failed scan that is most important when troubleshooting connection or permission issues).

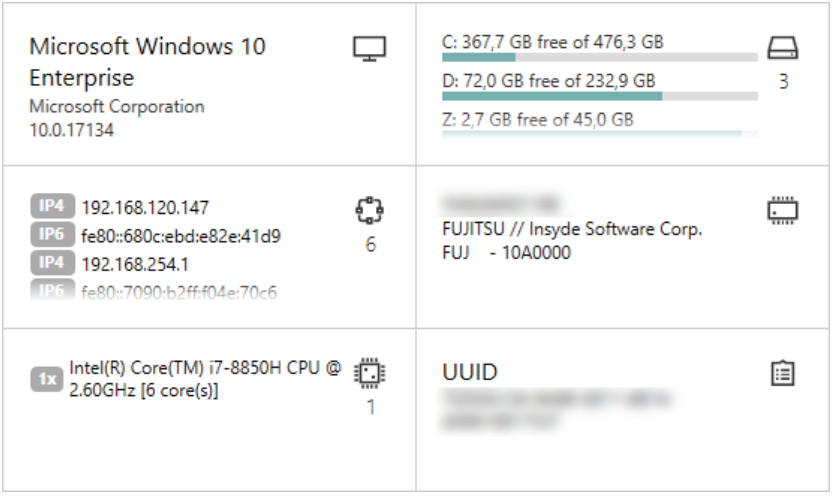
There is still a way to see the previous success result, which will give some additional information, including which method succeeded for the last time, when the last time was and which credentials were used at that time. This provided a valuable information, indicating whether the change in machine availability has been caused by changes in the configuration, password store, firewall and so.

## Viewing Inventory Details

Once a device has been successfully scanned for its software and hardware, a button will be shown on the sidebar allowing the user to show the content of a specified machine:




The button can be pressed to open a detailed overview of the machine software and hardware. For convenience, the summary of the machine asset is also shown directly in the sidebar:




For popular operating systems and distributions, RayVentory Scan Engine is able to show additional details and product logo, for example:

The details inventory overview contains several more tabs, which are contextually sensitive and display various information depending on the connection type.

Some most-used tabs are:

- SOFTWARE
- HARDWARE
- SERVICES
- SERVER FEATURES
- DOCKER
- RAW DATA

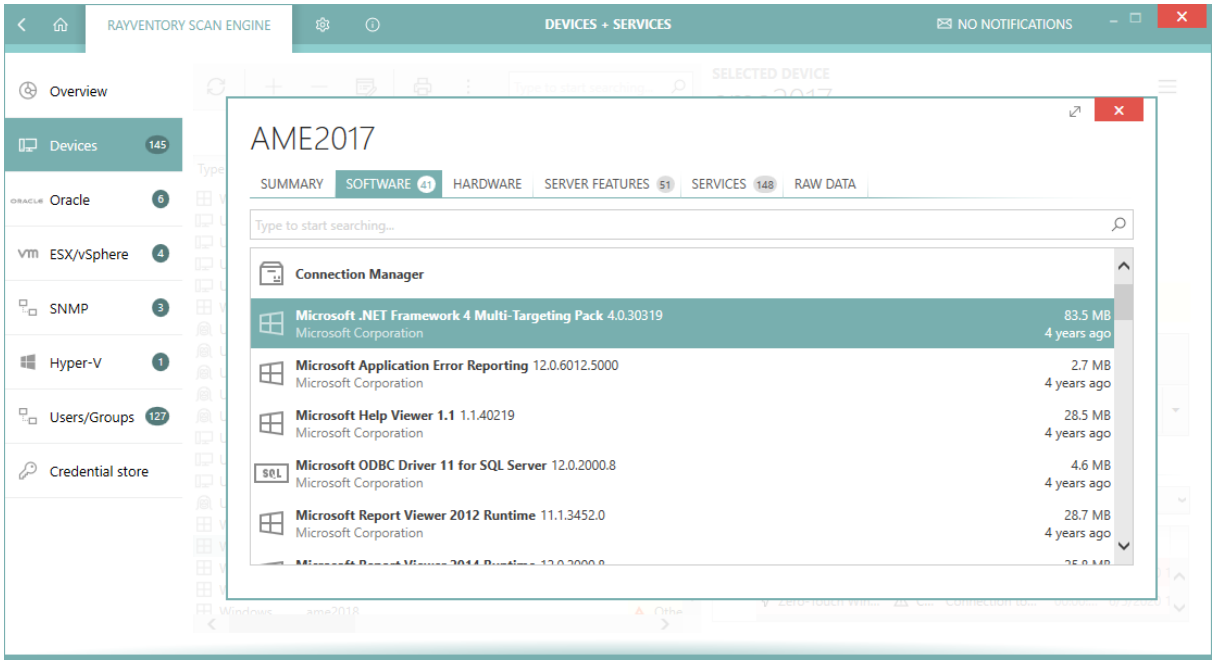
# Undocking the Inventory View

It is possible to "undock" the view to a separate window which can be freely moved and stacked with other windows. In order to do that, press the little **undock** button next to the **CLOSE** button of the inventory overlay.

**Note:**
After undocking, the window cannot be docked in the main window anymore.

## Software

The **SOFTWARE** tab contains an aggregated information about the software and packages, which were detected during the last inventory scan.



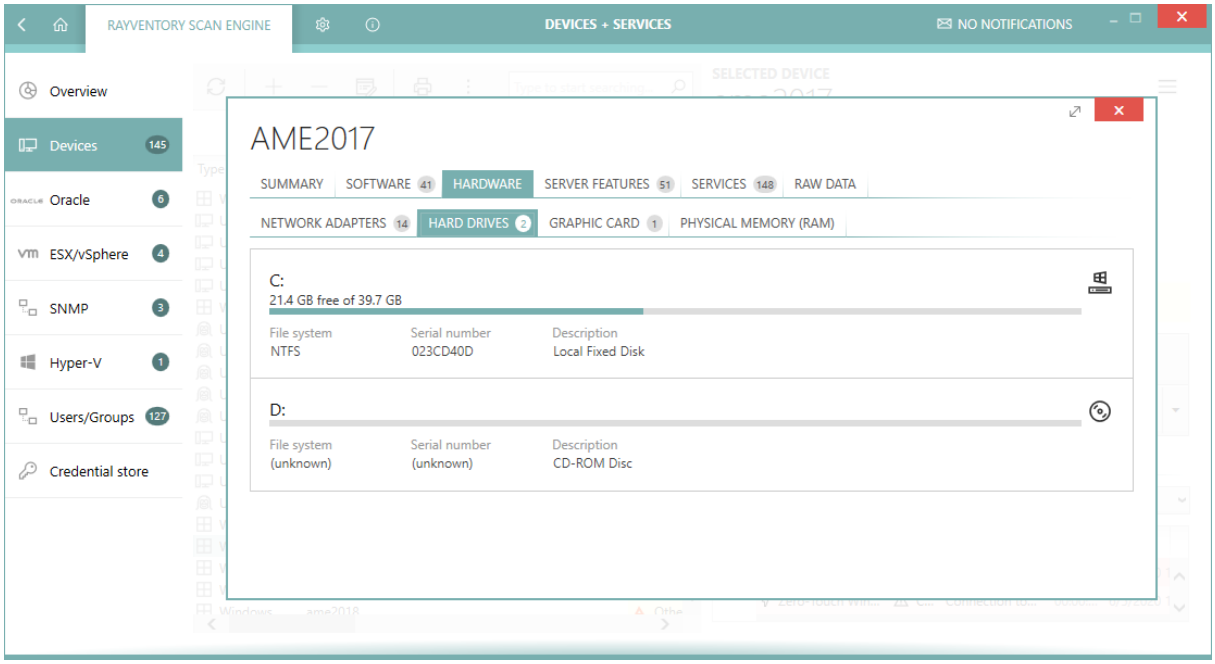

Depending on the target system, this view may differ and show various information, applicable to its source.

- Windows systems: This is a merged view of the ARP information (Add/Remove Programs registry keys) and the MSI evidence (Windows Installer Database). The duplicates are eliminated. Some products (for example SQL Server, Java, Visual Studio etc.) may already have dedicated icons. Estimated size, the actual version and installation date are shown, based on Windows evidence (usually ARP registry entries).
- Non-Windows systems: Usually only the data from package managers is shown, depending on the actual distribution being scanned. Due to differences between data reported back by various systems, the information here may vary. For most of supported package managers, at least name, publisher, size and versions are displayed.

You can use the built-in search box to filter the list and find the application of interest.

## Hardware

This tab shows the list of recognized hardware features of the target device.

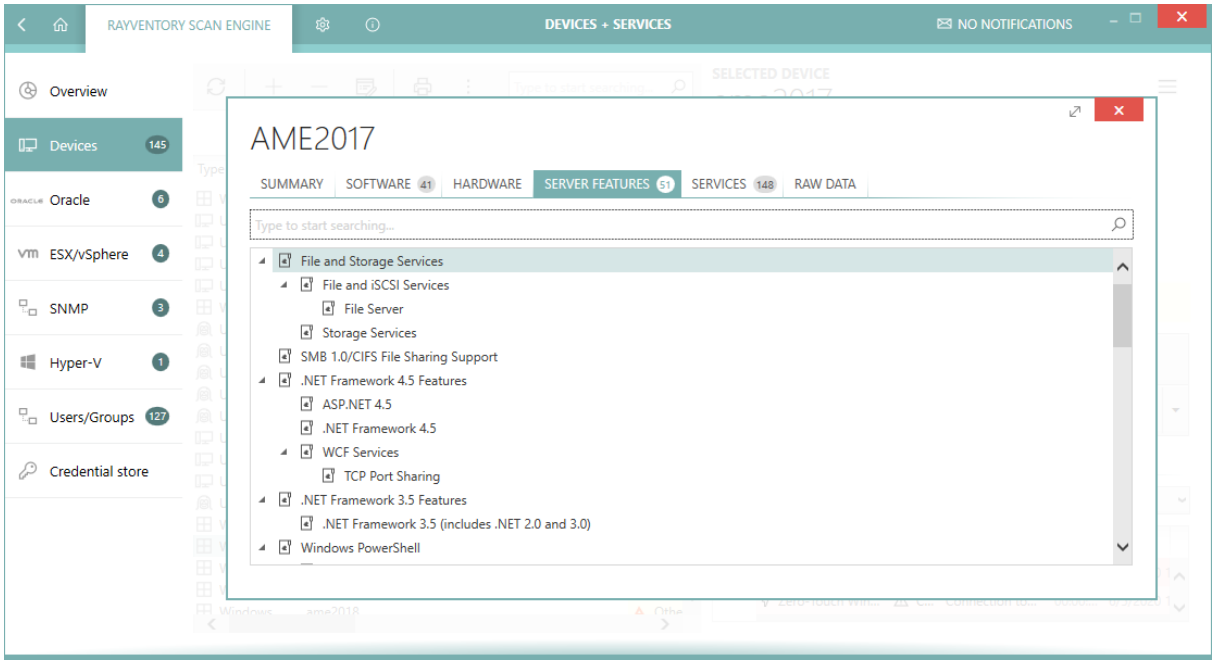



For most of supported devices, the following information should be available:

- The list of network adapters and their MAC addresses + IP addresses,
- The list of hardware drives, partitions, and mount points,
- The information about the graphic card,
- The amount of physical memory (RAM).

## Server features

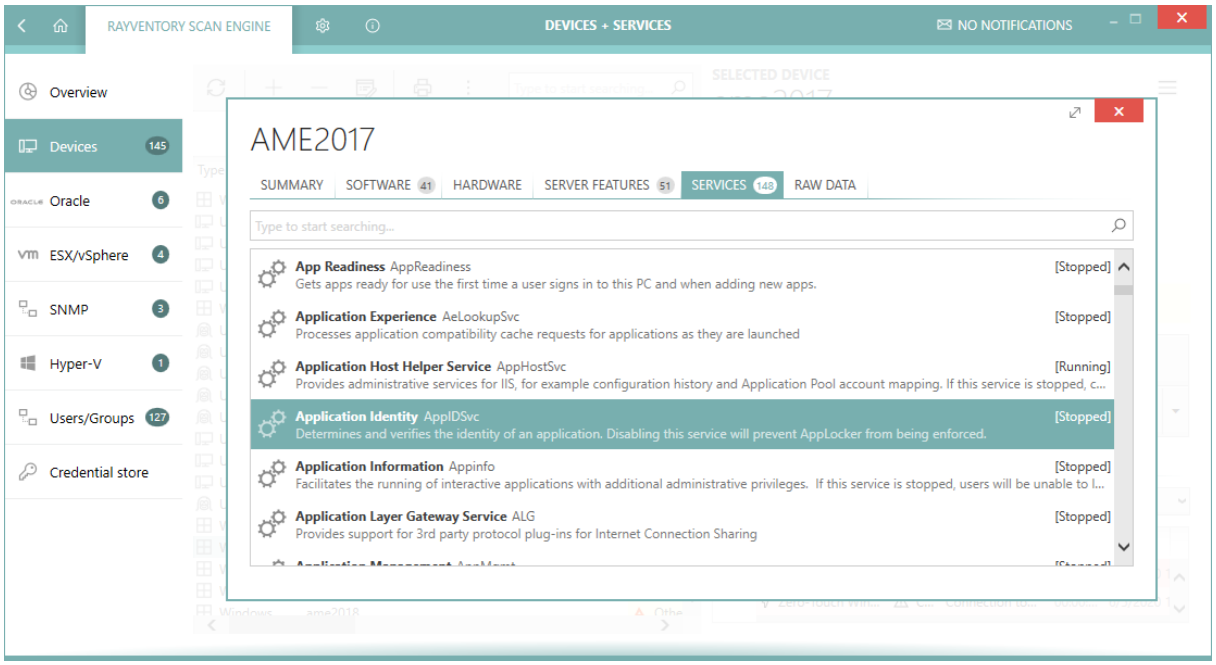
This tab shows the list of Windows Server features which were enabled at the time of the scan.




You can use the built-in search box to filter the list and find the feature of interest.

## Services

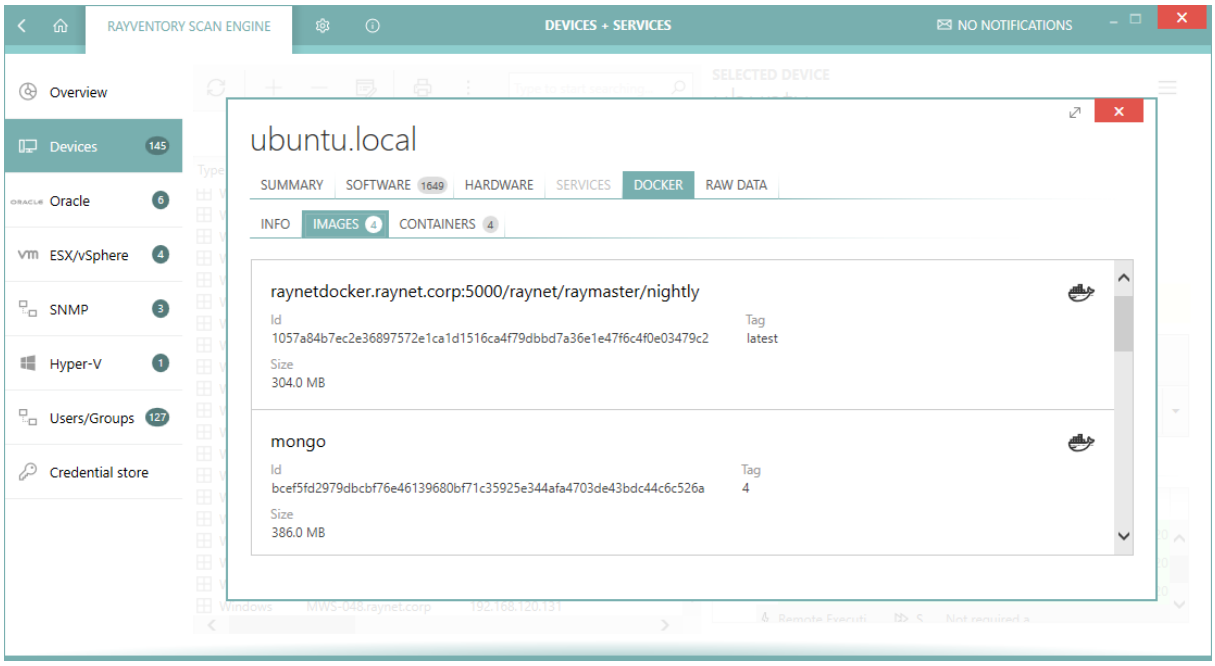
This tab shows the list of Windows services which were present on the machine at the time of the last inventory scan. Both started and stopped services are included.




You can use the built-in search box to filter the list and find the service of interest.

## Docker

The **DOCKER** tab contains an aggregated information about the available images, containers and meta data of a Docker instance, found on the particular device.



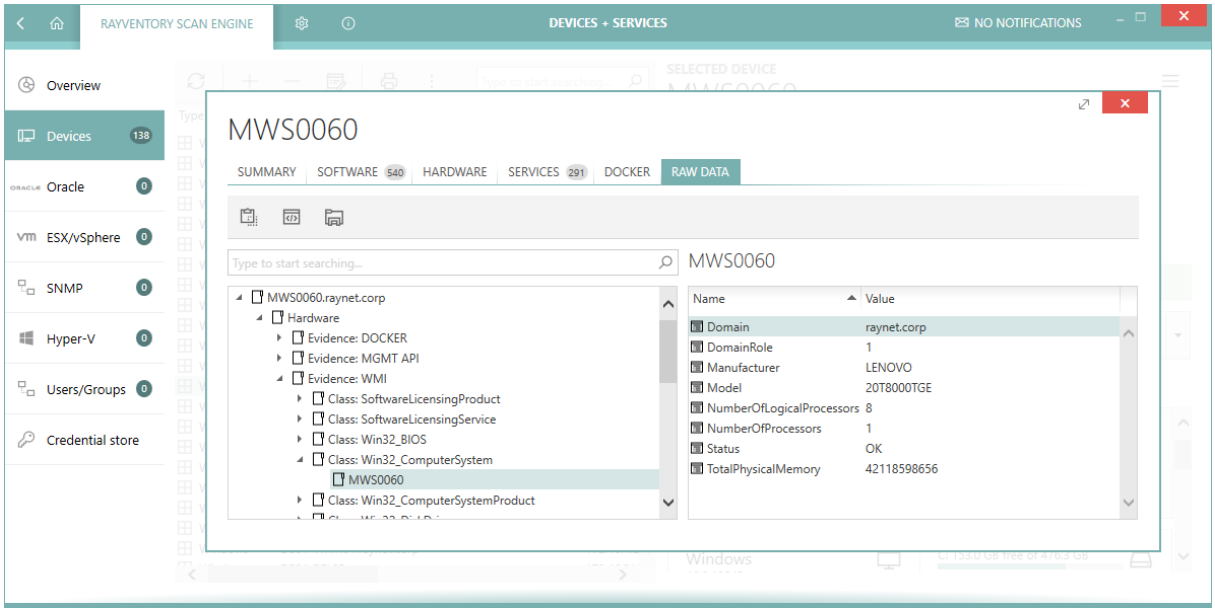

- **INFO**
  This is the place where client information is shown, for example the type of docker engine, its version and the architecture.
- **IMAGES**
  Locally available images are listed in this tab. For each image, the name, SHA-256 ID, size and tag are displayed.
- **CONTAINERS**
  This shows the list of containers, both running and stopped. Each container is identifier by the image it was run from, an entry command, exposed ports and information about whether it was running at the time of the scan.

**Note:**
This and much more information about docker entities can be found in the RAW view.

### Raw data

## Working with Raw Data

IT professionals and administrators having experience with the data structures of the RayVentory Scan Engine object domain can also work directly with the underlying inventory files (NDI).




To see the raw content:

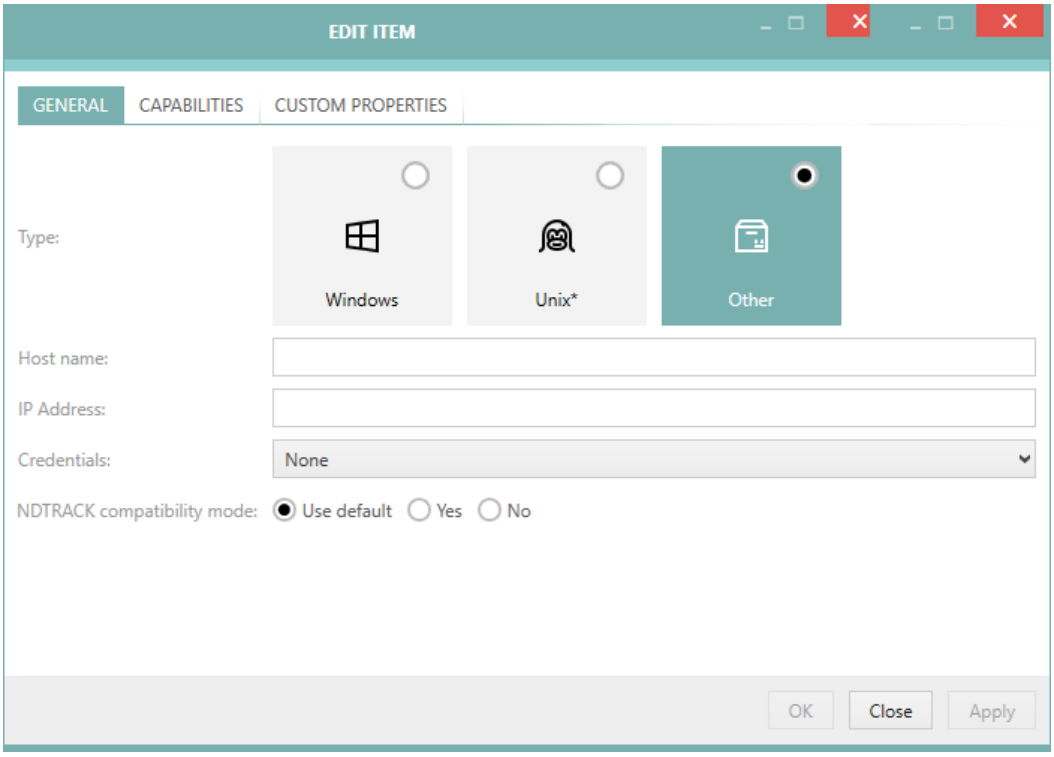
1. In the inventory overview select the last tab which is called **RAW DATA**.
2. The tree shows the logical structure of the underlying XML data contained within the `.ndi` file.
3. The trees can be expanded in order to reach the node containing the required content.
4. The three buttons in the toolbar have the following meaning:

- Copy the full path to the `.ndi` file to the clipboard.
- Open the `.ndi` file in the default editor.
- Open Windows Explorer and highlight the `.ndi` file.

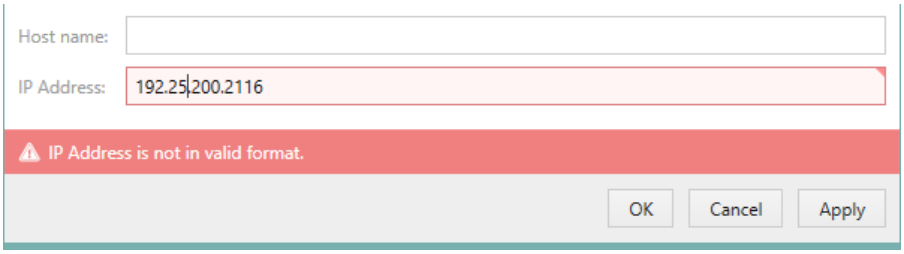
# Adding a New Device

## In order to add a new device execute the following steps:

1. Press the **+** button in the top toolbar or click the **Add..** menu item from the context menu.
2. A new empty dialog will be shown.



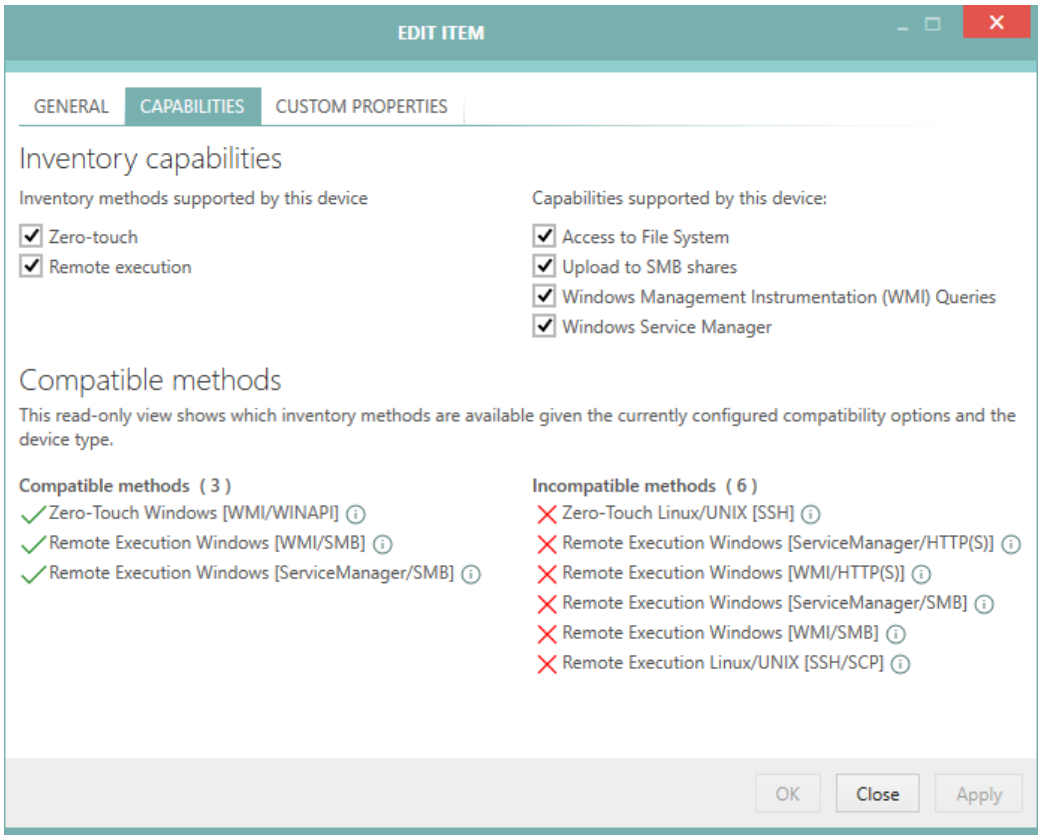

3. At minimum, specify the device DNS name and / or IP address. It is also possible to select the device family type (Windows / Unix / other) which will be respected by RayVentory Scan Engine later during the discovery step.
4. Optionally, it is possible to select the preferred credentials used by this device. If this is left empty, RayVentory Scan Engine applies a special logic which is described in the chapter Advanced Topics.
5. In the **CAPABILITIES** tab, the capabilities of the newly added device can be limited.
6. In the **CUSTOM PROPERTIES** tab the device-specific attributes can be configured.
7. Press **OK** to accept the changes and close the window or **Apply** to save them immediately.
8. If any mandatory field is not specified or is in the wrong format a validation error is shown:




Fix the issues indicated by the red error bar and press **OK** / **Apply** to apply the changes.

## Device Capabilities

The **CAPABILITIES** section allows users to precisely configure which low-level capabilities each device supports.



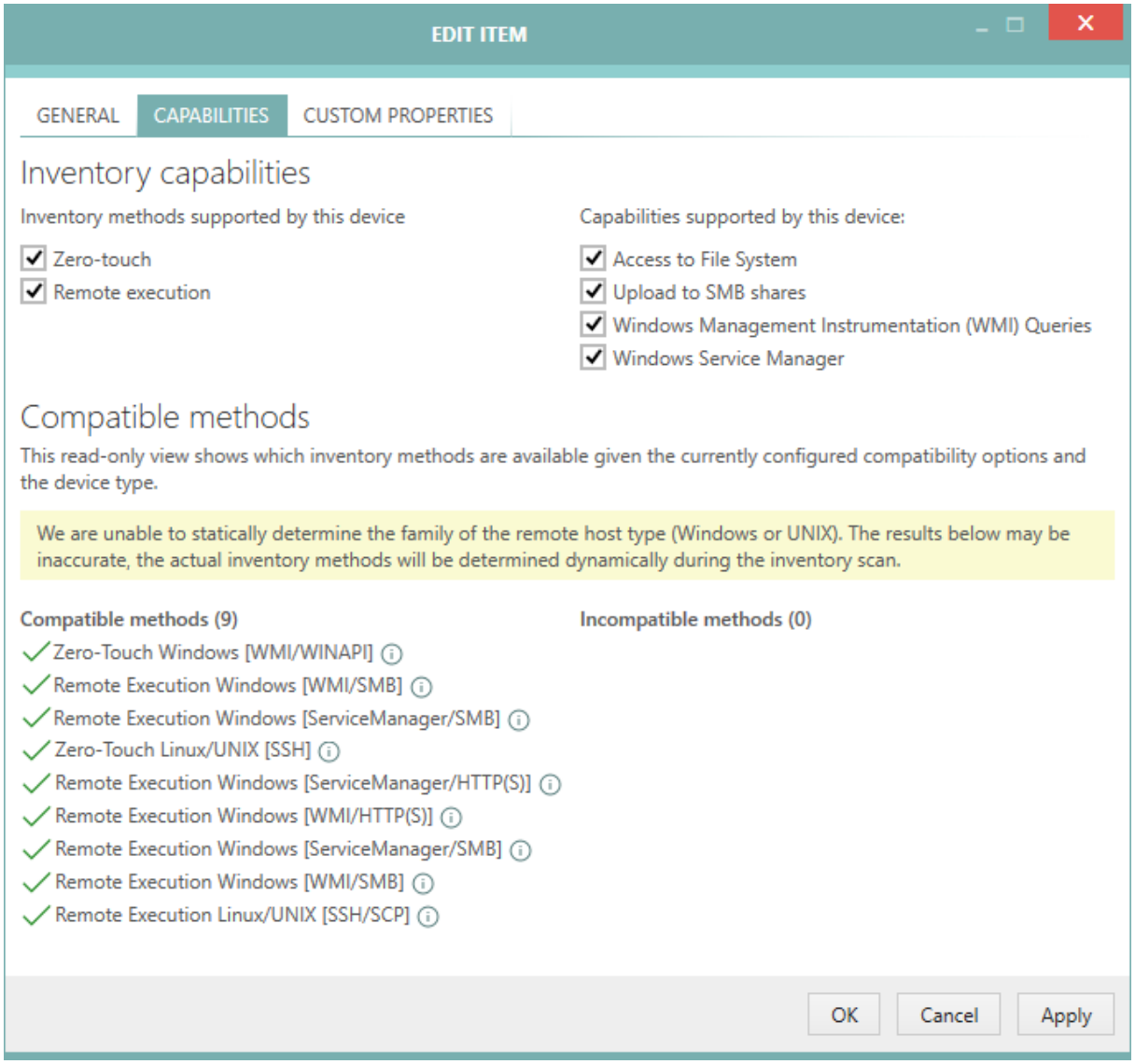

The combination of supported capabilities is used by the **Inventory** Wizard to determine which methods are compatible with which target. For example, unchecking the Zero-Touch option for a device will render it scannable by remote execution methods only.

The tab consists of three panels:

- The selection of general inventory methods supported by this device (Zero-Touch and / or Remote-Execution).
- More low-level capabilities which determine whether particular features should be supported by this device.
- The read-only view showing in real-time which inventory methods are applicable given the current state of the device.

The Zero-Touch execution does not require any additional capabilities other than certain RayVentory Scan Engine settings. On the other hand, the Remote-Execution may have different prerequisites depending on the method type. By ticking and unticking the checkboxes next to the methods, the lower view is refreshed and shows dynamically which methods are applicable for a given device. By hovering a mouse over the info icon next to a name, a detailed information is shown explaining which factors affect the static availability of the current inventory method.

For some devices, the view may not be shown accurately. For example, if the device type is **Unknown**, RayVentory Scan Engine is unable to determine which set of methods (Windows- or Unix-based) should be used. In this case, they are all shown as compatible methods, but a warning like below may be shown:




The options set here affect which methods are available later on when doing inventory scanning. Refer to the advanced topic Inventory Methods Overview to find out about the prerequisites and required settings.

## Preventing a Device from Being Scanned

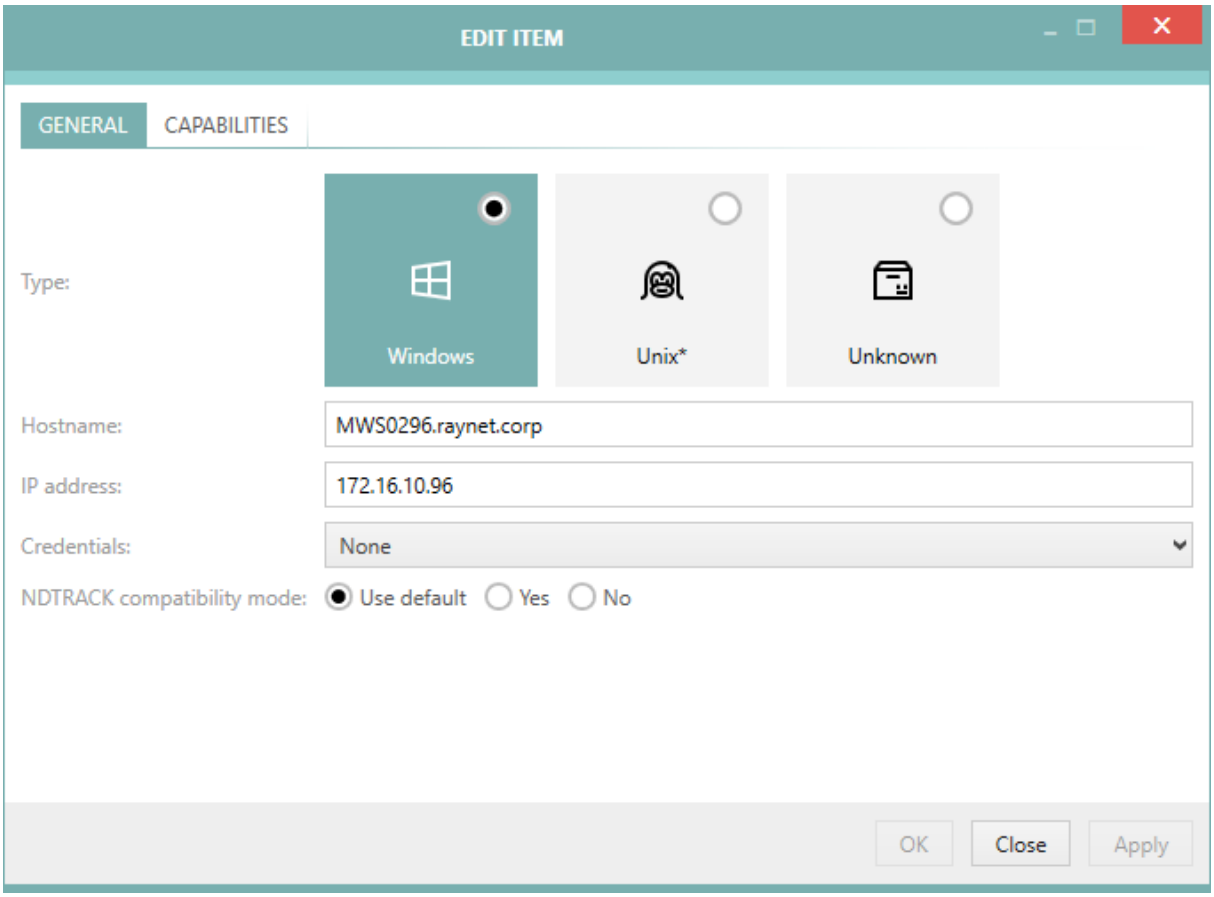
You can opt-out for any further scans of the device, thus preserving its current inventory state. To do that, disable the Zero-Touch and the Remote-Execution inventory. The list that is shown underneath these options should reflect this by saying that currently there is no compatible

method. This effectively means, that when the user executes an inventory job (from the Inventory wizard, PowerShell command let or from a scheduled task), the device is never scanned and its current inventory files and details are not affected by the new scan.

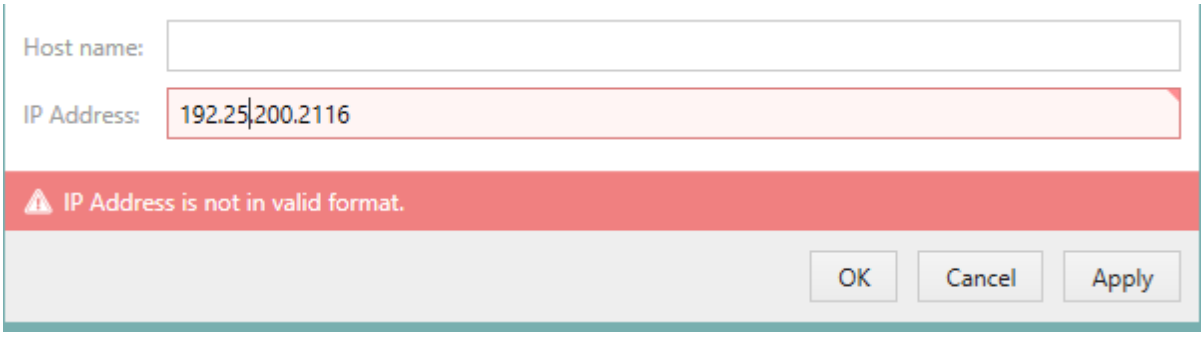
## Editing Devices

### In order to edit a device execute the following steps:

1. Highlight an entry in the list and press the **Edit selected...** button, the **Add..** menu item from the context menu, or the **Edit device...** button from the sidebar.
2. A new dialog will be shown with the details of the current selection:




3. Before the device can be saved, at minimum the device DNS name and / or IP address must be specified. It is also possible to select the device family type (Windows / Unix / other) which will be respected by RayVentory Scan Engine later at the discovery step.
4. Optionally, it is possible to select preferred credentials used by this device. If this is left empty, RayVentory Scan Engine applies a special logic which is described in the chapter Advanced Topics.
5. In the **CAPABILITIES** tab, you can also limit the capabilities of the edited device.
6. In the **CUSTOM PROPERTIES** tab the device-specific attributes can be configured.
7. Press **OK** to save the change and close the window or **Apply** to immediately save them.
8. If any mandatory field is not specified or is in the wrong format, a validation error is shown:

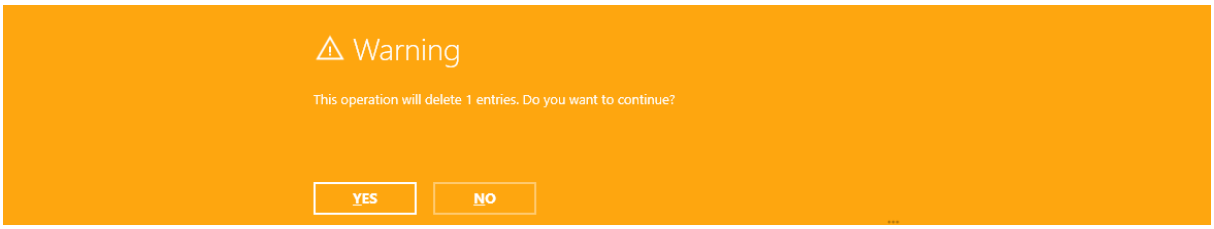



Fix the issues indicated by the red error bar and press **OK** / **Apply** to apply the changes.

## Removing Devices

### In Order to Remove a Device

1. Highlight an entry in the list and press the **-** button or click the **Remove...** menu item from the context menu.
2. Confirm the deletion by pressing **YES** in the confirmation dialog.

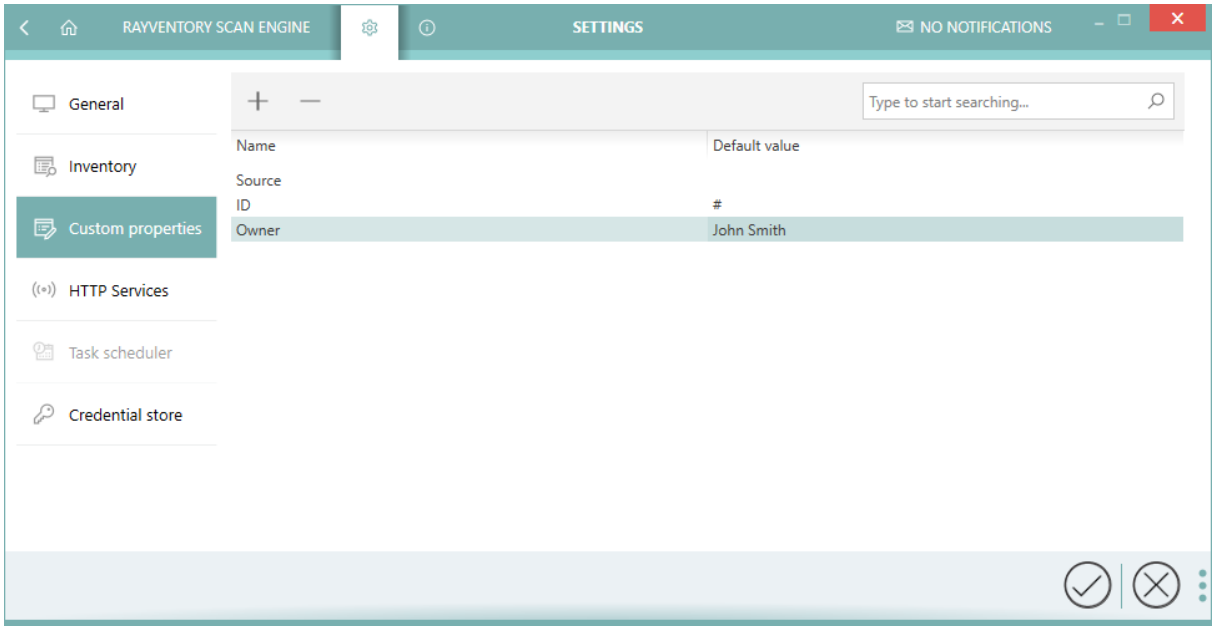



**Note:**
This operation is irreversible. Any existing inventory files which were assigned to the device will stay, though.

## Working with Custom Attributes

### Defining custom attributes

Custom attributes can be managed from the Settings screen, tab **Custom attributes**.




Each custom attribute consists of two fields:

- **Name**
  This is the name that will be used for value headers. When combined with NDI Upload, the custom attributes are written using the name as the key. This fields is required and must be unique.

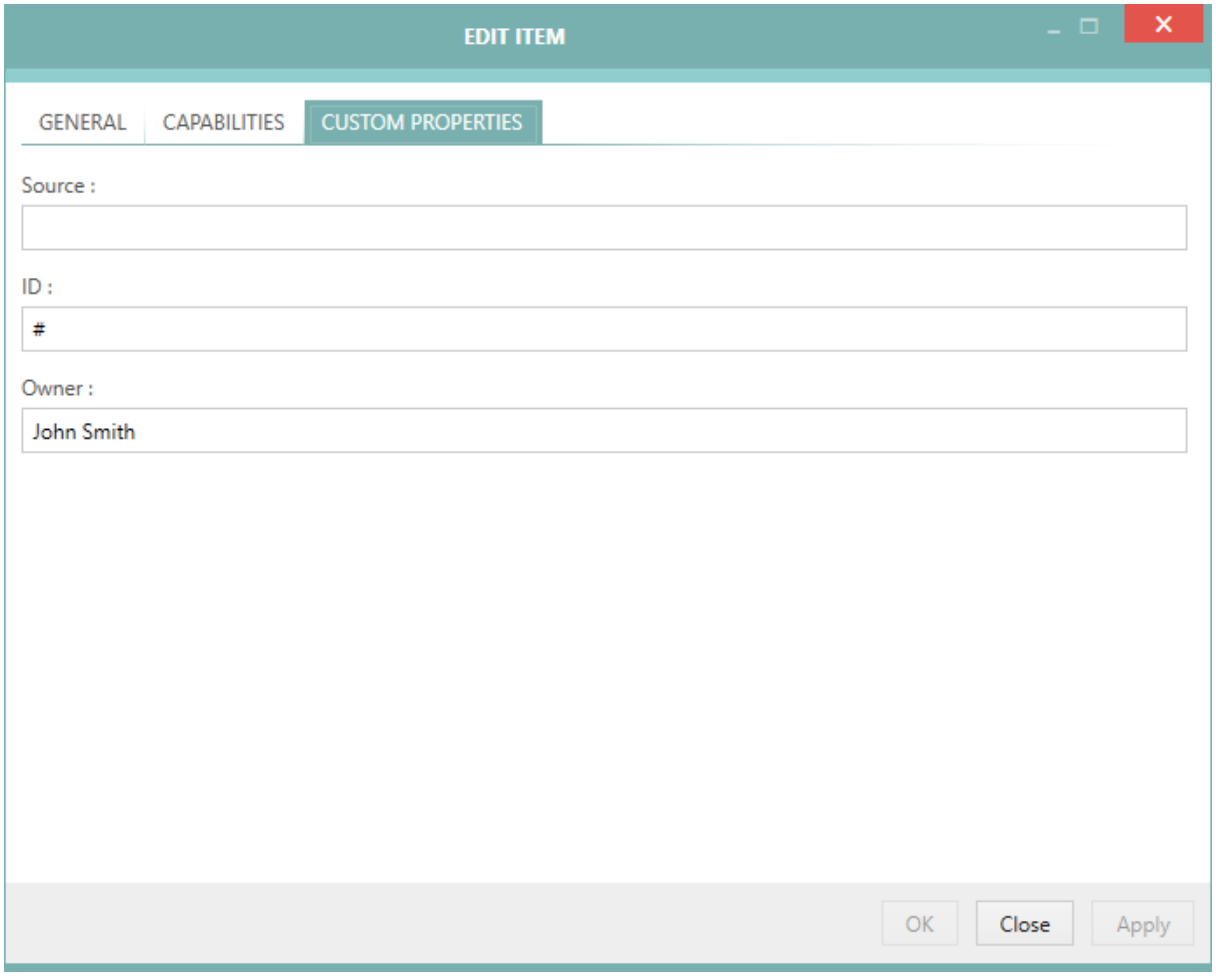
- **Default value**
  This is the default value used for attributes, for which the user did not provide anything yet. This field can be left empty to indicate that a simple empty string should be used as a default value.

The toolbar contains function buttons: **+** to add a new custom attribute, and **-** to remove the currently selected one.

### Editing custom attributes for devices

Once a set of custom attributes is defined, it is possible to define the actual values on a device level.

The editor is available in the **Edit device dialog**, tab **Custom attributes**.



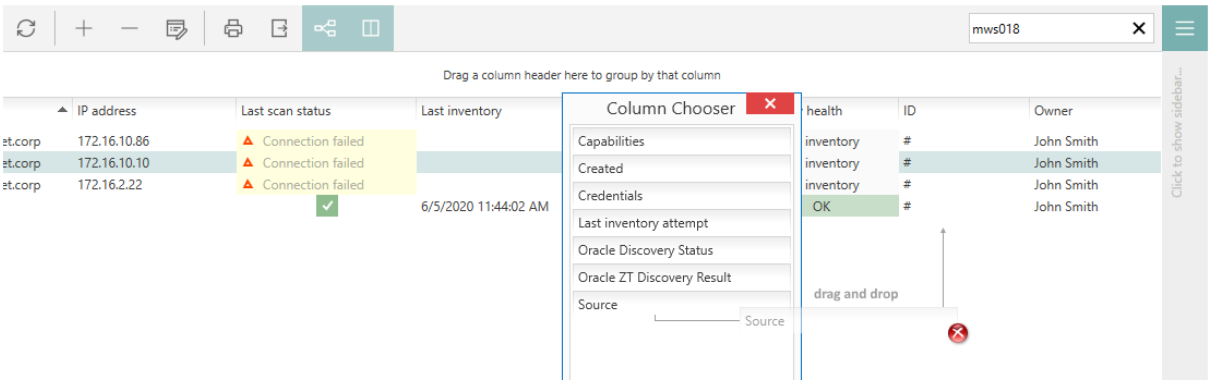

The number of fields and their names will be different, based on your current configuration.

If a default value is defined, for every device without a value configured explicitly by the user, the default one will be shown instead.

All custom properties are optional.

### Displaying and data-shaping

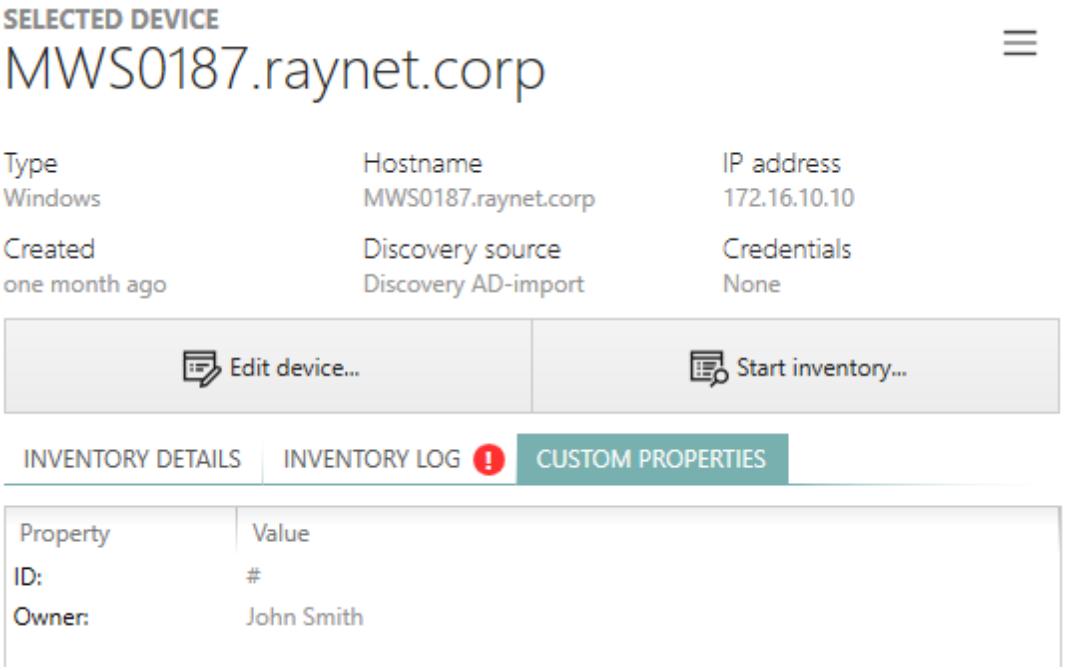
The device grid supports displaying custom attributes in a tabular way.




The attributes should be by default visible in the grid, each having a separate column. You can order, filter and search them using the same way as all other predefined columns.

To hide the custom column, drop it to the column chooser. In order to show the column again, drag it from the column chooser into the required place and release the mouse button to drop it.

Custom properties are also displayed in the device sidebar:




## Import Devices Wizard

The wizard can be executed by using the **Import devices connections from CSV file...** button on the bottom of the **Overview** section in the **Devices + Services** screen. The wizard will guide users through the process of importing devices from a `.csv` file.

### Format of the File

A **comma-separated values** (CSV) file is a text file that uses a comma to separate values. Each line of the file is a data record. Each record consists of one or more fields, separated by commas.

RayVentory Scan Engine uses a semicolon instead of a comma to separate the fields. The data for each device that is about to be imported should be in one line.

A data record can contain several values:

| Field | Description | Example |
|---|---|---|
| IP Address | The IPv4 address of the imported device, based on which it will be possible to connect to the target device. | 192.168.120.170 |
| Hostname | The device name identifying the computer in the network. | 0123.example.corp |
| Type | The type of the device to be added. RayVentory Scan Engine support three device types: **Windows**, **UNIX**, and **Unknown**. | Windows |
| Username | The username with which it will be possible to connect to the device and gain access to it in order to make an inventory. | Administrator |
| Password | The password for the account used to authenticate with the device. | Example!@3# |
| Credentials | The logical credential name that will be used to save the credentials. | 0123Credentials |

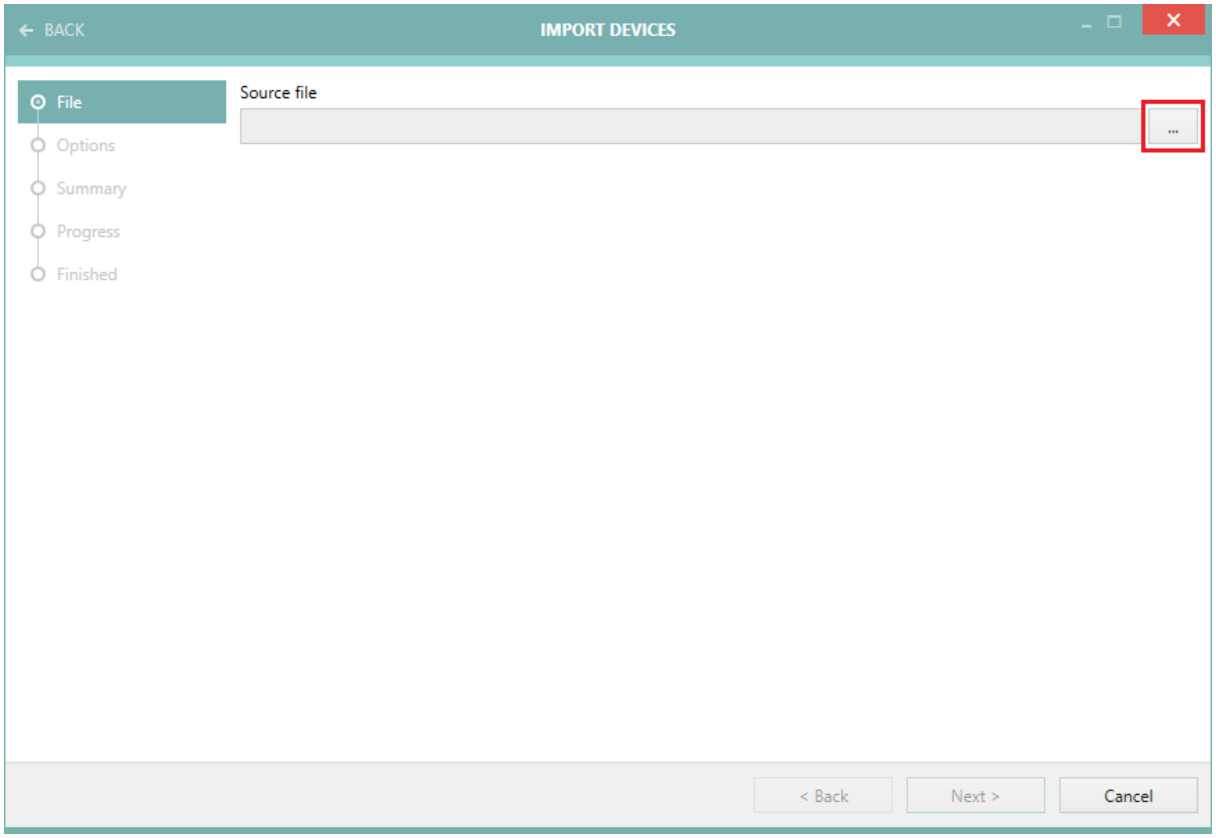
The minimum data required to import a device is the IP Address.

This is an example of a line based on the table above:

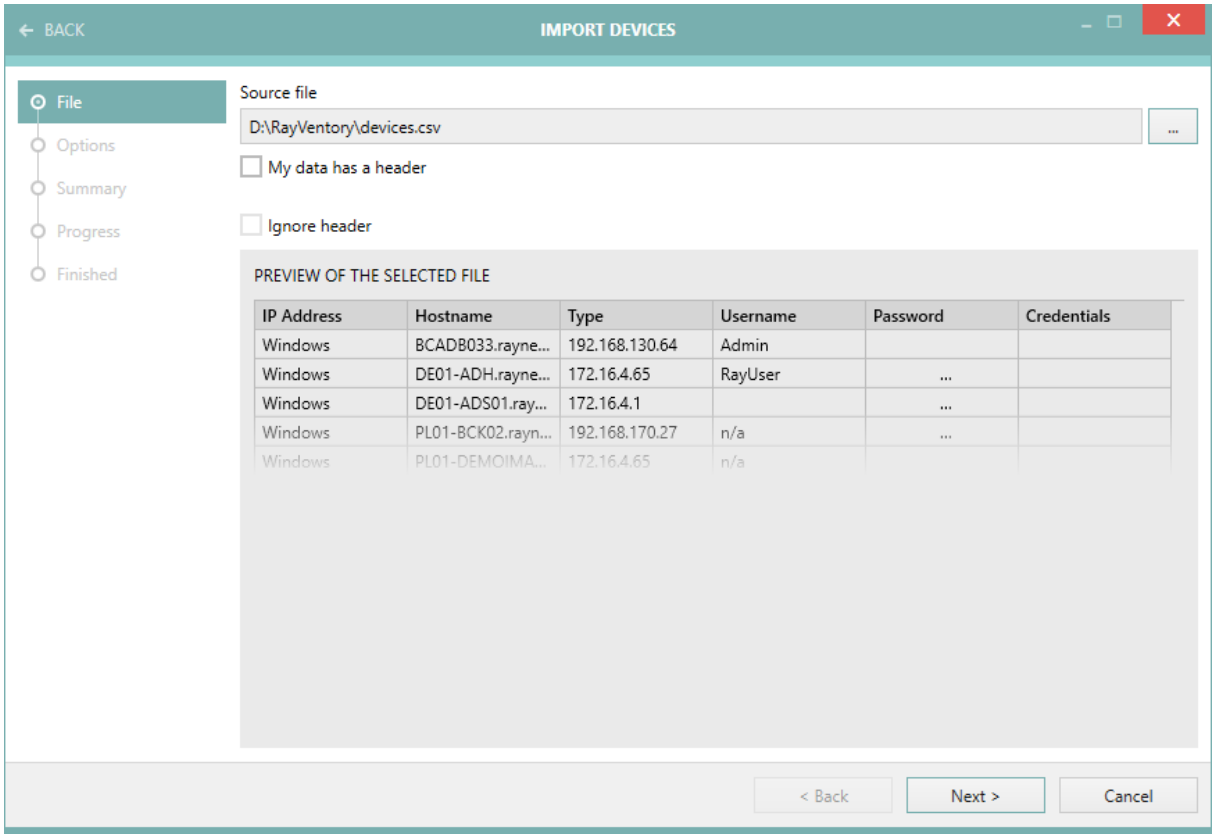
- 192.168.120.170;0123.example.corp;Windows;Administrator;Example!@3#;0123Credentials

### File Page

To select the file for the import, click the **Browse** [**...**] button. This will open the file seletion.



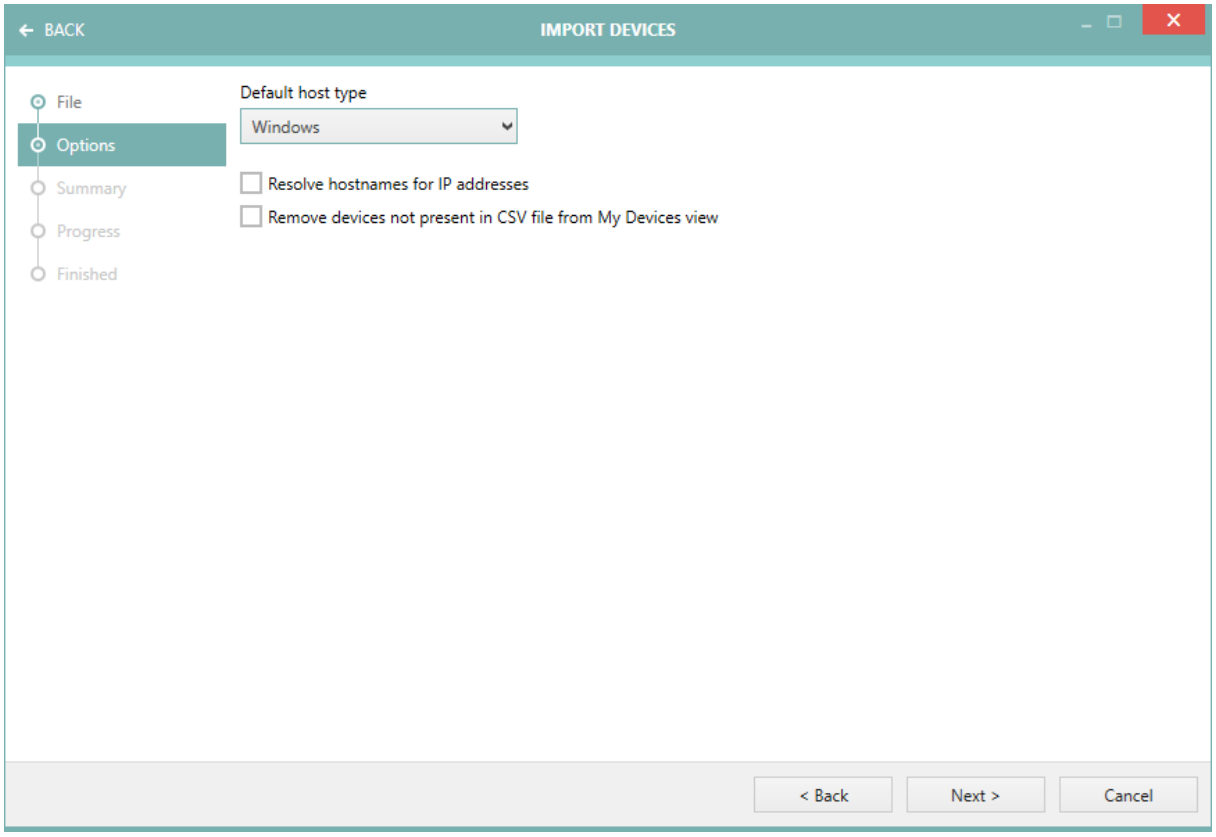

After selecting the file, a table with a preview of the first few rows will be displayed. The headers will show the name of the imported property.




The file data may be provided in a different order than indicated in the table. In this case, the first line of the file should contain the column names matching the names in the table. If this is the case, select **My data has a header** and verify in the **Preview of the selected file** table, that the columns match the parameters provided in the file. User can also ignore the file header from devices and force the data to be matched in the order given in the table above.

### Options Page

This page allows the user to select the default behavior for the import of an unknown device type and to specify additional tasks to be performed on the import.




If the device type is not recognized or the property is not specified in the device file, RayVentory Scan Engine may set the device type by default to the device type specified in the user preferences.

To change the type, select it from the drop-down list. There are three options to choose from:

- **Windows**
- **UNIX**
- **Not specified**

If the user selects **Not specified**, devices that do not contain the **Type** property will be marked as **Unknown** in the **My Devices** view. Users can change them later, by manually editing the device properties.

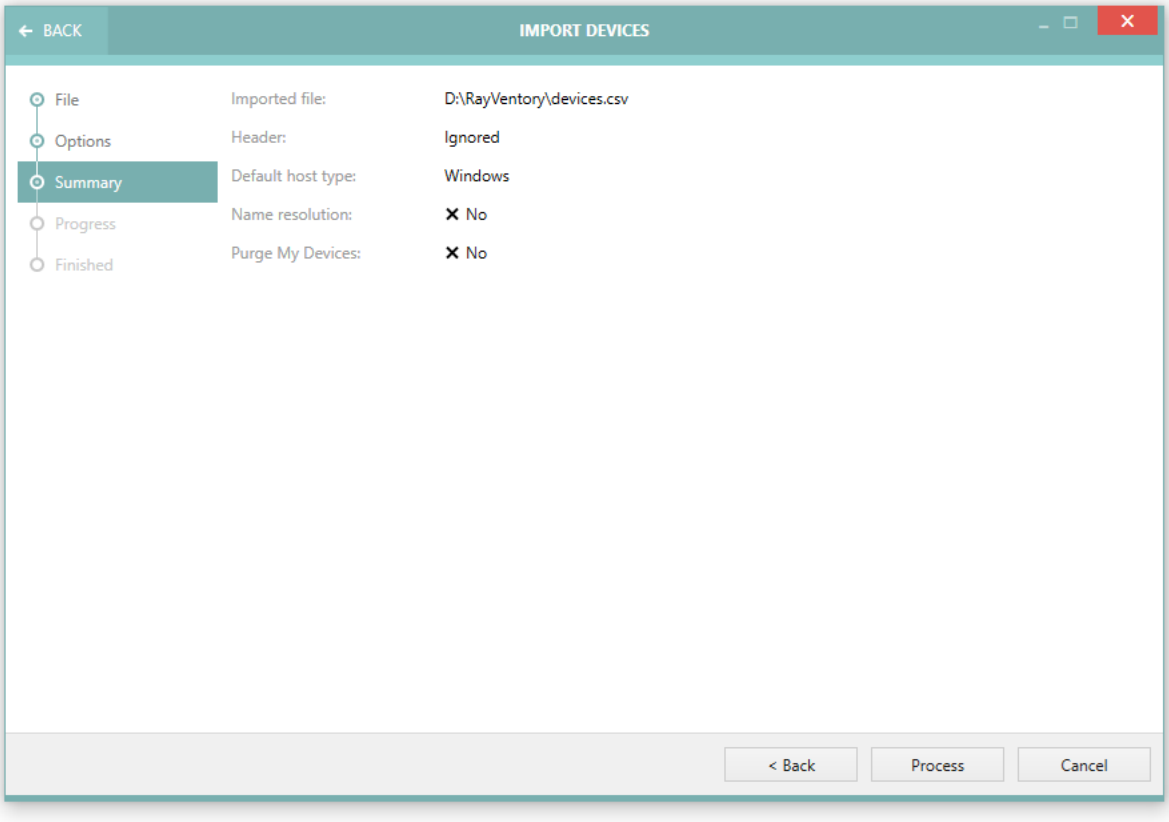
It is possible to order the wizard to perform additional actions during the device import:

- **Resolve hostnames for IP addresses**: In case the device being imported does not contain the hostname, RayVentory Scan Engine can try to find its name using the IP address of the machine. To resolve the hostname successfully, the device must be in the same network and be visible to the machine to which the user imports devices.
- **Remove devices not present in the CSV file from the My Devices view**: This option allows the user to remove all devices that are not included in the CSV file. If the device is already in the My Devices view from the CSV file, its inventory will remain available to the user without

any changes.

### Summary Page

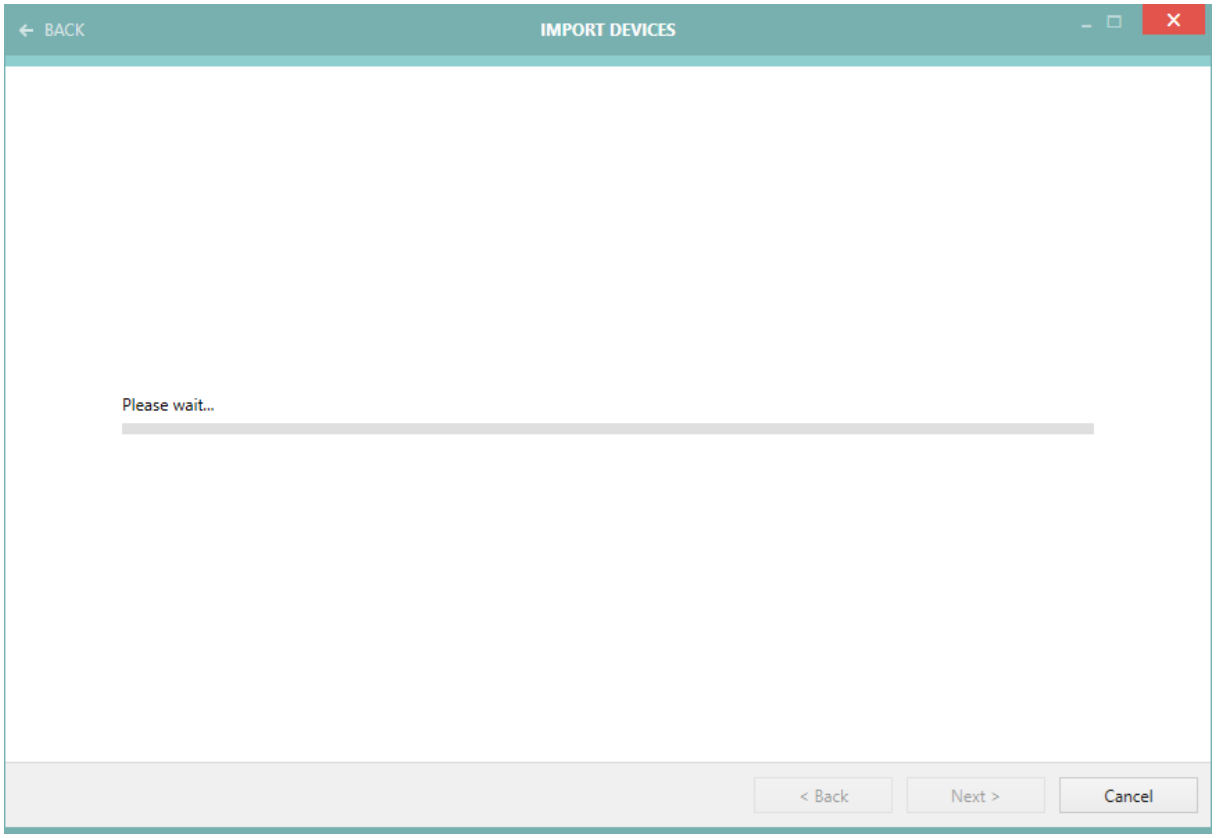
The **Summary** page shows the overview of all choices defined in the previous pages.



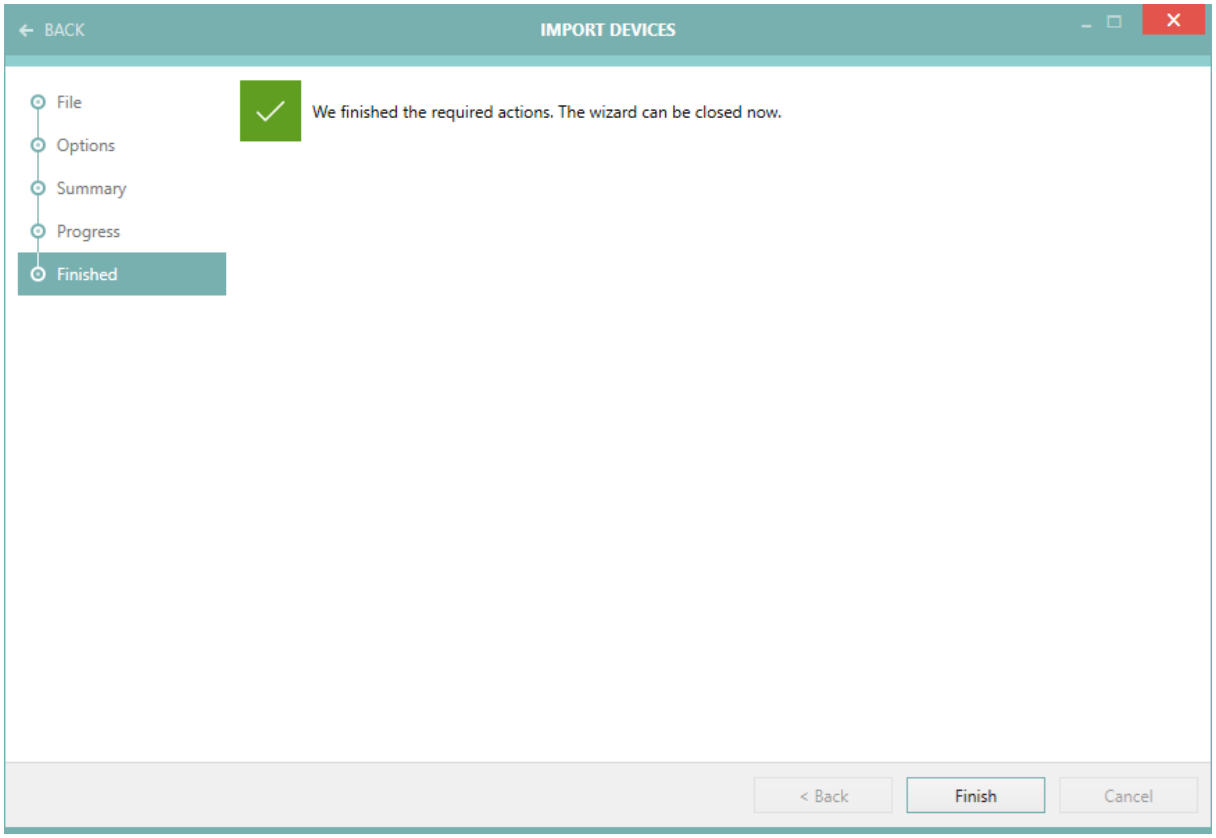

Click on **Process** to start the import.

### Progress and Results

The **Progress** page shows the current activity and the real-time results of the import.



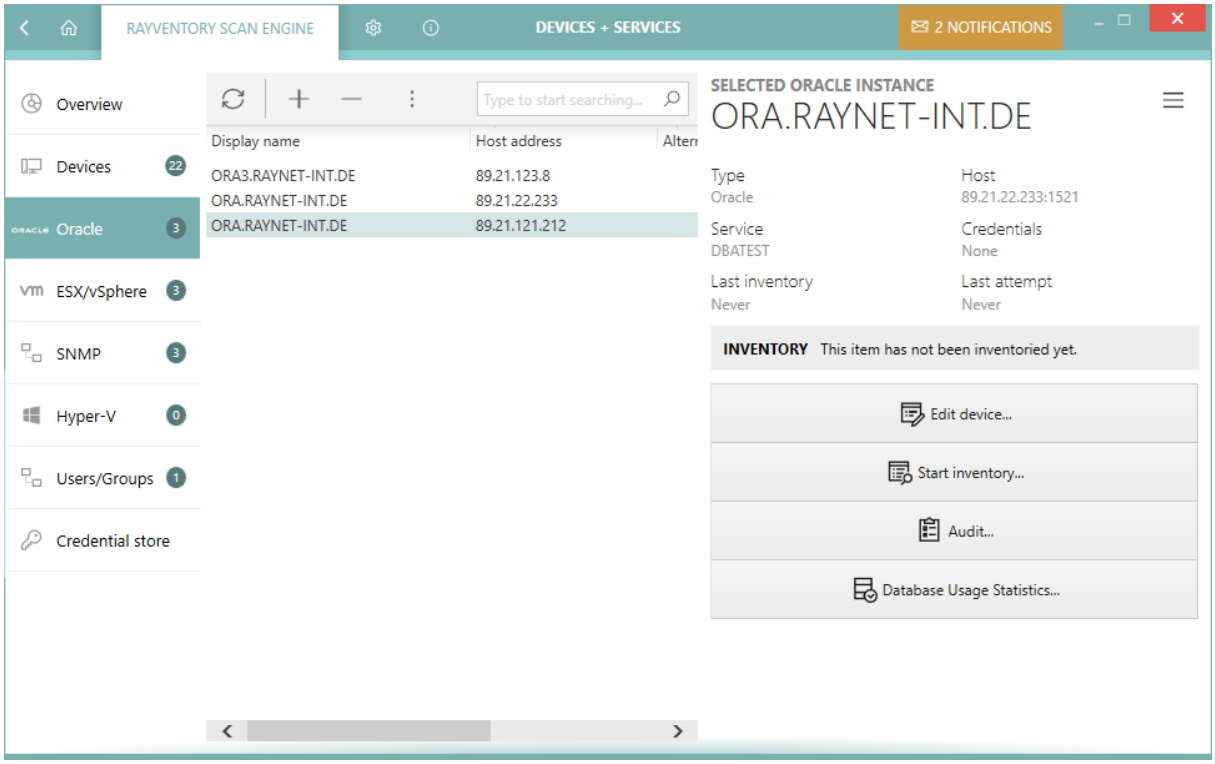

Once the import has been finished, a confirmation page will be shown.




Click on **Finish** to finalize the process.

# Oracle

This view aggregates the discovery and inventory data of your Oracle databases.




The view is divided into three parts:

- **Toolbar** - showing buttons to perform quick operations like adding, editing, and removing entries.
- **Main grid** - showing the list of all saved devices (physical and virtual).
- **Sidebar** - showing the details of the current selection including inventory data if available.

## Columns

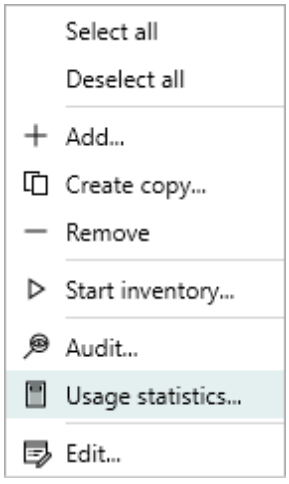
The grid supports a predefined set of columns. Out of these columns a handful is visible by default. It is possible to select more columns by pressing the **Column chooser** button in the grid toolbar. The following columns are available:

- **Type** - A static column with a connection icon (Oracle).
- **Host name** - The DNS hostname or IP address of the connection.
- **Service** - The name of the service.
- **Port** - The port used to communicate with the service.
- **Status** - The inventory status. There are 3 possible values:
  - n/a - The device has not been inventoried yet.
  - OK - The device has been inventoried and returned some results.
  - In any other case, the column **Status** contains a short description of the most recent issue.
- **Last inventory attempt** - The last time the inventory has been started (whether successful or not).

- **Credentials** - The logical names of credentials used when scanning this device.
- **Capabilities** - The short version of enumeration of allowed inventory capabilities for this device. The string consists of two-letter tokens, with the following meaning:
  - *ZT* = Zero-Touch
  - *RE* = Remote-Execution
  - *FS* = Access to File System
  - *SMB* = Upload to SMB Shares
  - *WMI* = Windows Management Instrumentation (WMI) Queries
  - *WSM* = Windows Service Manager
- **Last inventory attempt (time)** - The last time the inventory has been started (whether successful or not) (contains full date and time).
- **Last inventory** - The last time the inventory has been successfully started (only date part).
- **Last inventory (time)** - The last time the inventory has been successfully started (contains full date and time).
- **Last successful inventory method** - The name of the last inventory method that worked for this device. This method will be preferred in future scans.
- **Last failed inventory methods** - The names of inventory methods that failed the last time the device was scanned.
- **Last failed inventory method (details)** - Details about the methods that failed when the device was scanned for the last time.
- **Show inventory** - Shows the inventory details (only available if an inventory has been already performed).
- **Created** - The date when the device was created or imported.

## Context Menu

Pressing Right-Mouse-Button after highlighting an item opens a context menu for it.




- **Select all** - Selects all visible entries in the grid.
- **Deselect all** - Deselects all visible entries in the grid.
- **Add...** - Opens the New Database Dialog.

- **Create copy...** - Opens the New Database Dialog where the default values are automatically set to the values from the current selection.
- **Remove** - Removes the currently selected devices (see Removing Oracle connections).
- **Start inventory...** - Opens the Inventory Wizard for the currently selected devices.
- **Audit...** - Starts an audit for the currently selected connections.
- **Usage statistics... -** Starts **Database Feature Usage Statistics** for the currently selected connections.
- **Edit...** - Opens the Edit Database connection

**Note:**
Some options may be conditionally hidden or disabled (for example the menu item **Edit...** is disabled if more than one device is selected).

## Recent Scan Details

The results of the most recent scan are shown here. If the last scan was not successful, the last successful scan is also shown and can be selected from the drop-down.

For more information about working with Inventory log, refer to the following chapter: Recent Scan Details.

## Viewing Inventory Details

Once a database has been scanned successfully, a button which allows to show the details of scanned inventory assets will be shown in the sidebar. The button can be used to open a detailed overview of Oracle database instance details. For convenience, the summary of the database connection is also shown directly in the sidebar.

The **Details Inventory** overview contains several more tabs which are context-sensitive and which display various information depending on the connection type.

### Undocking the Inventory View

It is possible to "undock" the view to a separate window which can be moved freely and stacked with other windows. In order to do that, press the little **undock** button located next to the **CLOSE** button of the inventory overlay.

**Note:**
After undocking, the window cannot be docked in the main window anymore.

### Working with Raw Data

IT professionals and administrators having experience with data structures of RayVentory Scan Engine object domain can also work directly with underlying inventory files (NDI).

To see the raw content:

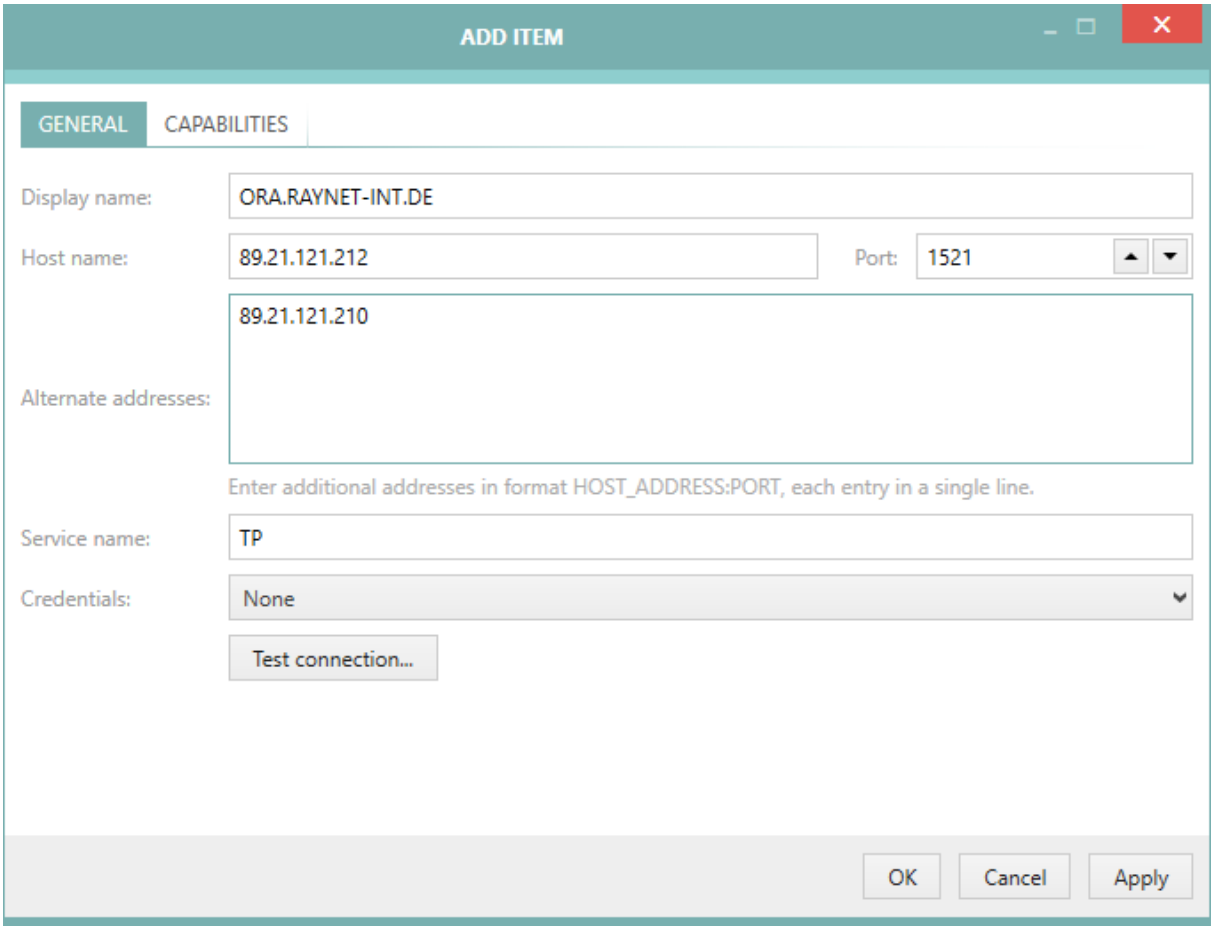
1. In the inventory overview select the last **RAW DATA** tab.
2. The tree shows the logical structure of the underlying XML data contained within the `.ndi` file.
3. It is possible to expand the trees to reach the node containing the required content.
4. The three buttons in the toolbar have the following meaning:

- Copy the full path to `.ndi` file to the clipboard.
- Open the `.ndi` file in the default editor.
- Open Windows Explorer and highlight the `.ndi` file.

## Adding New Database Connections

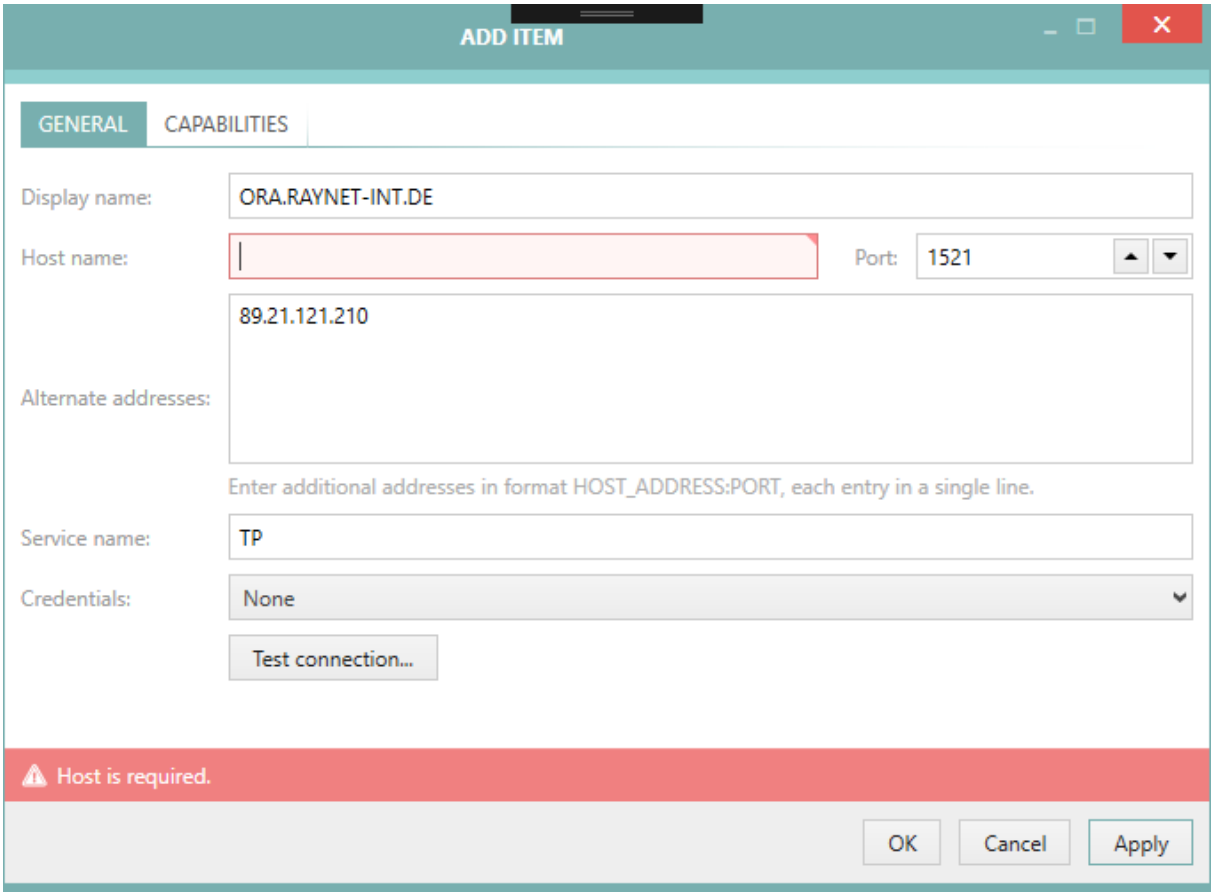
### In Order to Add a New Database Connection

1. Press **+** button in the top toolbar or click **Add..** menu item from the context menu.
2. A new empty dialog will be shown.




3. At the minimum, specify the host name and the service name.
4. It is possible to optionally select the preferred credentials used by this database connection. If this is left empty, RayVentory Scan Engine applies a special logic which is described in the chapter Advanced Topics.

5. In the **CAPABILITIES** tab, it is possible to limit the capabilities of the newly added device.
6. Press **OK** to accept the changes and close the window, or click **Apply** to immediately save them.
7. If any required field is not specified or is in the wrong format a validation error is shown:

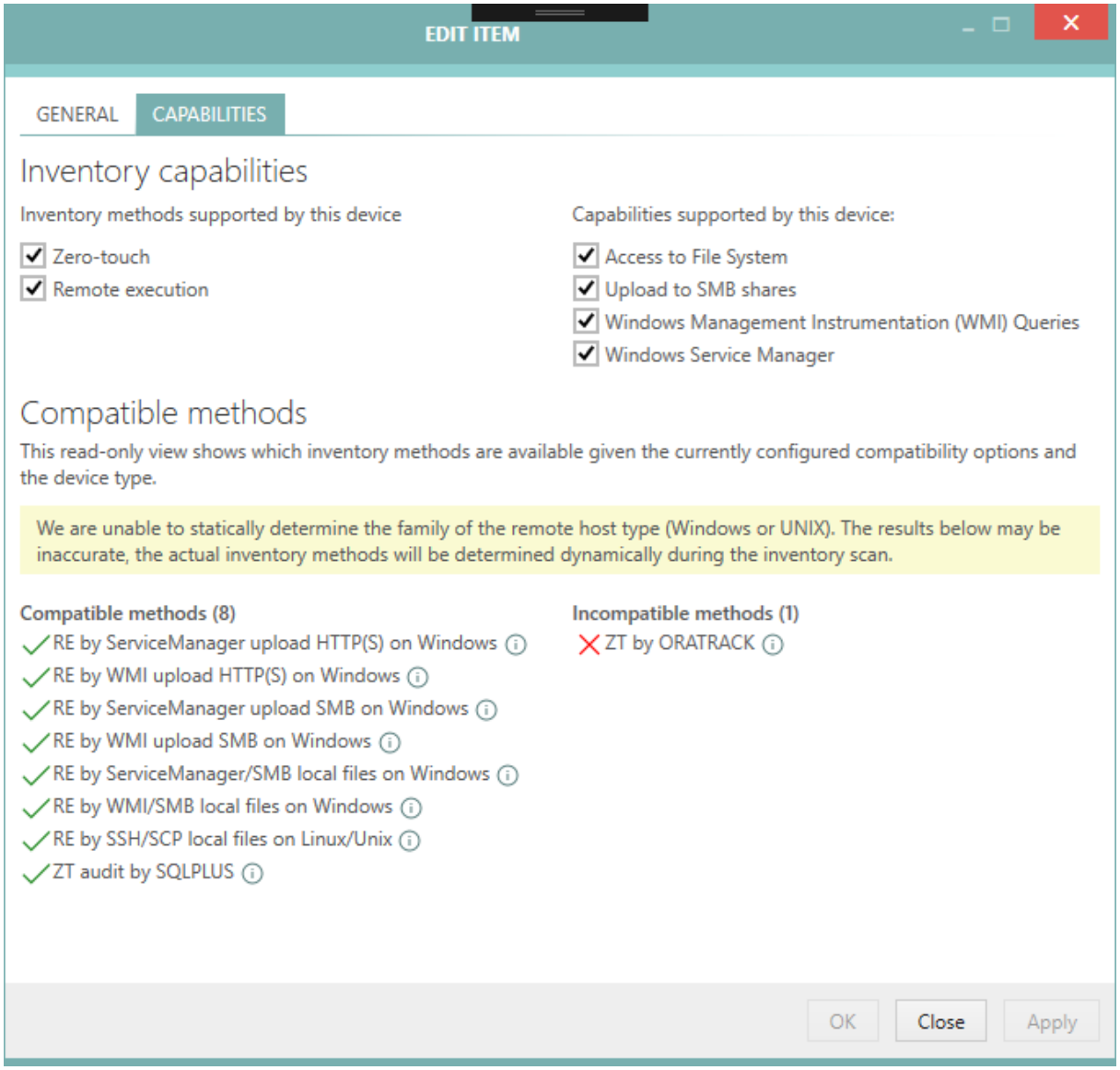



Fix the issues indicated by the red error bar and press **OK** / **Apply** to apply the changes.

**Note:**
The connection properties dialog that is being used to create and edit a vSphere / EX connection features a **Test connection...** button that allows to test the connection (and the credentials in the credential store) immediately.

## Database Connection Capabilities

The capabilities section allows users to precisely configure which low-level capabilities each device supports.




The combination of supported capabilities is used by the **Inventory** Wizard to determine, which methods are compatible with which target. For example, unchecking Zero-Touch options for a device will render it only scannable by remote execution methods.

The tab consists of three panels:

- The selection of general inventory methods supported by this device (Zero-Touch and / or Remote-Execution).
- More low-level capabilities, determining whether particular features should be supported by this device.
- The read-only view showing in real-time which inventory methods are applicable given the current state of the device.

Zero-Touch execution does not require any additional capabilities other than certain RayVentory

Scan Engine settings. On the other hand, the Remote-Execution may have different prerequisites depending on the method type. By ticking and unticking the checkboxes next to the methods the lower view is refreshed and shows dynamically which methods are applicable for a given device. By hovering a mouse over the info icon next to a name a detailed information is shown explaining which factors affect the static availability of the current inventory method.

The target platform cannot be statically determined for Oracle connections. The methods that depend on a particular host type are always shown as compatible as long as their other requirements are met.

The options set here affect which methods are available later on when doing inventory scanning. Refer to advanced topic Inventory Methods Overview to find out about the prerequisites and required settings.

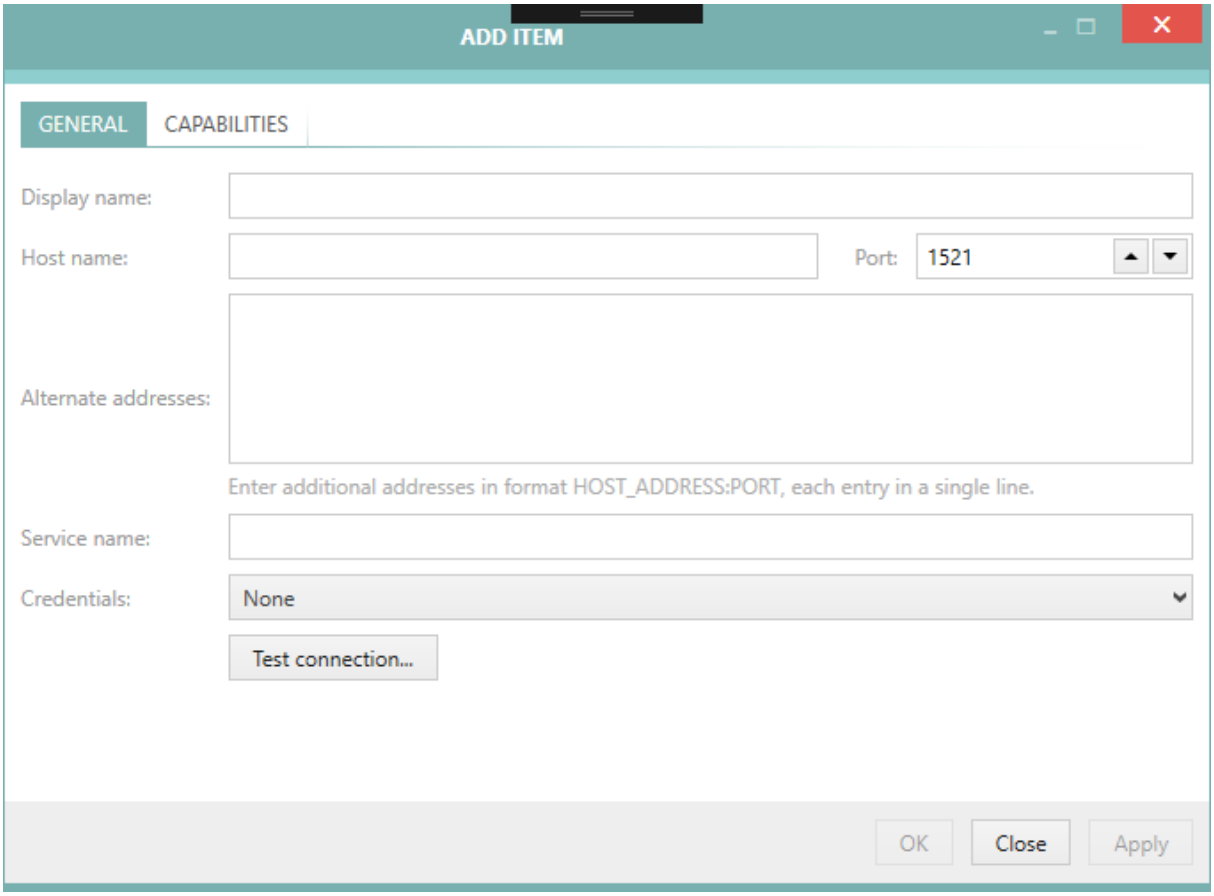
### Preventing a Connection from Being Scanned

You can opt-out for any further scans of the connection, thus preserving its current inventory state. To do that, disable the Zero-Touch and the Remote-Execution inventory. The list that is shown underneath these options should reflect this by saying that currently there is no compatible method. This effectively means that, when the user executes an inventory job (from the Inventory wizard, PowerShell command let or from a scheduled task), the database is never scanned and its current inventory files and details are not affected by the new scan.

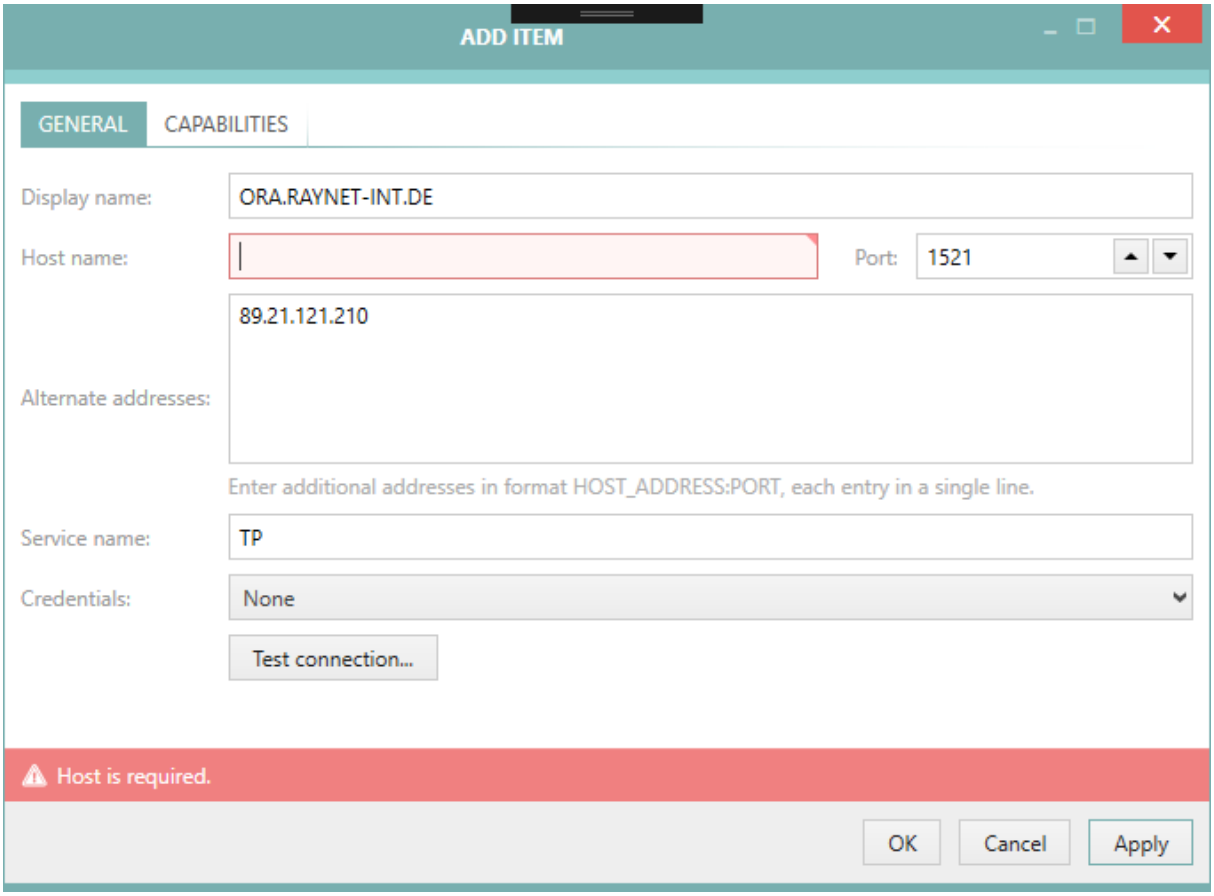
## Editing Database Connections

### In Order to Edit a Database Connection

1. Highlight an entry in the list and press the **Edit selected...** button, click the **Add..** menu item from the context menu, or press the **Edit device...** button in the sidebar.
2. A new dialog will be shown with the details of the current selection:




3. Before the connection can be saved it is necessary to specify at least the device host name and the service name.
4. Optionally, the preferred credentials used by this database can be selected. If this is left empty, RayVentory Scan Engine applies a special logic which is described in the chapter Advanced Topics.
5. In the **CAPABILITIES** tab, it is also possible to limit the capabilities of the edited database connection.
6. Press **OK** to save the change and close the window or **Apply** to immediately save them.
7. If any mandatory field is not specified or is in the wrong format a validation error is shown:




Fix the issues indicated by the red error bar, and press **OK** / **Apply** to apply the changes.

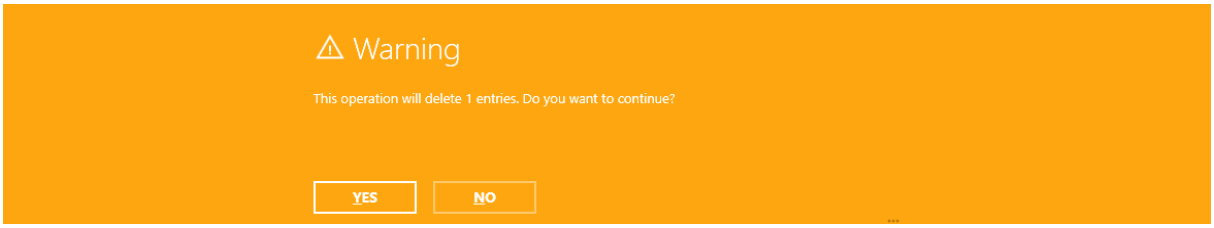
**Note:**

- The connection properties dialog that is being used to create and edit a vSphere / ESX connection features a **Test connection...** button that allows to test the connection (and the credentials in the credential store) immediately.

## Removing Database Connections

### In Order to Remove a Database Connection

1. Highlight an entry in the list and press the **-** button or click the **Remove...** menu item from the context menu.
2. Confirm the deletion by pressing **YES** in the confirmation dialog.



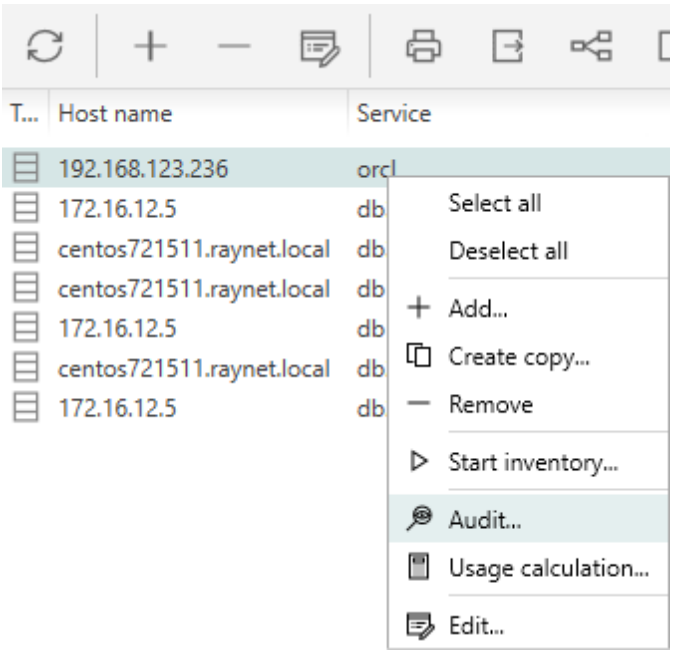
**Note:**
This operation is irreversible. Any existing inventory files which were assigned to the connection stay, though.

## Support for Review Lite Script

With RayVentory Scan Engine, it is possible to run the **Oracle's Review Lite Script** on many target databases at once and collect the output files.

To use this option in the settings under **Review Lite Script path** the path to the copy of the **Review Lite Script** needs to be set. Furthermore, the path to the SQLplus executable in the local Oracle client installation needs to be set (see Oracle settings for more information on that).

To perform an audit on one or more Oracle databases, select them in the Oracle view, press the **Right Mouse Button** to open a context menu, and select **Audit...**.



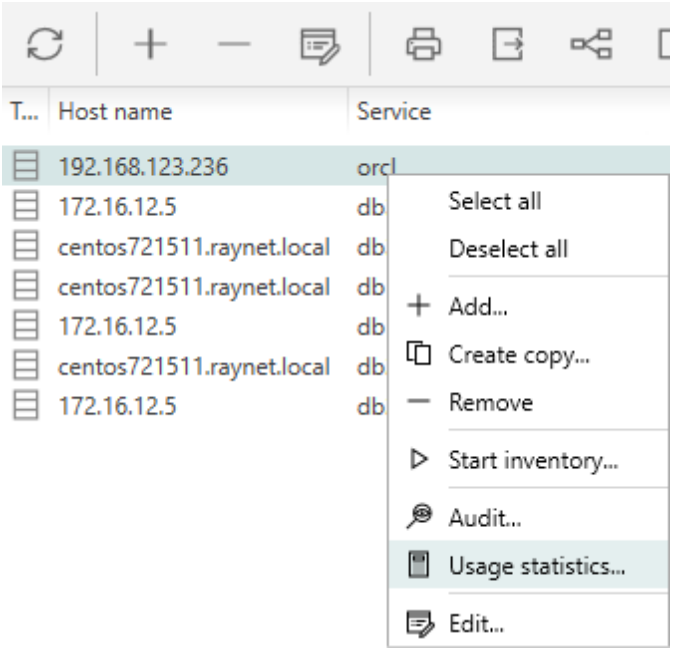

Alternatively, it is possible invoke the same functionality by pressing the button **Audit** from the right hand sidebar. This applies to the currently selected item(s) as well.

## Support for Database Feature Usage Script

RayVentory Scan Engine can be used to more easily run the **Oracle's Database Feature Usage Statistics** script (DFUS) on multiple target databases at once and collect the output as `.ndi` file.

To use this option in the settings under **Oracle Database Feature Usage Statistics script path** the path to a copy of the DFUS script needs to be set. Furthermore, it is necessary to set a path to the SQLplus executable located in the local Oracle client installation.

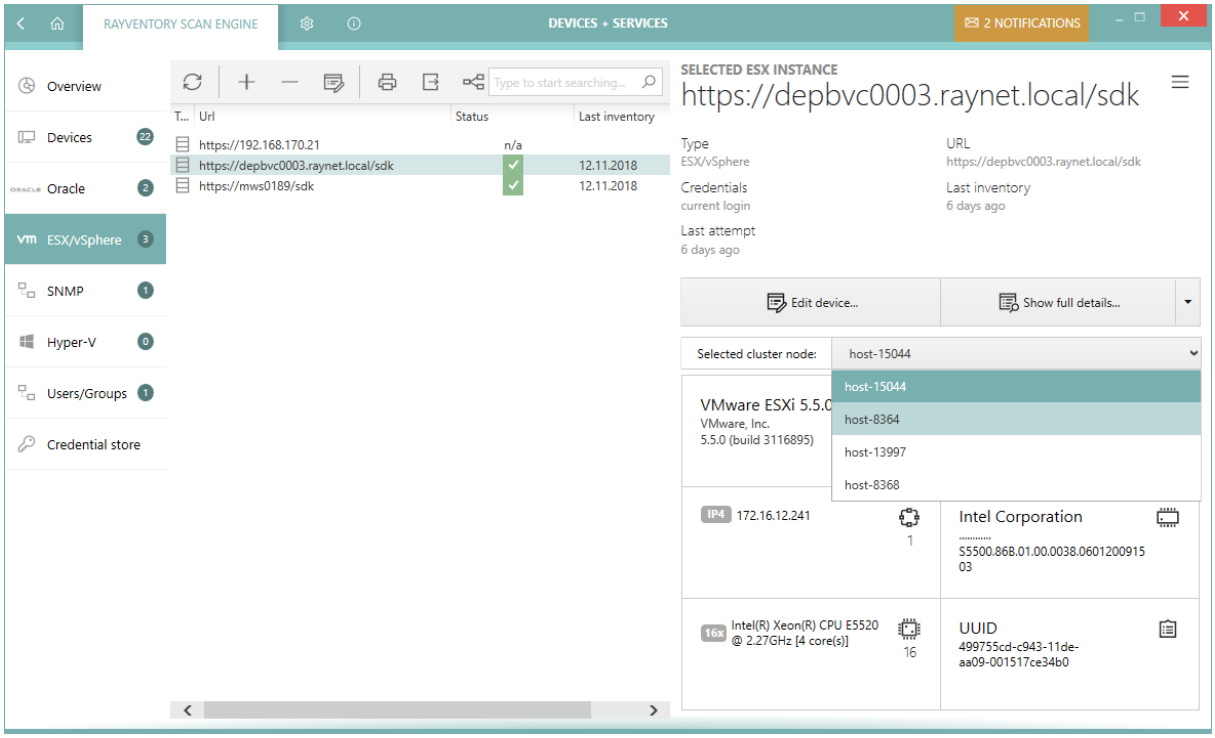
To execute feature usage calculation on one or more Oracle databases select them in the Oracle view, press the **Right Mouse Button** to open a context menu, and select **Usage statistics...**.




Alternatively, it is possible invoke the same functionality by pressing the button **Database Usage Statistics...** from the right hand sidebar. This applies to the currently selected item(s) as well.

# ESX / vSphere

This view aggregates discovery and inventory data of your vSphere / ESX instances.




The view is divided into three parts:

- **Toolbar** - showing buttons to perform quick operations like adding, editing, and removing entries.
- **Main grid** - showing the list of all saved devices (physical and virtual).
- **Sidebar** - showing the details of the current selection, including inventory data if available.

## Columns

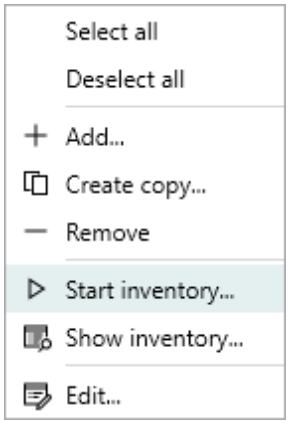
The grid supports a predefined set of columns, out of these columns a handful is visible by default. It is possible to select more columns by pressing the **Column chooser** button in the grid toolbar. The following columns are available:

- **Type** - A static column with the connection icon (vSphere / ESX).
- **URL** - The DNS hostname or the IP address of the instance.
- **Status** - The inventory status. There are 3 possible values:
  - n/a - The device has not been inventoried yet.
  - OK - The device has been inventoried and returned some results.
  - In any other case, the column **Status** contains a short description of the most recent issue.

- **Last inventory attempt** - The last time the inventory has been started (whether successful or not).
- **Credentials** - The logical names of the credentials used when scanning this device.
- **Capabilities** - The short version of enumeration of allowed inventory capabilities for this instance. The string consists of two-letter tokens, with the following meaning:
  - *ZT* = Zero-Touch
  - *RE* = Remote-Execution
  - *FS* = Access to File System
  - *SMB* = Upload to SMB Shares
  - *WMI* = Windows Management Instrumentation (WMI) Queries
  - *WSM* = Windows Service Manager
- **Last inventory attempt (time)** - The last time the inventory has been started (whether successful or not) (contains full date and time).
- **Last inventory** - The last time the inventory has been successfully started (only date part).
- **Last inventory (time)** - The last time the inventory has been successfully started (contains full date and time).
- **Last successful inventory method** - The name of the last inventory method that worked for this instance. This method will be preferred for future scans.
- **Last failed inventory methods** - The names of inventory methods that failed the last time the instance was scanned.
- **Last failed inventory method (details)** - Details about the methods that failed when the instance was scanned for the last time.
- **Created** - The date when the connection was created or imported.

## Context Menu

Pressing the Right-Mouse-Button after highlighting an item opens a context menu for it.




- **Select all** - Selects all visible entries in the grid.
- **Deselect all** - Deselects all visible entries in the grid.
- **Add...** - Opens the New vSphere Dialog.
- **Create copy...** - Opens the New vSphere Dialog where the default values are automatically set to the values from the current selection.

- **Remove** - Removes the currently selected devices (see Removing vSphere connections).
- **Start inventory...** - Opens the Inventory Wizard for the currently selected devices.
- **Show inventory** - Shows the details inventory (only available if an inventory has been performed).
- **Edit...** - Opens the Edit vSphere connection connection.

**Note:**
Some options may be conditionally hidden or disabled (for example the menu item **Edit...** is disabled if more than one device is selected).

## Recent Scan Details

The results of the most recent scan are shown here. If the last scan was not successful, the last successful scan is also shown and can be selected from the drop-down.

For more information about working with Inventory log, refer to the following chapter: Recent Scan Details.

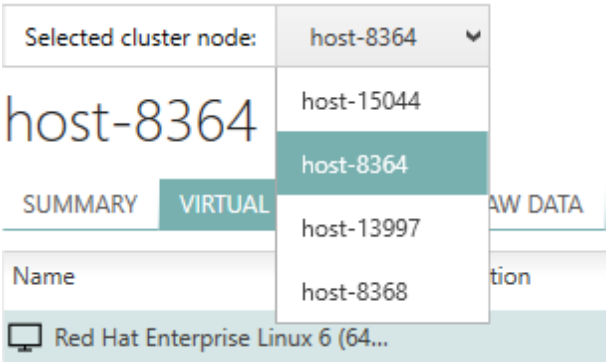
## Viewing Inventory Details

Once an vSphere / ESX instance has been successfully scanned, a button will be shown in the sidebar allowing the user to show the details of the scanned inventory asset including the virtual machines running on it. It is possible to use the button to open a detailed overview of machine software, hardware, and virtualized guests. For convenience, the summary of the vSphere / ESX instance is also shown directly in the sidebar.

The details inventory overview contains several more tabs, which are contextually sensitive and display various information depending on the connection type.

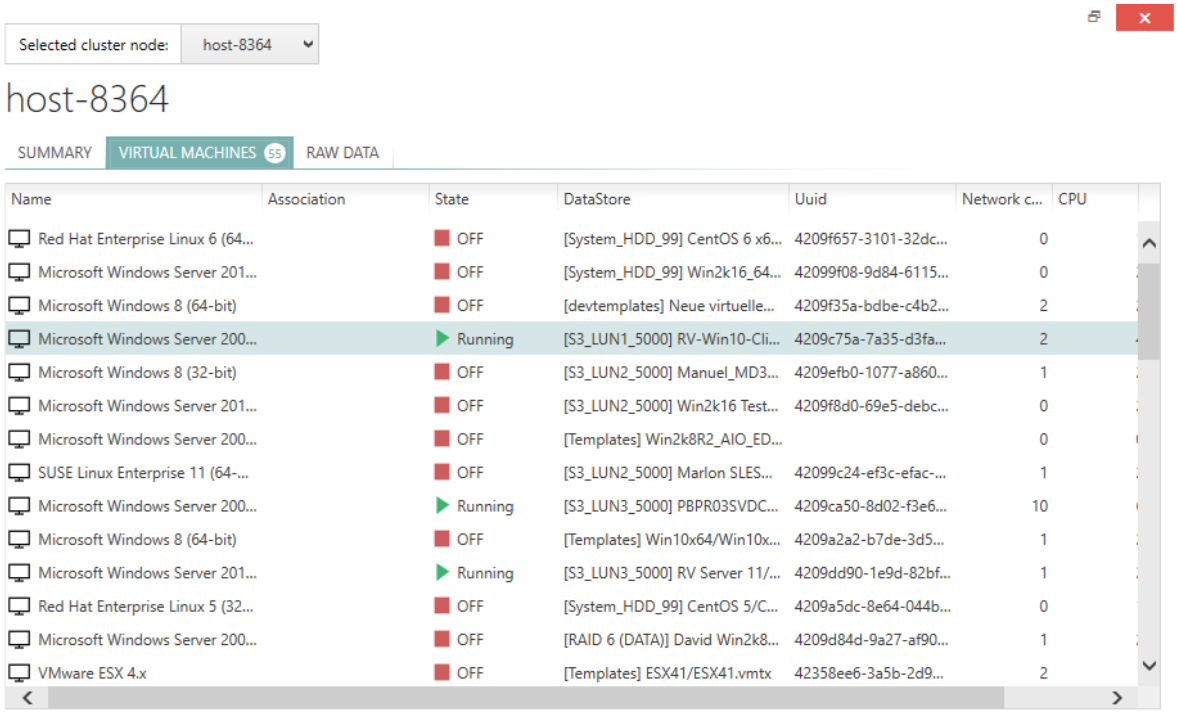
### Working with Clusters

For vSphere / ESX connections a selector of clusters is available if there is more than one cluster in the inventory data. The inventory view always shows the details of the current cluster. To change the current cluster, select its name from the drop down menu.



## Accessing Details about Guest Virtual Machines

Once in the inventory details, the tab **VIRTUAL MACHINES** can be used to view the details of hosted virtual guests:



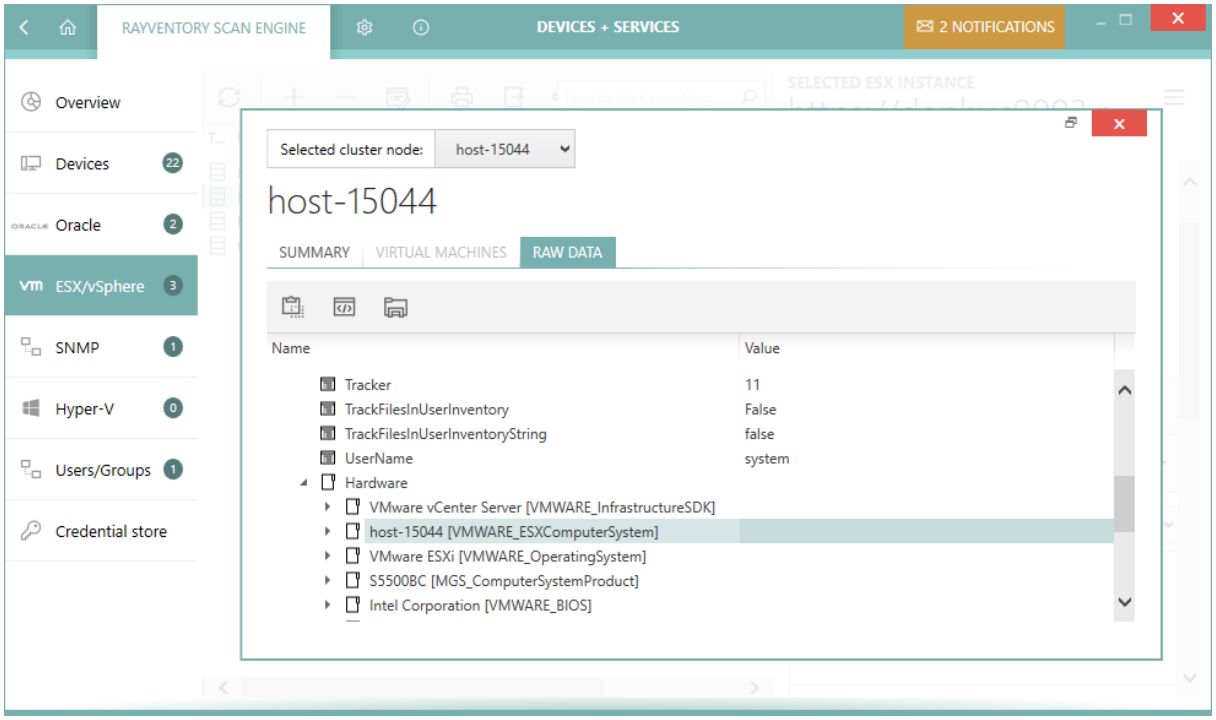

## Undocking the Inventory View

It is possible to "undock" the view to a separate window which can be freely moved and stacked with other windows. In order to do that, press the little **undock** button next to the **CLOSE** button of the inventory overlay.

**Note:**
After undocking, the window cannot be docked to the main window anymore.

## Working with Raw Data

IT professionals and administrators having experience with data structures of the RayVentory Scan Engine object domain can also work directly with the underlying inventory files (`.ndi`).




To see the raw content:

1. In the inventory overview select the last tab **RAW DATA**.
2. The tree shows the logical structure of the underlying XML data contained within the `.ndi` file.
3. It is possible to expand the trees to reach the node containing the required content.
4. The three buttons in the toolbar have the following meaning:

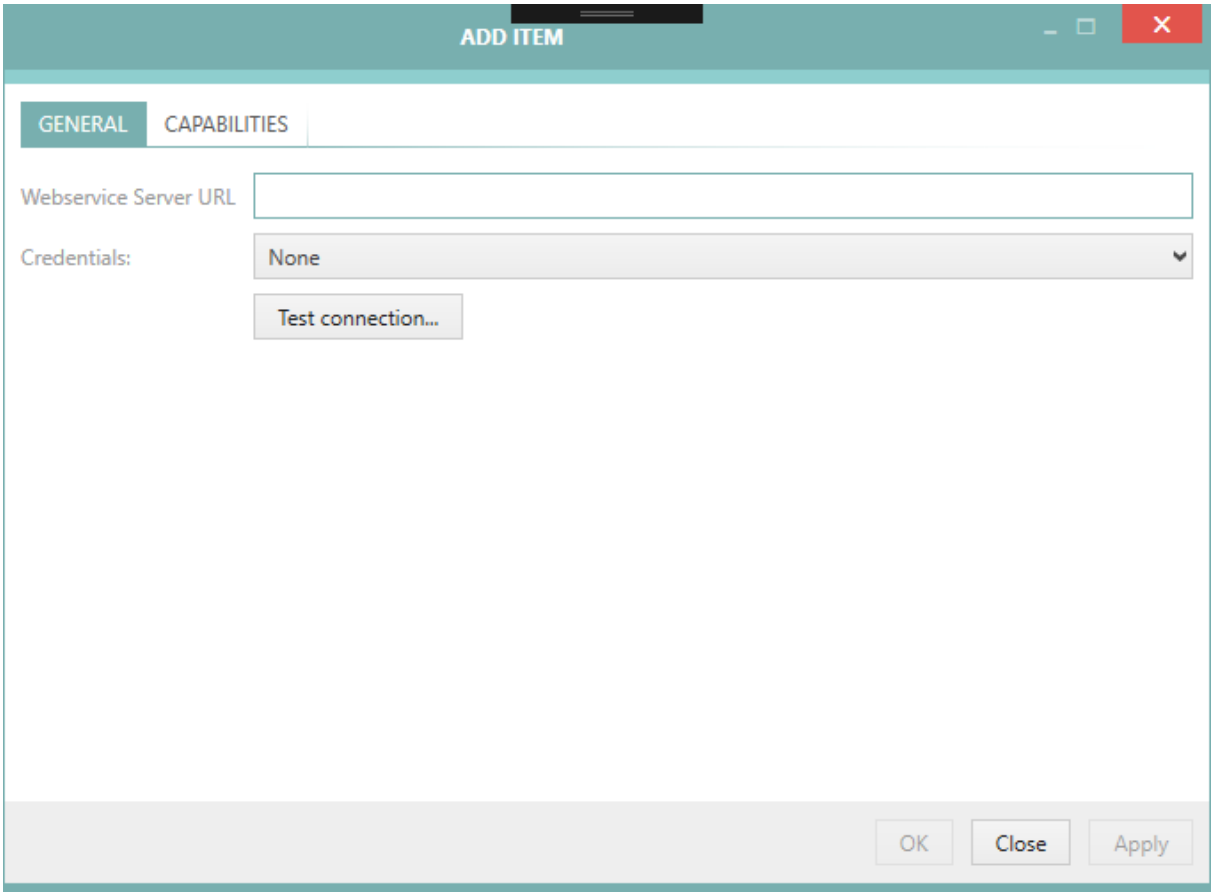
- Copy the full path to the `.ndi` file to the clipboard.
- Open the `.ndi` file in the default editor.
- Open the Windows Explorer and highlight the `.ndi` file.

**Note:**
After undocking, the window cannot be docked to the main window anymore.

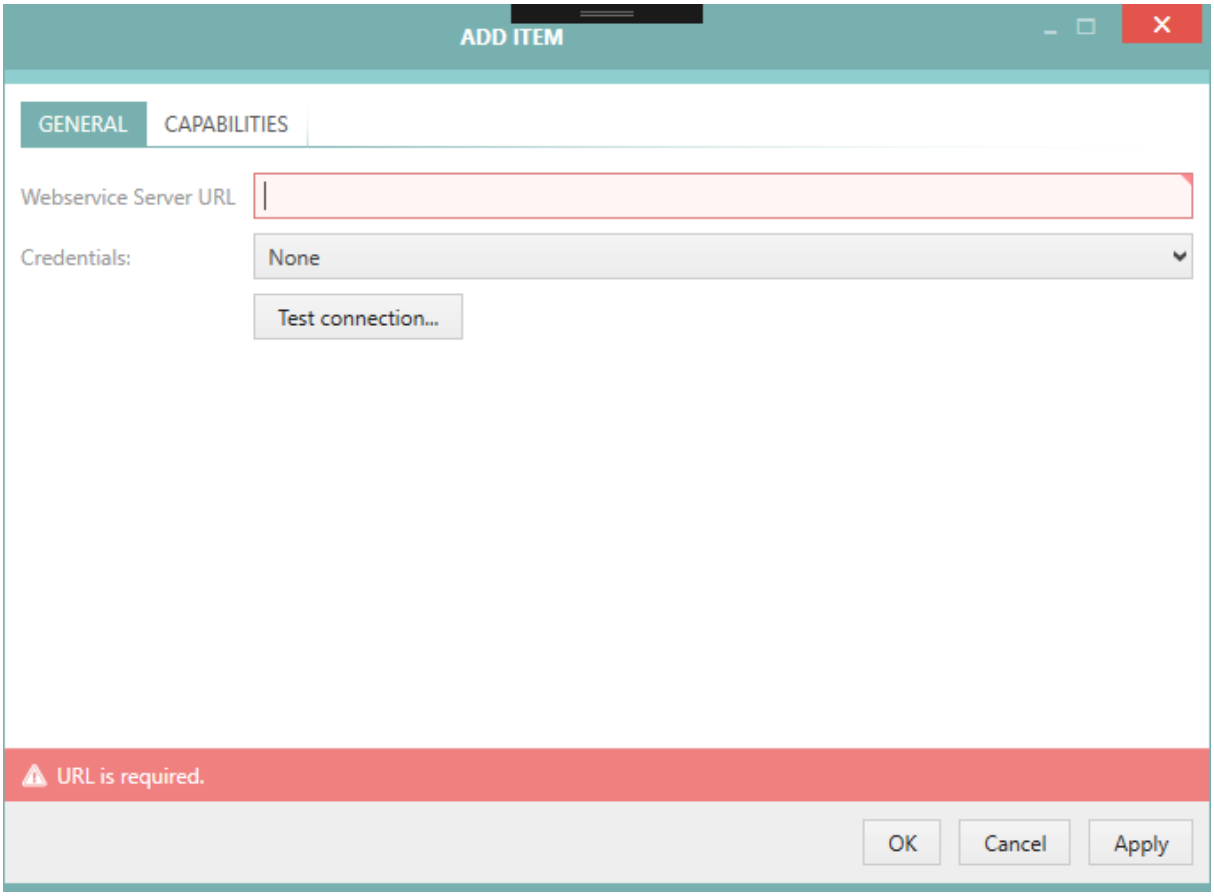
## Adding a new ESX / vSphere Connection

### In Order to Add a New vSphere Connection

1. Press the **+** button in the top toolbar or click the **Add..** menu item from the context menu.
2. A new empty dialog will be shown.




3. At least the URL address needs to be specified.
4. Optionally, the preferred credentials used by this vSphere connection can be selected. If this is left empty, RayVentory Scan Engine applies a special logic which is described in the chapter Advanced Topics.
5. In the **CAPABILITIES** tab, the capabilities of the newly added device can be limited.
6. Press **OK** to accept the changes and close the window, or **Apply** to immediately save them.
7. If any mandatory field is not specified or is in the wrong format, a validation error is shown:



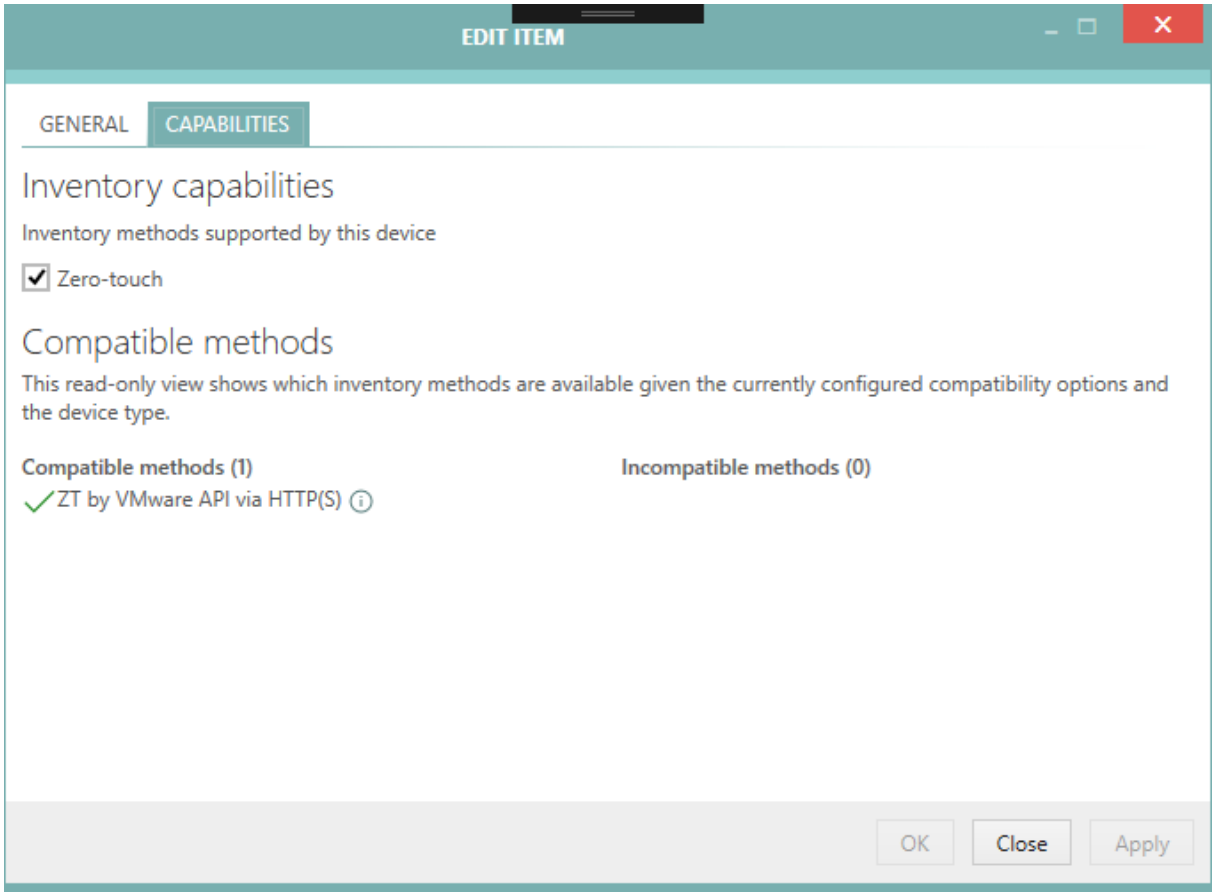

Fix the issues indicated by the red error bar and press **OK** / **Apply** to apply the changes.

**Note:**

- The URL for the connection to a vSphere / ESX SDK service endpoint usually has the following form: `https://youresxhost/sdk`.
- The connection properties dialog that is being used to create and edit a vSphere / ESX connection features a **Test connection...** button that allows to test the connection (and the credentials in the credential store) immediately.

### vSphere Connection Capabilities

The capabilities section allows users to precisely configure which low-level capabilities each device supports. Currently, only Zero-Touch execution is available for vSphere / ESX connections.




Whether Zero-Touch is enabled or not is used by the Inventory Wizard to determine whether the device can be scanned.

The tab consists of three panels:

- The selection of general inventory methods supported by this device (Zero-Touch and / or Remote-Execution).
- More low-level capabilities determining whether particular features should be supported by this device.
- The read-only view showing in real-time which inventory methods are applicable given the current state of the device.

Refer to the advanced topic Inventory Methods Overview to find out about the prerequisites and required settings.

## Preventing a Connection from Being Scanned

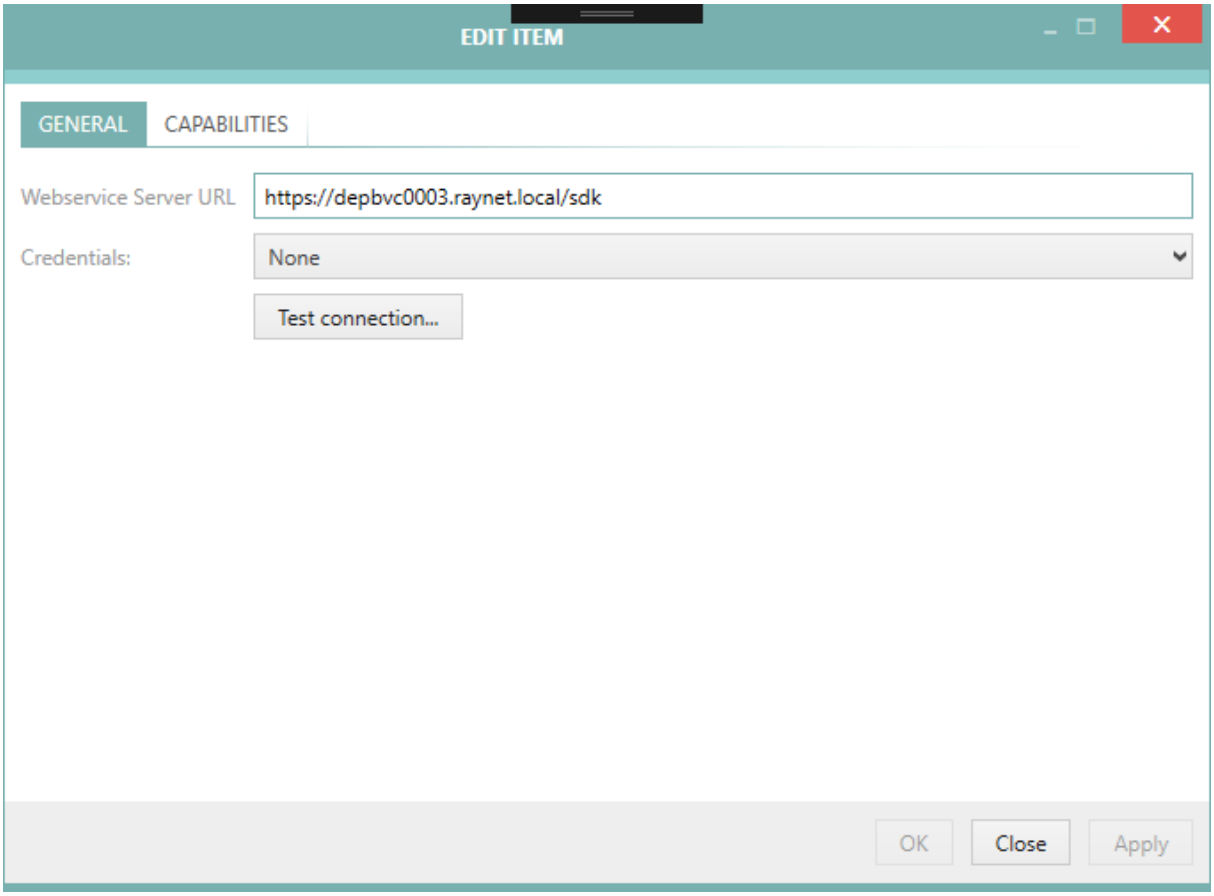
It is possible to opt-out for any further scans of the connection, thus preserving its current inventory state. To do that, disable Zero-Touch scanning. The list that is shown underneath these options should reflect this by saying that currently there is no compatible method. This effectively means, that when the user executes an inventory job (from the Inventory wizard,

PowerShell command let, or from a scheduled task), the instance is never scanned and its current inventory files and details are not affected by the new scan.

## Editing vSphere Connections

### In Order to Edit a vSphere Connection

1. Highlight an entry in the list and press the **Edit selected...** button, click the **Add..** menu item from the context menu, or press the **Edit device...** button in the sidebar.
2. A new dialog will be shown with the details of the current selection:




3. Before the connection can be saved, at least the device host name and the service name must be specified.
4. Optionally, the preferred credentials used by this database can be selected. If this is left empty, RayVentory Scan Engine applies a special logic which is described in the advanced topics.
5. In the **CAPABILITIES** tab, it is possible to limit the capabilities of the edited database connection.
6. Press **OK** to save the change and close the window or **Apply** to immediately save them.
7. If any required field is not specified or is in the wrong format, a validation error is shown. Fix the issues indicated by the red error bar and press **OK** / **Apply** to apply the changes.

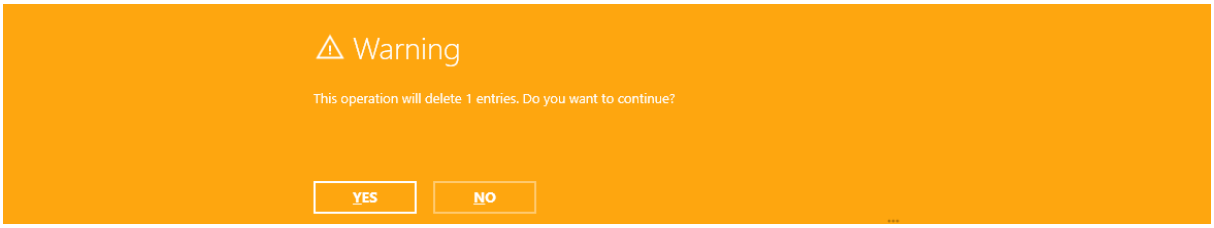
**Note:**

- The URL for the connection to a vSphere / ESX SDK service endpoint usually has the following form: `https://youresxhost/sdk`.
- The connection properties dialog that is being used to create and edit a vSphere / ESX connection features a **Test connection...** button that allows to test the connection (and the credentials in the credential store) immediately.

### Removing vSphere Connections

#### In Order to Remove a vSphere Connection

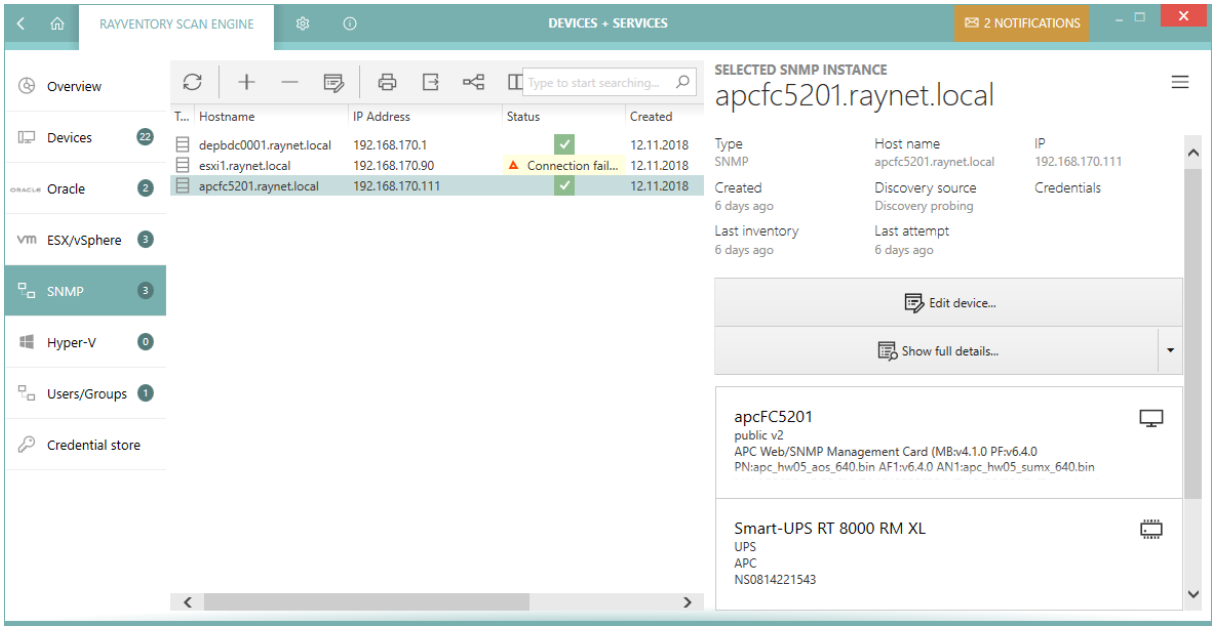
1. Highlight an entry in the list and press the **-** button or click **Remove...** menu item from the context menu.
2. Confirm the deletion by pressing **YES** in the confirmation dialog.




**Note:**
This operation is irreversible. Any existing inventory files which were assigned to the connection stay, though.

## SNMP

This view aggregates discovery and inventory data of your SNMP connections.


RAYVENTORY SCAN ENGINE
DEVICES + SERVICES
2 NOTIFICATIONS
Overview
Devices 22
Oracle 2
ESX/vSphere 3
SNMP 3
Hyper-V 0
Users/Groups 1
Credential store
Type to start searching...
T... Hostname IP Address Status Created
depbdc0001.raynet.local 192.168.170.1 12.11.2018
esxi1.raynet.local 192.168.170.90 Connection fail... 12.11.2018
apcfc5201.raynet.local 192.168.170.111 12.11.2018
SELECTED SNMP INSTANCE
apcfc5201.raynet.local
Type
SNMP
Host name
apcfc5201.raynet.local
IP
192.168.170.111
Created
6 days ago
Discovery source
Discovery probing
Credentials
Last inventory
6 days ago
Last attempt
6 days ago
Edit device...
Show full details...
apcFC5201
public v2
APC Web/SNMP Management Card (MB:v4.1.0 PF:v6.4.0
PN:apc_hw05_aos_640.bin AF1:v6.4.0 AN1:apc_hw05_sumx_640.bin
Smart-UPS RT 8000 RM XL
UPS
APC
NS0814221543

The view is divided into three parts:

- **Toolbar** - showing buttons to perform quick operations like adding, editing, and removing entries.
- **Main grid** - showing the list of all saved devices (physical and virtual).
- **Sidebar** - showing the details of the current selection including inventory data if available.

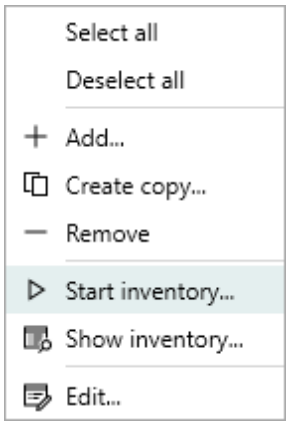
## Columns

The grid supports a predefined set of columns, only a handful of which are visible by default. It is possible to select more columns by pressing the **Column chooser** button in the grid toolbar. The following columns are available:

- **Type** - A static column with the connection icon (SNMP).
- **Hostname** - The DNS hostname of the instance.
- **IP Address** - The IP Address of the instance.
- **Status** - The inventory status. There are 3 possible values:
  - n/a - The device has not been inventoried yet.
  - OK - The device has been inventoried and returned some results.
  - In any other case, the column **Status** contains a short description of the most recent issue.
- **Last inventory attempt** - The last time the inventory has been started (whether successful or not).
- **Credentials** - The logical names of credentials used when scanning this instance.
- **Capabilities** - The short version of enumeration of allowed inventory capabilities for this instance. The string consists of two-letter tokens, with the following meaning:
  - *ZT*= Zero-Touch
  - *RE*= Remote-Execution
  - *FS*= Access to File System
  - *SMB*= Upload to SMB Shares
  - *WMI*= Windows Management Instrumentation (WMI) Queries
  - *WSM*= Windows Service Manager
- **Last inventory attempt (time)** - The last time the inventory has been started (whether successful or not) (contains full date and time).
- **Last inventory** - The last time the inventory has been successfully started (only date part).
- **Last inventory (time)** - The last time the inventory has been successfully started (contains full date and time).
- **Last successful inventory method** - The name of the last inventory method that worked for this instance. This method will be preferred in future scans.
- **Last failed inventory methods** - The names of inventory methods that failed the last time the instance was scanned.
- **Last failed inventory method (details)** - Details about the methods that failed when the instance was scanned for the last time.
- **Created** - The date when the connection was created or imported.

## Context Menu

Pressing Right-Mouse-Button after highlighting an item opens a context menu for it.




- **Select all** - Selects all visible entries in the grid.
- **Deselect all** - Deselects all visible entries in the grid.
- **Add...** - Opens the New SNMPDialog.
- **Create copy...** - Opens the New SNMPDialog where the default values are automatically set to the values from the current selection.
- **Remove** - Removes the currently selected devices (see Removing SNMP connections).
- **Start inventory...** - Opens the Inventory Wizard on the currently selected devices.
- **Show inventory** - Shows the details inventory (only available if an inventory has been already performed).
- **Edit...** - Opens the Edit SNMP connection connection.

**Note:**
Some options may be conditionally hidden or disabled (for example the menu item **Edit...** is disabled if more than one device is selected).

## Recent Scan Details

The results of the most recent scan are shown here. If the last scan was not successful, the last successful scan is also shown and can be selected from the drop-down.

For more information about working with Inventory log, refer to the following chapter: Recent Scan Details.

## Viewing Inventory Details

Once a device has been successfully scanned for its software and hardware, a button will be shown in the sidebar allowing the user to show the content of specified machine:


The button can be used to open a detailed overview of the machine software and hardware. For convenience, the summary of machine assets is also shown directly in the sidebar.

The details inventory overview contains several tabs which are contextually sensitive and display various information depending on the connection type.

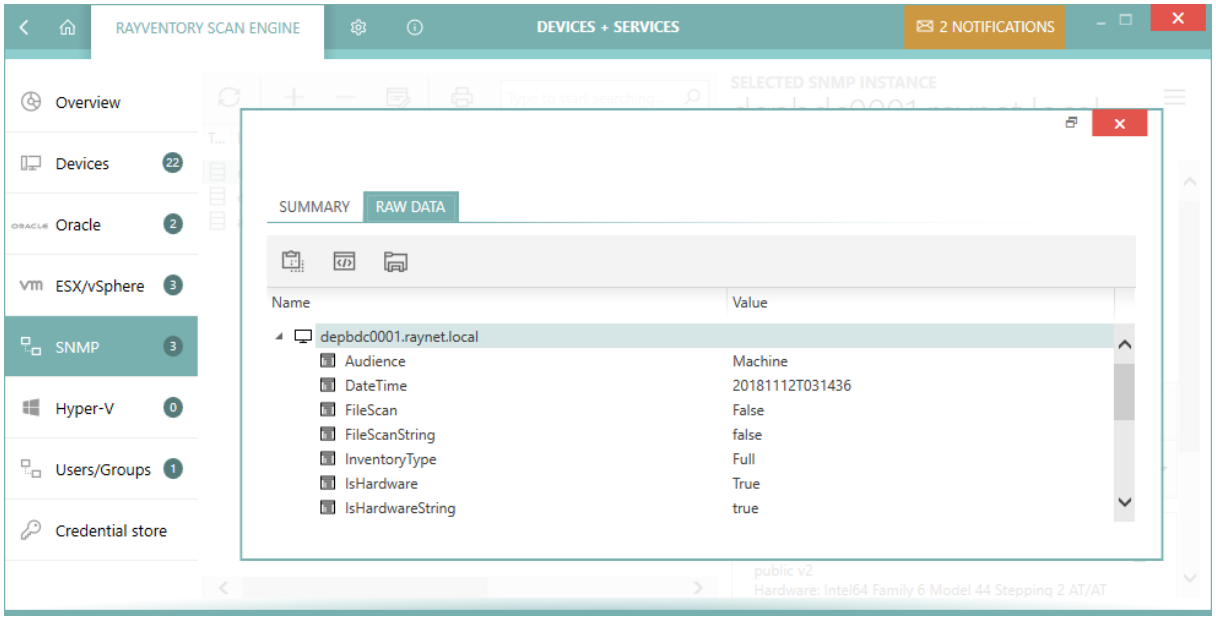
## Undocking the Inventory View

It is possible to "undock" the view to a separate window which can be freely moved and stacked with other windows. In order to do that, press the little **undock** button next to the **CLOSE** button of the inventory overlay.

**Note:**
After undocking, the window cannot be docked to the main window anymore.

## Working with Raw Data

IT professionals and administrators having experience with the data structures of the RayVentory Scan Engine object domain can also work directly with underlying inventory files (.ndi).




To see the raw content:

1. In the inventory overview select the last tab **RAW DATA**.
2. The tree shows the logical structure of the underlying XML data contained within the `.ndi` file.
3. The trees can be expanded to reach the node containing the required content.
4. The three buttons in the toolbar have the following meaning:

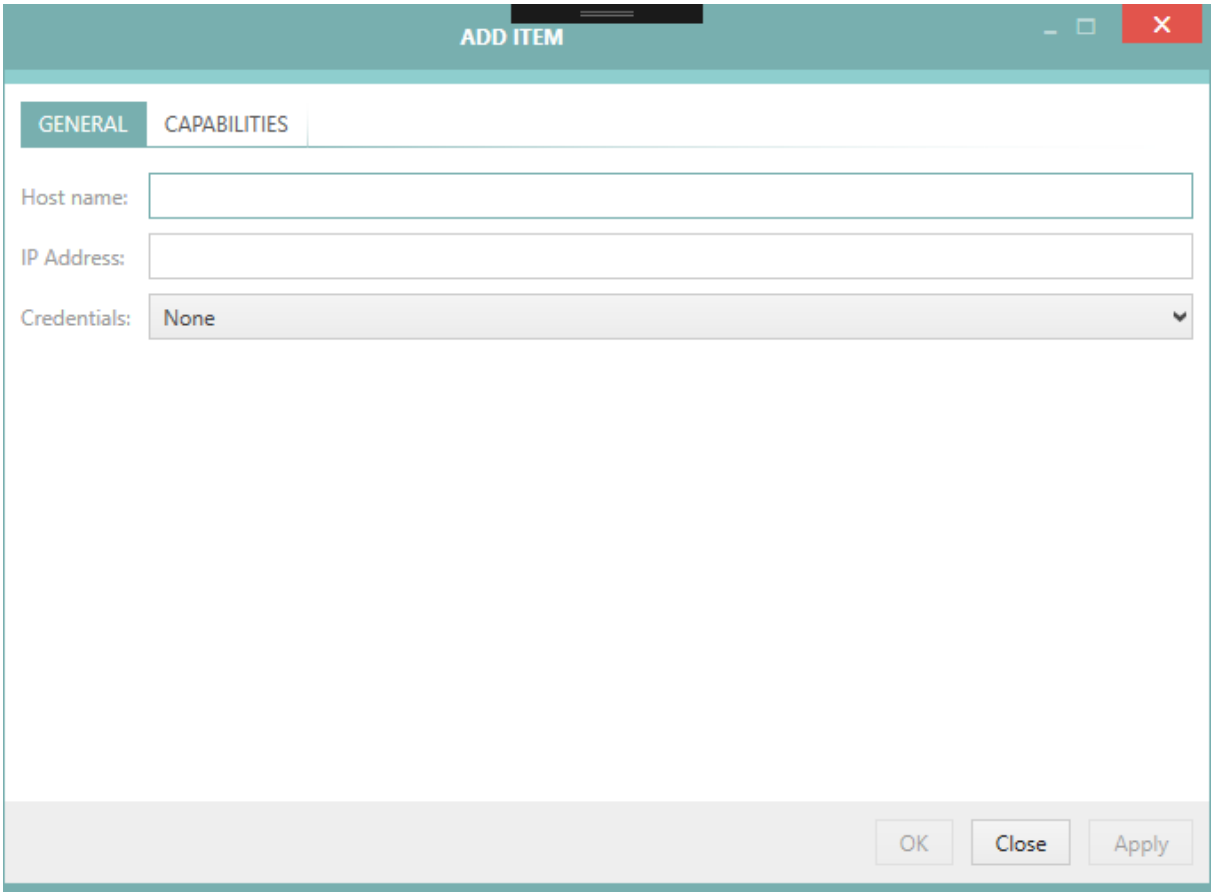
- Copy the full path to the `.ndi` file to the clipboard.
- Open the `.ndi` file in the default editor.
- Open Windows Explorer and highlight the `.ndi` file.

**Note:**
After undocking, the window cannot be docked to the main window anymore..

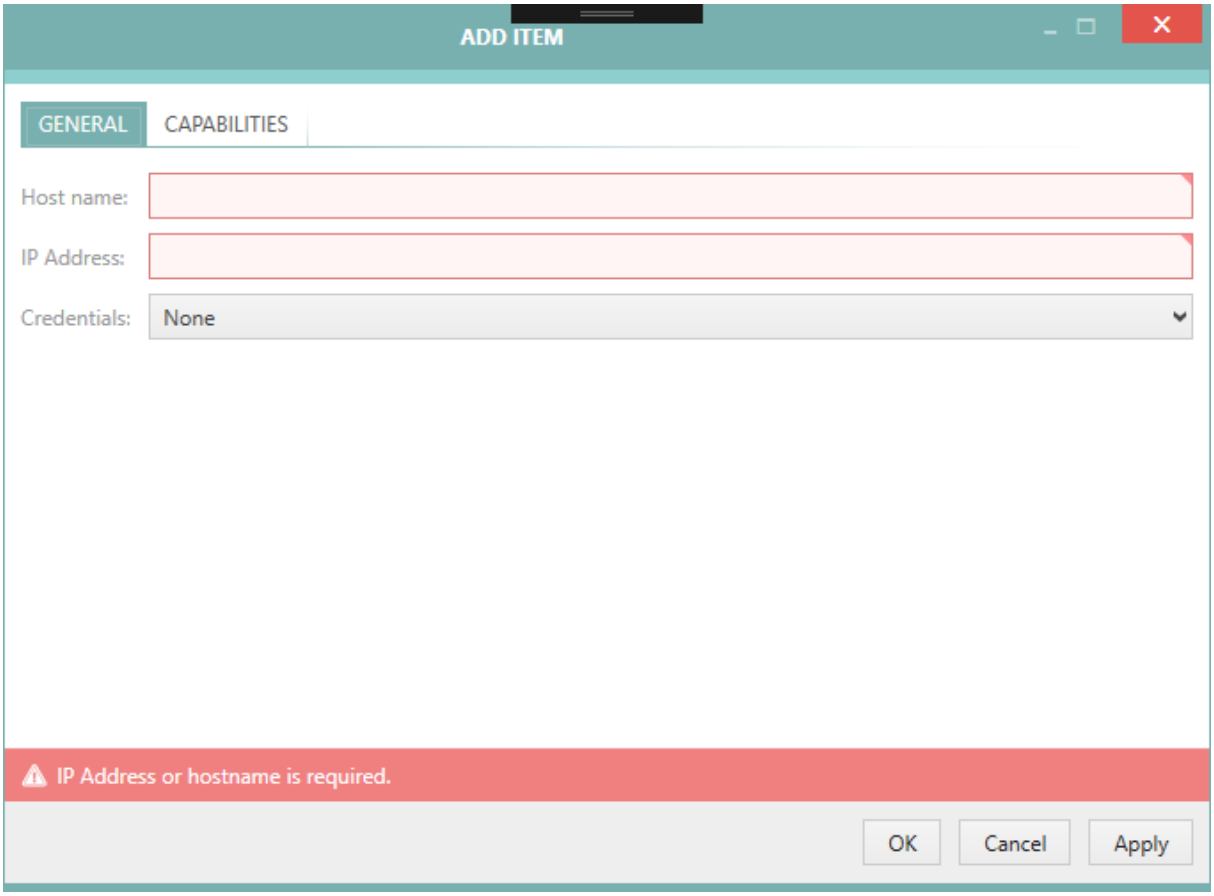
## Adding a New SNMP Connection

### In Order to Add a New SNMP Connection

1. Press the **+** button in the top toolbar or click the **Add..** menu item from the context menu.
2. A new empty dialog will be shown.



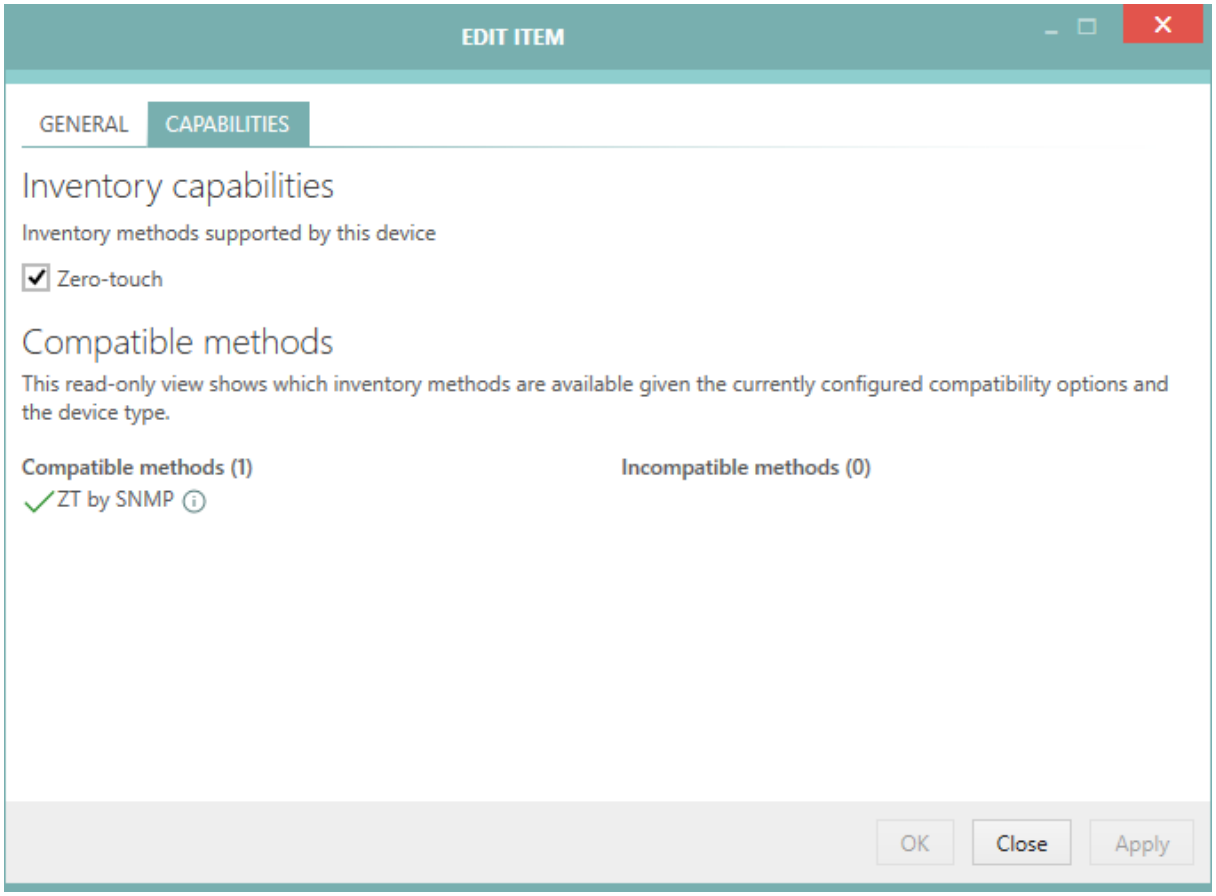

3. At least, the Host name and / or the IP Address need to be specified.
4. Optionally, the preferred credentials used by this vSphere connection can be selected. If this is left empty, RayVentory Scan Engine applies a special logic which is described in the chapter Advanced Topics.
5. In the **CAPABILITIES** tab, it is possible to limit the capabilities of the newly added device.
6. Press **OK** to accept the changes and close the window, or **Apply** to immediately save them.
7. If any required field is not specified or is in the wrong format, a validation error is shown:




Fix the issues indicated by the red error bar and press **OK** / **Apply** to apply the changes.

### SNMP Connection Capabilities

The capabilities section allows users to precisely configure which low-level capabilities each device supports. Currently, only Zero-Touch execution is available for SNMP connections.




Whether Zero-Touch is enabled or not is used by the Inventory Wizard to determine if the device can be scanned.

The tab consists of three panels:

- The selection of general inventory methods supported by this device (Zero-Touch and / or Remote-Execution).
- More low-level capabilities, determining whether particular features should be supported by this device.
- The read-only view showing in real-time which inventory methods are applicable given the current state of the device.

Refer to advanced topic Inventory Methods Overview to find out about the prerequisites and required settings.

## Preventing a Connection from Being Scanned

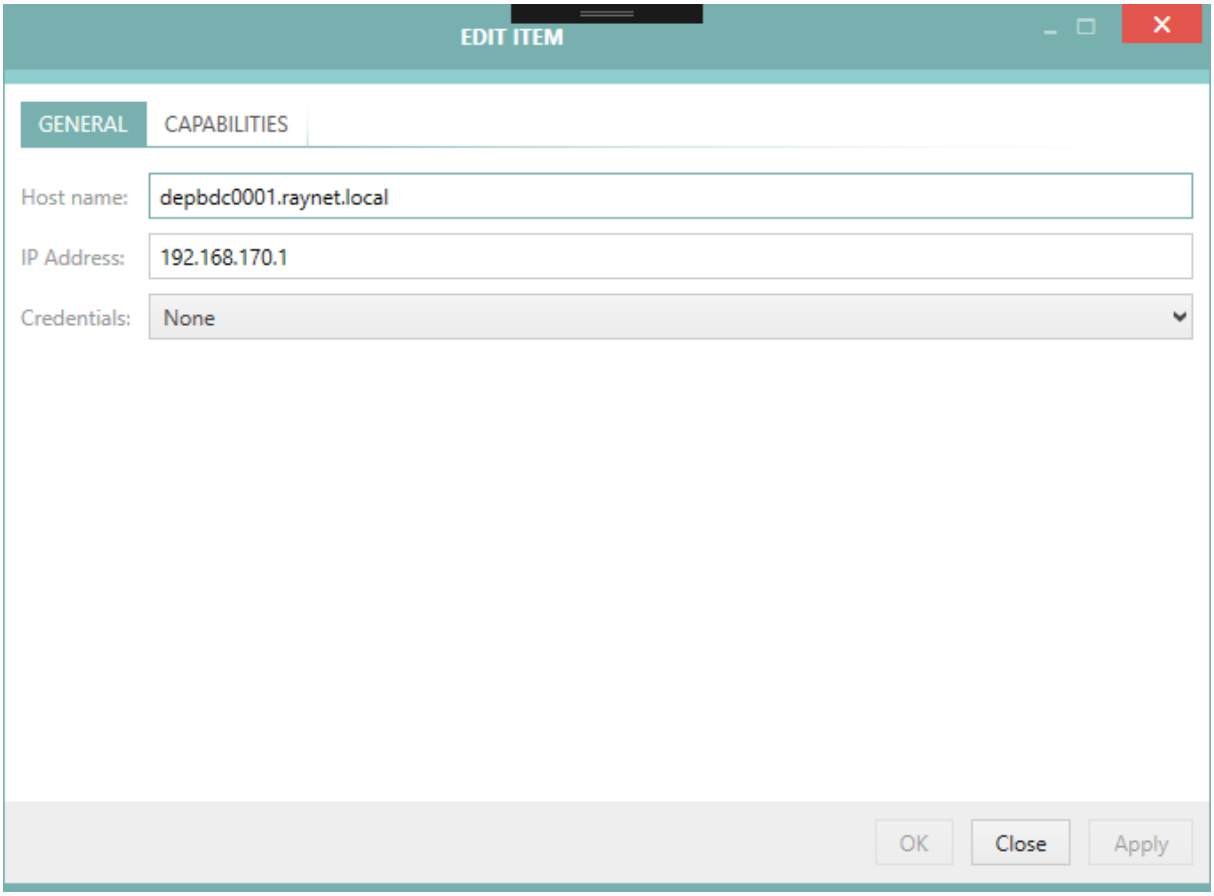
It is possible to opt-out for any further scans of the connection, thus preserving its current inventory state. To do that, disable Zero-Touch scanning. The list that is shown underneath these options should reflect this by saying that currently there is no compatible method. This effectively means, that when the user executes an inventory job (from the Inventory wizard,

PowerShell command let, or from a scheduled task), the instance is never scanned and its current inventory files and details are not affected by the new scan.

## Editing SNMP Connections

### In Order to Edit an SNMP Connection

1. Highlight an entry in the list and press the **Edit selected...** button, click the **Add..** menu item from the context menu, or press the **Edit device...** button in the sidebar.
2. A new dialog will be shown with the details of the current selection:

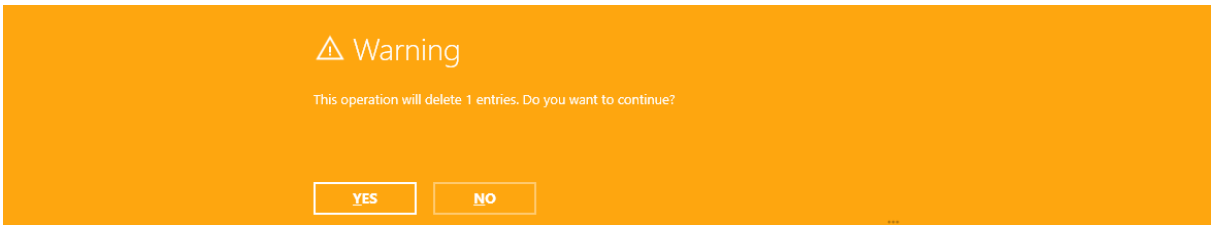



3. Before the connection can be saved, at least the device host name and the service name must be specified.
4. Optionally, the preferred credentials used by this database can be selected. If this is left empty, RayVentory Scan Engine applies a special logic which is described in the advanced topics.
5. In the **CAPABILITIES** tab, it is possible to limit the capabilities of the edited database connection.
6. Press **OK** to save the change and close the window or **Apply** to immediately save them.
7. If any required field is not specified or is in the wrong format, a validation error is shown. Fix the issues indicated by the red error bar and press **OK** / **Apply** to apply the changes.

## Removing SNMP Connections

### In Order to Remove an SNMP Connection

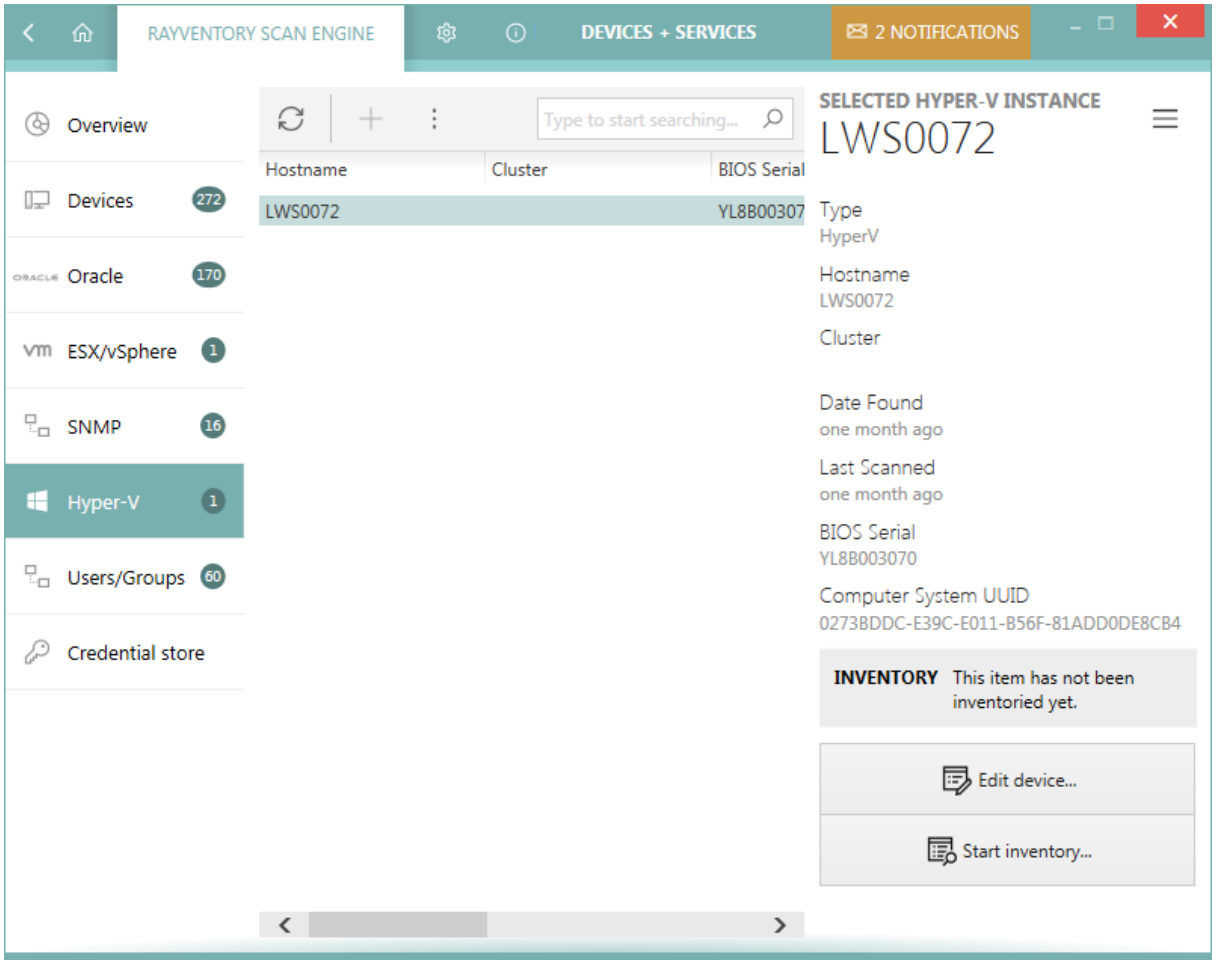
1. Highlight an entry in the list and press the **-** button or click the **Remove...** menu item from the context menu.
2. Confirm the deletion by pressing **YES** in the confirmation dialog




**Note:**
This operation is irreversible. Any existing inventory files which were assigned to the connection stay, though.

# Hyper-V

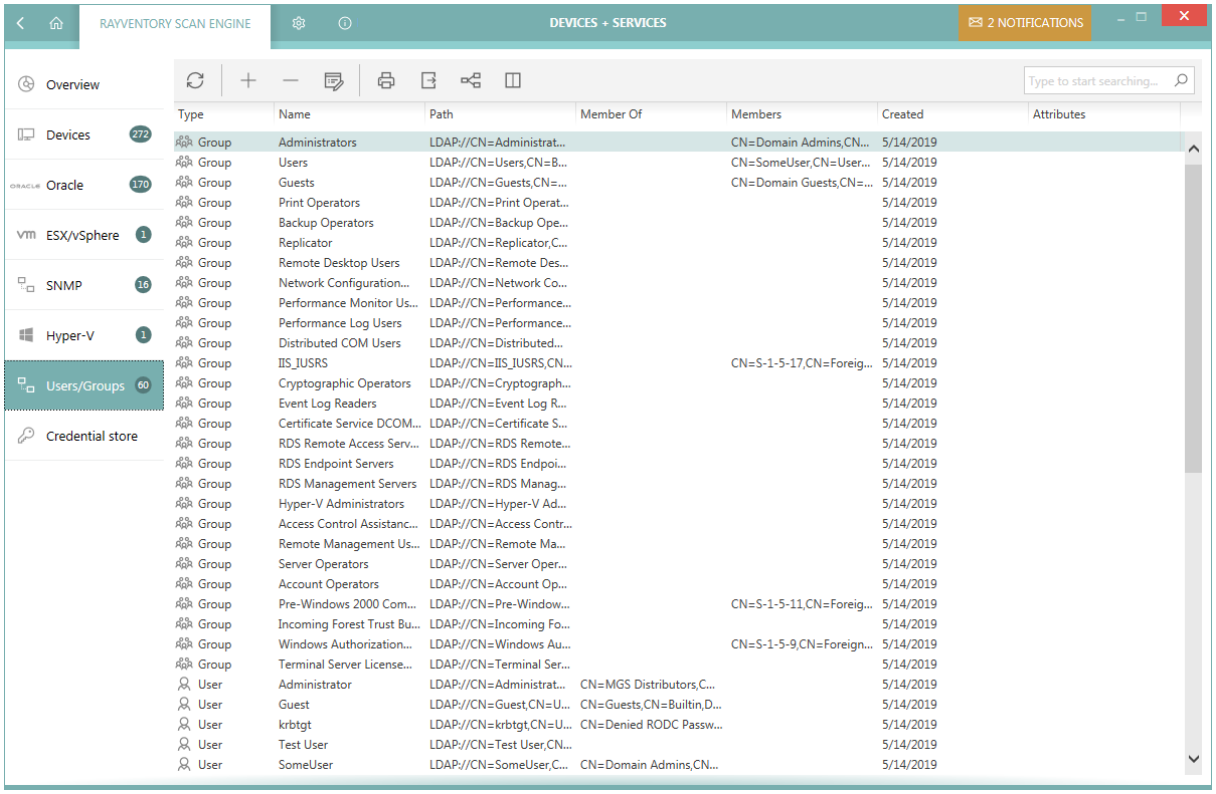
This view aggregates discovery and inventory data of existing Hyper-V connections.




This screen will be populated automatically. Therefore, it is not possible to add, edit, or remove any entries. This screen only shows the data that has been gathered from the OS related inventory data. The data will be present in RayVentory Server after an OS inventory import no matter if an explicit Hyper-V inventory has been executed or not.

# Users / Groups

This view lists all imported **Users** and **Groups**.




All **Users** and all **Groups** imported from the **Active Directory** are shown here. It is possible to add, remove, and edit entries. Two optional text files which can be used to customize the Active Directory Users and Groups features are available in RayVentory Scan Engine.

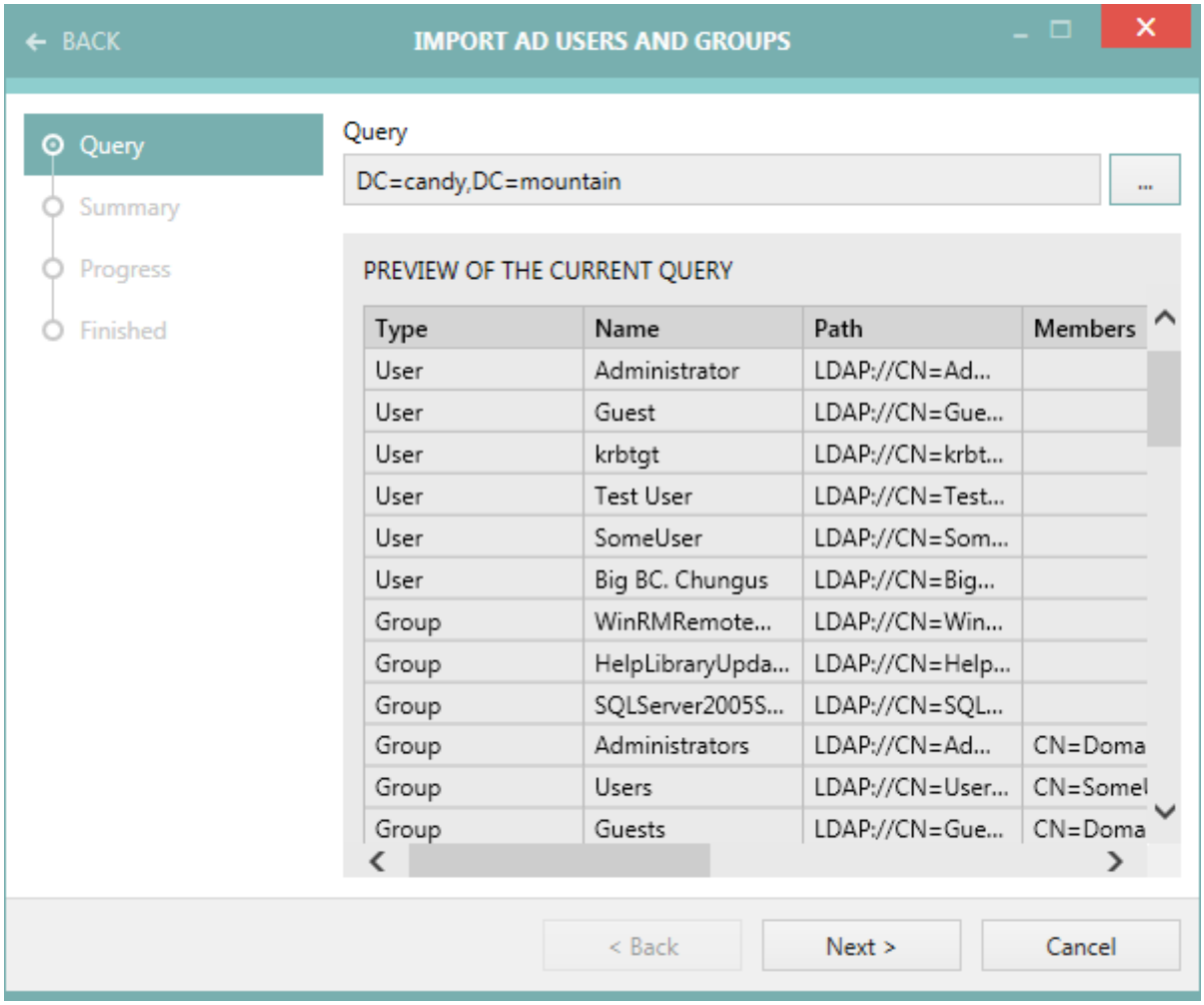
It is possible to import and view additional attributes from the Active Directory by creating the optional files `ADImportUserAttributes.lst` and `ADImportGroupAttributes.lst` and adding the attribute names to the files (one per line).

The following links contain lists of attribute names that can be used for the attribute file lists:

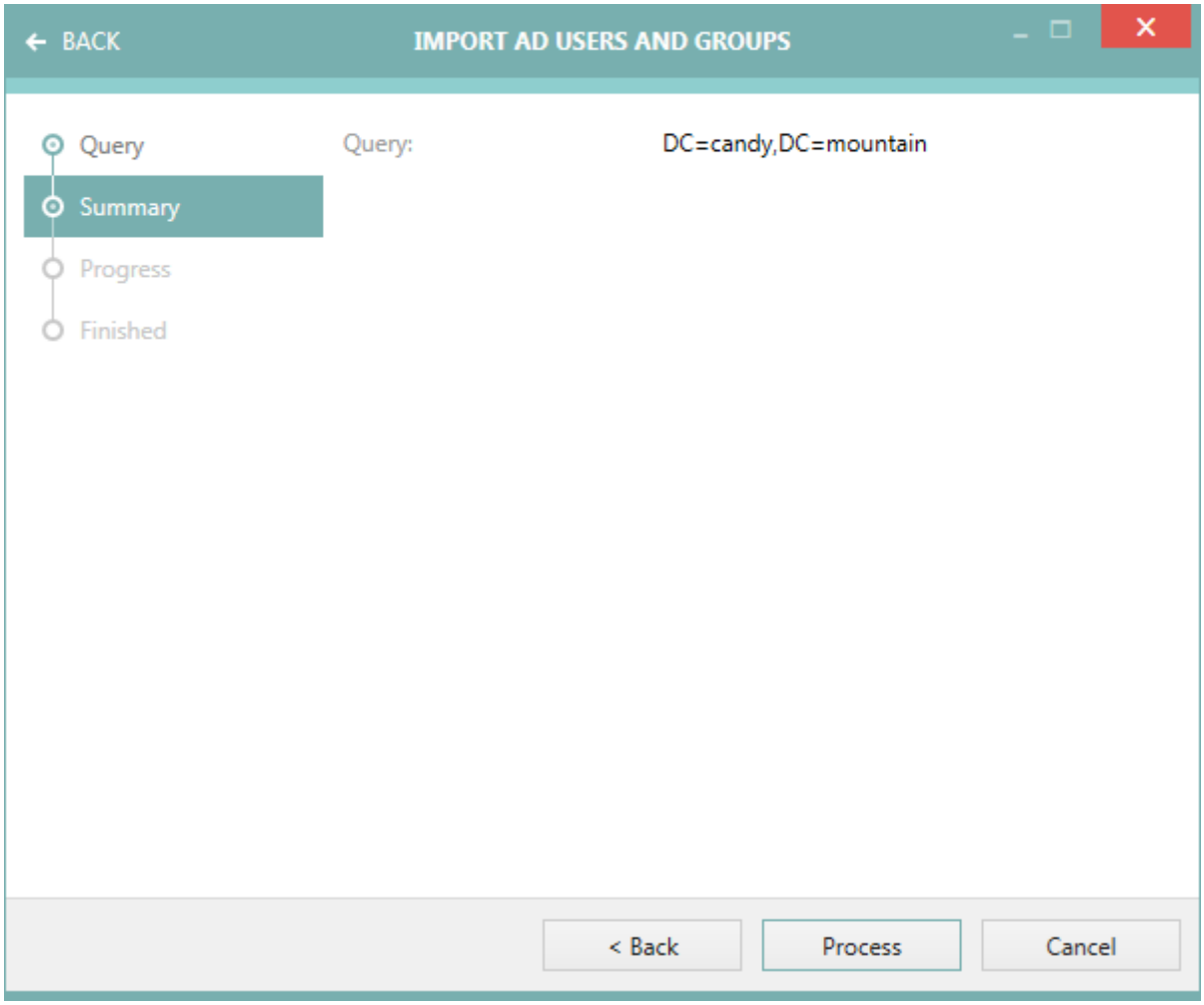
- https://docs.microsoft.com/en-us/windows/desktop/adschema/c-user#windows-2000-server-attributes
- https://docs.microsoft.com/en-us/windows/desktop/adschema/c-group#windows-2000-server-attributes

## Import AD Users and Groups Wizard

The **Import AD Users and Groups** wizard will show a preview of the data for the current query.



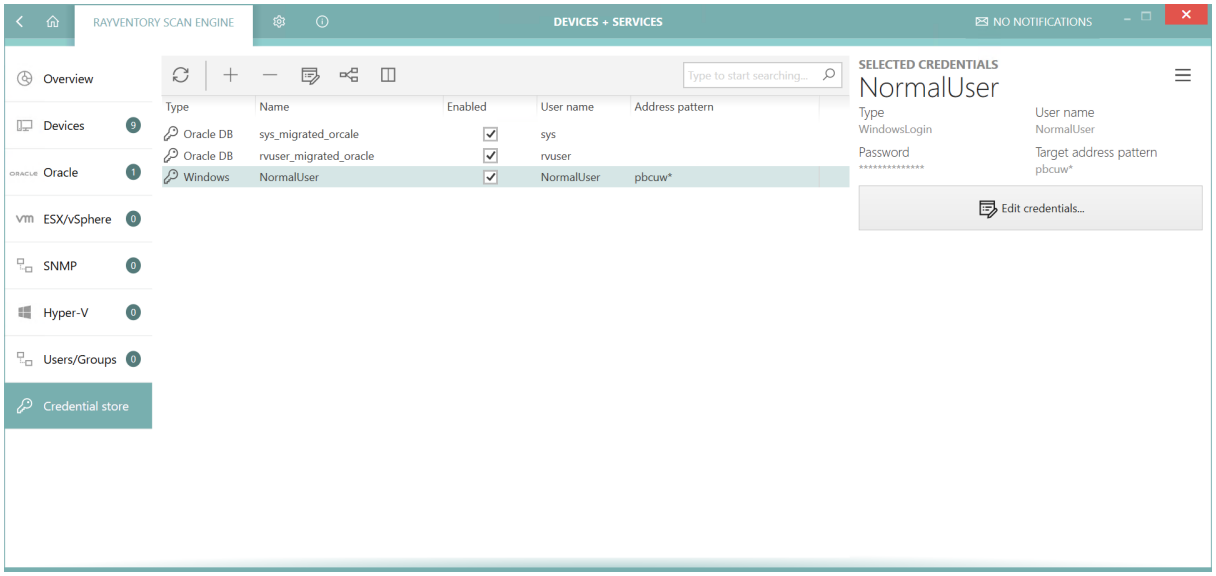

To create a query from the Active Directory browser, click on the **Browse** button [**...**]. To continue to the next step, click on the **Next >** button.




In the **Summary** step of the wizard the query will be shown once more. Click on the **Process** button to execute the query.

# Credential Store

In the credential store all credentials that are used by RayVentory Scan Engine are consolidated. The credentials are stored in an encrypted form. By default, the store uses its default encryption key. It is possible to use the custom encryption key as described in the **Settings** > **General** section.




The credential store distinguishes between different types of credentials:

- Windows
- SSH
- Oracle
- vSphere / ESX
- SNMP

Different types of credentials are used for different inventory operations and the Windows credentials are additionally used for the upload operation.

All credentials have a logical name that is used in the drop-down menus which are available in the connection properties dialogs. The drop-down menu in the **Credentials** field which is present in all connection properties dialogs is used to configure the preferred credentials for a connection. This means that the selected credentials are the first ones that are being tested. If no preferred credentials have been defined, the credentials are tested in the order in which they are shown in the credential store screen and filtered by matching the optional **Target Address Pattern** field to all credentials.

All credentials have a **Target Address Pattern** field. This field is optional and it is used to filter the credentials for the usage with a certain host or set of hosts. If the field is empty, then the credentials are applicable for all hosts. The field can be set to a specific hostname or address and will then be applied to this hostname or address only. It is also possible to use a regular expression for the **Target Address Pattern** in order to match multiple hosts that fit this expression.

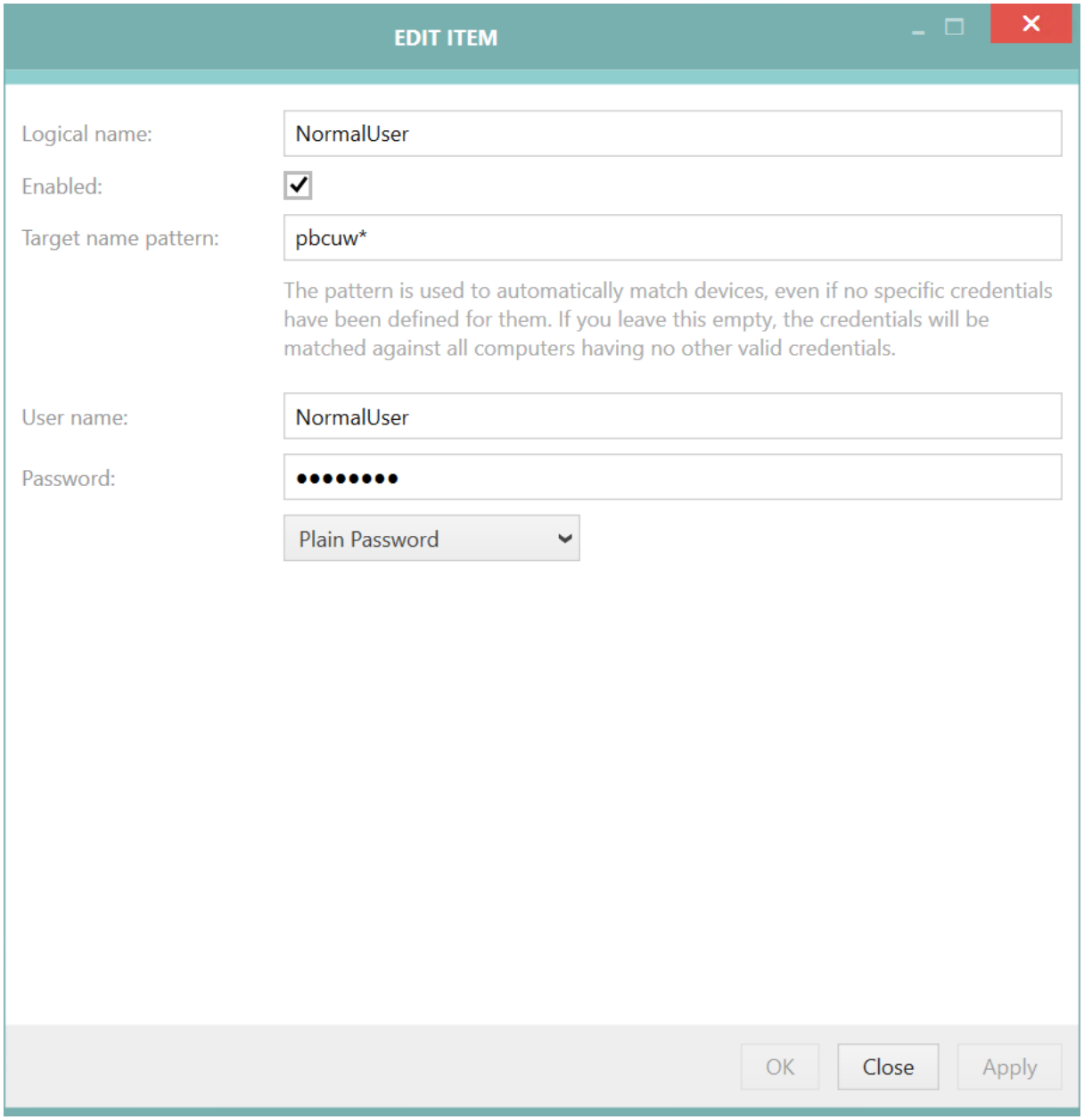
It is possible to disable credentials without completely deleting it by clicking on **Edit credentials...** and unchecking the **Enabled** checkbox.

## Credential Type Windows

When running an inventory on a remote Windows host, the Windows credential are used for authentication.

**Note:**
There is an option to use a Windows type login for authentication for the upload operation.

Windows credentials consist of a user name and a password. The user name might include a domain name (NT domain name). An example for this is `mydomain/myusername`.



## Credential Type SSH

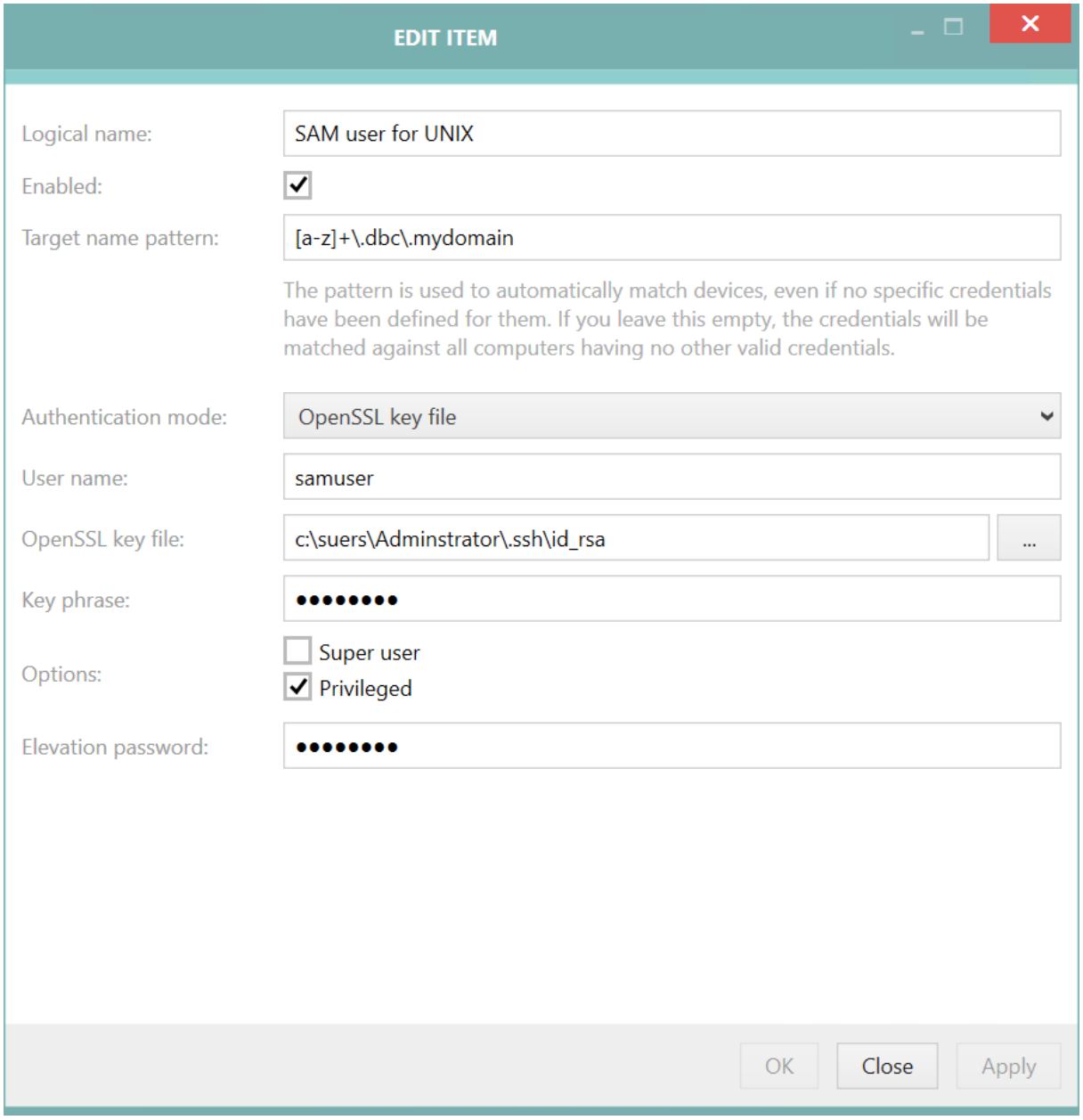
The SSH credentials are used for a remote inventory on Linux and unix-like platforms.

The credential store offers three different types of credentials for SSH authentication:

- user name / password
- user name / private key
- user name / private key file

A private key or a private key file can be secured by a passphrase. A passphrase can be set in the **Authentication Key Passphrase** field. The passphrase is stored in the credential store in an encrypted form. A private key without a passphrase is stored in the credential store in an encrypted form.

The SSH credentials include the information on how the elevation of privileges is done or if the user specified for credentials is assumed to be a super user on the target host. It is possible to specify a password for the elevation of privileges for targets where, for example, sudo would prompt for a password when using it to run a command with elevated privileges.

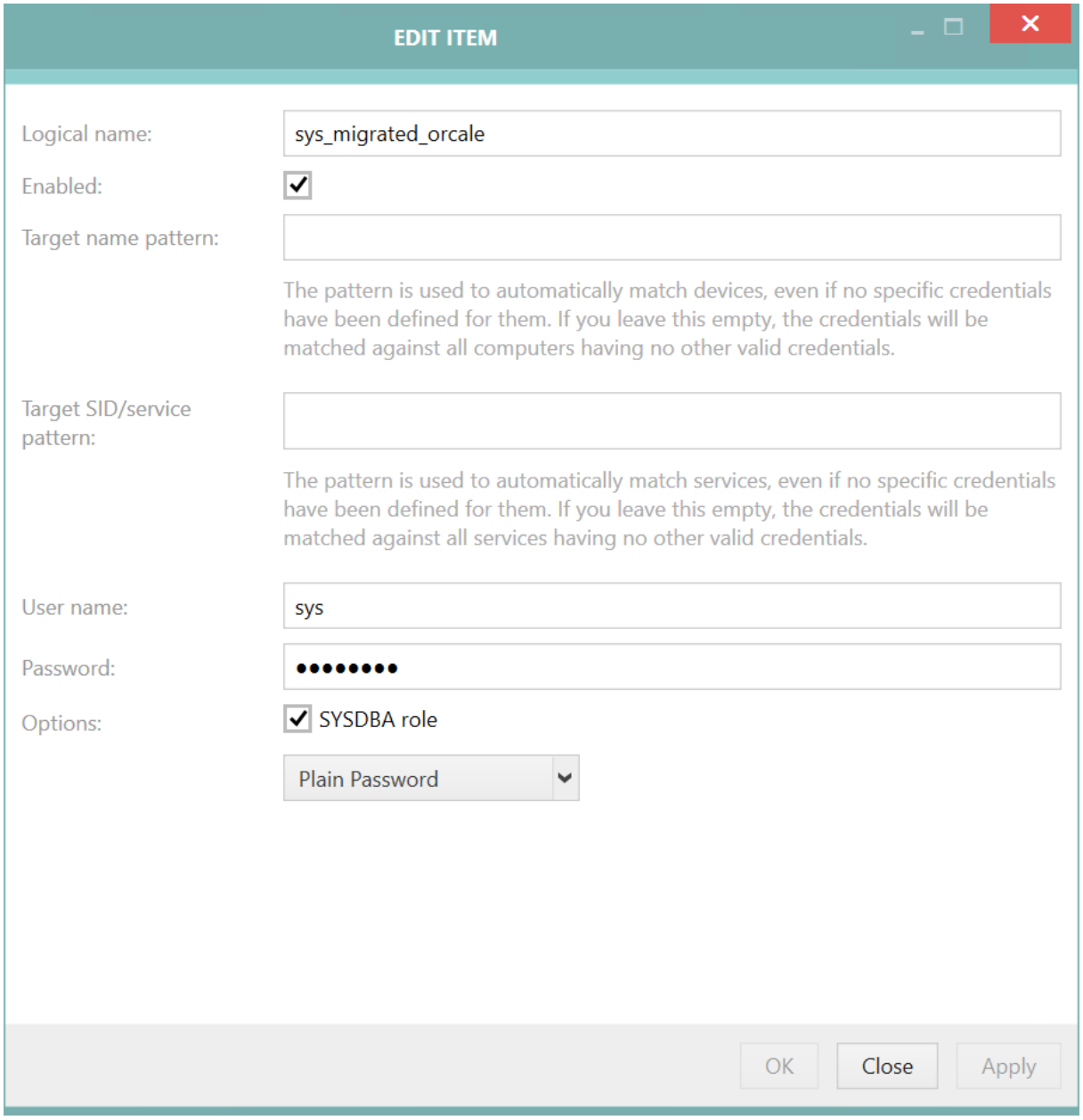



## Credential Type Oracle

The Oracle credentials are used for inventory operations on Oracle databases and are for using the Review Lite Script or DFUS script support.

The credentials consist of a user name / password pair. The password is optional. For special logins like the build-in user SYS the flag SYSDBA role can be set.

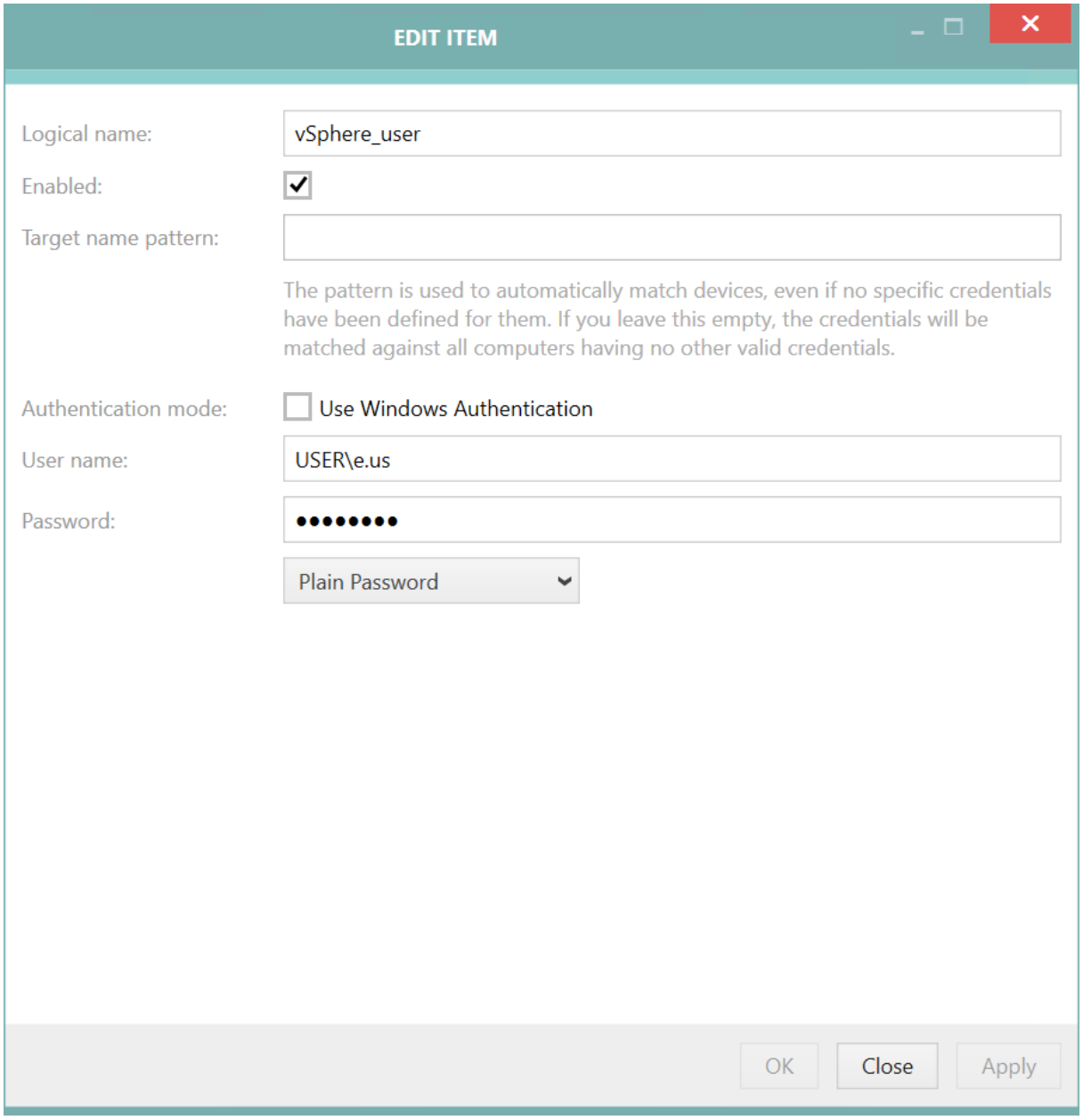
In addition to the Target Address Pattern, the DB credentials have another optional field. The **Target SID/Service Pattern** field which matches with the server name or the SID. Just like the Target Address Pattern, this field may stay empty, a concrete SID / service name, or a regular expression which matches multiple SIDs / service names can be enter.




## Credential Type VMware

The VMware credentials are used for authentication when running an inventory on a VMware infrastructure.

A set of VMware credentials consists of a user name and a password. In case that the implementation involves a Windows Domain Controller, the user name may include a domain name (NT domain name). An example for this is `mydomain/myusername`. User names which are local to a VMware infrastructure should work without any prefix or suffix like username@esxhost.

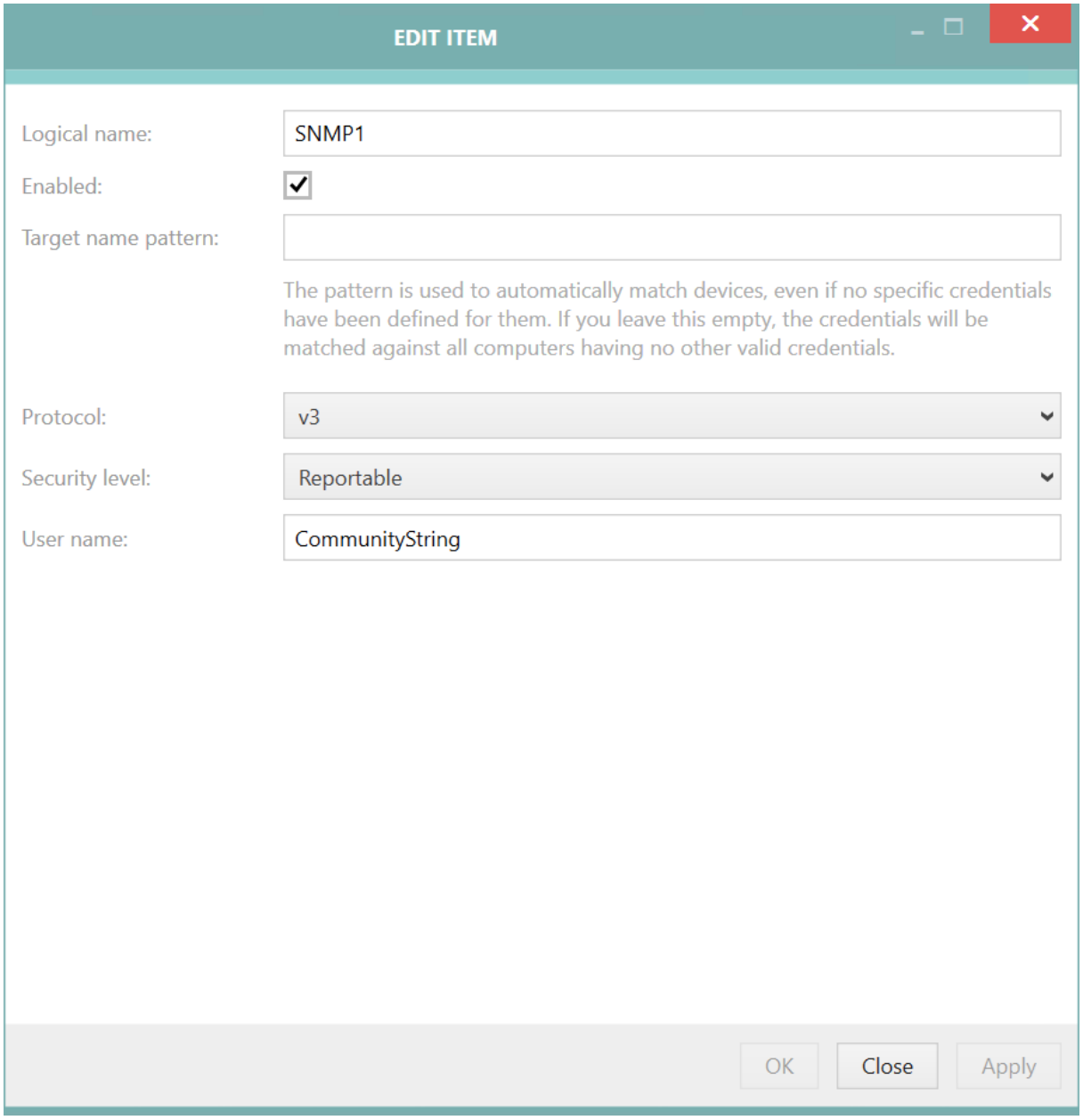



It is possible to set the **Use Windows session credentials** flag instead of entering a username and password if all the following conditions are met:

- It is an implementation that allows for authentication to be delegated to a Windows Domain Controller.
- The RayVentory host is part of the domain.
- The user that runs RayVentory Scan Engine and its scheduling service is a domain user.

## Credential Type SNMP

The SNMP credentials are used for authentication when running an inventory on an SNMP connection.




## Adding New Credentials

To add new credentials press the **+** button in the toolbar or right click the credentials grid and choose **Add...** from its context menu (visible after pressing **Right Mouse Button**).

RayVentory Scan Engine uses a wizard-based approach guiding you through all steps and ensuring the correctness of gathered information.

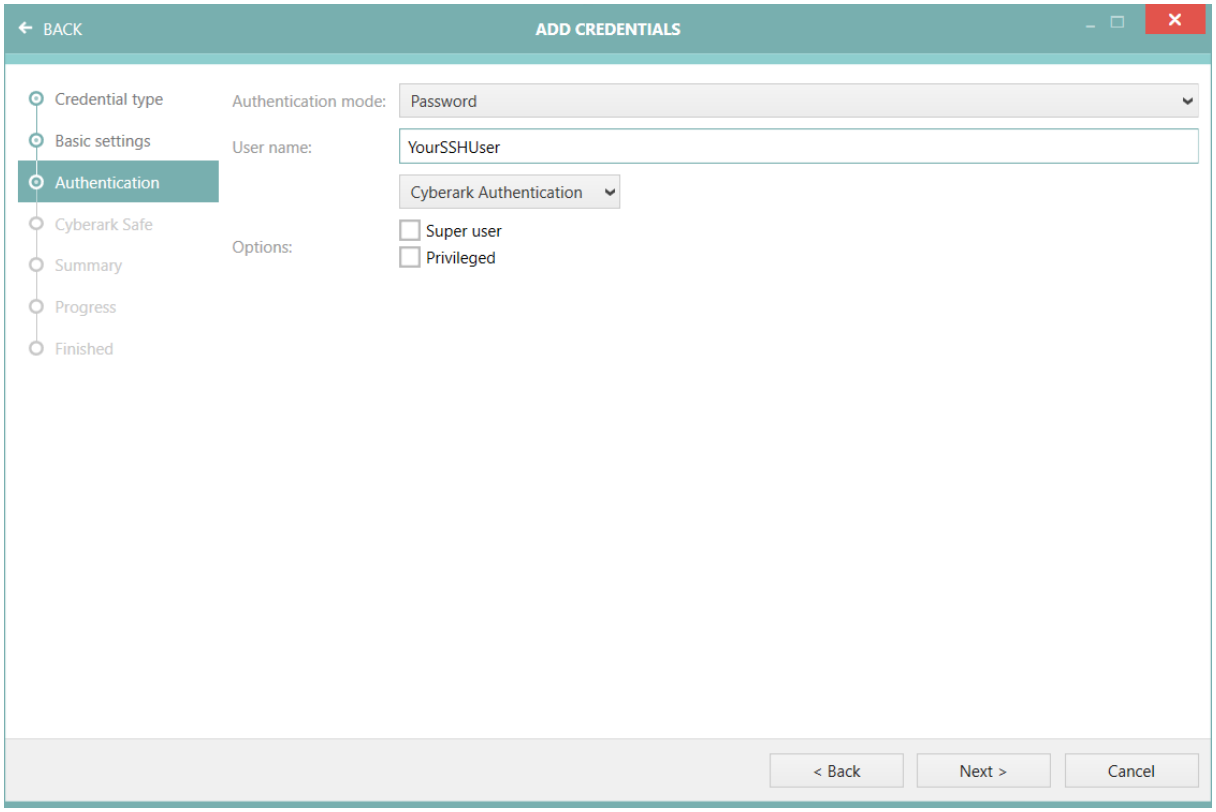
## Authentication Using CyberArk

This feature is using a service for the handling and the creation of credentials that is part of the enterprise solution of CyberArk. The service is used to create username/password pairs that can be used with the RayVentory Scan Engine inventory methods. When using this service, credentials in the RayVentory Scan Engine Credential Manager are saved with additional information for the usage with the service. The data in the Credential Manager is used to authenticate against the CyberArk service and to specify from where the credentials will be requested. When executing the inventory method, the credentials that are needed for authorization will be requested just in time.

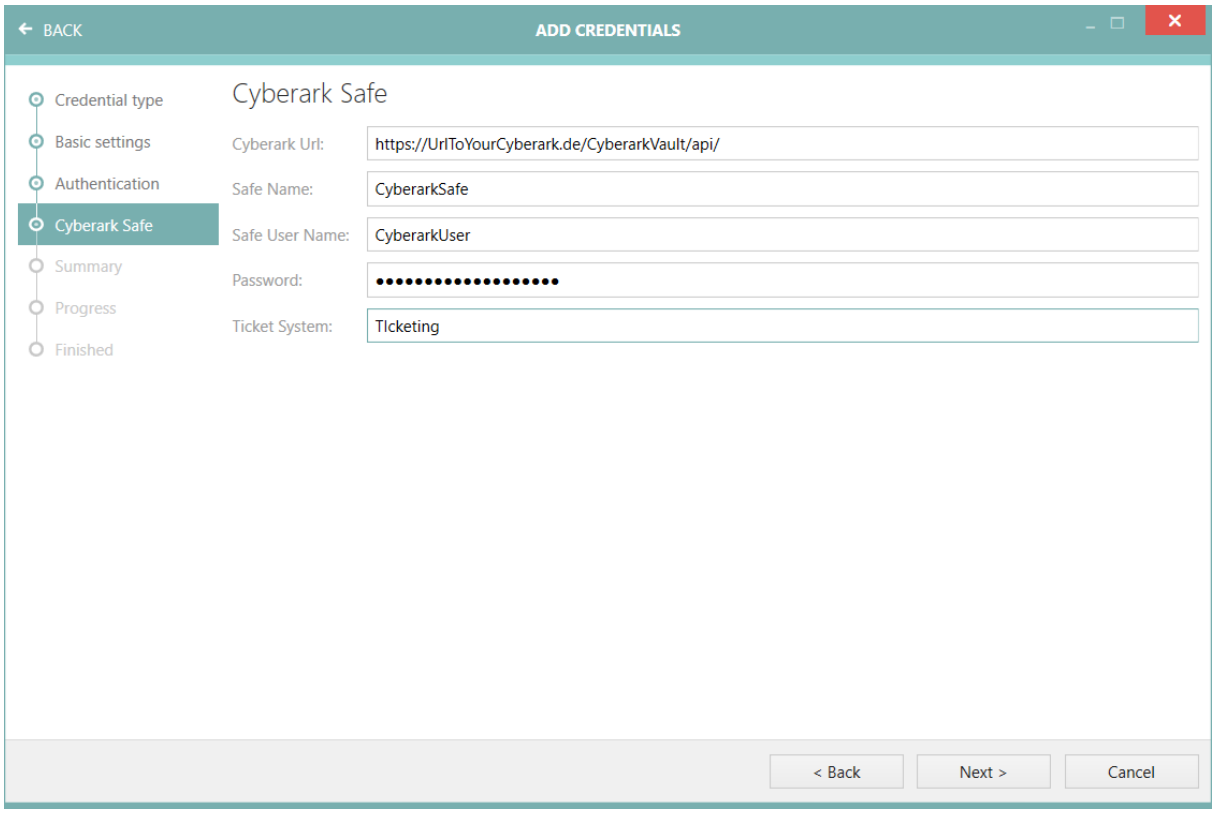
The authentication using CyberArk is available for the credential types listed below. If multiple authentication methods are available for a credential type, it can only be selected for the password authentication method.

- Windows credentials
- SSH credentials (The usage of the CyberArk authentication is not possible if **Key** or **OpenSSL key file** have been chosen as authentication method)
- Oracle credentials
- VMware credentials (The usage of the CyberArk authentication is not possible if the **Use Windows session credentials** option has been activated)

In order to configure CyberArk Authentication select the CyberArk Authentication option from the dropdown menu that is available for the all the authentication methods for which CyberArk authentication is available.




The **CyberArk Safe** step will now be available in the wizard.



In the **CyberArk Safe** step the following settings need to be configured.

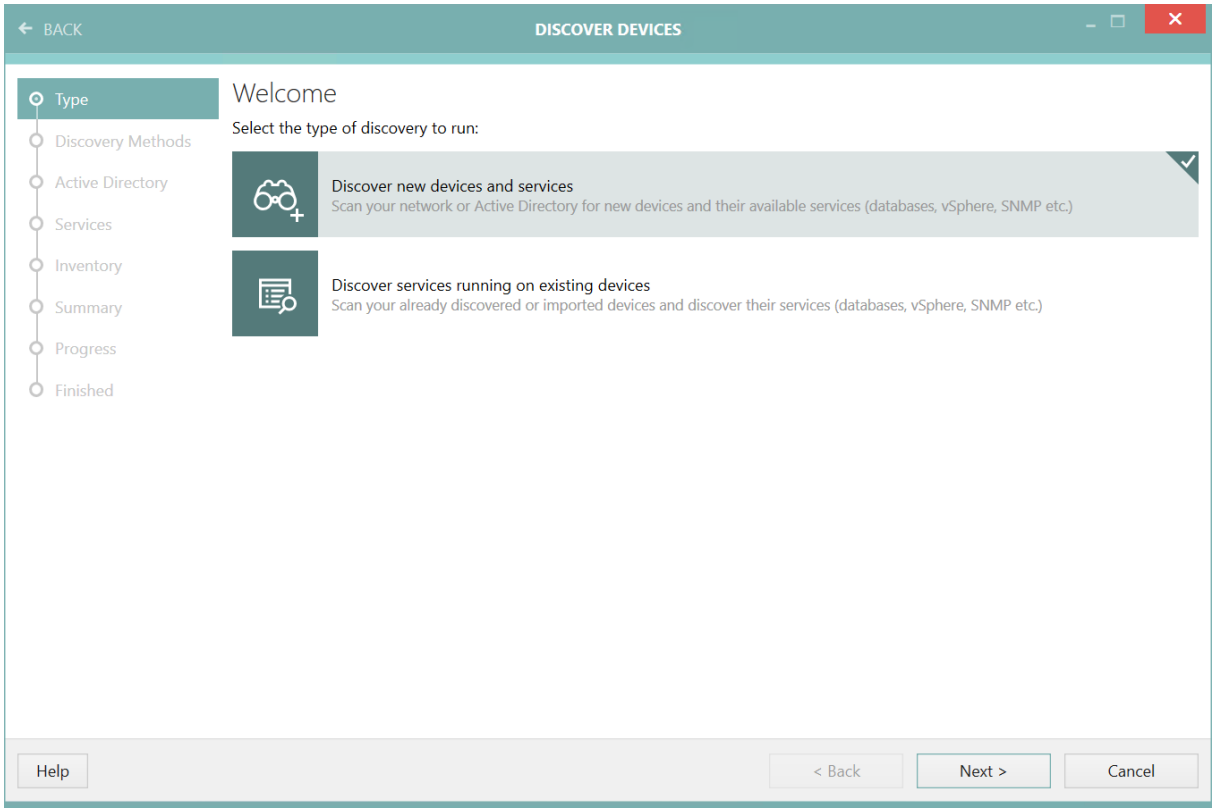
- **CyberArk URL**: Enter the URL to the CyberArk vault that will be used. The CyberArk vault is an environment in which it is possible to create the different safes which can be used to authorize access for users.
- **Safe Name**: Enter the name of the specific safe that will be used. The safes in a CyberArk vault can be used to store and organize authorized users according to the specific requirements needed.
- **Safe User Name**: The name of the user that will be used. This name must match the name with which the user is stored in the specific safe.
- **Password**: Enter the password of the user that will be used.
- **Ticket System**: Enter the name of the ticket system that will be used.

**Be aware:**
Errors when calling the credentials will be logged though there will not be any visible feedback in the RayVentory Scan Engine GUI.

More information about CyberArk can be found here.

# Discovery Wizard

The **Discovery** wizard is the primary tool for finding running devices and services and / or importing them from Active Directory.




There are two mutually exclusive options available in the first page of the **Discovery** Wizard.

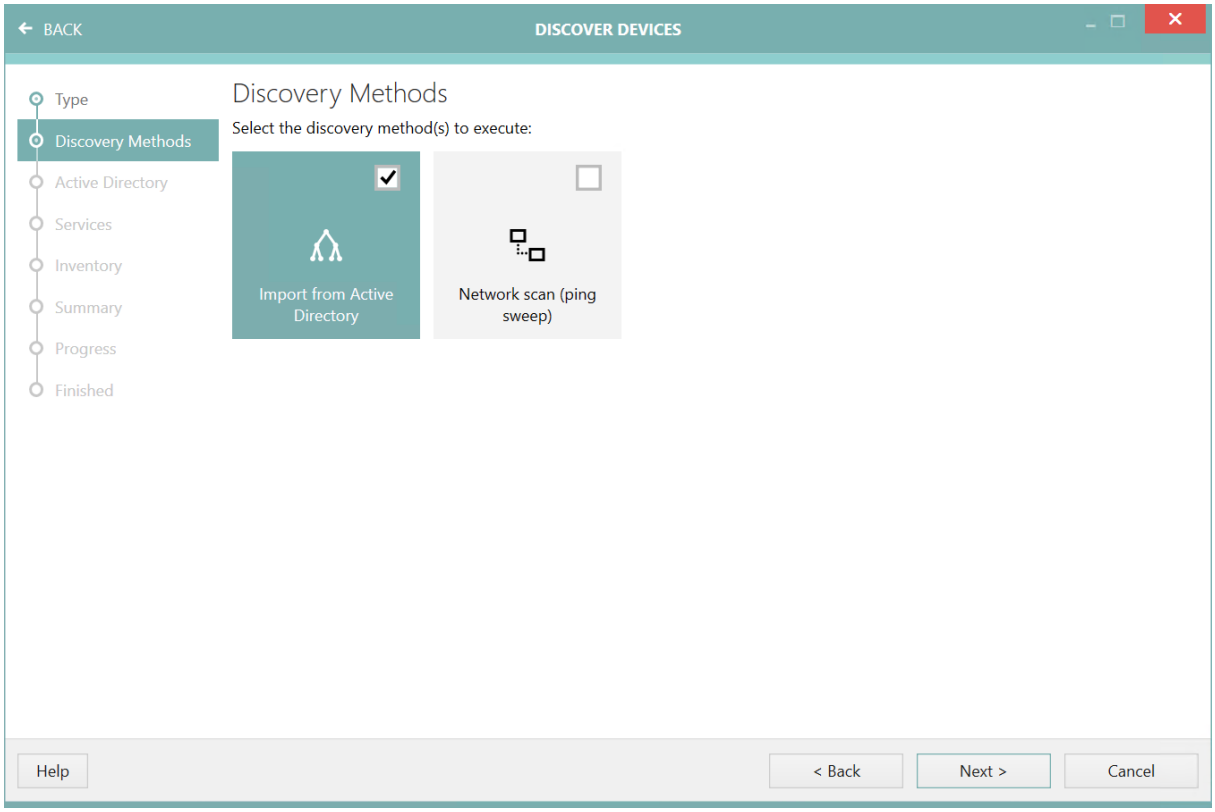
- **Discover new devices and services** - Select this option to perform a discovery of new devices and / or services.
- **Discover services running on existing devices** - Select this option to perform a discovery of services on devices that have already been added to the **Devices** lists.

Depending on the choice selected, the number and the content of wizard pages that are following may vary.

# Discovering New Devices

This section describes the steps of the **Discovery** wizard, provided that the user selected the option **Discover new devices and services** as his required Discovery type.

## Methods




There are two complementary options which determine the source of the data for the discovery operation. In order to continue to the next page, at least one of them needs to be selected.

- **Import from Active Directory** - use this option to import devices from your Active Directory. You will be able to specify the domain and the filtering in the next step.
- **Network scan (ping sweep)** - use this option to perform a ping sweep on your network. You will be able to specify the IP address range in the next step.

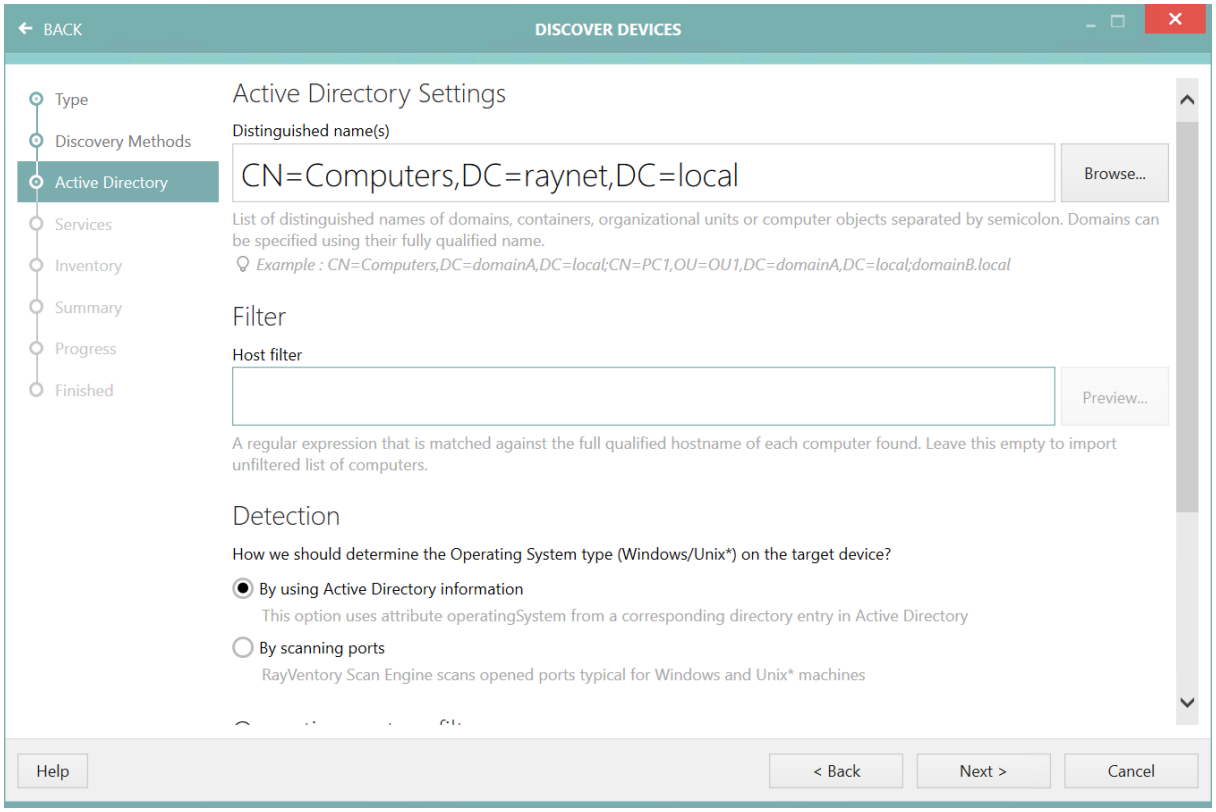
**Note:**
The network scan tries to contact each device in the specified network range and checks for predefined ports to determine the device family (Windows or UNIX). Contact your network administrator before executing scans on a wide range of addresses.

Selecting **Network scan (ping sweep)** adds an extra page **Network scan** to the list, where the user can configure the IP address range. Selecting **Import from Active Directory** adds an extra page **Active Directory** where the user can configure scanned domain(s), filters, and default behaviors. The options are complementary which means that both can be selected at the same

time. In this case, RayVentory Scan Engine merges information from both sources and discovers devices respecting both ping sweep results and extra bits of information from the Active Directory.

## Active Directory

This page is shown only if the user selected the **Import from Active Directory** option on the previous screen.




### Distinguished Name(s)

The domain name (dotted domain) or a LDAP query sting into the Active Directory Domain field. It is possible to enter multiple domain names or LDAP queries by separating them using a semicolon. You can also press the **Browse...** button to open a simple LDAP browser, that enables you to visually select the required container.

**Be aware:**
The LDAP browser supports multiselection. Hold thr **CTRL** button when selecting items to select more than one item at once.

### Filter

This is an optional field, which can be used to perform extra filtering of devices. Regular expressions are supported. If the field is left empty, all entries from the Active Directory will be considered when performing the import, otherwise only the devices with matching full qualified host names will be imported.

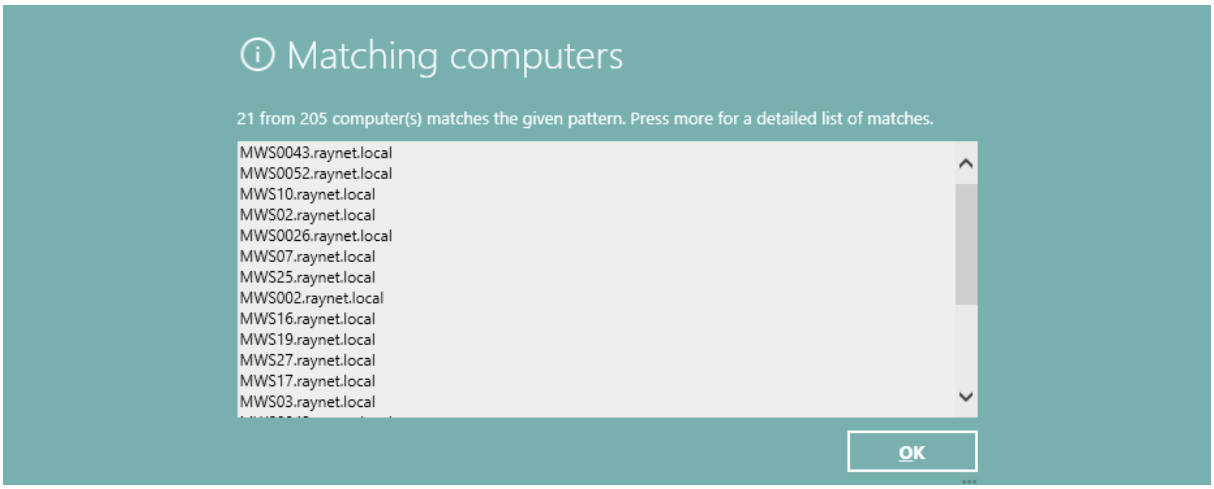
**Be aware:**
Regular expressions can be complex to write and validate. If an invalid regular expression is entered, nothing will be imported. Also, pay attention to the fact that some characters and sequenced may have special meaning, for example a dot "." means any character inside the Regular Expression. To escape special characters, prepend them with backslash (\). Information about how to use regular expression refer to the Microsoft Documentation.

**Example:**
[a-z]+\.sitea\.mydomain

- [a-z]: all lowercase letters from "a" to "z".
- +: the preceding element can be matched one or more times.
- \.: matches a "."

Therefore this regular expression can be used to filter for everything that starts with letters only and ends with ".sitea.mydomain".

You can always test the filter string by pressing the **Preview...** button. It will scan the currently selected node(s) and report back with a list of devices that matched your filter. For example, for a filter `^mws` the following is reported:




(as a side note: `^mws` means any computer name that starts with the string `mws`).

The regular expression engine is case insensitive and does not enforce string boundaries (you have to enforce them by `^` and `$` respectively).

### Detection

This setting determines how RayVentory Scan Engine figures out which Operating System is installed on the imported device. There are two different ways of solving this:

- Select **By using Active Directory information** to use Active Directory member attributes and resolve Operating System from their values. This is recommended if you want to avoid physical scans of the network and it is much faster then the other method.

- Select **By scanning ports** if your Active Directory does not contain the required information OR if you want the results to be reliable based on the actual status of open / closed ports. If this option is selected, RayVentory Scan Engine checks whether typical ports for SSH or RPC are opened and based on their status the right device family is assigned.



**Note:**
Option **Scanning ports** tries to contact each device in the specified network range and checks for predefined ports to determine the device family (Windows or UNIX). Contact your network administrator before executing scans on wide range of addresses.

**Operating system filter**

These options define which types of operating systems should be detected. It is possible to select multiple types.

- **Client**: Select this option to detect all clients.
- **Server**: Select this option to detect all servers.
- **Custom**: When this option is selected, it is possible to define a custom type of system in the **Operating systems** field.

**Fallback**

These options define the behavior of RayVentory Scan Engine should an unrecognized or disabled device be found.

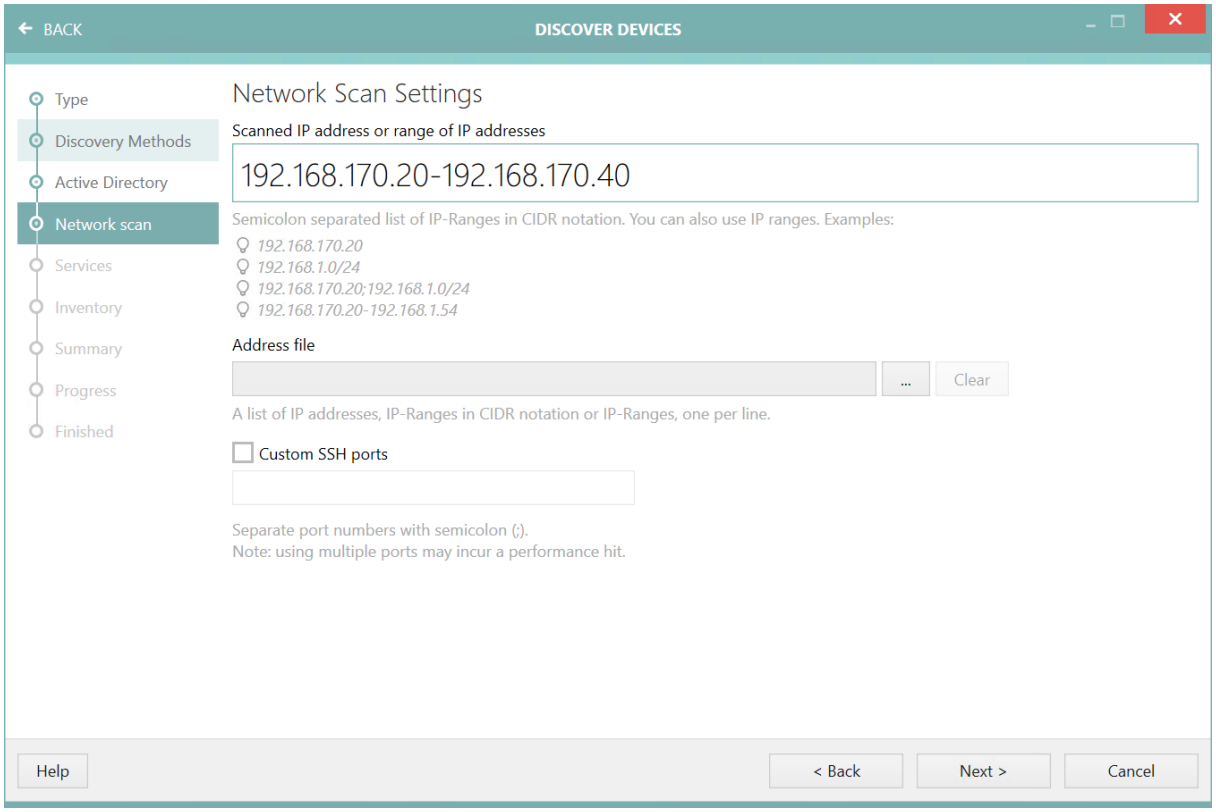
- **Include disabled devices** - Select this option to include disabled devices. If you leave it unchecked, the devices that are disabled in Active Directory will not be imported (recommended).
- **If there is no proper information in Active Directory, treat the device as Windows-based machine** - Select this option to automatically fallback to Windows, if the information in Active Directory is not sufficient to determine the device family. If you uncheck this option, the devices will be imported as "unknown" types. This option is only available if the detection is set to **By using Active Directory information**.
- **If device is unreachable, treat it as Windows-based machine** - Select this option for an automatically fallback to Windows if the set of open / closed ports was not enough to identify the device family. If you uncheck this option, the devices will be imported as "unknown" types. This option is only available if the detection is set to **By scanning ports**.



**Note:**
Make sure that the current user has read permissions for the Active Directory.

## Network Scan

This page is shown only if the user selected the **Network scan (ping sweep)** option on the previous screen.




**Scanned IP Address or Range of IP Addresses**
This is a value which is represented in one of the following notations:

- CIDR notation (for example `192.168.170.0/24`),
- Single IP address (for example `192.168.170.21`),
- IP address range (for example `192.168.170.1-192.168.172.200`),
- Any combination of previous three separated by semicolon.

The discovery will try to reverse-lookup the DNS name for the IP addresses that have been found responding during the scan.

## CIDR Examples

| CIDR | First IP | Last IP |
|---|---|---|
| 192.168.40.0/24 | 192.168.40.0 | 192.168.40.255 |
| 192.168.40.10/32 | 192.168.40.10 | 192.168.40.10 |

**Address file**

When specified, the list of IP addresses is read from a plain text file (where each line is a separate address to scan).

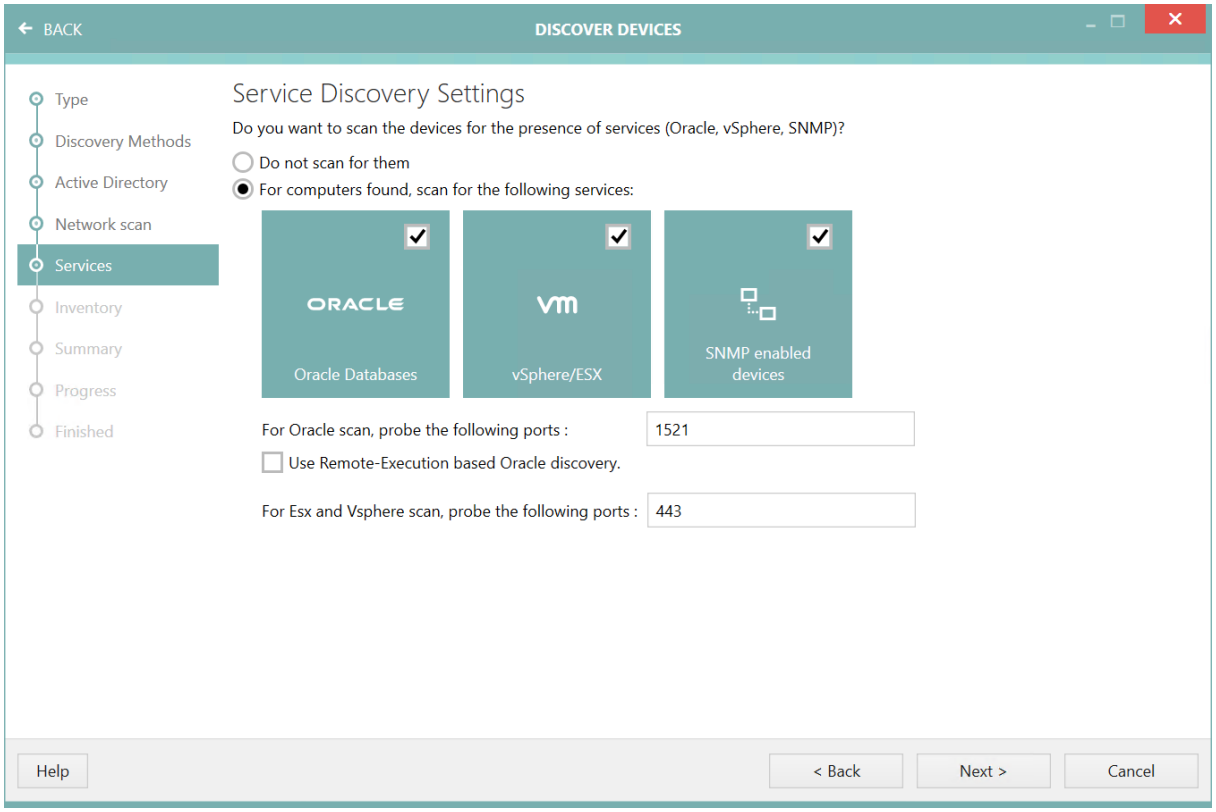
### Custom SSH ports

If a non-standard SSH port should be used for non-Windows devices, it should be entered here. More than one value can be entered by using a semicolon.

**Note:**
Providing more than one port may have negative impact on the scan speed.

## Services

This page contains a configuration of an optional step, allowing the user to perform extra-scanning of services likes vSphere / ESX instances, SNMP devices, and Oracle databases on newly found devices.



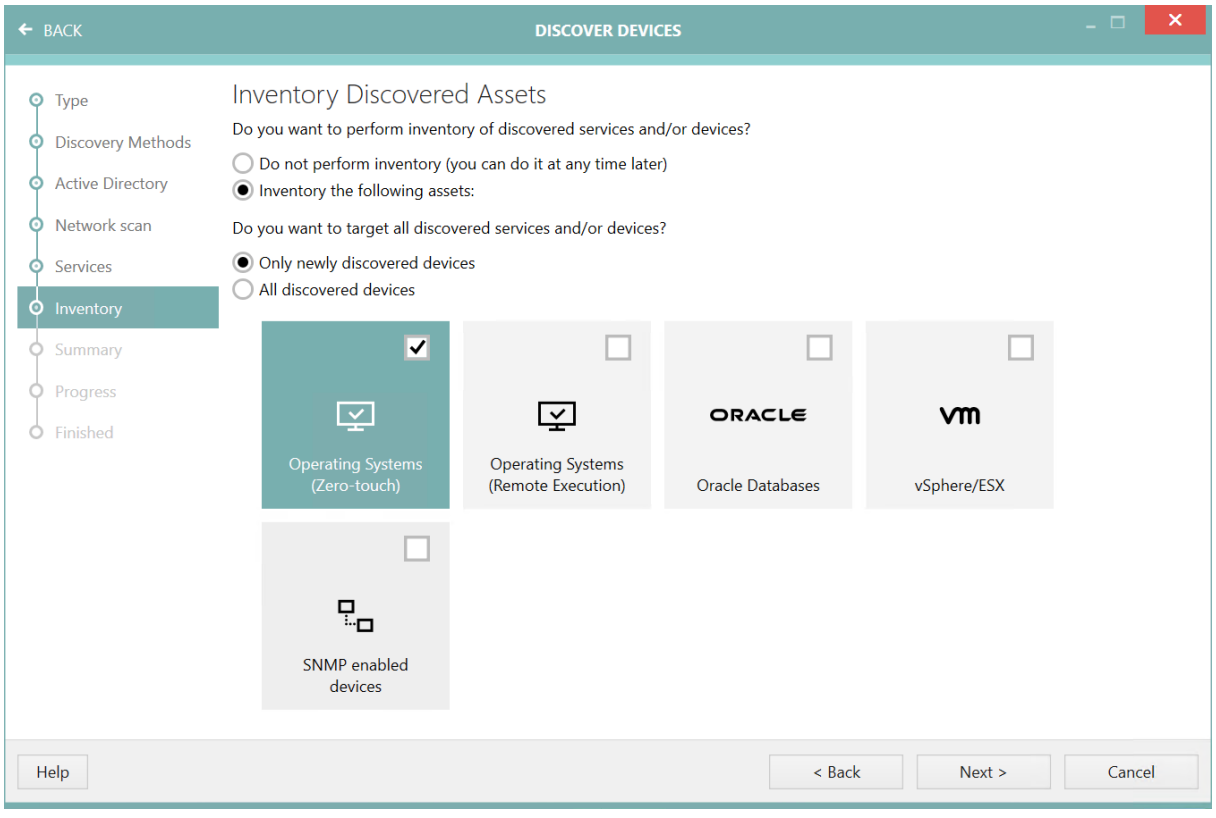

In order to scan for the services, tick the radio button **For computers found, scan the following services** and then select the options for the scan. The options are complementary, if the radio button is checked then at least one option must be enabled in order to continue. By default, none of these options are selected.

For Oracle inventory, there are two extra options on this page that can be configured:

- **For Oracle scan probe the following ports** - This is the port that will be used for probing for Oracle TNS listeners. Multiple ports can be probed by entering a list of port numbers, separated by commas or semicolons. If not defined otherwise, the default port **1521** is going to be used.
- **For Esx and vSphere scan, prove the following port**: This is the port that will be used for probing for vSphere. Multiple ports can be probed by entering a list of port numbers, separated by commas or semicolons. If not defined otherwise, the default port **443** is going to be used.
- **Use Remote-Execution based Oracle discovery** - By default, the discovery of the instance will be done in the "zero-touch" mode. Using this option enables remote execution of scanning utilities on a target server, which usually provides better scanning results. For more information about differences between inventory methods, refer to the Inventory Methods chapter.

## Inventory

This page contains a configuration of an optional step which allows users to perform an inventory of software and hardware on newly found devices.




In order to scan the software and the hardware of new devices, tick the radio button **Inventory the following assets** and then select the options which are to be scanned. The options are complementary, if the radio button is checked then at least one option must be enabled in order to continue. By default, these options are not selected.

The number of options may vary, depending on which services were selected on the Services

page. For the **Operating Systems** scan, two options are available:

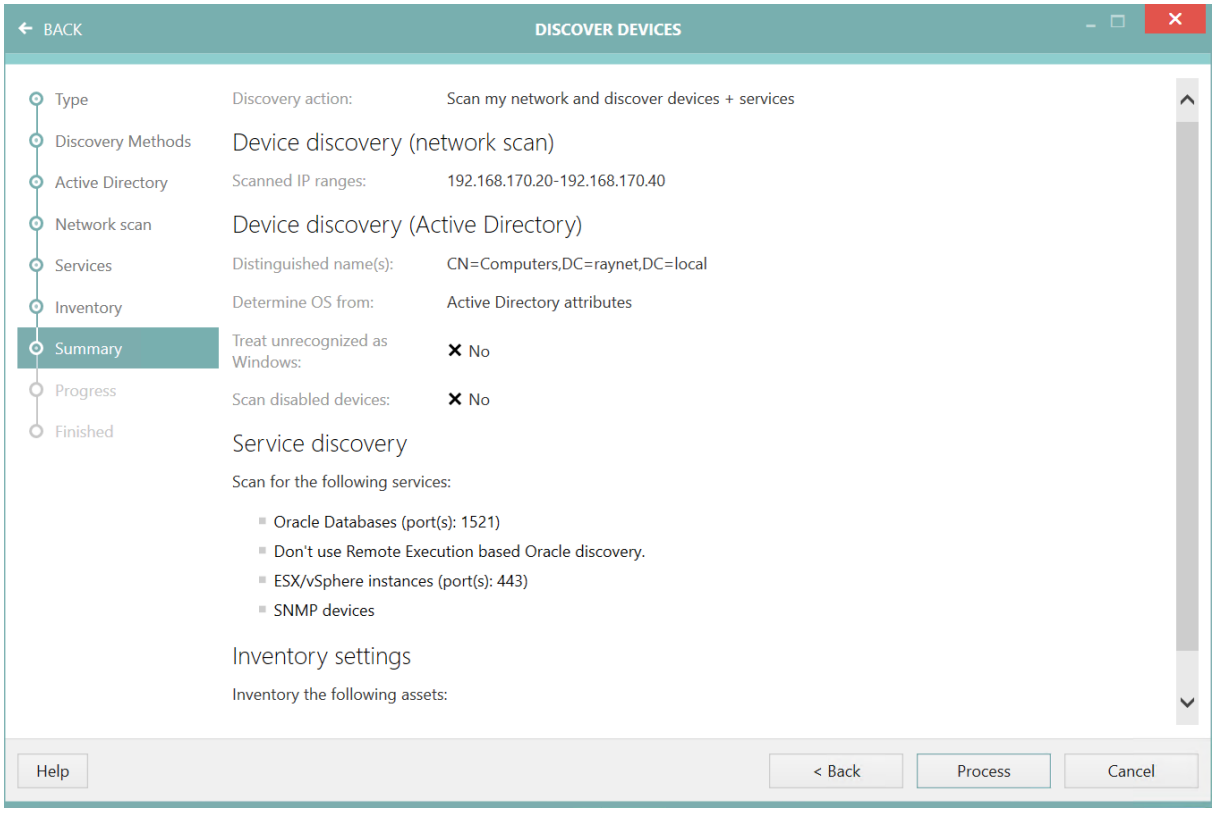
- **Zero-touch** - which uses agentless, zero-touch execution leaving no traces on the target device,
- **Remote Execution** - which uses agentless execution of scanning utilities which run on the target device and report the results back to RayVentory Scan Engine.

Since the devices that are to be found are not known when the wizards starts, the Zero-Touch and Remote-Execution are offered separately. For more controls on how an inventory is executed on target devices, we recommend to skip the inventory at the discovery page and instead let it run and discover the devices. Then, adjust the devices and / or settings and start the Inventory Wizard for fine-grained control on all inventory aspects.

If the optional scan for ESX / vSphere has been enabled, the option **Populate devices** will be shown. In the summary screen of the wizard this will be shown as **Populate the devices table from ESX/vSphere guests**.

## Summary

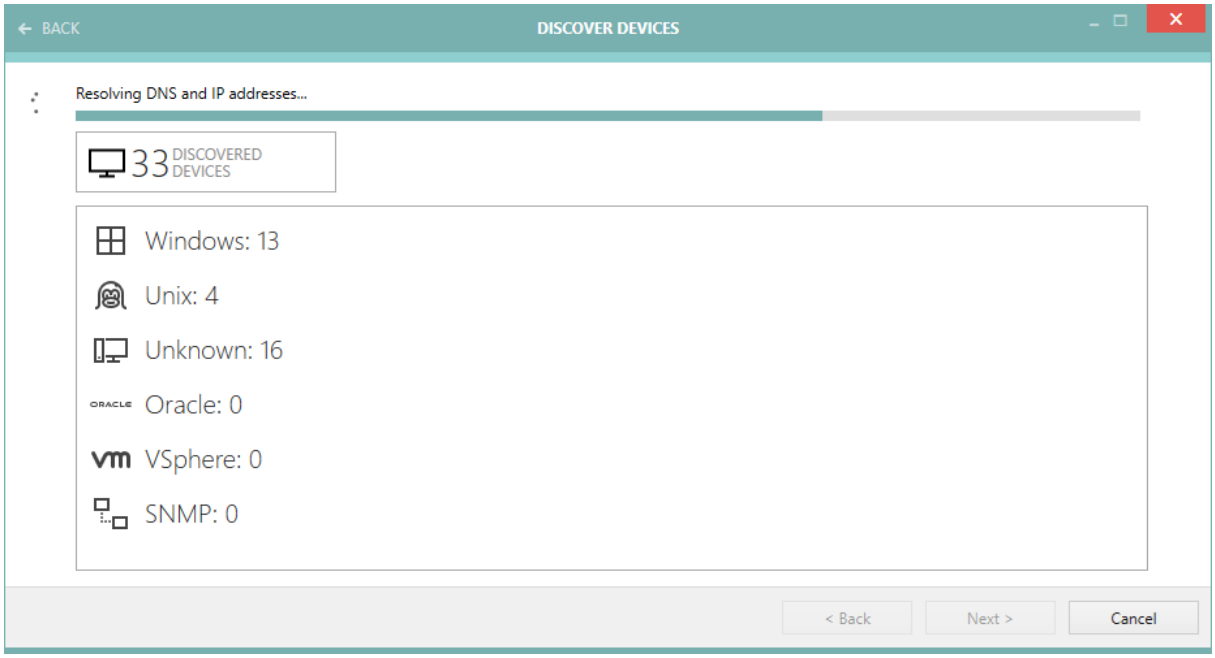
The **Summary** page shows the overview of all choices defined in the previous pages. You can click on the settings to go back to their corresponding places and change them.




Press **Process** to start the discovery.

## Progress and Results

The **Progress** page shows the current activity and real-time results of the scanning.

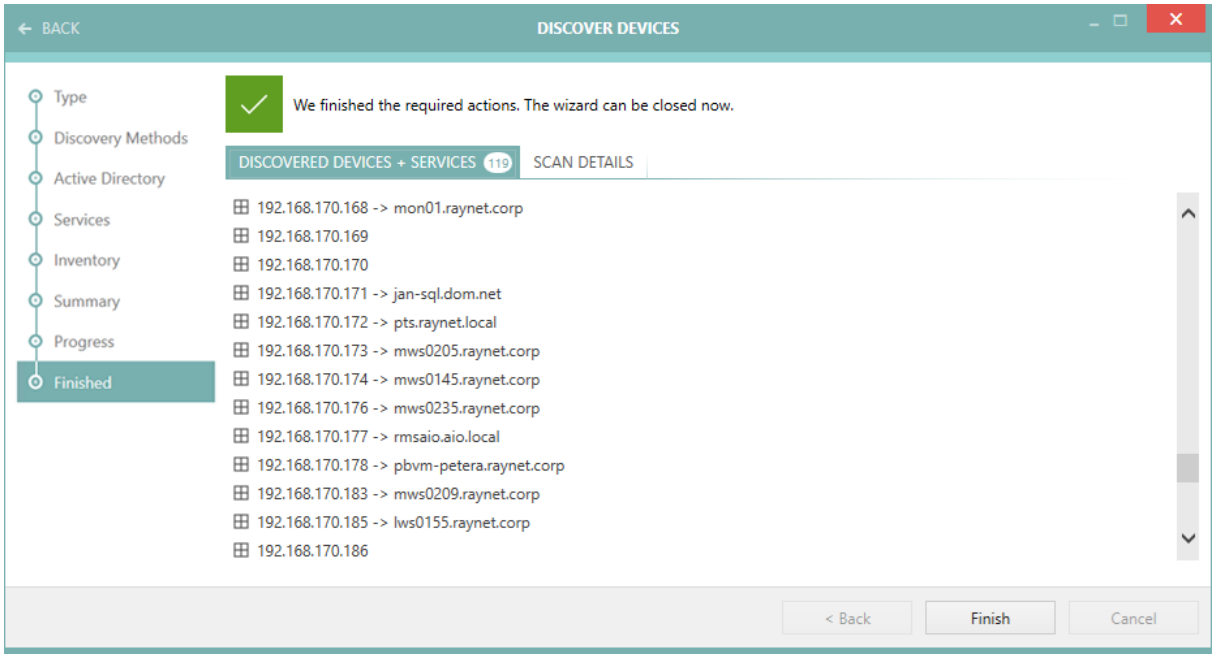



The grid shows the IP addresses and / or host names of devices (depending on the options selected on previous pages) and it runs basically in two steps:

1. Ping sweep and / or Active Directory import of devices.
2. DNS lookup / Reverse lookup.

Once an IP address is resolved to a host name (or vice versa), the content of the entry is updated (see picture above for resolved IP addresses). The icon next to the IP address identifies the device family (Windows or UNIX).

Once the work is done, a confirmation page is shown:




This shows the overview of the discovered devices and extra scan details plus the detailed descriptions in case of errors during scanning.

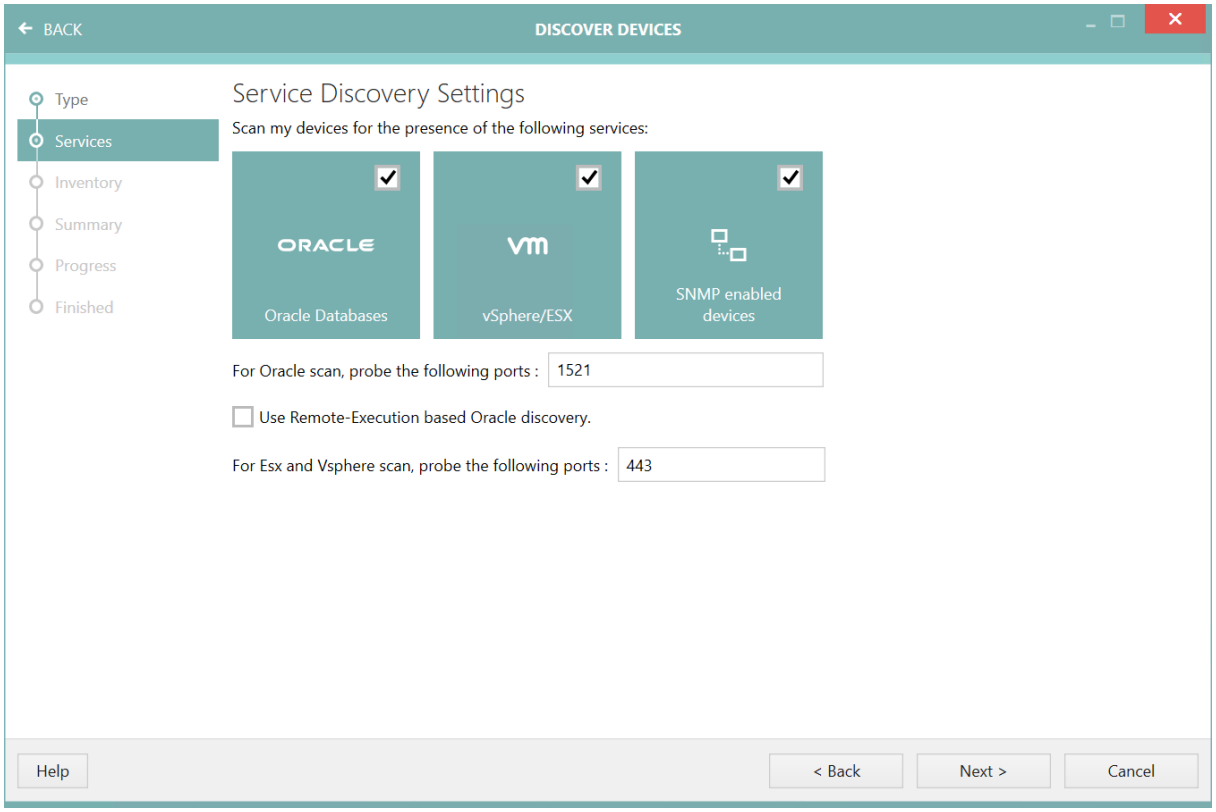
Once the wizard is closed, all discovered devices are automatically imported into the **Devices** view. All discovered services are imported to the **Oracle**, **SNMP**, or **vSphere / ESX** views.

# Discovering Services on Existing Devices

This section describes the steps of the **Discovery** wizard, provided that the user selected the option **Discover services running on existing devices** as his required Discovery type.

## Services

This page contains a configuration of an optional step, allowing the user to perform scanning of services likes vSphere / ESX instances, SNMP devices, and Oracle databases on already existing devices.



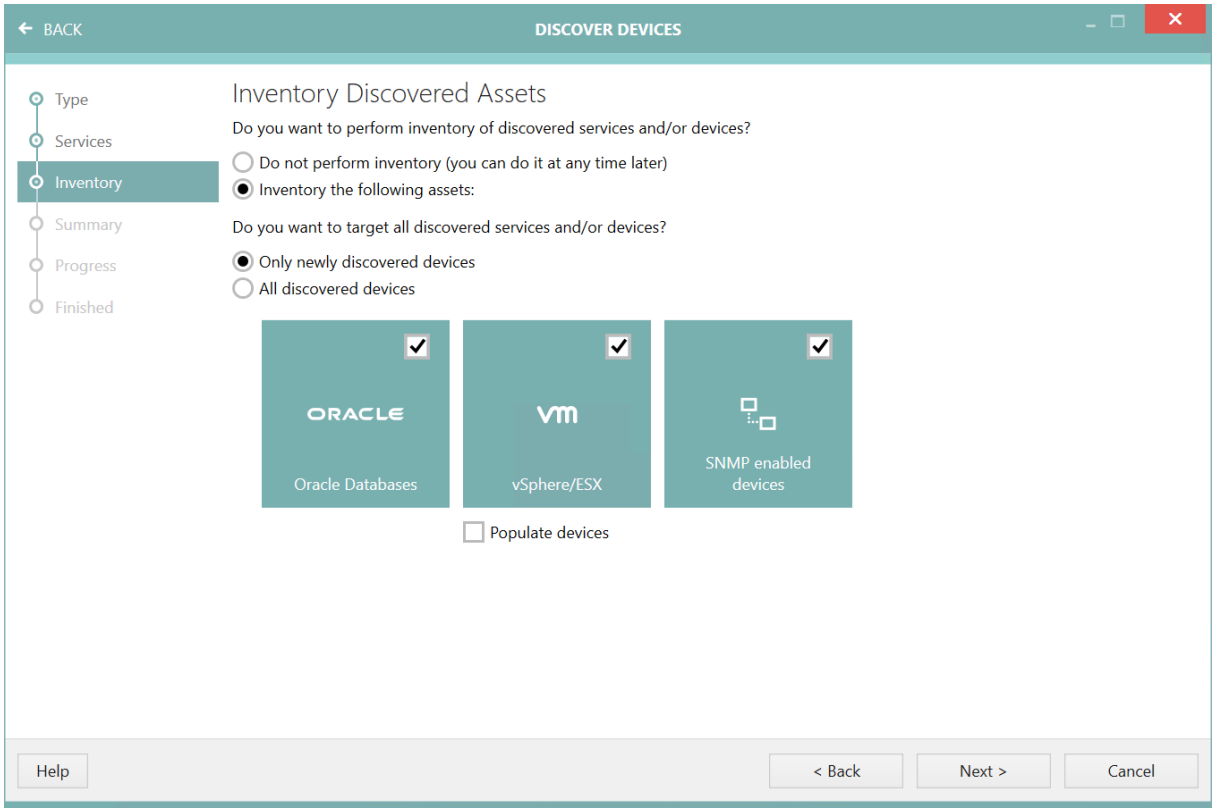

The options are complementary, if the radio button is checked then at least one option must be enabled in order to continue. By default, these options are not selected.

For the Oracle inventory, there are two extra options to configure on this page:

- **For Oracle scan, probe the following ports** - This is the port that will be used for probing for Oracle TNS listeners. Multiple ports can be probed by entering a semicolon separated list of port numbers. If not defined otherwise, the default port **1521** is going to be used.
- **Use Remote-Execution based Oracle discovery.** - By default, the discovery of the instance will be done in the "zero-touch" mode. Using this option enables remote execution of scanning utilities on a target server, which usually provides better scanning results. For more information about the differences between inventory methods, refer to the Inventory Methods chapter.
- **For Esx and Vsphere scan, probe the following ports:** - This is the port used for probing for ESX and vSphere environments. Multiple ports can be probed by entering a semicolon separated list of port numbers. If not defined otherwise, the default port **443** is going to be used.

## Inventory

This page contains a configuration of an optional step, allowing users to perform inventories of newly found services.




In order to scan software and hardware of new devices, tick the radio button **Inventory the following assets** and then select the options to scan. The options are complementary, if the radio button is checked then at least one option must be enabled in order to continue. By default, no options are selected.

The **Do you want to target all discoverd services and/or devices?** option can be used to define if all or only newly discovered devices will be inventoried. To inventory all discovered devices select the **All discovered devices radio** button. To inventory only the newly discovered devices, select the **Only newly discovered devices** radio button.

If the **Populate devices** checkbox is checked, the VMs discovered in vSphere will automatically be added as devices.

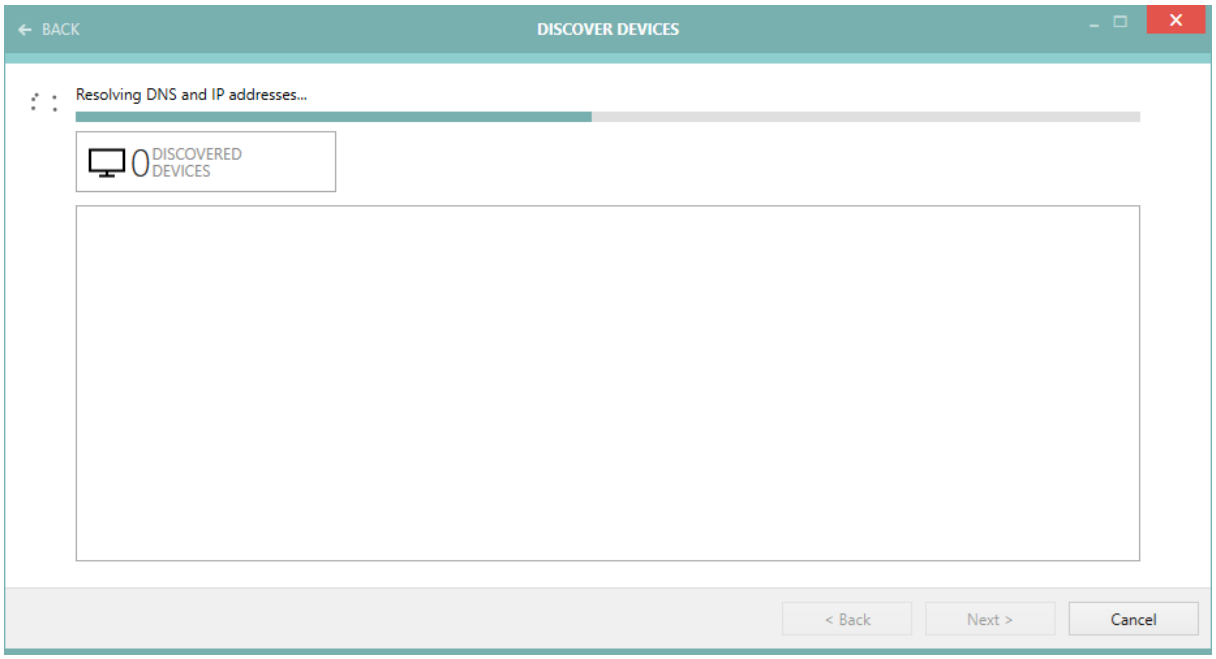
The number of options may vary, depending on which services were selected on the Services page.

For more controls on how the inventory is executed on the target devices, we recommend to skip the inventory at the discovery page and instead let it run and discover the devices. Then, adjust the devices and / or settings and start the Inventory Wizard for fine-grained control on all

inventory aspects.

## Progress and Results

The progress page shows the current activity and real-time results of the scanning.




Once the work is done, a confirmation page is shown. Once the wizard is closed, all discovered services are automatically imported into the **Oracle**, **SNMP**,or vSphere / ESX views.

# Inventory Wizard

The **Inventory** wizard is a primary tool of scanning for software and hardware assets on existing devices and services.

There are different inventory methods which are offered by RayVentory Scan Engine and which will generate data that can be imported by the RayVentory Server for further processing and reporting.

## Starting the Wizard

You can start the **Inventory** wizard from the start / home screen by pressing the tile **Inventory devices...** or from the context menu or the right side-bar of the Devices + Services screen. You may also run an inventory as part of the discovery on newly found devices / hosts and services.

## Introduction to Inventory Methods

RayVentory Scan Engine supports Zero-Touch inventory for all inventory target types. Zero-Touch is a general term for number of techniques which use exposed APIs and management functions without executing own code on the target system. Using Zero-Touch inventory usually involves faster scans, simplistic configuration, and zero impact on the target device (no leftovers, no code execution). As a trade-off, some functions must be enabled and some permissions must be set on the target device in order for the remote zero-scanning being possible at all.

Alternatively, RayVentory Scan Engine offers remote-execution based inventory methods for device / OS and Oracle inventory. Remote-Execution denotes that custom scan utilities must be executed on the scanned environment. This already implies a certain impact on the target device, even though all temporary files are gone once the process is done. Remote scans are more complex to configure, but may deliver more precise scanning results, as by executing a code locally there is usually a better access to lower-level features and APIs.

What inventory methods may be applied to the targets, can be restricted per device or for all devices in general. Restriction on a per-device base is done implicitly by disabling what technologies / capabilities of the target devices may be used. For example: Disabling the capability **Windows Service Manager** for a device will disable all remote-execution based inventory methods that use the Windows service manager. For completely excluding a device / service from inventory, disable all capabilities.

**Hint:**
For the current set of OS / device inventory and Oracle inventory methods, it is sufficient to disable the capabilities 'Zero-touch' and 'Remote-execution' to disable all respective inventory methods.

The restriction of inventory methods for all devices / services is done explicitly, by disabling

methods in the Settings > Inventory > Inventory Methods.

If an inventory is started and multiple inventory methods are available for a target, RayVentory Scan Engine will try each inventory method until it succeeds or runs out of available inventory methods.

Currently, RayVentory Scan Engine offers only one inventory method (zero-touch) for **Oracle Audit**, **vSphere,** and **SNMP**.

The inventory / discovery method **Oracle Auto Discovery and Inventory** is a remote-execution based discovery and inventory method. This method tries to find all available Java installations and determines their versions in order to pick the right version of ORATRACK for the combinations of Oracle DB versions and Java that are to be expected on a target device. This method is not restricted by capabilities and cannot be disabled in the **Settings** screen as it is not part of the inventory methods that will be automatically applied by RayVentory Scan Engine. This inventory / discovery method can only explicitly be triggered by the context menu in the **Devices** list or during discovery by enabling the option **Use Remote-Execution based Oracle discovery**.

## Usual Precedence of Inventory Methods

The inventory methods for device / OS inventory and Oracle inventory are tried in a specific order. They are usually sorted by their impact on the target device (for example Zero-Touch Inventory considered to have the least impact on the target devices are always tried first, unless they are disabled or incompatible).

### Operating System Inventory Methods for Windows Devices

- Zero-Touch by WMI / WINAPI on Windows
- Remote Execution by ServiceManager upload HTTP(S) on Windows
- Remote Execution by WMI upload HTTP(S) on Windows
- Remote Execution by ServiceManager upload SMB on Windows
- Remote Execution by WMI upload SMB on Windows
- Remote Execution by WMI / SMB local files on Windows
- Remote Execution by ServiceManager / SMB local files on Windows

### Operating System Inventory Methods for UNIX Devices

- Zero-Touch by SSH on Linux / Unix
- Remote Execution by SSH / SCP local files on Linux / Unix

### Oracle Inventory Methods

- Zero-Touch by ORATRACK
- Remote Execution by ServiceManager upload HTTP(S) on Windows
- Remote Execution by ServiceManager upload HTTP(S) on Windows
- Remote Execution by ServiceManager upload SMB on Windows
- Remote Execution by WMI upload SMB on Windows

- Remote Execution by ServiceManager / SMB local files on Windows
- Remote Execution by WMI / SMB local files on Windows
- Remote Execution by SSH / SCP local files on Linux/Unix

### Special Precedence of Inventory Methods

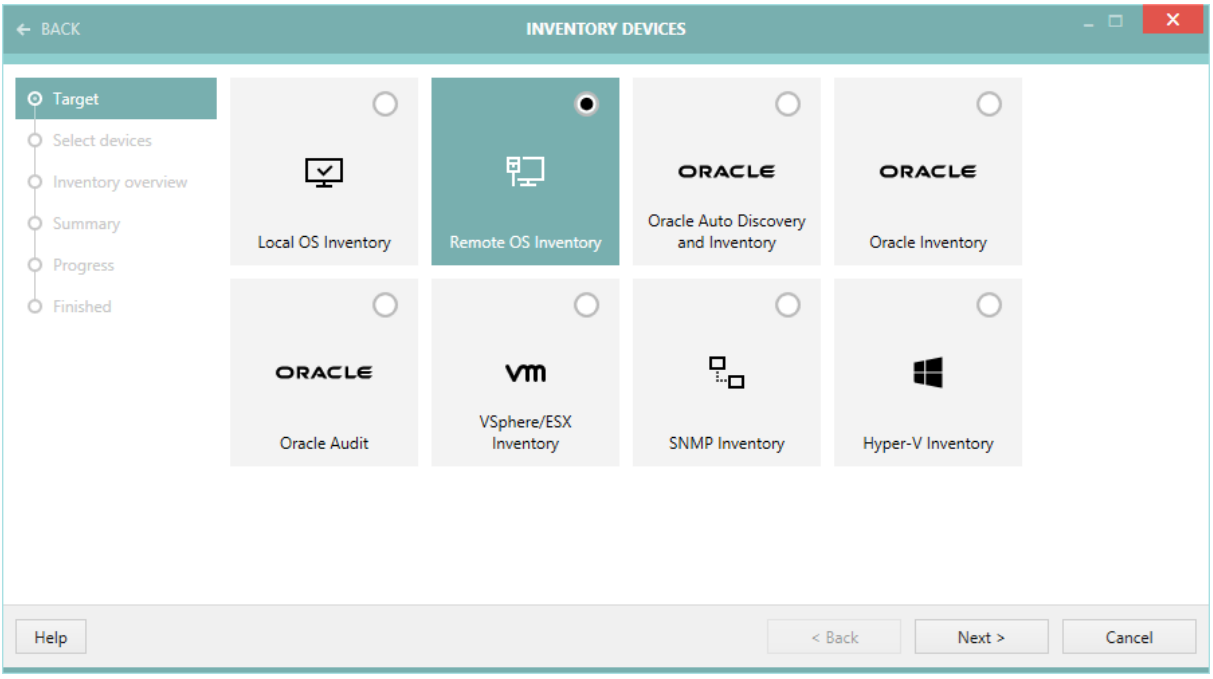
RayVentory Scan Engine remembers which inventory method worked for a each scanned device. If the previous scan was successful and RayVentory Scan Engine is able to determine which method worked for a device, that method is going to be preferred in the future and overrides the precedence outlined in the above lists. If the last successful method fails, then all other methods will be tried in the usual precedence. You can see the technical name for that method in the column **Last successful inventory method**. The column is hidden by default and must be selected from the column chooser. Similarly, technical names of failing methods are available in column **Last failed inventory methods**. The columns can be shown in the respective views in the Devices + Services screen.

## Optimization on Consecutive Runs

Certain inventory methods for device / OS inventory and Oracle inventory are platform-specific. If an inventory succeeds for a target of undetermined type, then the implied target type is set for its respective device. The next time an inventory is run, certain checks that are needed to determine which of the available inventory methods are considered (like which ports are open) will be skipped.

# Target

The first wizard page asks the user to select the type of inventory he would like to perform.




Based on the selected method, subsequent pages will receive slightly different context. For example, once **Remote OS Inventory** is selected, the next page shows the list of devices to be scanned. On the other hand, selecting **SNMP Inventory** shows the list of SNMP connections on the next page, while **Local OS Inventory** completely hides the second page.

The following options are available:

### Local OS Inventory

This runs a local inventory which will give an `.ndi` file as result. This inventory file includes data regarding the hardware and software of the host. The extend of the inventory can be controlled by adjusting the file `wmitrack.ini` (query by WMI) or by adding certain registry entries (including the results of special Visual Basic scripts).

This is the only method that can be started without any devices or services configured in the Devices + Services screen.

**Be aware:**
Running the local inventory is equivalent of starting `ndtrack.exe` manually with the following parameters `-t machine -o ShowIcon=false -o NetworkSense=false -o Upload=false -o InventoryDirectory="XXX"` where XXX is the path to the output folder.

This option is kept in RayVentory Scan Engine for compatibility reasons. The results of the scan cannot be seen in the **Devices** view. If you want to perform a scan on the local system and have the results managed by the Devices screen trigger an inventory on a target device with DNS name localhost or IP address of the current host.

### Remote OS Inventory

This creates an inventory of a remote machine using either pure Zero-Touch technique or Remote-Execution on the target device. RayVentory Scan Engine automatically chooses the right method for each device, based on various settings, capabilities and previous runs. After selecting remote devices, you will be able to verify which methods get executed on which device by visiting the Inventory Overview page.

Local and remote OS inventories support the following platforms:

- Windows
- Linux*
- HP-UX
- IBM AIX
- OSX
- Solaris

* - conditions apply, support for different Linux distribution may vary.

**Be aware:**
There may be certain versions or architectures of platforms that are not supported.

### Oracle Inventory

The actual inventory runs certain queries on an Oracle database in order to retrieve data and hints on license relevant configuration details. In conjunction with the OS inventory of the database host, this data is the source for the reports that help to figure out what kind of Oracle license is needed. If the path to the Oracle's DBFUS script is set, this inventory will also include results from this script. If such an inventory is being imported by a RayVentory Server, this will allow for the comparison on what database features were found to be license relevant by the Oracle Inventory and by the DBFUS script. See Support for Database Feature Usage Script for the requirements that are necessary to enable the DBFUS execution option.

**Be aware:**
In order to use this option a Java Runtime Environment (JRE) compatible with at least Java SE 6 must be installed.

### Oracle Auto Discovery and Inventory

Discovers database instances installed on a host and gathers data on enabled options and usage on Oracle database instances.

### Oracle Audit

With RayVentory Scan Engine, it is possible to run the **Oracle's Review Lite Script** on many target databases at once and collect the output files. To use this option, in the setting **Review Lite Script** path, the path to the copy of the **Review Lite Script** needs to be set. Furthermore, the path to the SQLplus executable in the local Oracle client installation needs to be set.

### vSphere / ESX Inventory

This gathers data regarding host / guest relationships and the host and guest configuration for

VMware ESX or vSphere virtual infrastructures. This inventory method requires VMware type credential in the credential store. Addressing the SDK service endpoint of an ESX will retrieve data on all hosts and guests in the cluster.

**SNMP Inventory**
Gathers basic data on network devices that support the SNMP protocol. This is intended for devices like printers, UPSs, and network equipment that does not expose its Operating Systems or a standardized interface besides SNMP.

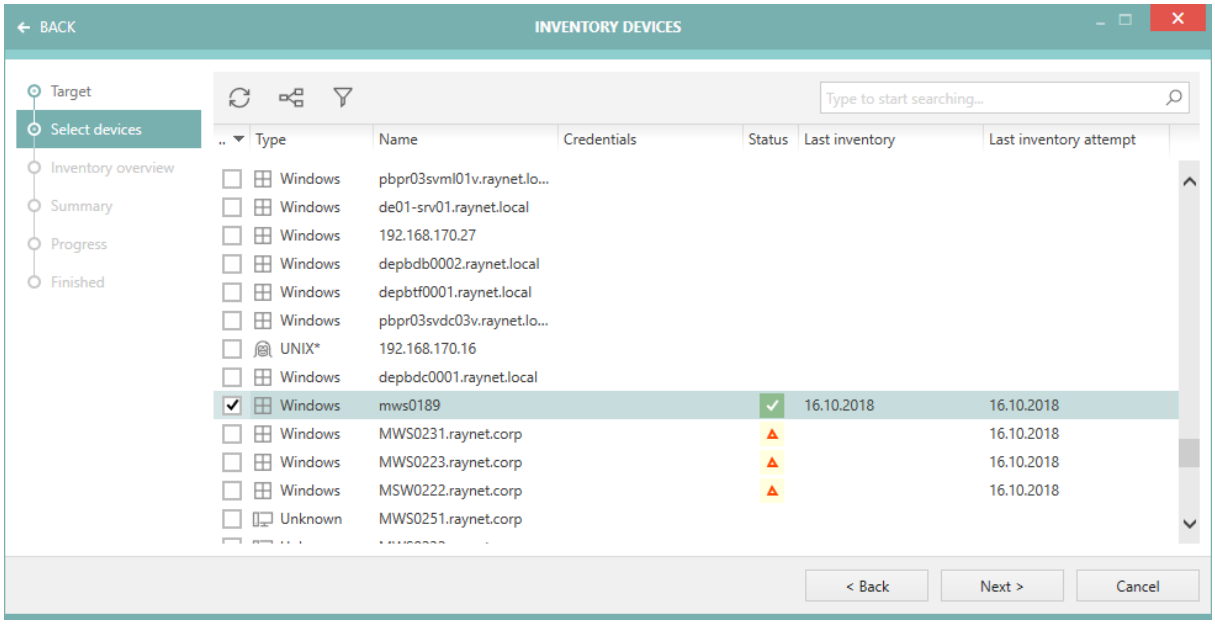
## Performing Oracle Audit and Discovery

There are two methods are not available in the Wizard, but can be called from other places:

- The inventory method **Oracle Audit** is not available from the inventory wizard or as an option in the discovery wizard but it is accessible in the context menu of the **Oracle** list as **Audit…**.
- The discovery / inventory method **Oracle Discovery** is not available in the inventory wizard but as an option in the Discovery wizard and in the context menu of the **Devices** list as **Oracle Discovery…**.

# Select Devices

This page allows the user to select which devices are to be scanned.




The entries and layout seen here may vary based on the inventory type selected on the first page. You can use similar filtering, searching, and grouping capabilities as in the Devices view. The same chapter describes how to interpret the values visible in the **Status** column.

Unlike the **Devices** view, the grid here does not support multiselection by highlighting items. Instead, a separate checkbox column is used. You can select all, deselect all, and invert the currently visible selection by accessing the grid context menu and choosing one of three options.

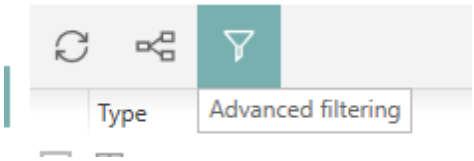
In order to go to the next page, at least one item should be selected.

**Note:**
RayVentory Scan Engine provides a usability shortcut - if no device is selected in this view and the user presses the **Next>** button, all devices will be included automatically.

## Filtering

The list of available devices can be filtered using advanced capabilities of the **Advanced filtering**.

To filter the devices, press the **Advanced filtering** button:




The **Advanced Filtering** dialog consists of the following parts:

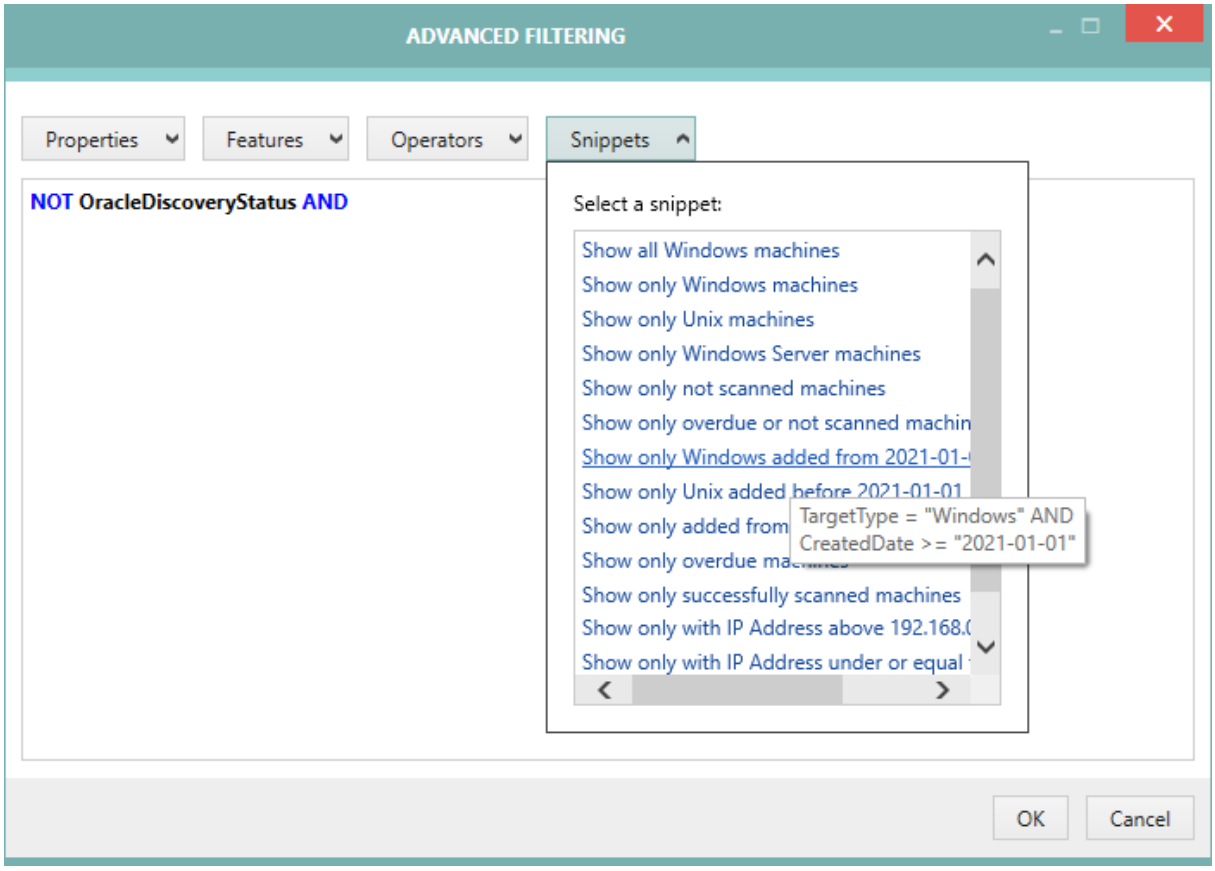
- **Expression editor**
  A multi-line text field supporting basic syntax highlighting. The content of this field can be edited manually, or changed by interacting with four functional buttons: Properties, Features, Operators and Snippets.

- **Properties dropdown**
  This is the list of properties belonging to devices. You can insert the property by selecting it and choosing the comparison/equality operator, followed by pressing the **Add to condition** button. Properties may be compared against values entered manually, or generated by the **Features** button.

- **Features**
  This is a generator of values for various properties. Using this editor ensures, that dates, boolean values, names etc. are used properly, with a right context and in the right format.

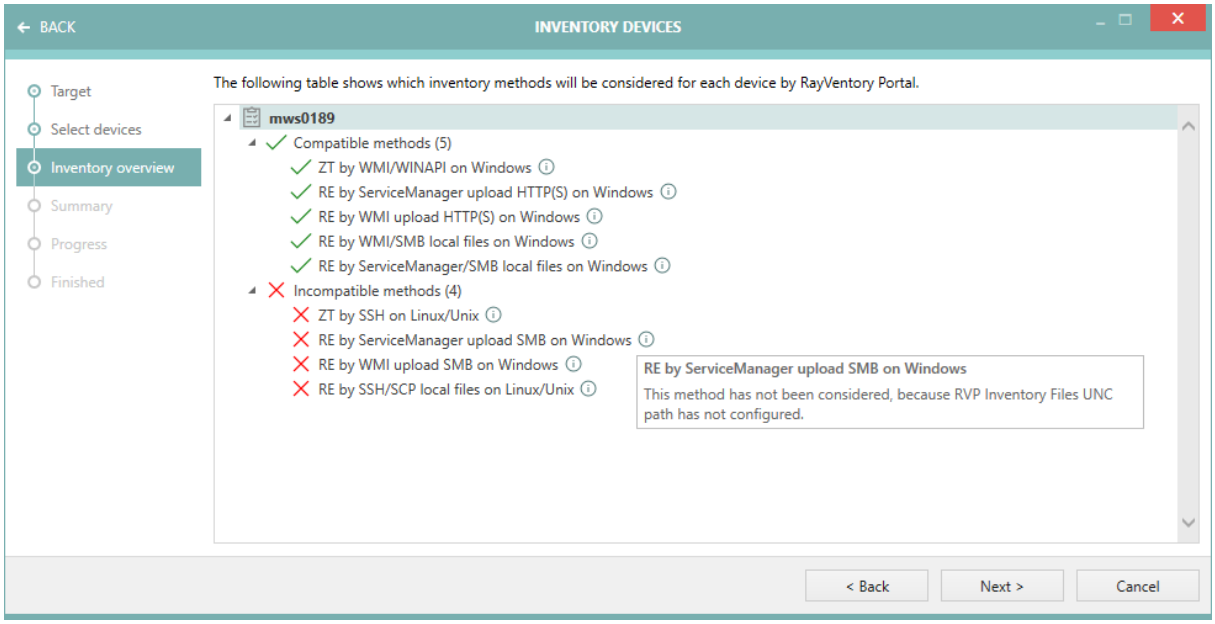
- **Operators**
  Various boolean operators for joining conditions.

- **Snippets**
  This is the list of predefined pieces of conditions, which provides some common conditions and serves as a basis and get-started help for customizing conditions.




# Inventory Overview

This page contains a detailed technical summary of compatible and incompatible methods for each selected device.



The main purpose of this view is to identify issues before the scan is even started. The administrator can utilize this information to find out devices which:

- ... have no compatible methods and thus will not be scanned at all,
- ... have certain required methods disabled,
- ... are otherwise misconfigured or use unwanted scanning methods.

For each method, RayVentory Scan Engine provides a tooltip help which explains why a given method was considered as **Incompatible**. The tooltip has also secondary function - for methods that are considered **OK** for a given device, the tooltip information explains why the method has been chosen (for example: the target OS is Windows and its capabilities are sufficient to handle the task).

To determine which methods are compatible 3 sources are primarily concerned:

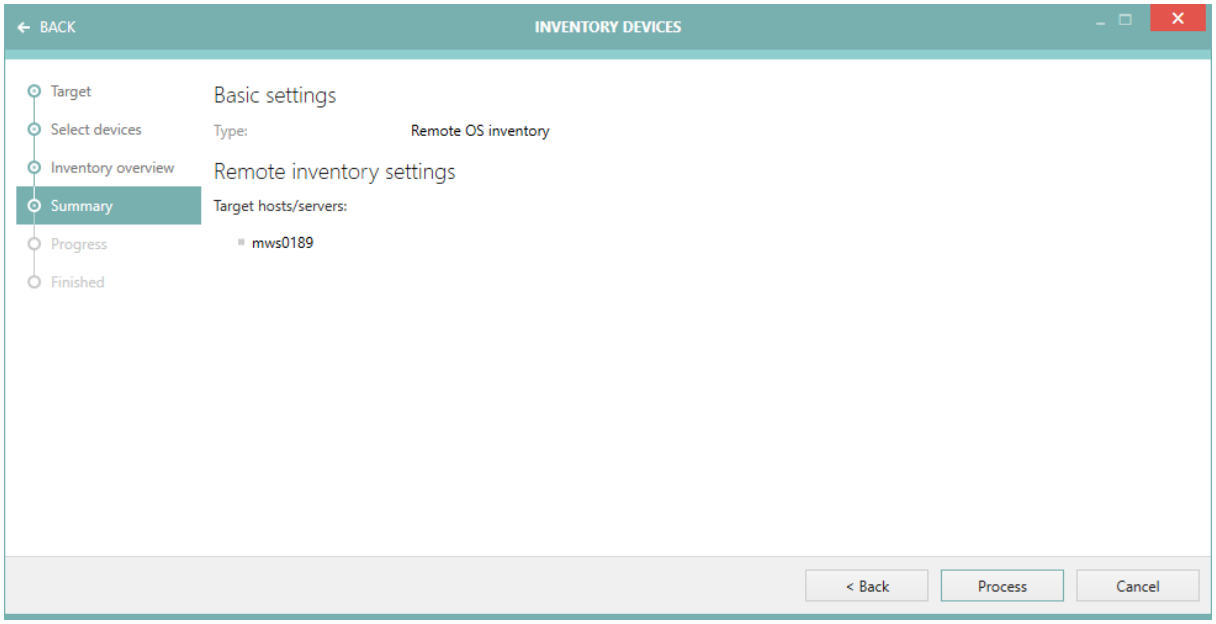
- Current RayVentory Scan Engine settings, for example globally excluded inventory methods, HTTP server, remote execution settings, etc.
- Device properties (for example type) and capabilities,
- Additional information gathered from port scanning (only if device type is **Unknown**).

You can ignore recommendations from this wizard page and continue to the next page, even if loading the data has not been finish yet. Keep in mind, that for every device of the type **Unknown** a separate port scan will be executed to determine which methods (Windows- or Unix-based) will be used. In order to avoid this extra ping sweep on your network, make sure to always set the device type to either Windows or Unix in the device properties.

**Be aware:**
This view provides read-only information. It is not possible to change or configure RayVentory Scan Engine or target devices from here.

# Summary

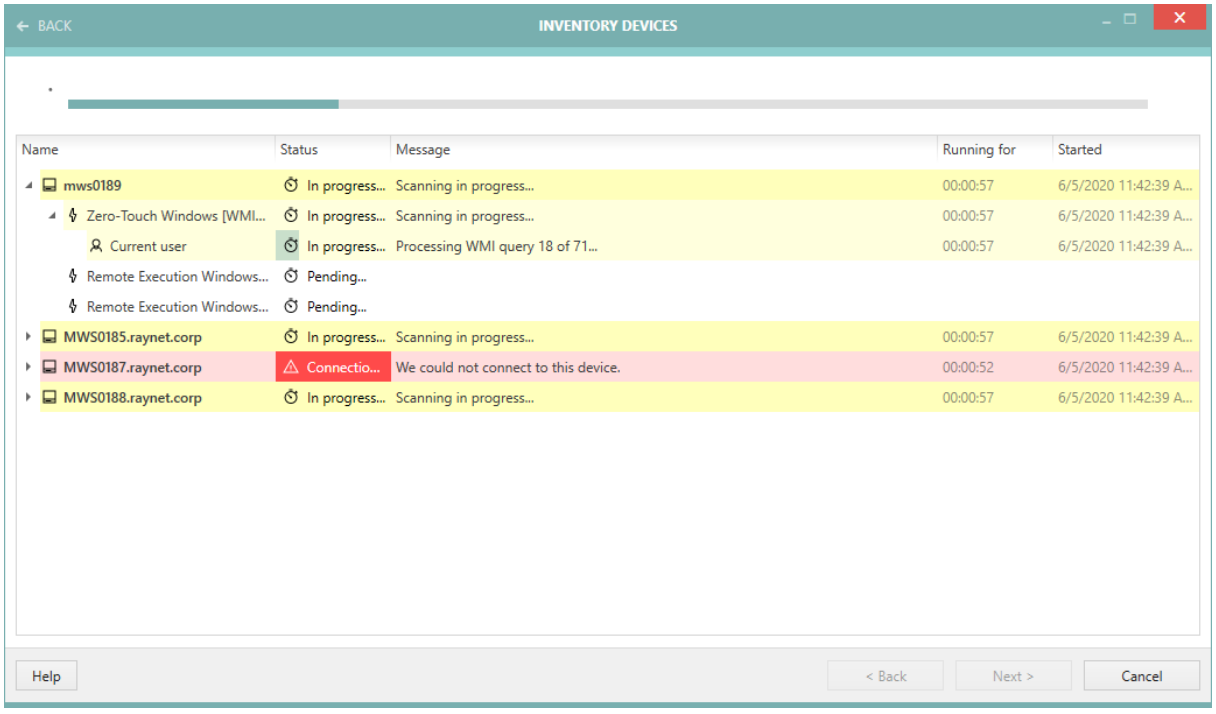
The **Summary** page shows the overview of all choices defined in the previous pages. You can click on the settings to go back to their corresponding places and change them.




Press **Process** to start the inventory scan.

# Progress and Results

The progress page shows the current activity and real-time results of the scan.




The progress is live, which means the status, time and progress/error feedback is immediately visible.

- Yellow cell - the item is currently being processed.
- Green cell - the item has been successfully processed and returned the inventory results back.
- Grey cell - the item has been skipped or canceled.
- Red cell - the item has failed.

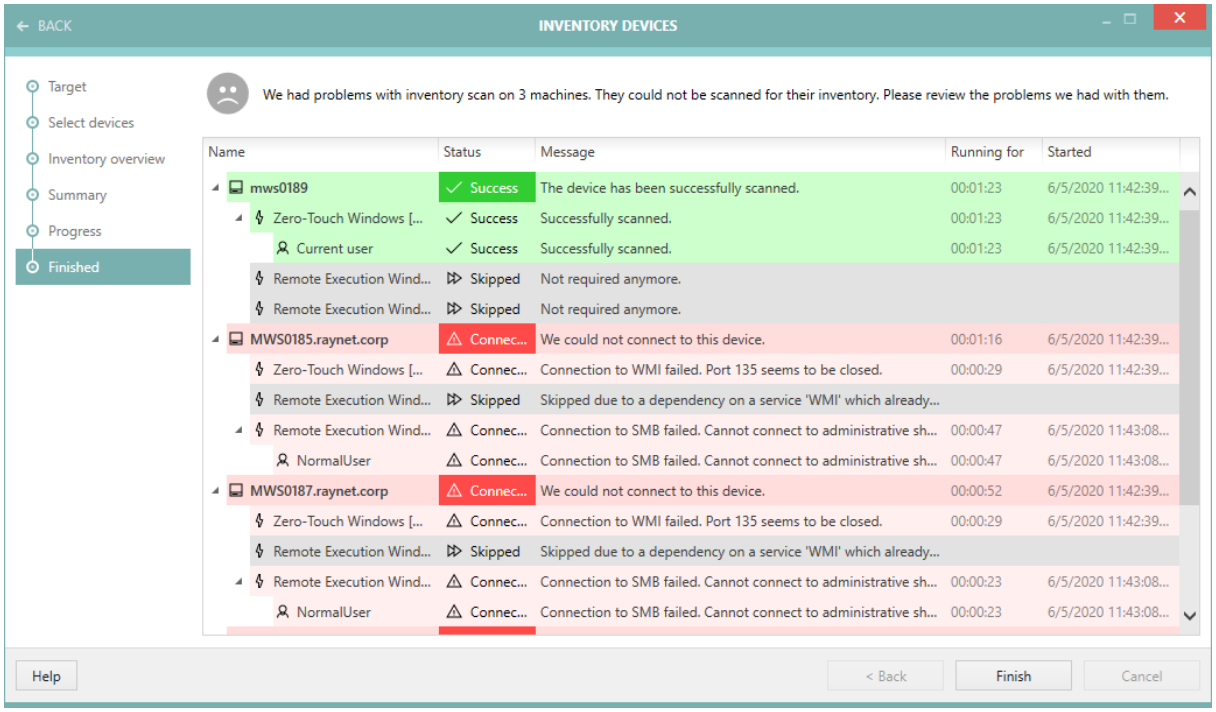
You can cancel the inventory at any time by pressing the **Cancel** button. Note that some operations may still need a few seconds to cancel properly and release the resources, close connections etc.

**Note:**
Chapter Recent Scan Details contains various tips how to interpret the results displayed in the inventory view.

By default, the treeview of device data is partially collapsed, which means only root entries (the devices itself) are displayed. You can expand the items using the little arrow icons to reveal more details about each of the item.

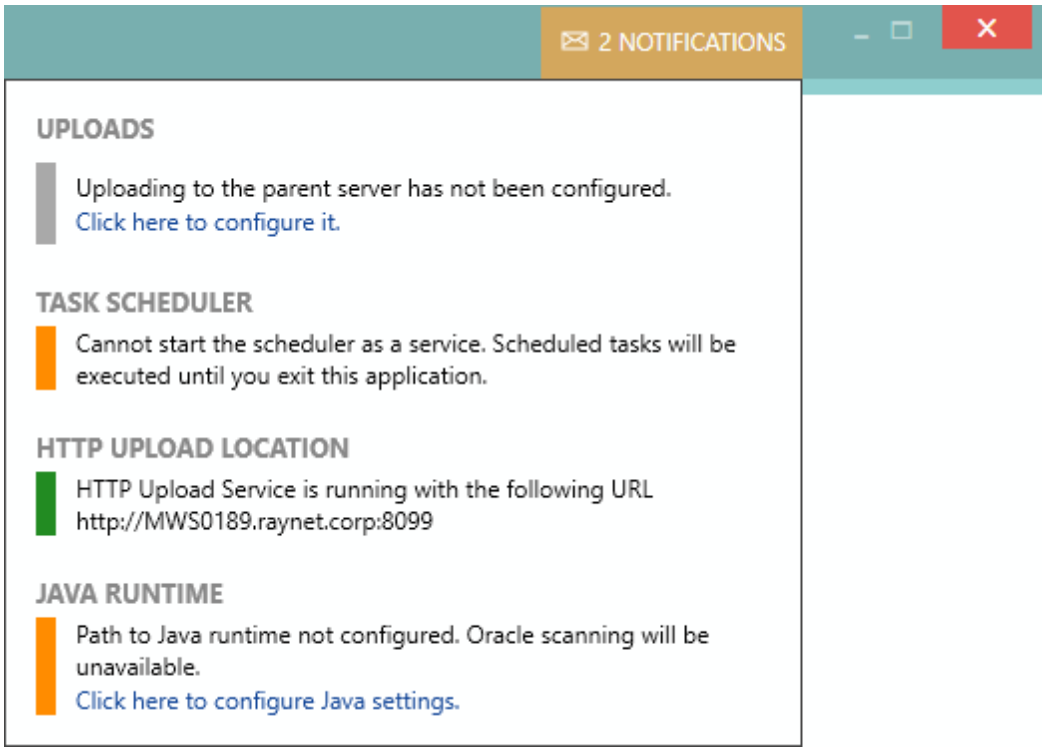
Once the scan is over, press **Finish** to close the wizard. If any errors are encountered during the scanning, they will be listed here.

The finished page contains a partially expanded view of the results. This way it is possible to review the results gathered during the finished inventory scan.



# Notification Center

The Notification Center is a central place where important messages and status are shown.




You can access it at anytime by clicking on its icon residing in the upper right corner of the window. Depending on the status, the notification center can contain the counter of important messages (for example **2 NOTIFICATIONS**) or just simply a text **NOTIFICATION CENTER** if there are no pending messages. Additionally, the color of the badge determines the importance of the messages:

- Green (transparent) - no urgent messages.
- Yellow - important messages (for example missing non-critical configuration, pending uploads etc.).
- Red - critical messages (errors and critical problems with product configuration).

The notification center is the place where file uploads are triggered. If your upload path is configured in Settings > HTTP Services > Upload Location, then the button to upload any pending files is displayed in the **UPLOADS** section.

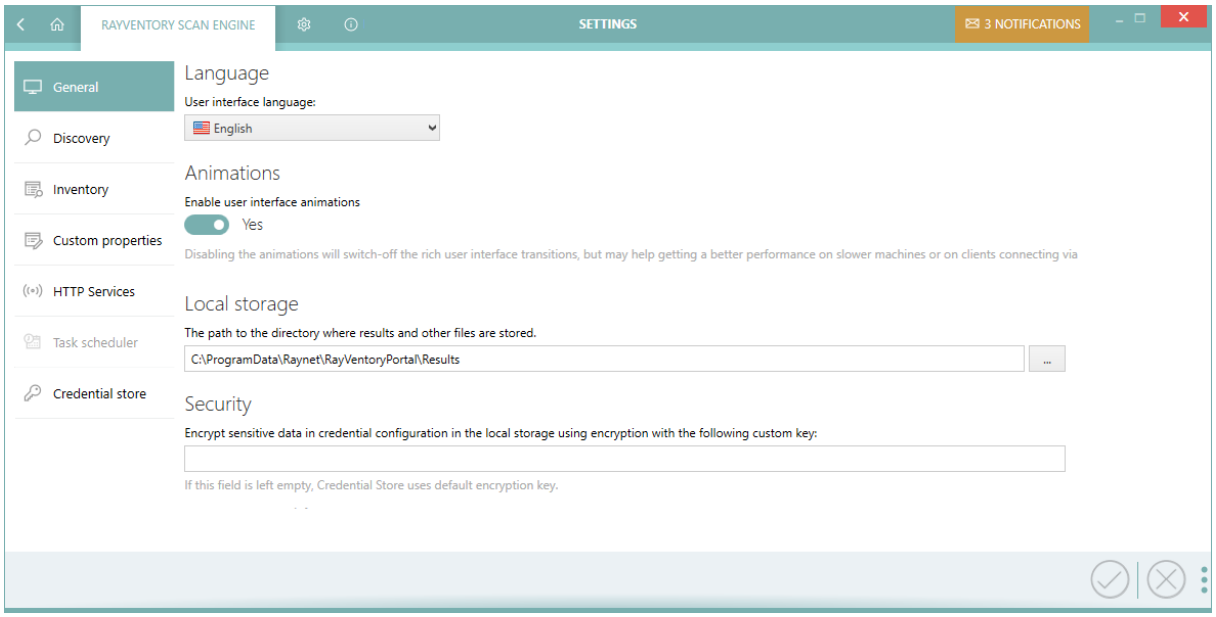
The notification center also informs about:

- The status of the task scheduler (whether it is running as a Windows Service or as private process).

- The status of the HTTP Upload location exposed to endpoints (and its full address).
- The status of the Java runtime (required for Oracle scans).

# Settings

In the **Settings** screen further settings for the customization of the inventory methods and the customization of RayVentory Scan Engine can be found.




The **Settings** screen is split into the following categories:

- General
- Discovery
- Inventory
- Custom Properties
- HTTP Services
- Task scheduler
- Credential store

## General

In the **GENERAL** section there are two settings which apply to the user interface of all the features.

### User interface language

This is used to change the language of RayVentory Scan Engine UI. In the current version, three languages are available (English, German and Russian).

### Animations

This is used to enable or disable the user interface animations like the transition between the different screens.

### Local Storage

This setting defines where RayVentory Scan Engine saves the results of software and hardware inventories. The default value is `C:\ProgramData\RaynetVentoryPortal\Results`.

**Note:**
It is not recommended to change the local storage after any inventory wizards are available.

### Security

This setting defines the encryption key used to encrypt sensitive data (like passwords and keys) inside the configuration files. If this value is left empty, a default encryption key (which is the same for all RayVentory Scan Engine users) will be used.

**Note:**
It is not recommended to change the local storage after any inventory wizards are available. Once the encryption key is changed, all credentials and keys must be corrected again, because no automatic decryption and encryption will take place once the key is changed.

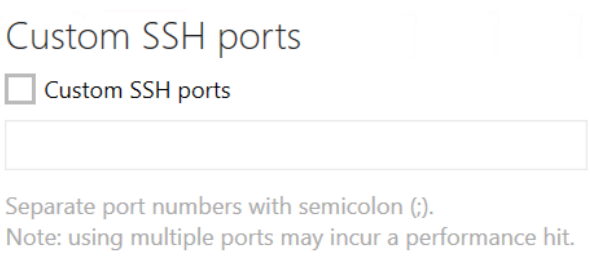
**WARNING**
The encryption key is saved in encrypted format in the RayVentory Scan Engine configuration file. For this the default encryption key is used. For increased security ensure that the permissions to the configuration file suit the respective security needs.

## Discovery

In the Discovery tab of the Settings it is possible to configure the ports that should be used for the discovery.

### Custom SSH ports

Check the **Custom SSH ports** checkbox to define custom ports that will be used for the discovery. If the checkbox remains unchecked, the default settings will be used.



When using custom ports, define those ports in the field located below the checkbox. If using multiple ports, the port numbers need to be separated by a semicolon (;).

**Note:**
Using multiple ports for the discovery may lead to performance degradation! If performance problems occur reduce the number of used ports.

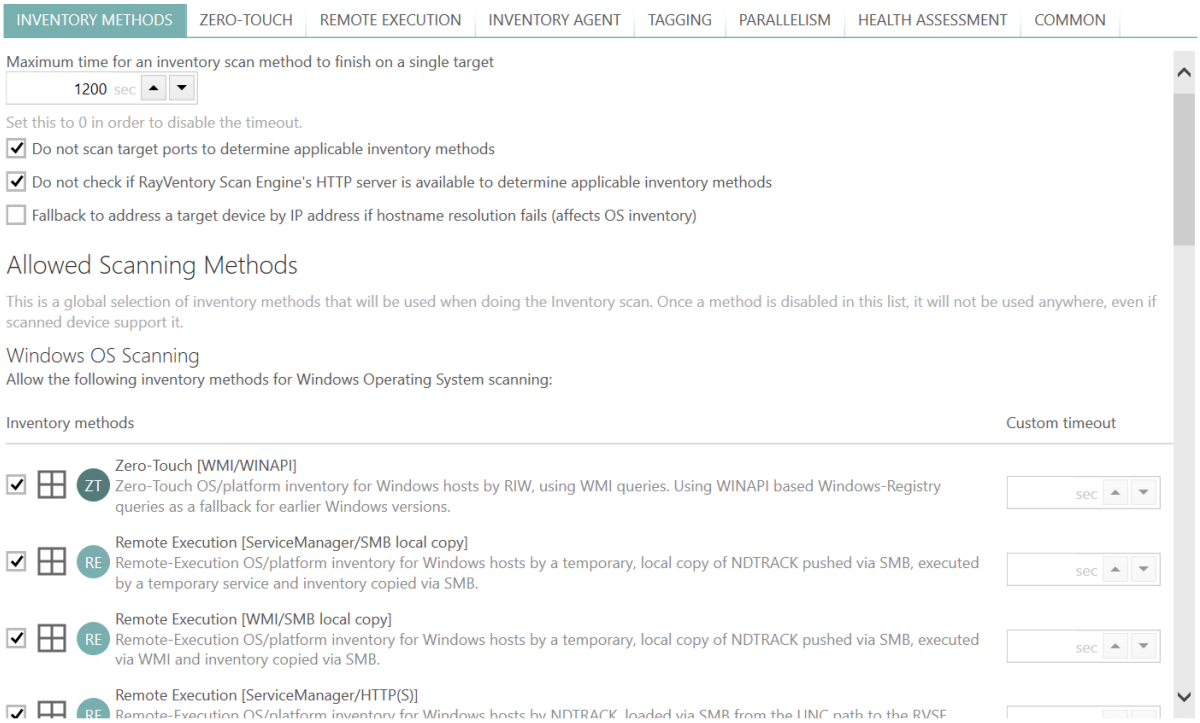
# Inventory

This tab contains eight subtabs that control which inventory methods are active and defines the settings that are used by the Zero-Touch and the Remote-Execution inventory scans.

This is also the place where the configuration of Inventory Agent can be managed and created.

## Inventory Methods

This tab is used to configure which inventory methods are globally enabled or disabled.




RayVentory Scan Engine supports up to 20 different methods which differ in:

- ... where the work is being done (zero-touch from remote machine, or remote execution on the actual target machine).
- ... the type of the Operating System (different set of techniques for Windows and Unix).
- ... the way results are received back (via HTTP, file share, file access etc.).
- ... the way in which remote execution is triggered (portable executable from share, a service, file access etc.).

The description next to each icon contains information about the target device family (in form of icon and textual description, either Windows or Unix), its name, and main characteristics of how it is actually working and where potential security pitfalls may lie. Within every group, the methods are sorted from what is agreed to be the "safest" method to some more invasive methods.

As a general rule:

- The methods described as "Zero-Touch" have minimal or zero impact on the target system. They usually connect to the target device via publicly available APIs or services and extract the data they need. No RayVentory Scan Engine process are executed on the host system. Methods from this family are less "intrusive" than remote execution, but may sporadically be limited by functionality of underlying APIs, services, and exposed endpoints. As a rule of thumb, you should try to start with zero-touch methods and verify whether the incoming results are satisfactory and move to the remote-execution in the next step, should some devices or networks require them.
- Methods described as "Remote Execution" may potentially have impact on the target system to some varying degree. Depending on the method type, some of them may try to install a service and almost all of them execute scan utilities bundled with RayVentory Scan Engine. After each execution a necessary clean-up is done (for example removing the temporary service and / or files), but the machine may be impacted in one or more ways just because of some processes being physically started on it. While these methods are considered more "intrusive", they tend to deliver better results as they have better access to the necessary resources and databases.



**Note:**
You can read more about differences between the different methods in the following chapter: Inventory Methods Overview. You may review them with your system network administrator to find out which methods should be allowed or disallowed in your environment.

Enabling or disabling any method is done by a mouse click:

- Tick a checkbox next to the name of the method to enable it.
- Untick it to disable it.

The icon and badge are dimmed if the given method is disabled.

Once the method is disabled here, it will no longer be considered by any inventory scan, even if the target device supports it. For example, if you disable remote execution on Windows and Unix from this screen, only zero-touch methods will be considered for your devices.

## Timeout per method

It is possible to define a timeout, which - regardless of global settings - will be then used for a particular family of methods. To define it, enter the required value into the **Custom timeout** column. Leave the column empty to not use any method-specific timeout.

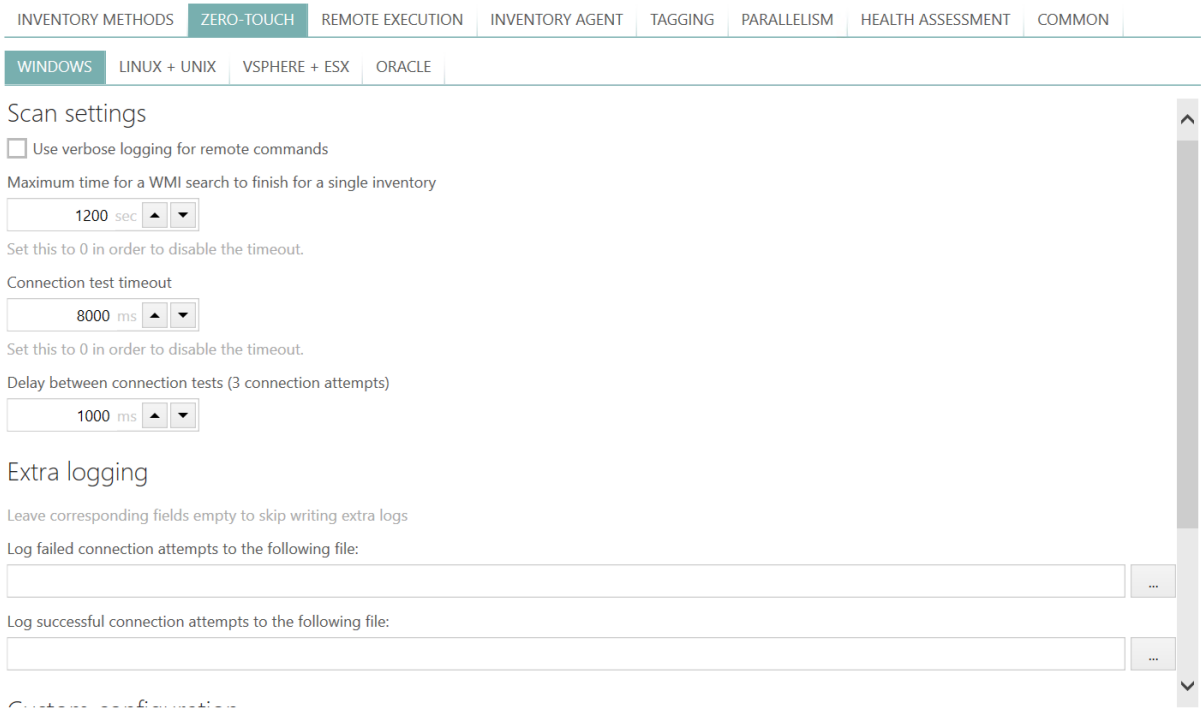
# Zero-Touch

This tab contains four sub-tabs that control how Zero-Touch inventory is being executed on the following platforms:

- Windows
- Linux + UNIX
- vSphere + ESX
- Oracle

## Windows

This tab controls the Zero-Touch scan settings for Windows machines.




### Scan Settings

- **Use verbose logging for remote commands**
  When this options is active, extra information about executed WMI queries are logged to the program log and / or console. You can use this option for the troubleshooting of custom Zero-Touch Windows scans.

- **Maximum time for the inventory scan to finish on a single computer**
  The value (in seconds) which denotes the timeout for a single operation. For a task to finish successfully, it must return to the caller within the specified time range. Setting this value to 0 means that RayVentory Scan Engine waits indefinitely for the results.

- **Maximum number of scanned computers by a single task**

Within the concept of RIW, this number denotes how many computers can be scanned in parallel by a single task.

- **Connection test time out**
  This number (expressed in milliseconds) denotes the timeout for a connection attempt. If the machine does not answer within this time range, the computer is considered to be offline or unavailable.

- **Delay between connection tests**
  This denotes the delay between repetitions of unsuccessful connection attempts. RayVentory Scan Engine performs up to 3 attempts before returning a failed result if none of the attempts succeeded.

## Extra Logging

- **Log failed connection attempts to the following file**
  Specifies the extra logging path for failed connection attempts. New entries are appended to the bottom of the file content.



**Note:**
This option is not equivalent to the various RayVentory Scan Engine log options.

- **Log successful connection attempts to the following file**
  Specifies the extra logging path for successful connection attempts. New entries are appended to the bottom of the file content.



**Note:**
This option is not equivalent to the various RayVentory Scan Engine log options.

## Custom Configuration

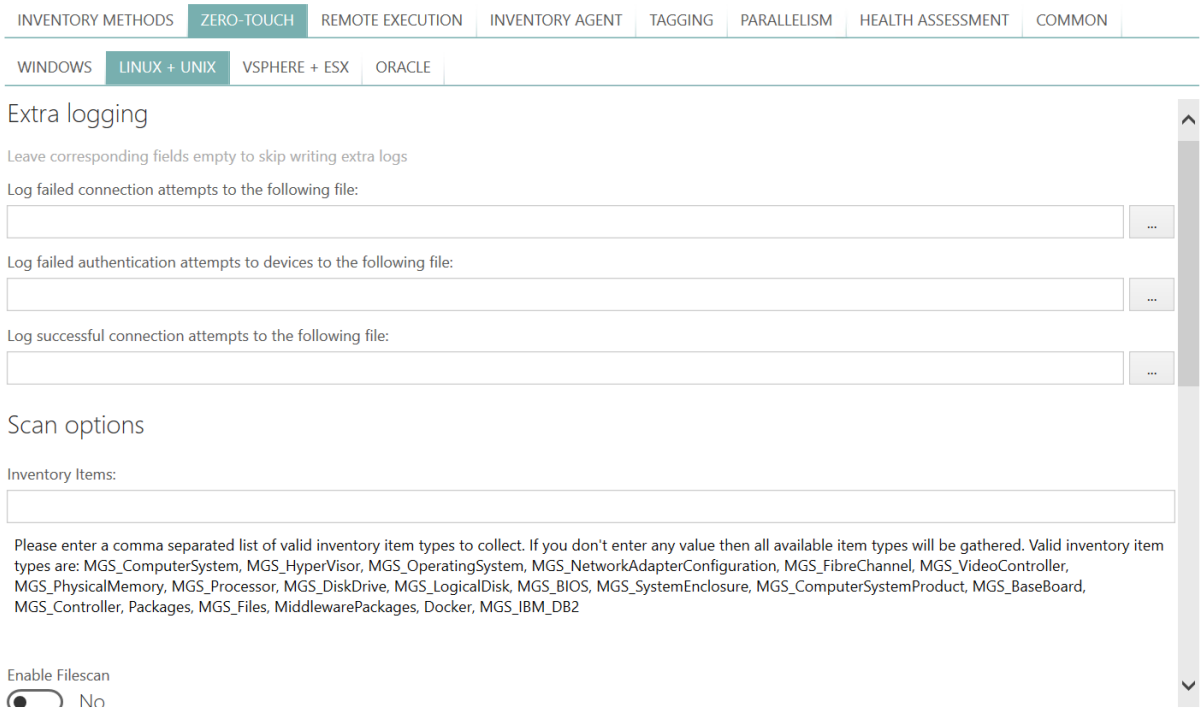
- **Custom inventory configuration file**
  This extra configuration file defines the custom scanning options. Configuring custom scans is a complex tasks, which is described in the chapter Custom Windows Scans.

## Options for Legacy Systems

- **When reading the registry, prefer WINAPI over WMI**
  When active, this indicates that RayVentory Scan Engine should prefer the WINAPI strategy over WMI for querying for registry entries. The option is designed for legacy systems where WMI is unavailable.

### Linux + UNIX

This tab controls the Zero-Touch scan settings for Linux machines.




## Scan Settings

- **Maximum number of scanned computers by a single task**
  Within the concept of RIU, this number denotes how many computers can be scanned in parallel by a single task.

- **Maximum time for the inventory scan to finish on a single computer**
  The value (in seconds) which denotes the timeout for a single operation. For a task to finish successfully, it must return to the caller within the specified time range. Setting this value to 0 means that RayVentory Scan Engine waits indefinitely for the results.

## Extra Logging

- **Log failed connection attempts to the following file**
  Specifies the extra logging path for failed connection attempts. New entries are appended to the bottom of the file content.

**Note:**
This option is not equivalent to the various RayVentory Scan Engine log options.

- **Log failed authentication attempts to devices to the following file**
  Specifies the extra logging path for failed authentication attempts. New entries are appended

to the bottom of the file content.

**Note:**
This option is not equivalent to the various RayVentory Scan Engine log options.

- **Log successful connection attempts to the following file**
  Specifies the extra logging path for successful connection attempts. New entries are appended to the bottom of the file content.

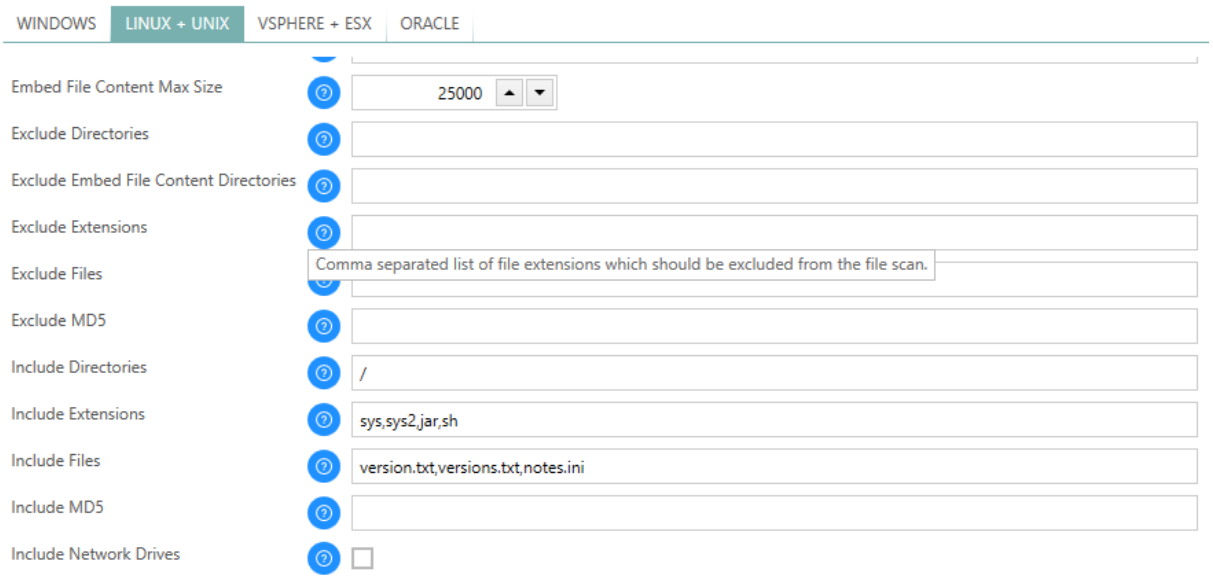
**Note:**
This option is not equivalent to the various RayVentory Scan Engine log options.

## Scan options

- **Inventory items**
  A list of comma-separated items of items to collect.
- **Enable Filescan**
  Activate this switch to enable file scanning on UNIX devices.

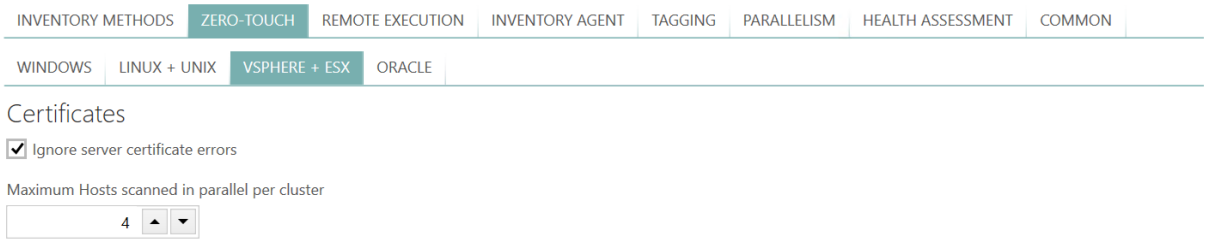
## File Scan Options

This section controls file scan on UNIX systems. Each setting has a dedicated help text, which can be revealed upon hovering with mouse on the question mark icon. To change these settings, enable file scan first.



## vSphere + ESX

This tab controls the vSphere / ESX Zero-Touch scan settings.




### Ignore Server Certificate Errors

When this option is enabled, invalid, missing, or expired SSL certificates of the vSphere server will be ignored by the scanning agent. If this option is deactivated, then the certificate of the server (in case of HTTPS connection) will be checked and must be therefore: valid, trusted by trusted authority on the current machine, and not expired.

### Maxium Hosts Scanned in Parallel per Cluster

This option enables the user to limit the maximum number of ESX hosts scanned in parallel per target (cluster).

## Oracle

This tab is used to configure various Oracle-related paths.


### Java Executable Path

The full path to the java runtime executable. You can type the path directly, use the **...** button to pick a file from your host, or use the auto-detection by pressing the **Detect** option.

If the Java runtime path is not configured here, RayVentory Scan Engine tries to look for it anyway any time it needs it. By providing a custom value in this field, you can cover the following use

cases:

- There are multiple instances of Java and you want to use a specific one for Oracle tasks or...
- You have a "private" instance of Java which is not registered in the system-wide location and you want to use that instance.



**Note:**
All Oracle-related methods require that Java is available to the RayVentory Scan Engine scan utilities, otherwise these method will fail.

### Oracle Review Lite Script Path

The full path to the Oracle Review Lite script, which gets executed when performing an audit on Oracle databases. You can type the path directly or use the **...** button to pick a file from your host.

### Oracle Database Feature Usage Statistics Script Path

The full path to the Oracle Database Feature Usage Statistics script, which gets executed when performing an DBFUS on Oracle databases. You can type the path directly or use the **...** button to pick a file from your host.

### SQL *Plus Executable Path

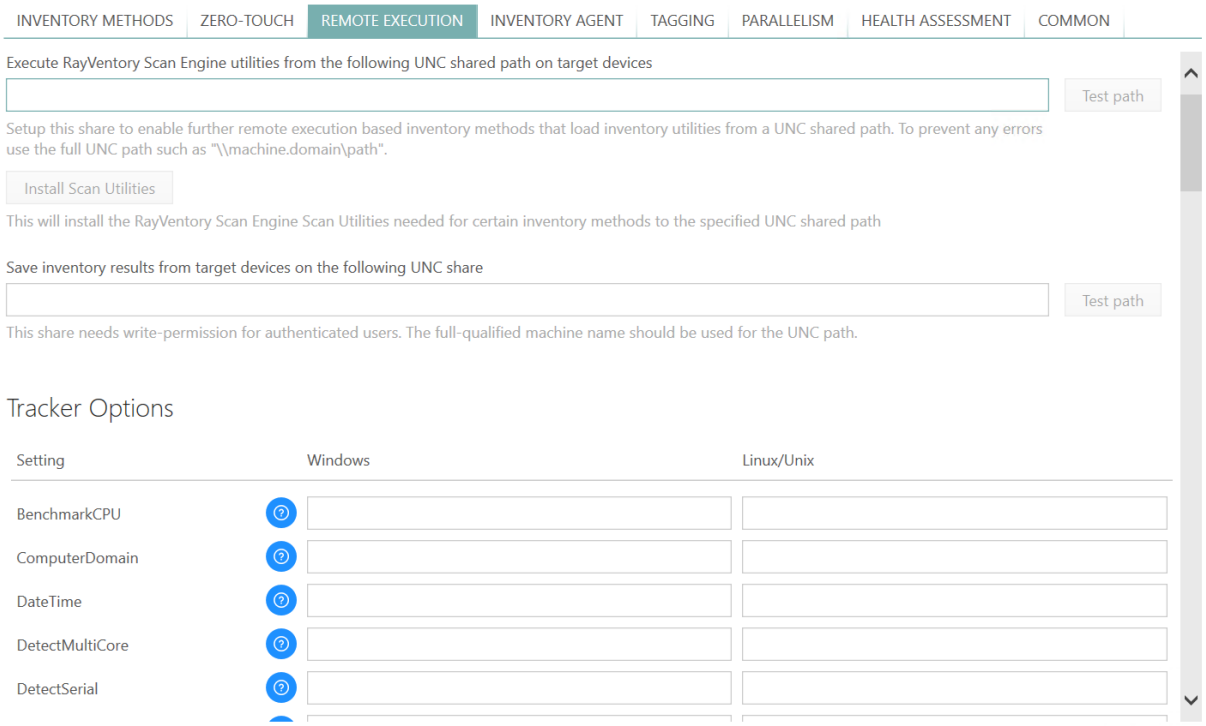
The full path to the SQL *Plus executable. This path is required to perform audit tasks (see more on that in the following chapter Oracle Audit).

### Ignore SIDs

The list of semicolon separated regular expressions, used to determine which DB instances (based on their SIDs) will be ignored during OracleDB Inventory/Discovery operations. The defaults should be reasonable for most of use-cases, but you may finetune your results by adding some more excluded items here.

## Remote Execution

This tab contains settings which are relevant for all remote-execution-based scans.




### Execute RayVentory Scan Engine Utilities from the Following UNC Shared Path on Target Device

This is a UNC share path from which RayVentory Scan Engine is available. The default installation does not install the tools, it is the responsibility of the administrator to set up a file share which is accessible by the machines scanned by the **Remote Execution Inventory Scan**.

Once the path is entered in the text field, the content of the share should be initialized. This process copied the required files (scan utilities) into the share specified by the user. To initialize the share, press **Install Scan Utilities**. This is a onetime operation, you can simply copy over the content of the folder and reuse it for another scan tools source paths.

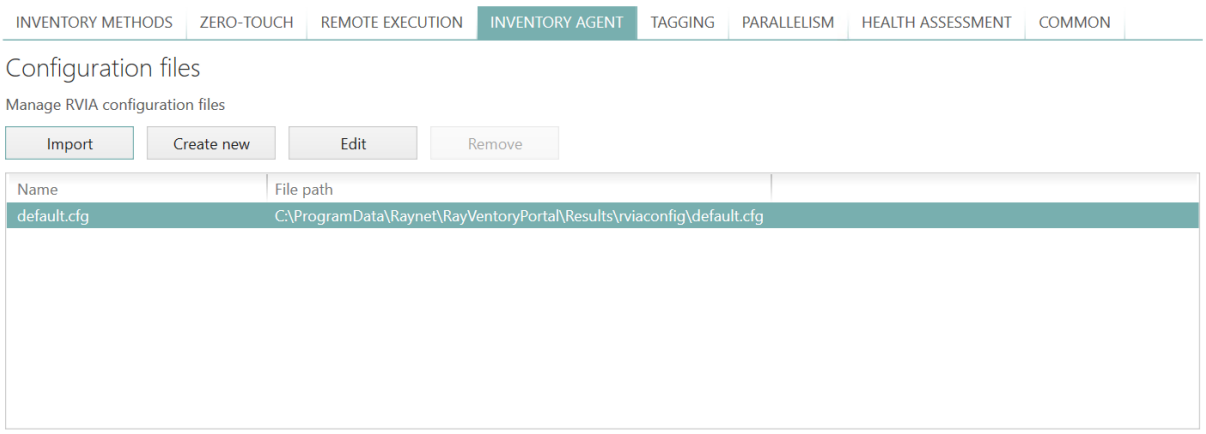
### Save inventory results from Target Devices on the Following UNC Share

This is the place where scanned devices save their inventory results. The default installation does not set up a file share for uploads, it is the responsibility of the administrator to configure it.

**Note:**
Certain inventory methods may require that all of some settings from this page are configured. Failing to configure them will render these inventory methods incompatible. You can find more details about the dependencies and requirements for each method in the following chapter: Inventory Methods Overview.

## Inventory Agent

These settings control the configuration, which is served for the purpose of the RayVentory Inventory Agent.




RayVentory Scan Engine serves default values, which are reasonable for most of simple use cases. To further control the settings, new configurations may be added.

## Creating or importing new configuration

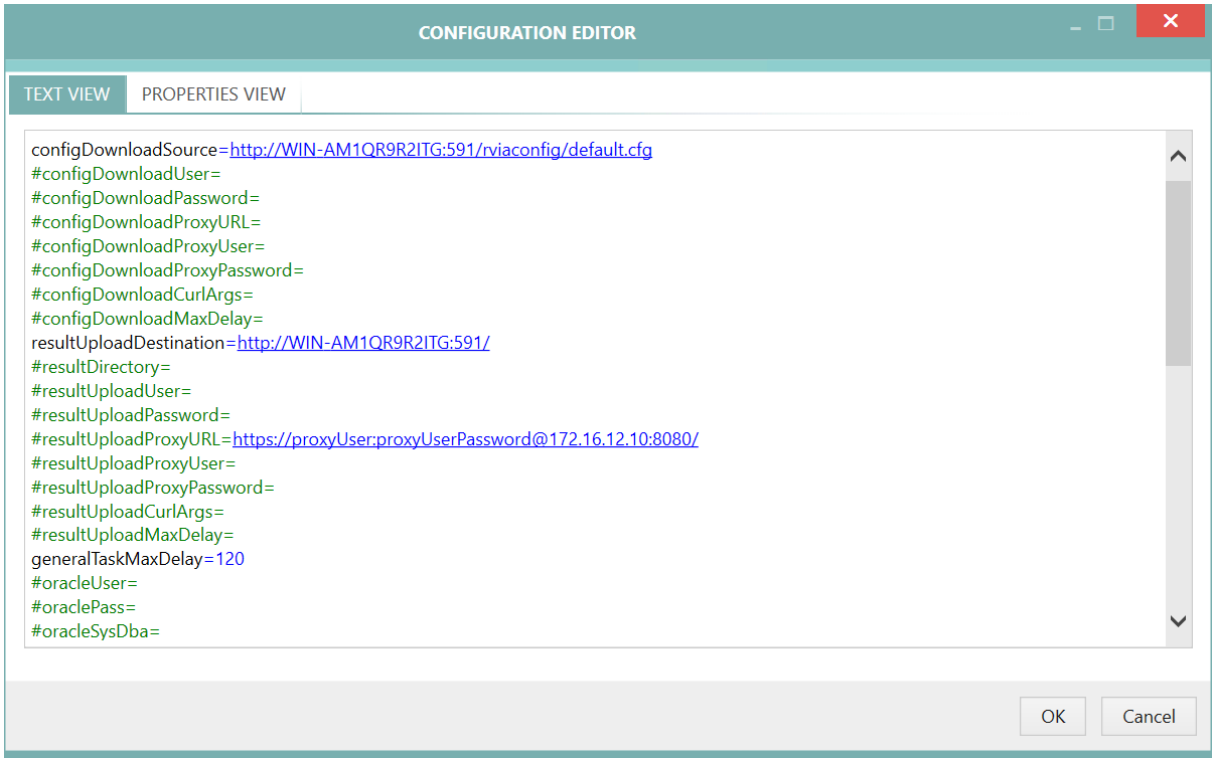
To add a new configuration:

1. Go to the **Settings** screen > **Inventory** > **Inventory Agent**
2. Press **Import** to import an existing file, or **Create new** to create a new one from a template
3. Edit the content of the file. The default values have many commented lines, which you can activate by removing the leading # character.
4. Once the configuration is finished, press **OK** to save the changes.

Chapter Inventory Agent (sections Configuration and command-line) explains in details how to control getting the settings by the Inventory Agent.

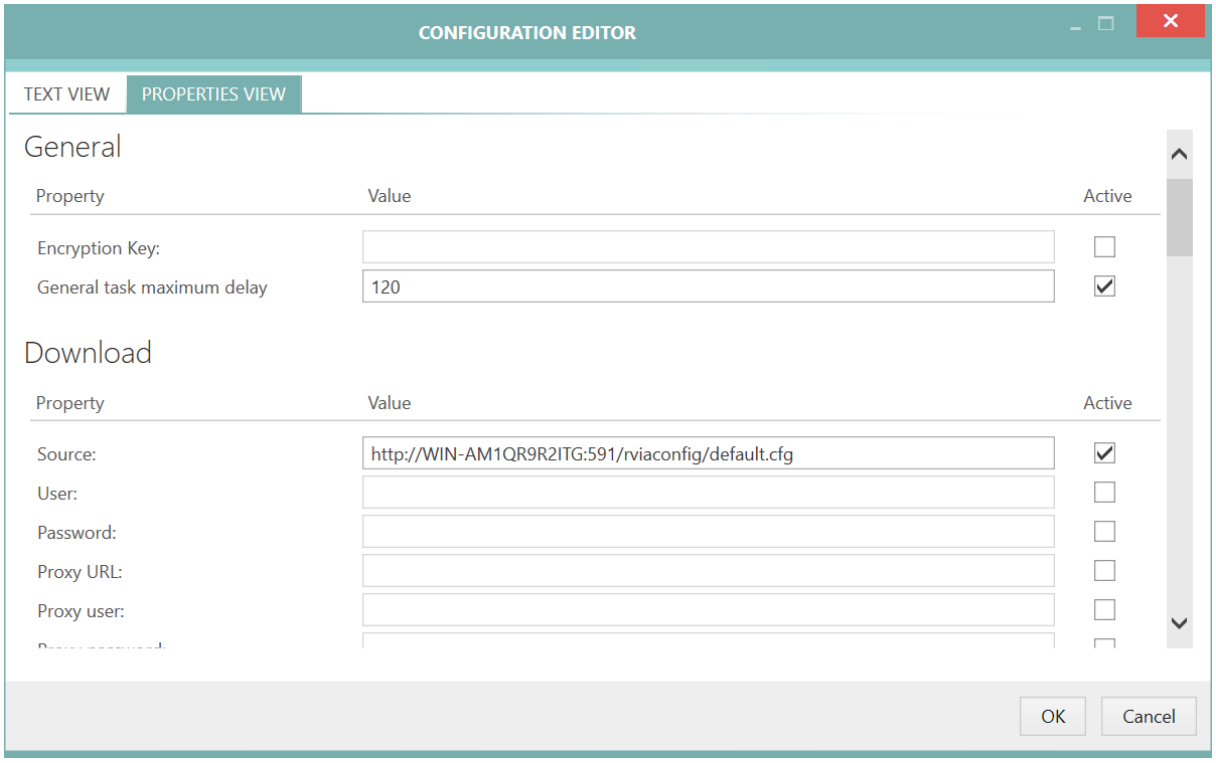
## Editing configuration with WYSIWYG editor

It is possible to toggle between raw editing (default, great for experienced users and for quick copy and paste between different dialogs and config files) in the **TEXT VIEW** tab and a more structured form which guides through the options exposed by the Inventory Agent in the **PROPERTIES VIEW** tab.




TEXT VIEW tab

In the **TEXT VIEW** tab the configuration files can be edited directly by changing the content of the file.



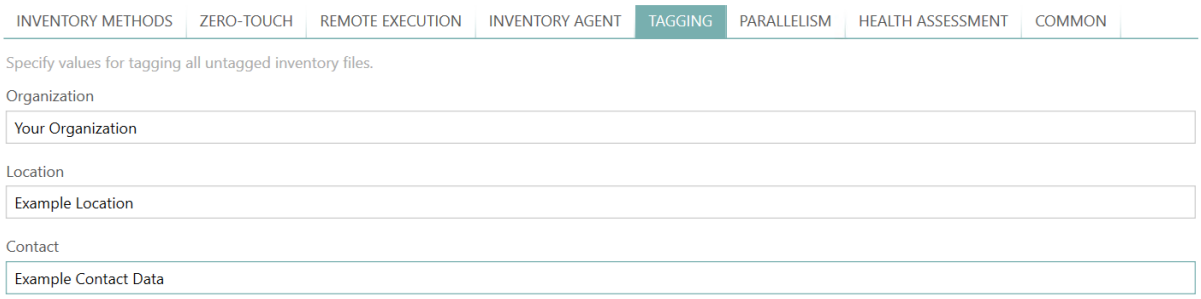

PROPERTIES VIEW tab

In the **PROPERTIES VIEW** tab the configuration file can be edited using the easily accessible GUI. The options in the **PROPERTIES VIEW** are divided into different sections. Options can be switched on and off by checking or unchecking the Active checkbox.

The different parameters that are used can be found in the chapter Parameters.

## Tagging

Tagging allows users to brand all incoming or outgoing NDI files with specific meta information, which can be used later on to identify the origin of each scan. This advanced feature is useful in case of multi-tier architecture of several RayVentory Scan Engine instances or many Distribution Servers / Upload locations.




This **Settings** screen contains three fields to be configured by the user:

### Organization

This is the name of the organization performing the scan. You can enter any value that uniquely identifies your company, branch, or division.

### Location

The location name. You can enter any value that uniquely identifies your physical, geographical, or infrastructure related location.

### Contact

The information about the Point-Of-Contact for that particular scan. This can be a name, e-mail address, telephone number or any other value that uniquely identify the person responsible for the scan results.

The information provided by the user are merged with a few dynamically generated values. All NDI files receive the following information:

- Automatically created values:
  - The date and time of the scan (property `RVP-Time`).
  - The name of the server that initiated the scan (property `RVP-Server`).
  - The version of RayVentory Scan Engine instance that initiated the scan (property `RVP-Server`).

- Values specified by the user (only non-empty values are being written):
  - The name of the organization (property `Organization`).
  - The name of the location (property `Location`).
  - The name of the contact (property `Contact`).

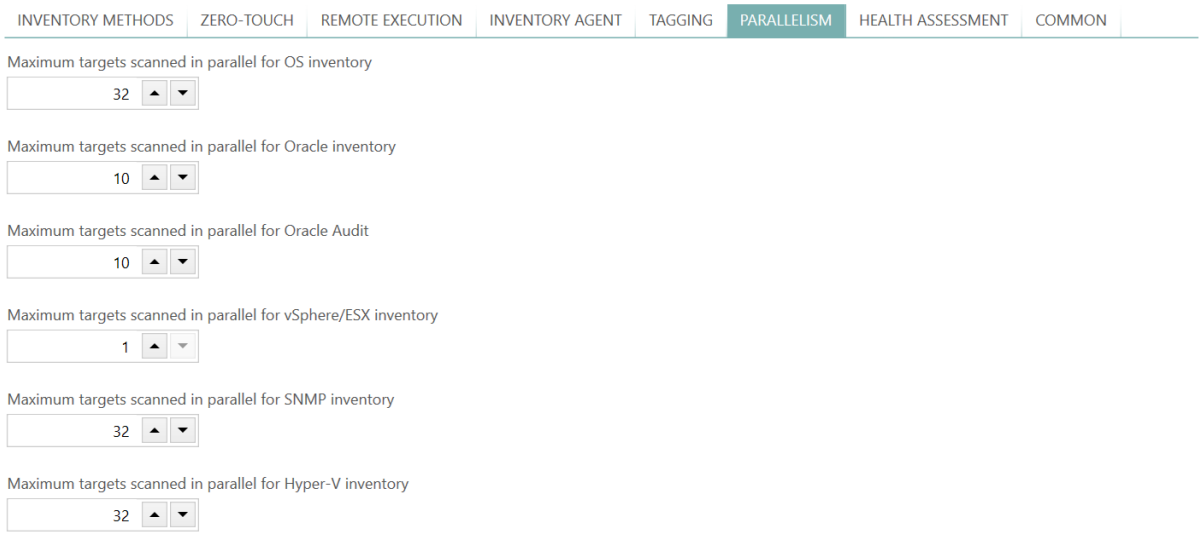
All values are saved in a special entry (Hardware), having `MGS_Identity` as its class name and `MGMT API` as its evidence.

### Conflict Handling

In case of processing of files that have already been branded by any other RayVentory Scan Engine, the original branding is kept intact. This way, the results always contain fine grained information about the low-level entity that initiated the scan.

## Parallelism

Parallelism enables the user to set the maximum number of targets processed in parallel during an inventory run for each inventory type.



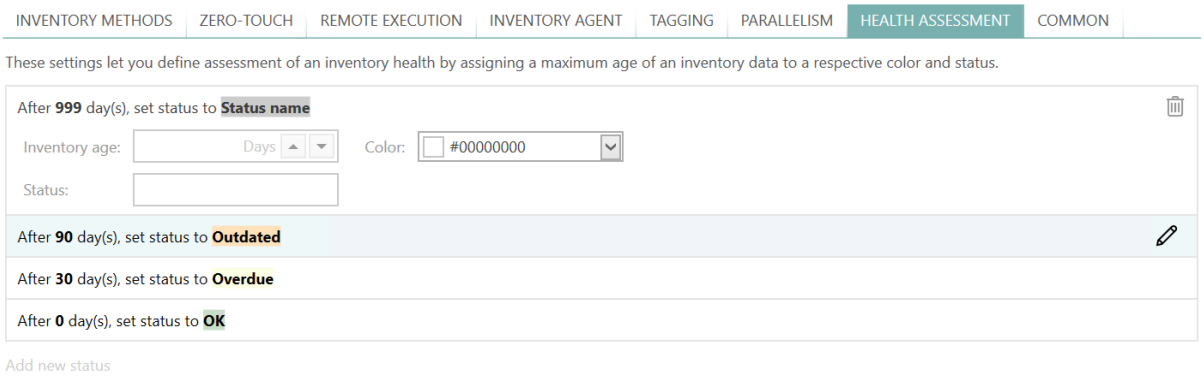

## Health Assessment

This tab allows you to edit custom rules for assessing the status of an inventory.

The default rules are:

- A device that has been scanned within last 30 days is considered to be up-to-date (status OK).
- Otherwise, if a device has been scanned not later than 90 days ago, it is considered to be outdated (status Outdated).
- Finally, if the last time a device has been scanned is more than 90 days ago, then the result is

considered to be obsolete (status Obsolete).




The list is read from the top to the bottom, and once the criteria are fulfilled the processing stops, and the result is determined from the defined name and color.

You can customize, remove or add custom statuses and rules.

## In order to adjust an existing status...

1. Click on the status
2. Enter the adjusted details (the lower bound of days, name and the color).
3. Accept the changes by pressing OK.

## In order to add a new status

1. Make sure no status is currently being edited.
2. Press **Add new status** link
3. Enter the required details (the lower bound of days, name and color).
4. Accept the changes by pressing OK.

## In order to remove an existing status...

1. Click on the status
2. Click on the **Trash** icon on the right side

**Note:**
It is not possible to manually sort the list. Once the lower-bound of days is changed, the list will be sorted automatically to reflect the processing order of how RayVentory Scan Engine determines the status for each device.

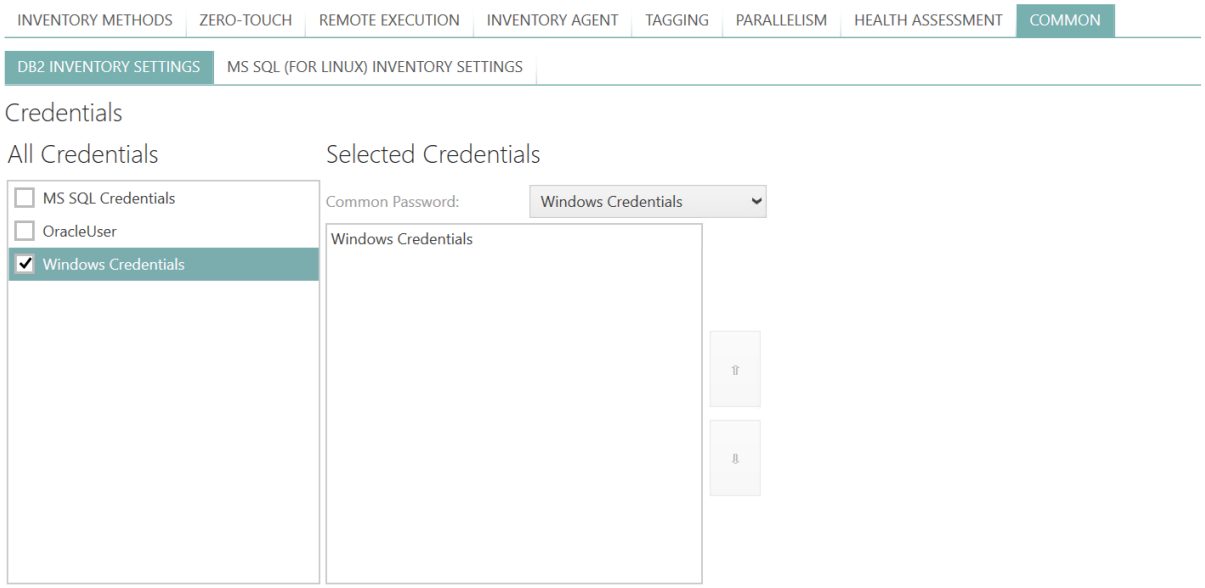
## Common

The Common tab in the Inventory section of the Settings is divided into further tabs.

- the **DB2 Inventory** tab
- the **MS SQL (For Linux) Inventory** tab

### DB2 Inventory

In the **DB2 INVENTORY SETTINGS** tab the credentials will be used for linux zero touch (RIU) or ndtrack scans and for windows zero touch (RIW) and ndtrack scans.




The tab is divided into two sections.

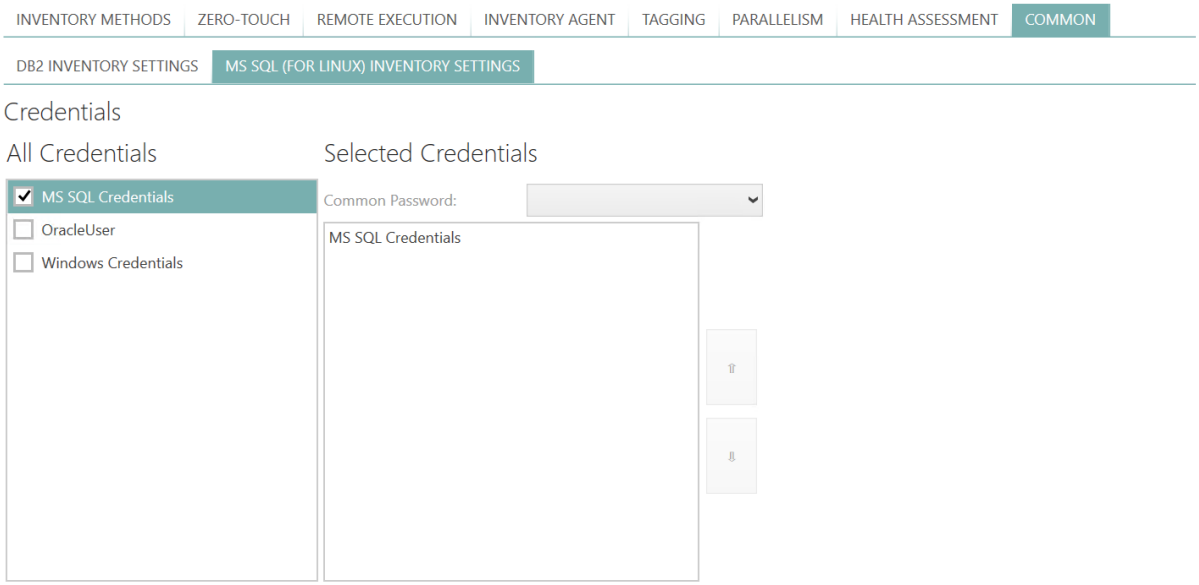
The first section consists of the **All Credentials** list, which contains a list of all the credentials that are stored in the Credential Store. Next to each of the credentials there is a checkbox which can be used to mark the credentials that will be used for the scans.

The second section consists of the **Selected Credentials** list, which contains the list of the selected credentials. Furthermore, above the list there is the **Common Password** dropdown menu. The dropdown menu offers all selected credentials for selection.

**Be aware:**
Newly added credentials will only be available in the **All Credentials** list after a restart of RayVentory Scan Engine

### MS SQL (For Linux) Inventory

In the **MS SQL (FOR LINUX) INVENTORY SETTINGS** tab the credentials which will be used for linux zero touch (RIU) or ndtrack scans.




The tab is divided into two sections.

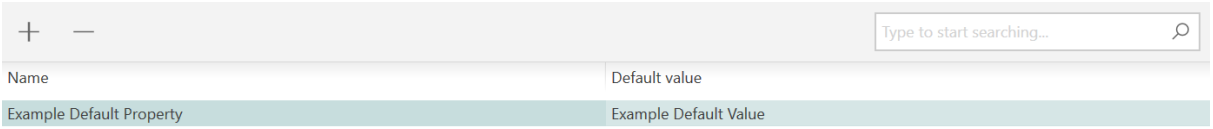
The first section consists of the **All Credentials** list, which contains a list of all the credentials that are stored in the Credential Store. Next to each of the credentials there is a checkbox which can be used to mark the credentials that will be used for the scans.

The second section consists of the **Selected Credentials** list, which contains the list of the selected credentials. Furthermore, above the list there is the **Common Password** dropdown menu. The dropdown menu offers all selected credentials for selection.

**Be aware:**
Newly added credentials will only be available in the **All Credentials** list after a restart of RayVentory Scan Engine

# Custom properties

This screen manages the custom properties, which are visible on device-base in the Devices screen.



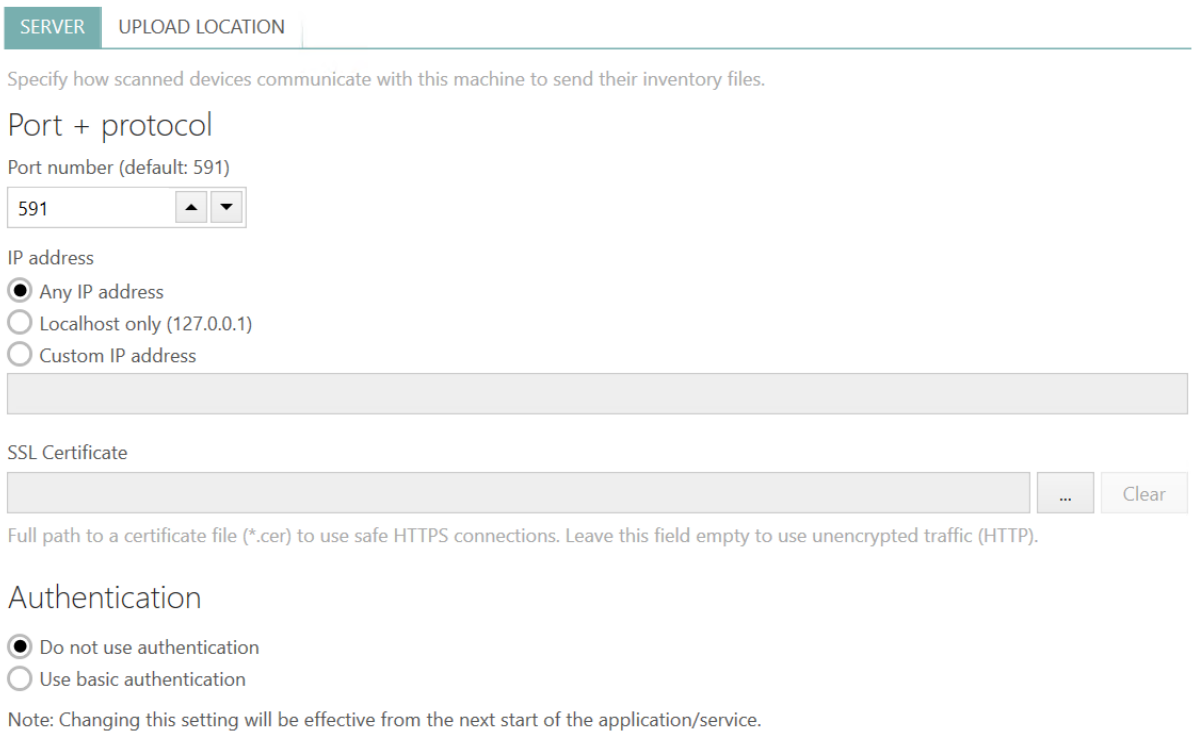

For more details about how to work with custom properties, refer to the following chapter: Defining custom attributes.

# HTTP Services

This tab contains two sub-tabs that control the incoming (Server tab) and outgoing (Upload Location tab) uploads.

## Server

These settings control how the built-in HTTP Upload Server is working.




### Port Number (Default: 591)

This is the port number that the HTTP server is listening on. You can set it on any value not used by any other process. Typical values are `80` for HTTP connections and `443` for HTTPS, but to avoid any conflicts the default that RayVentory Scan Engine uses is set to `591`.

### IP address

This is a setting which controls the addresses on which the built-in HTTP server listens. The default option (before version 12.2 the only available one) is to listen on any IP address, but this can be changed to only listen on `localhost` (`127.0.0.1`) or on a specific IP address.

### SSL Certificate

In order to set up an encrypted traffic between the target devices and RayVentory Scan Engine, provide a full path to the `.cert` file containing your SSL certificate. The certificate authority has to be trusted on clients connecting to the server. Once a certificate is selected, HTTPS will be the default protocol for incoming connections. In order to revert back to unencrypted traffic, press the **Clear** button to make the path empty.

### Authentication

Configuration of authentication options is not relevant for daily tasks started from RayVentory Scan Engine. These settings should be reviewed and applied when using custom scans triggered from local copies of `ndtrack.exe` combined with upload options.

The communication between target devices and RayVentory Scan Engine can optionally require authentication. RayVentory Scan Engine supports two modes:

- No authentication
- Basic authentication

Selecting **Use basic authentication** requires that the target devices send user name and encoded password over the wire. If **No authentication** is chosen, then any device can upload its data to the local HTTP Upload server.



**Note:**
Basic authentication does send the credential in an unencrypted, encoded form. HTTPS connections should be used to secure the connection.

**Be aware:**
Depending on whether you installed the HTTP Upload Service or started a portable version of RayVentory Scan Engine, changing the HTTP settings requires a restart of the Service (installed instance) or of the main application (portable).

**Also be aware:**
After changing the credentials for the basic authentication, it will still be necessary to change these settings in the `.cfg` files used by the inventory agent.

## Upload Location

RayVentory Scan Engine is able to push software and hardware inventory results to a parent instance. The following parent instances are supported:

- RayVentory Scan Engine
- RayManageSoft Reporting Location
- RayVentory Reporting Location

RayVentory Scan Engine also supports the following upload locations:

- Local directory
- UNC folder path
- FTP address

These settings enables the user to control which parent instance is in use and how to upload data to it.

Setting up these values is not relevant if you do not intend to upload the data to any parent

server (for example if your RayVentory Scan Engine instance is already self-contained root).

### URL to Upload Inventory Files

The full URL to the parent upload location (see supported types section). This value should include the protocol and the port number. RayVentory Scan Engine uses this path for uploads of inventory files (`.ndi`). If you do not know this value, ask your administrator.

### URL to Upload Discovery Data

The full URL to the parent upload location (see supported types section). This value should include the protocol and the port number. RayVentory Scan Engine uses this path for uploads of discovery files (`.disco`). If you do not know this value, ask your administrator.

### Upload Discovery Data during Manually Triggered Inventory Upload

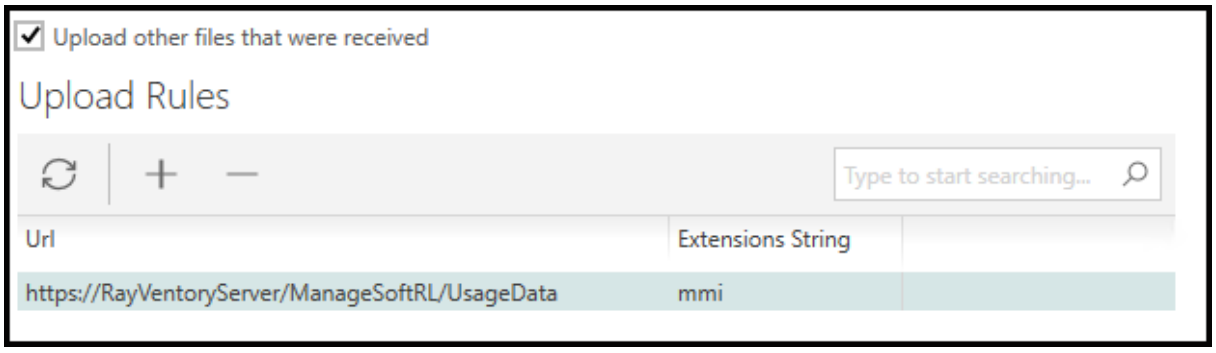
If this checkbox is checked, `.disco` files will be uploaded to parent URL at the beginning of every inventory upload. If this checkbox is unchecked, the files will not be uploaded automatically during a manually triggered inventory upload.

### Delay in seconds between discovery and inventory upload

The delay after uploading discovery data and before uploading inventory data. The delay is supposed to ensure that if the upload target was a RayVentory Server with direct import enabled for discovery and inventory then the server has enough time to process the discovery data before inventory import, in order to avoid missing inventory status updates for the network devices based views and reports.

### Upload rules

By default, RayVentory Scan Engine only uploads `.ndi` files to the HTTP server. For these files, the built-in routing is applied to ensure that they are correctly parsed. The **Upload Rules** option can be used to also upload other file types. In order to enable the upload of other files, check the **Upload other files that were received** checkbox. Click on the **+** button to open the **Add Upload Rule** dialog. Enter the upload location into the **URL** field and the extension of the files to the **File Extensions** field. Either click on the **Apply** button to add the rule to the **Upload Rules** and enter the next one or click on the **OK** button to add the rule and close the dialog. Upload rules can be created for `.mmi` and `.xml` files.




As `.gz` files will always be uncompressed by RayVentory Scan Engine, therefore no additional upload rules for compressed files are necessary.

**Ignore Server Certificate Errors**
If this checkbox is checked and HTTPS is used as the communication protocol, any SSL-related errors will be ignored.

**Note:**
The connection will be not secure if SSL-errors are ignored.

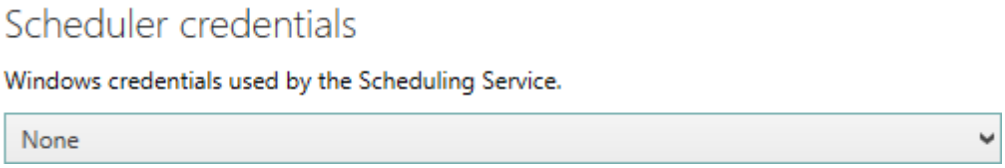
**Certificate for Authentication against the Upload Endpoint**
A custom certificate (`.cer`) which is used for encrypted connections via HTTPS protocol. If no value is provided, the standard Windows Certificate Store will be used.

**Use Credentials from the Credential Store for Authentication to the Upload Endpoint**
In case the parent location requires basic authentication, you can select the required credentials from the list. Credentials can be defined in the Credential Store screen.

**Proxy Settings**
Optional proxy settings used for communication with the parent upload location.

# Task Scheduler

This screen enables the user to configure which credentials are used by the built-in Scheduling Service.

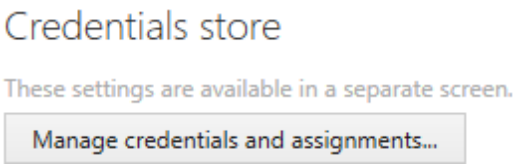



You can leave the default option **None** in order to force RayVentory Scan Engine to use the current user identity.

If you want to execute the tasks impersonating another user, first ensure that his credentials (type Windows) are configured in the Credential Store and then select them from the drop-down list in this view.

# Credential Store

This view is merely a shortcut to the **Credential Store Manager**, which is available in the **Devices + Services** screen.




Press **Manage credentials and assignments...** to go to the configuration grid.

# Scheduling

RayVentory Scan Engine offers a schedule for automation of operations.

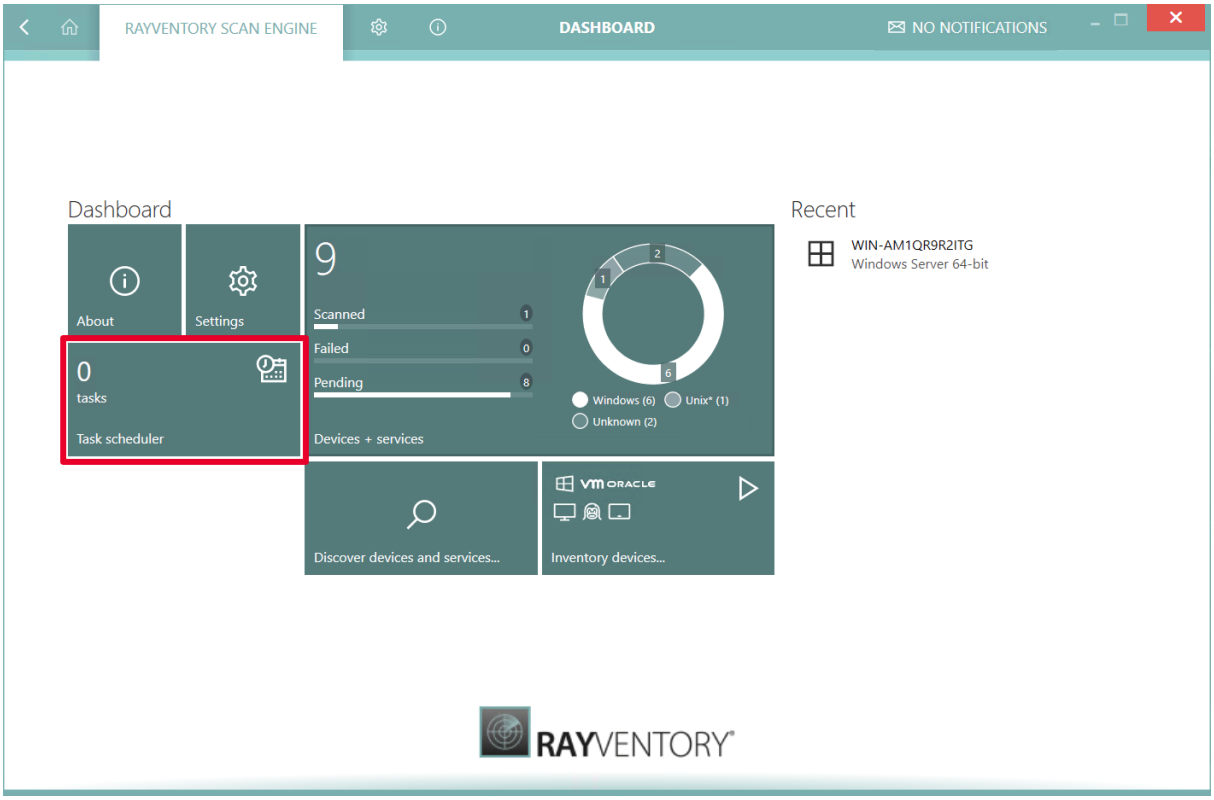
The following tasks and operations can be scheduled:

- Remote OS Inventory
- Oracle Inventory
- vSphere / ESX Inventory
- Discovery (with automatic inventory option)
- Inventory Upload

During the setup of RayVentory Scan Engine, the schedular is installed as a service. It executes scheduled tasks in the background even if the RayVentory Scan Engine application is not running.

During the start-up of the RayVentory Scan Engine application, if no running scheduling service is found and if the scheduling service cannot be installed with the permissions of the currently logged in user, a warning is shown in the Notification Center and an instance of the scheduler is run within the RayVentory Scan Engine application itself. This instance of the scheduler will be stopped if the RayVentory Scan Engine application is closed. Therefore, the execution of the schedule will also be stopped when exiting RayVentory Scan Engine.

In the Schedule screen it is possible to create and edit the schedule. The Schedule screen can be opened by clicking on the Schedule tile which is located on the dashboard. The tasks which are listed in the schedule can also be triggered manually.

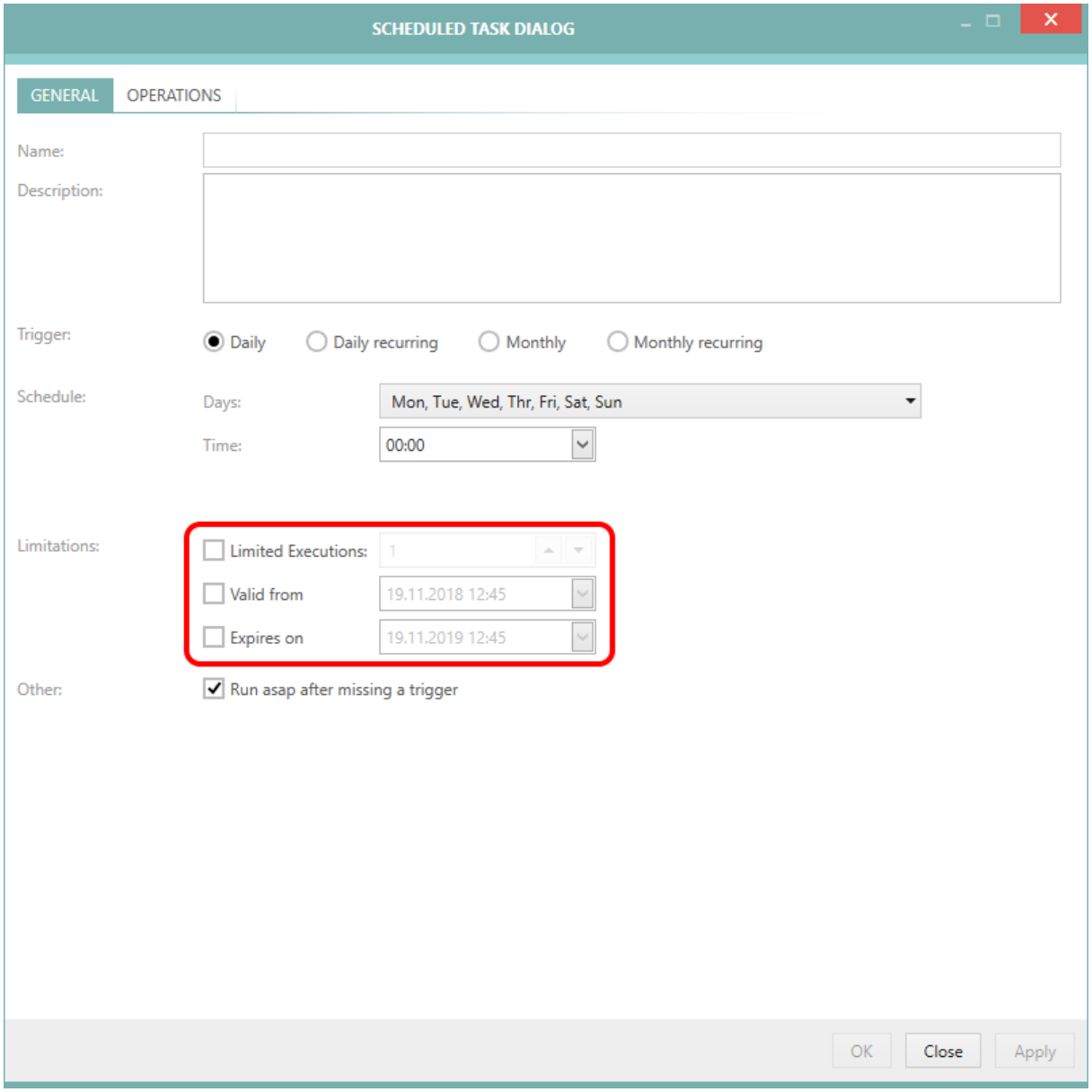



# Triggers

Each scheduled task is triggered by one of four triggers:

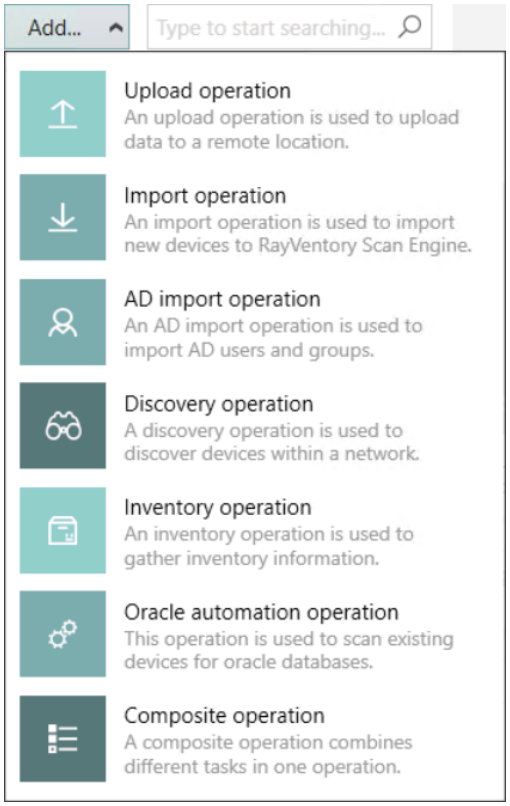
- **Daily**: Once on a certain day at a certain tie for all or certain days of the week.
- **Daily recurring**: Multiple times a day, delayed by a certain number of minutes between the end of and the next run, for all or certain weekdays, from a certain time of day to a certain time of day.
- **Monthly**: Once a month on a certain day, on a certain time, on all or certain weekdays.
- **Monthly recurring**: Multiple times on a certain day of month, delayed by a certain number of minutes, between the end of a previous run and the next run, for all or certain weekdays from a certain time of day to a certain time of day.

For all triggers a date from when to begin and a date when the trigger will expire can be configured. It is also possible to set a limit of the number of times that a task is run. These settings are optional.




# Scheduled Operations

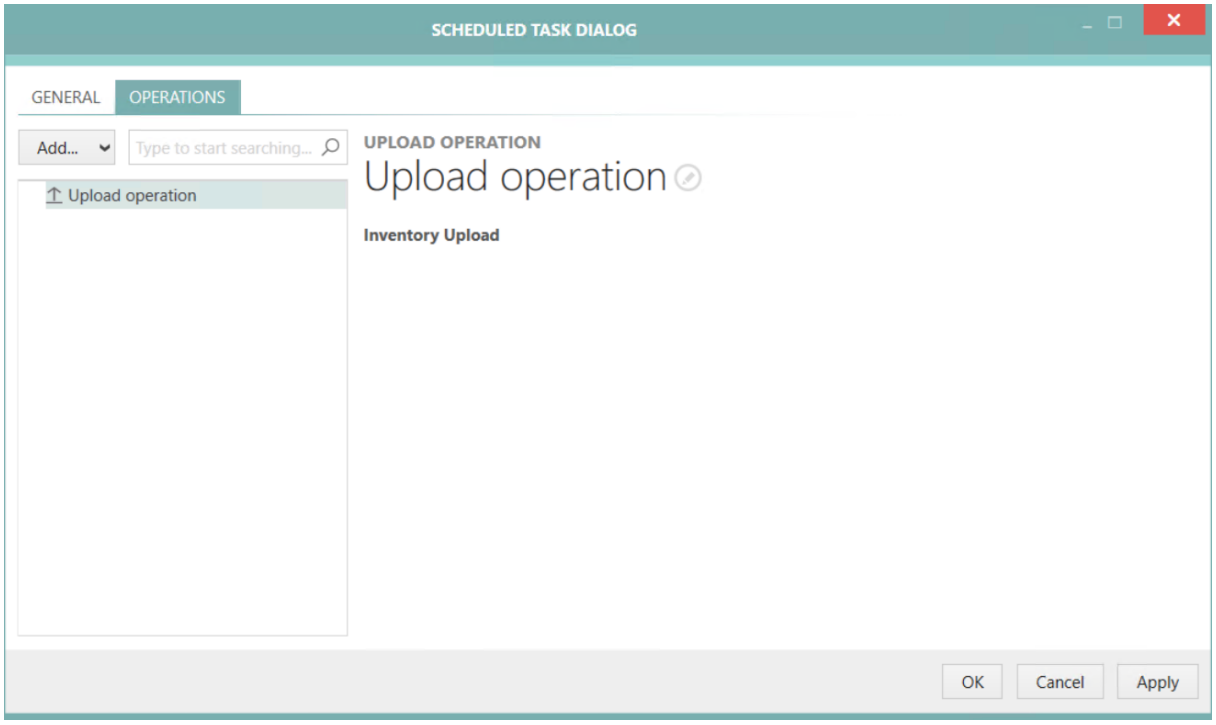
For each action or operation it is necessary to set a caption. A scheduled task consists of at least one action or operation. The following operations can be added to a scheduled task:




- Upload operation
- Import operation
- AD import operation
- Discovery operation
- Inventory operation
- Oracle automation operation
- Composite operation

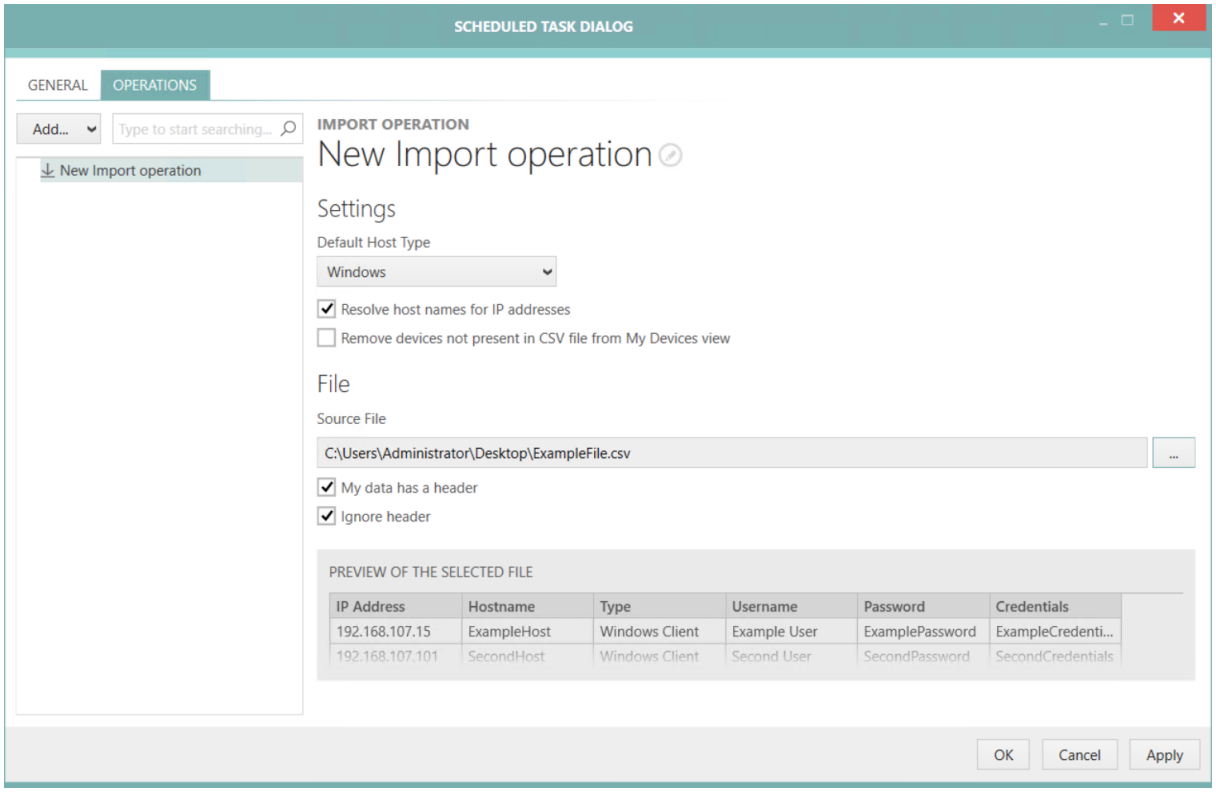
## Upload Operation

An **Upload operation** is used to upload data to a remote location. There are no parameters for the **Upload operation** available in the schedule user interface. The **Upload operation** will trigger an inventory upload identical to the operation that is triggered when the **Upload** tile in the **Notification Center** is being used.




## Import Operation

An **Import operation** is used to import new devices to RayVentory Scan Engine. The settings for the **Import operation** are divided into to areas. The **Settings** area that contains the settings related to the devices and the **File** area where the information about the `.csv` are specified and where a preview of the file is available.



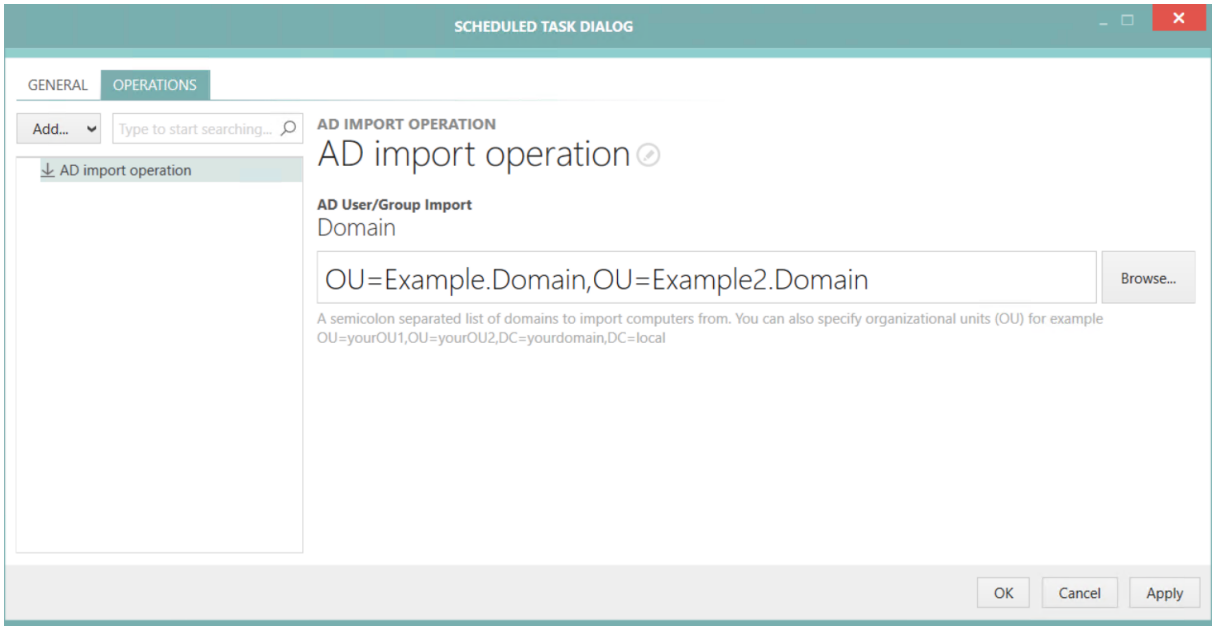

In the **Settings** area the default host type for the import can be defined. The **Default Host Type** dropdown menu contains the following options: **Windows**, **UNIX***, and **Unknown**. Furthermore, there is the option to automatically check for the hostname of a given IP address if the file does not contain the hostname and the option to automatically remove those devices which are not present in the file from the **My Devices** view.

In the **File** area the **Source File** field is used to select the file for the import. Click on the **Browse** [**...**] button on order to select the `.csv` file from the file browser. Once a file is selected, a preview of the file will be shown at the bottom of the window. Check the **My data has a header** checkbox if the `.csv` file is using a header. To completely ignore the header, check the **Ignore header** checkbox.

For further details on the available options and features refer to the Import Devices Wizard chapter.

## AD Import Operation

An **AD import operation** is used to import AD users and groups from the defined domains.




The domains can be defined manually by entering a list of domains separated by semicolons into the Domain field. It is also possible to select the domains available in Scan Engine by clicking on the **Browse...** button and selecting the target domains from the domains listed in the **Active Directory Browser**.

## Discovery Operation

The **Discovery operation** can be used to run the **Active Directory** import, the network scan, the service discovery / probing, and the automatic inventory in the same way as the discovery in the discovery screen, but the presentation differs.

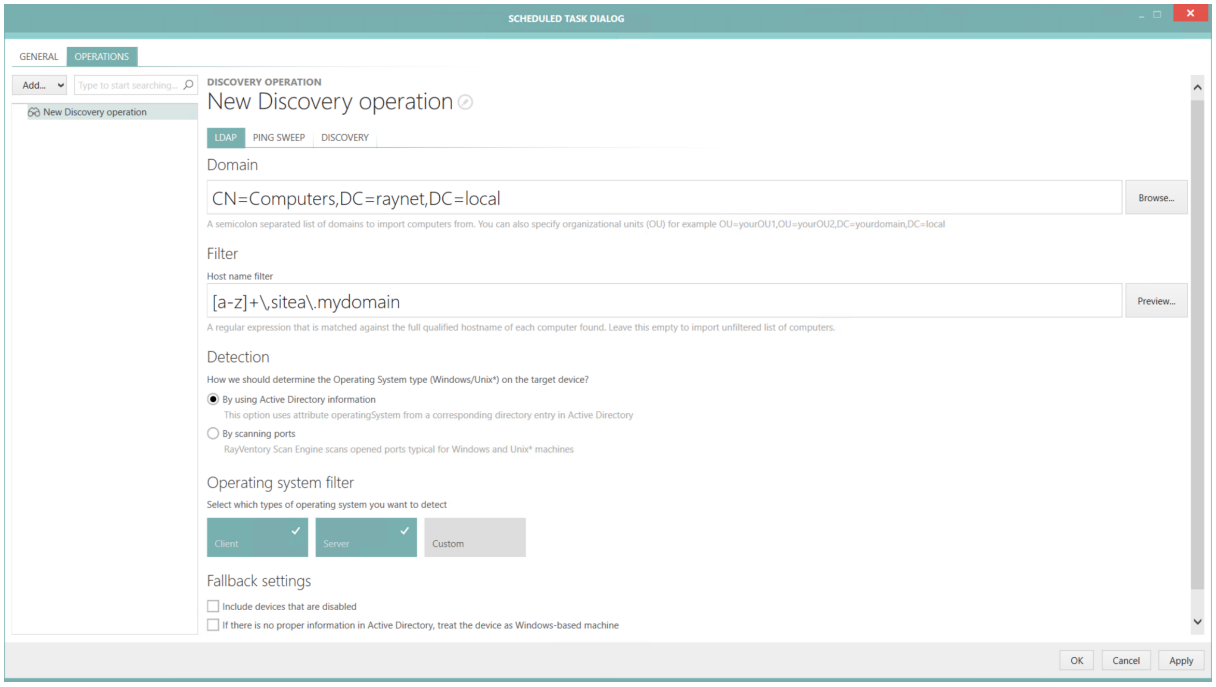
The dialog for the Discovery operation is subdivided into different tabs. The following tabs are available:

- LDAP
- Ping Sweep
- Discovery

For further details on the available options and features, refer to the Discovery chapter.

### LDAP

In the **LDAP** tab the settings for the discovery from the Active Directory can be configured.




The domains can be defined manually by entering a list of domains separated by semicolons into the **Domain** field. It is also possible to select the domains available in Scan Engine by clicking on the **Browse...** button and selecting the target domains from the domains listed in the **Active Directory Browser**.

In the **Filter** filter field a regular expression that is matched against the hostnames of the found computers can be entered. If an unfiltered list of computers is to be imported, the field can be left empty. Information about how to use regular expression refer to the Microsoft Documentation.

**Example:**
[a-z]+\.sitea\.mydomain

- [a-z]: all lowercase letters from "a" to "z".
- +: the preceding element can be matched one or more times.
- \.: matches a "."

Therefore this regular expression can be used to filter for everything that starts with letters only and ends with ".sitea.mydomain".

Select one of the radio buttons located under Detection in order to define how the OS type is being determined. If the **By using Active Directory information** radio button is selected, it is further necessary to select the target operating system. This options is hidden if the **By scanning ports option** is selected.

Furthermore the following fallback settings are available:

- **Include devices that are disabled** - Select this option to include disabled devices. If left unchecked, the devices that are disabled in Active Directory will not be imported

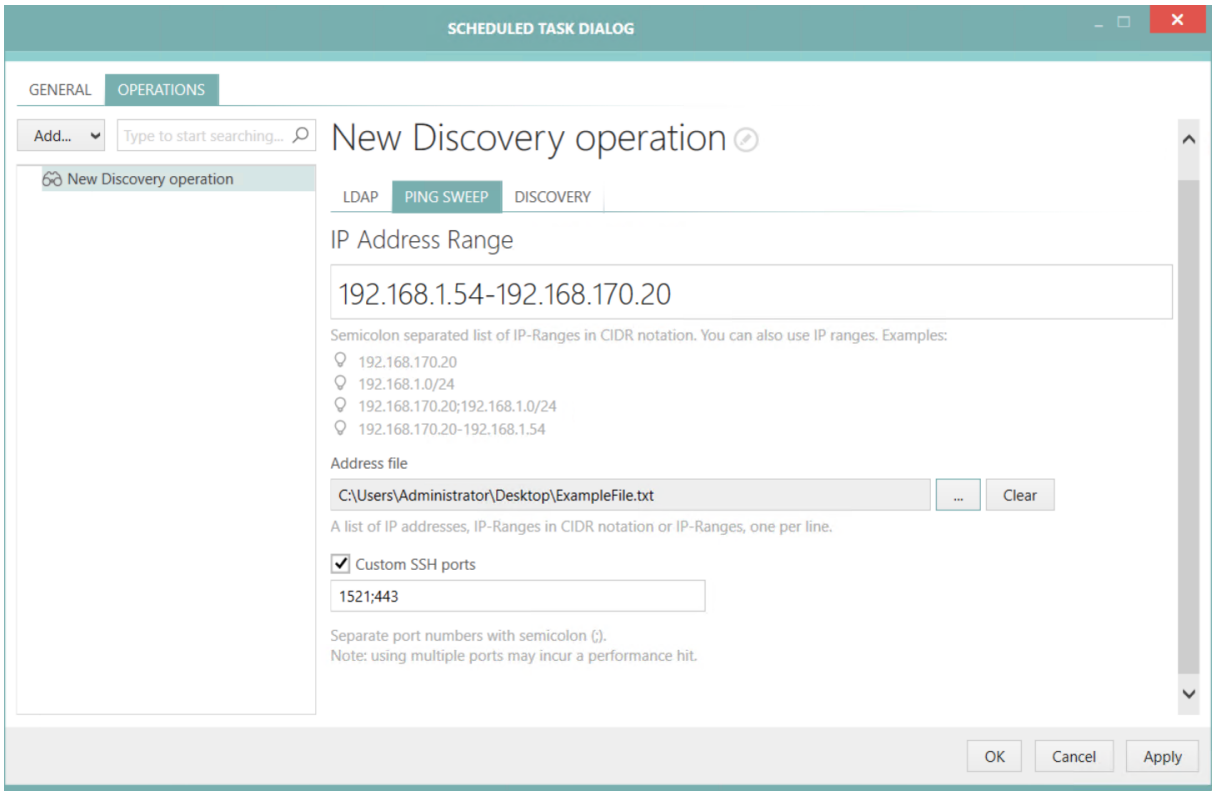
(recommended).

- **If there is no proper information in Active Directory, treat the device as Windows-based machine** - Select this option to automatically fallback to Windows, if the information in Active Directory is not sufficient to determine the device family. If left uncheck, the devices will be imported as **unkown** types.

For further details on the available options and features, refer to the Active Directory chapter.

### Ping Sweep

In the **Ping Sweep** tab, the settings for the discovery using a network scan can be configured.




In the IP Address Range field a list of IP addresses and IP-ranges separated by semicolon can be configured. The following notations are possible:

- CIDR notation (for example `192.168.170.0/24`),
- Single IP address (for example `192.168.170.21`),
- IP address range (for example `192.168.170.1-192.168.172.200`),
- Any combination of previous three separated by semicolon.

It is also possible to import a prepared list containing IP addresses and IP-ranges. Add the file to the Address file field by clicking on the **Browse** [**...**] button. If a file has already been added, the field can be cleared by clicking on the **Clear** button. This button is only available if a file has been added.

If a non-standard SSH port should be used for non-Windows devices, check the **Custom SSH**

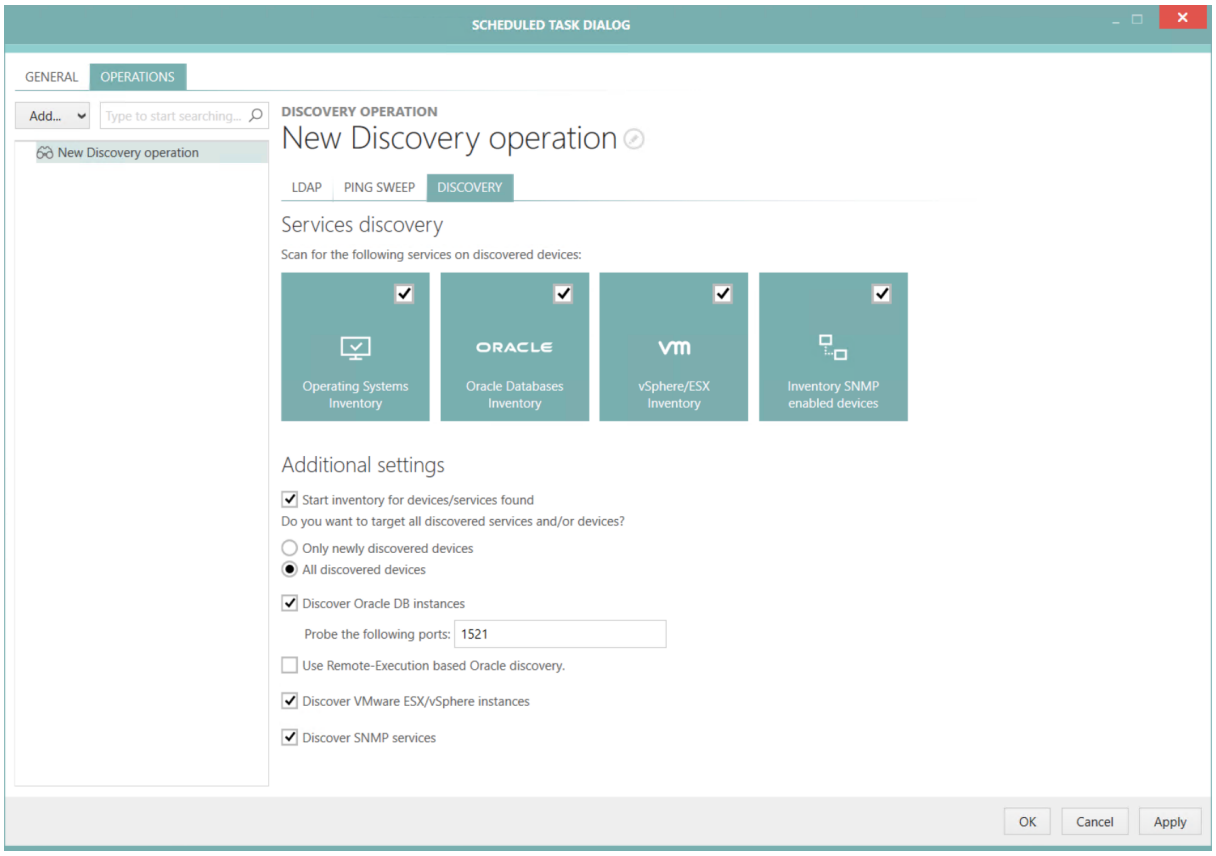
**ports** checkbox and enter the ports into the field located underneath the checkbox. More than one value can be entered by using a semicolon.

**Note:**
Providing more than one port may have negative impact on the scan speed.

For further details on the available options and features, refer to the Network Scan chapter.

### Discovery

In the **Discovery** tab, the settings for the discovery of services on devices can be configured.




Select the services that should be included into the scan by clicking on the respective tile. The following services are available:

- Operating Systems Inventory
- Oracle Databases Inventory
- vSphere/ESX Inventory
- Inventory SNMP enabled devices

There are also additional settings available.

- **Start inventory for devices/services found**: Check this checkbox in order to start an

inventory for the devices and services that are found.

- **Do you want to target all discovered services and/or devices?**: This option is used to define if all discovered devices will be targeted or only those newly discovered. This can be defined by selecting one of the available radio buttons.
  - **Only newly discovered devices**: Select this radio button to only target newly discovered devices. Devices that were already known will be ignored.
  - **All discovered devices**: Select this radio button to target all devices that have been discovered.
- **Discover Oracle DB instances**: Check this checkbox in order to activate discovery for Oracle databases. Specify a port or a range of ports in the **Probe the following ports** field.
- **Use Remote-Execution based Oracle discovery.**: Check this checkbox in order to use remote execution for the Oracle discovery.
- **Discover VMware ESX/vSphere instances**: Check this checkbox in order to also discover VMware ESX/vSphere instances.
- **Discover SNMP services**: Check this checkbox in order to also discover SNMP services.

For further details on the available options and features, refer to the Services chapter.

## Inventory Operation

An **Inventory operation** is used to gather inventory information. There are different types of inventories that can be used for the **Inventory operation**. The following inventory types can be selected:

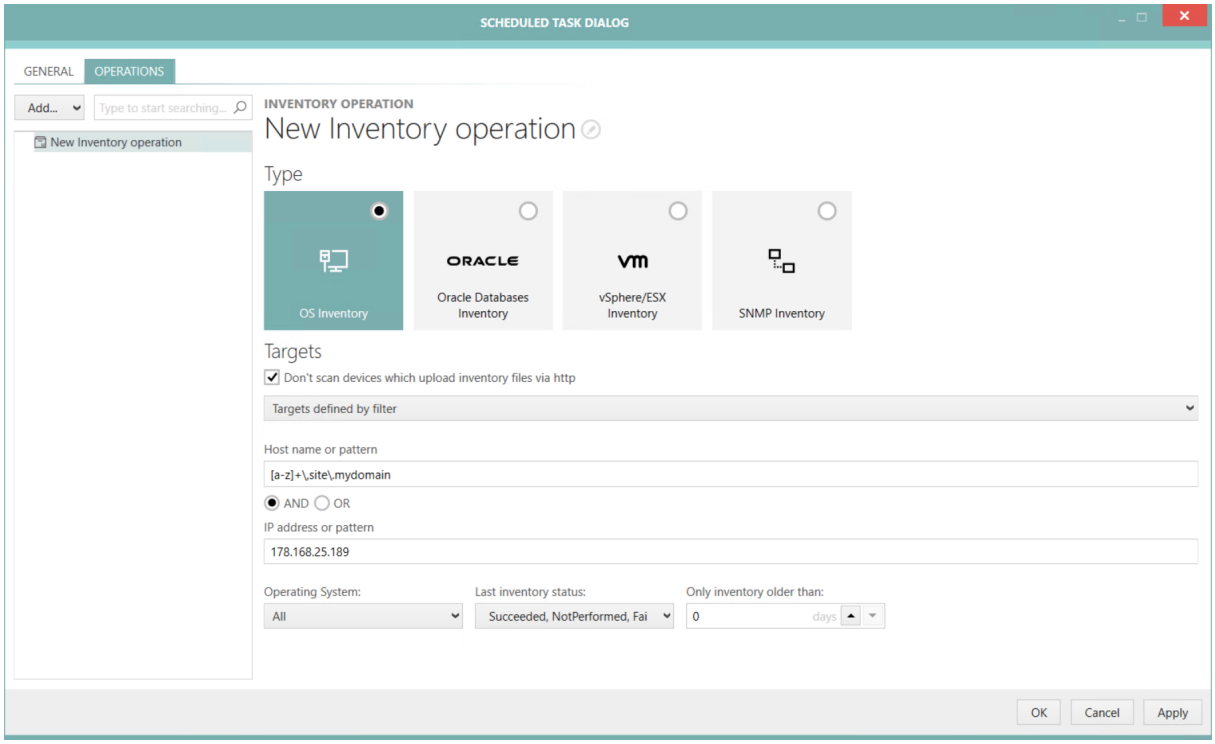
- OS Inventory
- Oracle Databases Inventory
- vSphere/ESX Inventory
- SNMP Inventory

### OS Inventory

A remote OS inventory is performed by the OS Inventory action. The target host for this operation is defined by one of the following options that can be configured using the dropdown menu underneath the **Don't scan devices which upload inventory files via http checkbox**:

- Targets defined by filter
- Targets defined by expression
- Targets defined by list
- Targets filtered by file

#### Targets Defined by Filter




For the **Targets defined by filter** option, the following settings are available:

- **Host name or pattern**: This field can contain a concrete host name or a regular expression. Information about how to use regular expression refer to the Microsoft Documentation.

**Example:**
[a-z]+\.sitea\.mydomain
- [a-z]: all lowercase letters from "a" to "z".
- +: the preceding element can be matched one or more times.
- \.: matches a "."

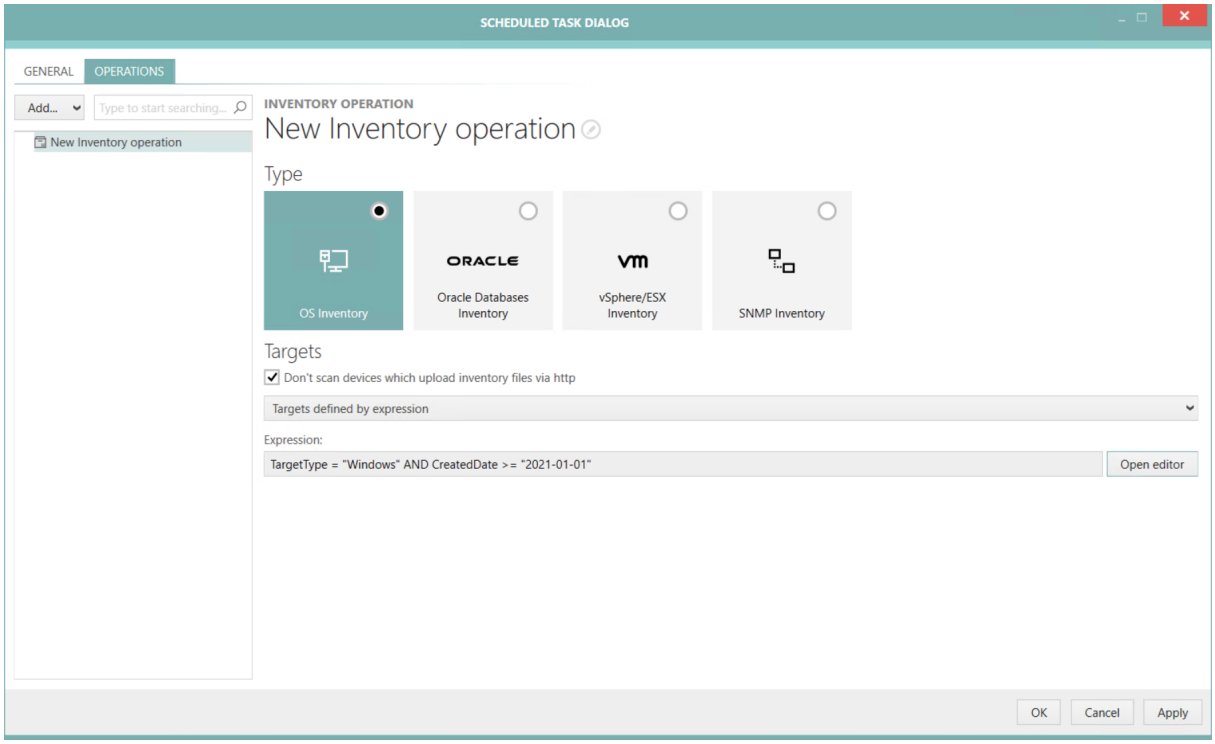
Therefore this regular expression can be used to filter for everything that starts with letters only and ends with ".sitea.mydomain".

The **AND** and the **OR** radio button can be used to defined if the Host name or pattern and the IP address or pattern fields are used together or separately.

- **IP address or pattern**: This field can contain a concrete IP address or a regular expression.

Furthermore, the following additional filter options are available.
- **Operating System**: This filter is used to filter after the operating system. It can be set to **All**, **Windows**, **UNIX***, and **Unknown**.
- **Last inventory status**: This filter is used to filter after the status of last inventory. It can be set to any combination of **Succeeded**, **NotPerformed**, or **Failed**.
- **Only inventory older than**: This filter is used to filter according to the age of the inventory. Configure the minimum age in days of the last inventory.

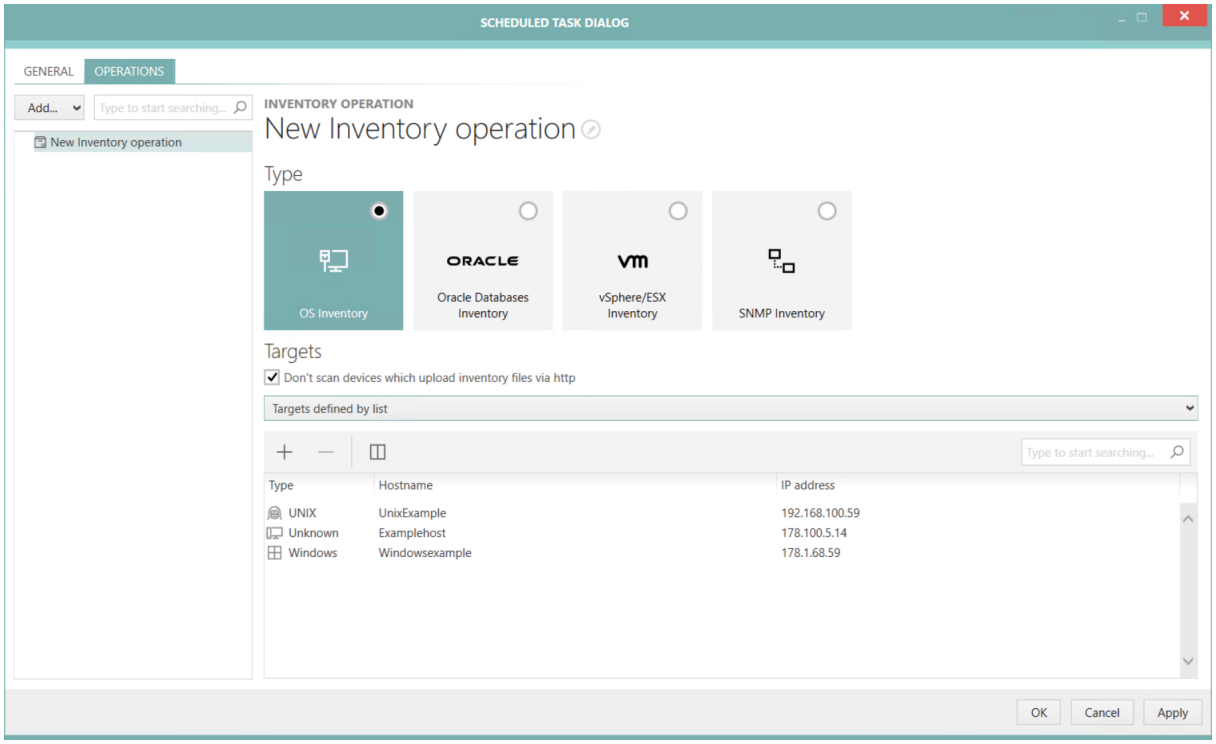
## Targets Defined by Expression




For the **Targets defined by expression** option, the following settings are available:

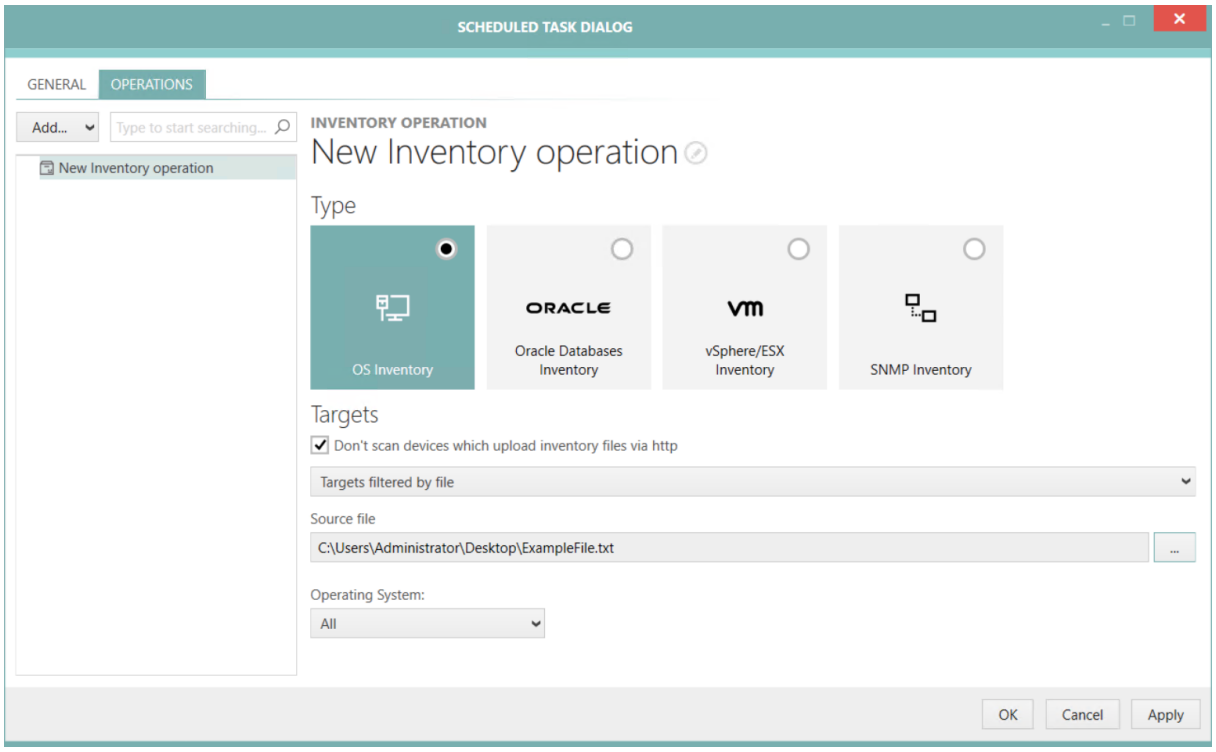
- **Expression**: Enter the expression that is used as filter into this field. The expression can be created by using the **Advanced Filtering** editor. To use the editor, click on the **Open editor** button located behind the **Expression** field.

## Targets Defined by List




The **Targets defined by list** option allows to pick one or multiple hosts from the **Devices** list. In order to add a new host, click on the **+** button located at the top of the list. Hosts can be removed by selecting on host and clicking on the **-** button.

## Targets Filtered by File



For the **Targets defined by file** option, the following settings are available:

- **Operating System**: Use this filter to define which operating systems will be considered. It can be set to **All**, **Windows**, **UNIX***, and **Unknown**.
- **Source file**: Select a file containing the IP addresses or the hostnames by clicking on the **Browse** [...] button and selecting the target file (either `.txt` or `.csv`).

A simple text file is sufficient as source file. Values in the file can either be IP addresses, hostnames or both. The values need to be separated by either a semicolon or a (NewLineCharacter). The following examples are valid entries for the file:

**Example 1:**

```
Hostname1;Hostname2
```

**Example 2:**

```
192.168.170.1;192.168.170.2
```

**Example 3:**

```
Hostname1
Hostname2
```

**Example 4:**

```
192.168.170.1
192.168.170.2
```

**Example 5:**

```
192.168.170.1;Hostname1
```

**Example 6:**

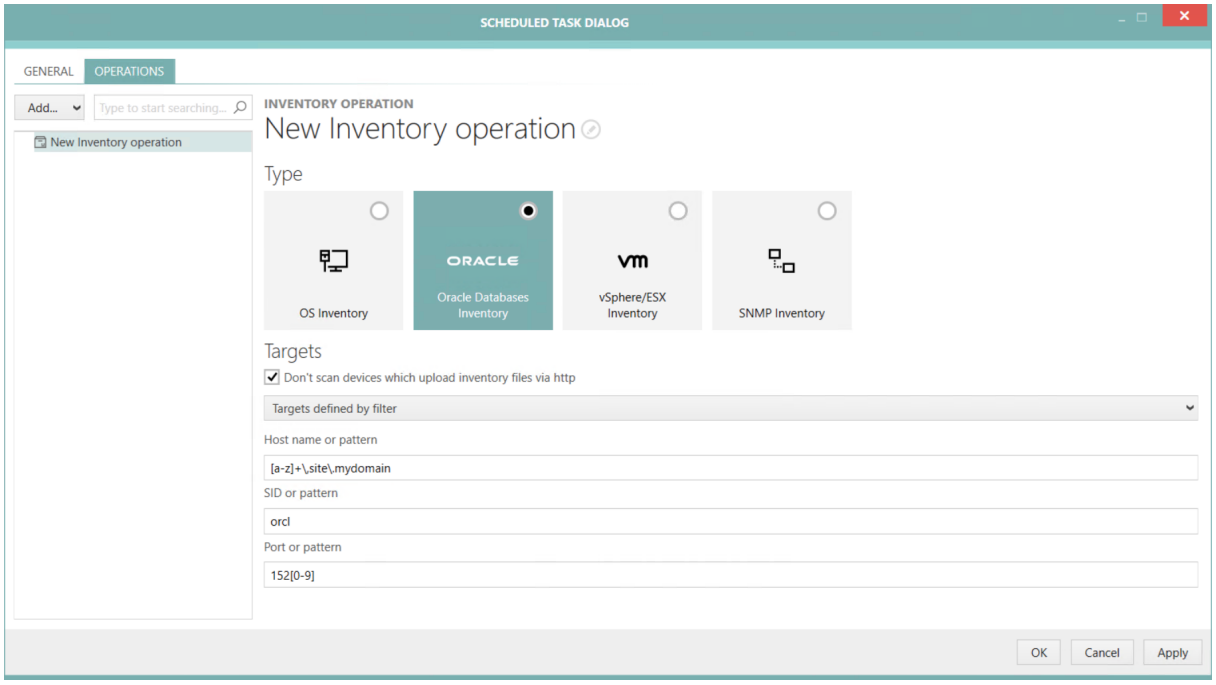
```
192.168.170.1
Hostname1
```

### Oracle Databases Inventory

An Oracle inventory is performed by the Oracle action. The target host for this operation is defined by one of the following options that can be configured using the dropdown menu underneath the **Don't scan devices which upload inventory files via http checkbox**:

- Targets defined by filter
- Targets defined by expression
- Targets defined by list
- Targets filtered by file

## Targets Defined by Filter




For the **Targets defined by filter** option, the following settings are available:

- **Host name or pattern**: This field can contain a concrete host name or a regular expression.
- **SID or pattern**: This field can contain either a specific SID or a pattern that can be used to form SIDs.
- **Port or pattern**: This field can contain either a specific port or a pattern that can be used to form ports.

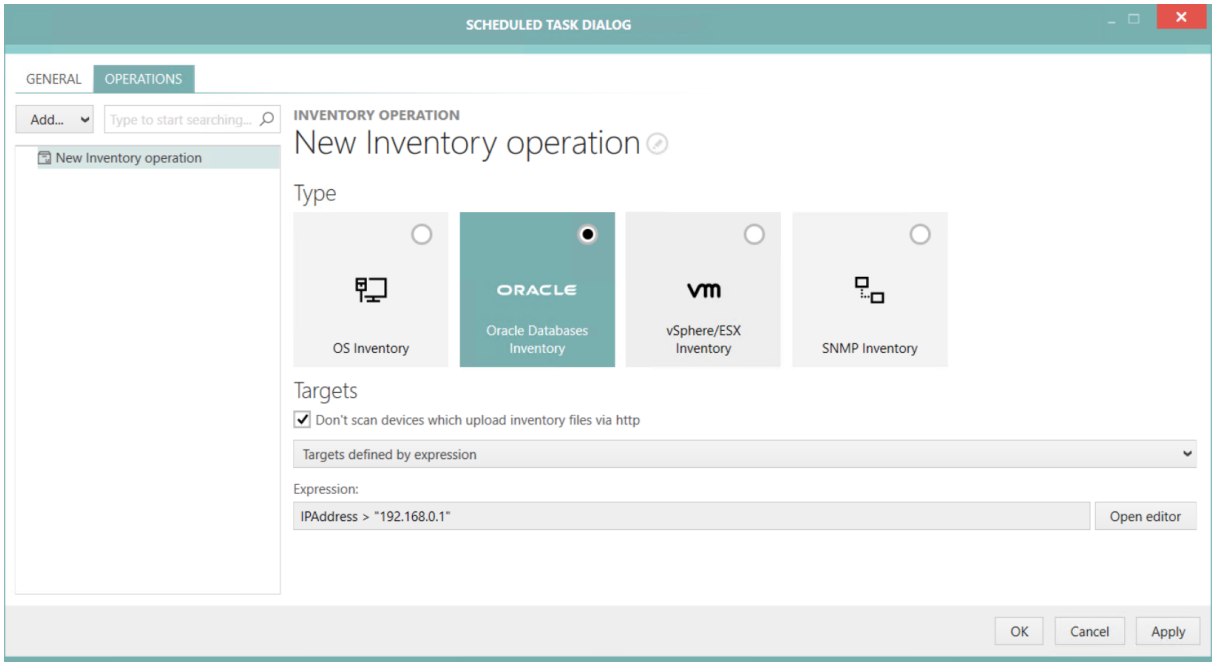
**Be aware:**
Regular expressions can be complex to write and validate. Pay attention to the fact that some characters and sequenced may have special meaning, for example a dot "." means any character inside the Regular Expression. To escape special characters, prepend them with backslash (\). Information about how to use regular expression refer to the Microsoft Documentation.

**Example:**
[a-z]+\.sitea\.mydomain

- [a-z]: all lowercase letters from "a" to "z".
- +: the preceding element can be matched one or more times.
- \.: matches a "."

Therefore this regular expression can be used to filter for everything that starts with letters only and ends with ".sitea.mydomain".

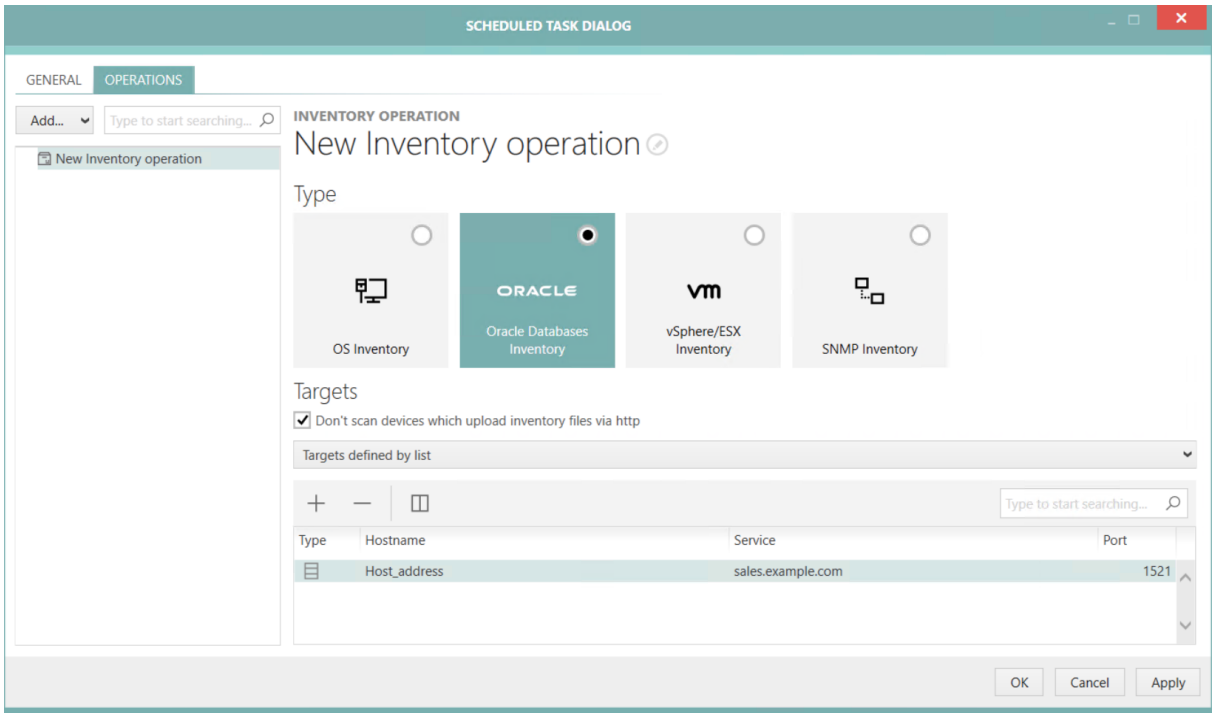
## Targets Defined by Expression




For the **Targets defined by expression** option, the following settings are available:

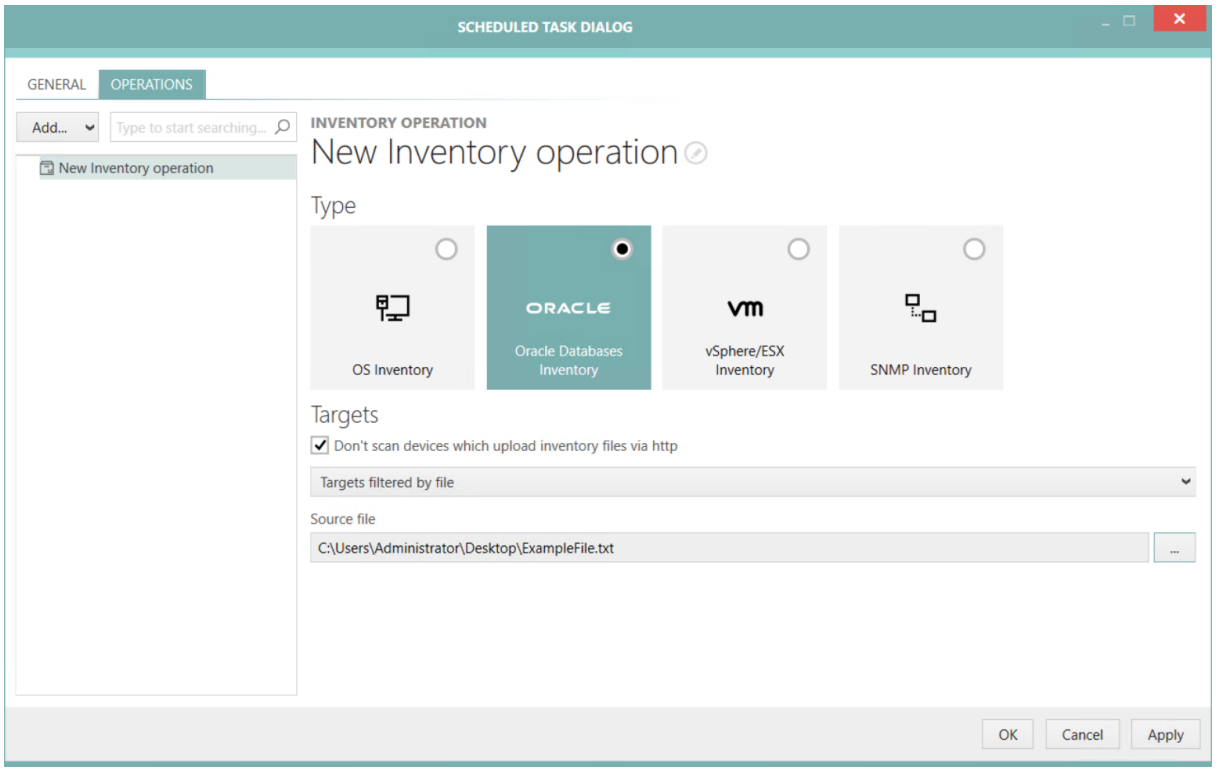
- **Expression**: Enter the expression that is used as filter into this field. The expression can be created by using the **Advanced Filtering** editor. To use the editor, click on the **Open editor** button located behind the **Expression** field.

## Targets Defined by List



The **Targets defined by list** option allows to pick one or multiple hosts from the **Oracle** list. In order to add a new host, click on the **+** button located at the top of the list. Hosts can be removed by selecting on host and clicking on the **-** button.

## Targets Filtered by File




For the **Targets defined by file** option, the following settings are available:

- **Source file**: Select a file containing the hostnames by clicking on the **Browse** [**...**] button and selecting the target file (either `.txt` or `.csv`).

A simple text file is sufficient as source file. The values need to be separated by either a semicolon or a (NewLineCharacter). The following examples are valid entries for the file:

**Example 1:**

```
Hostname1;Hostname2
```

**Example 2:**

```
Hostname1
Hostname2
```

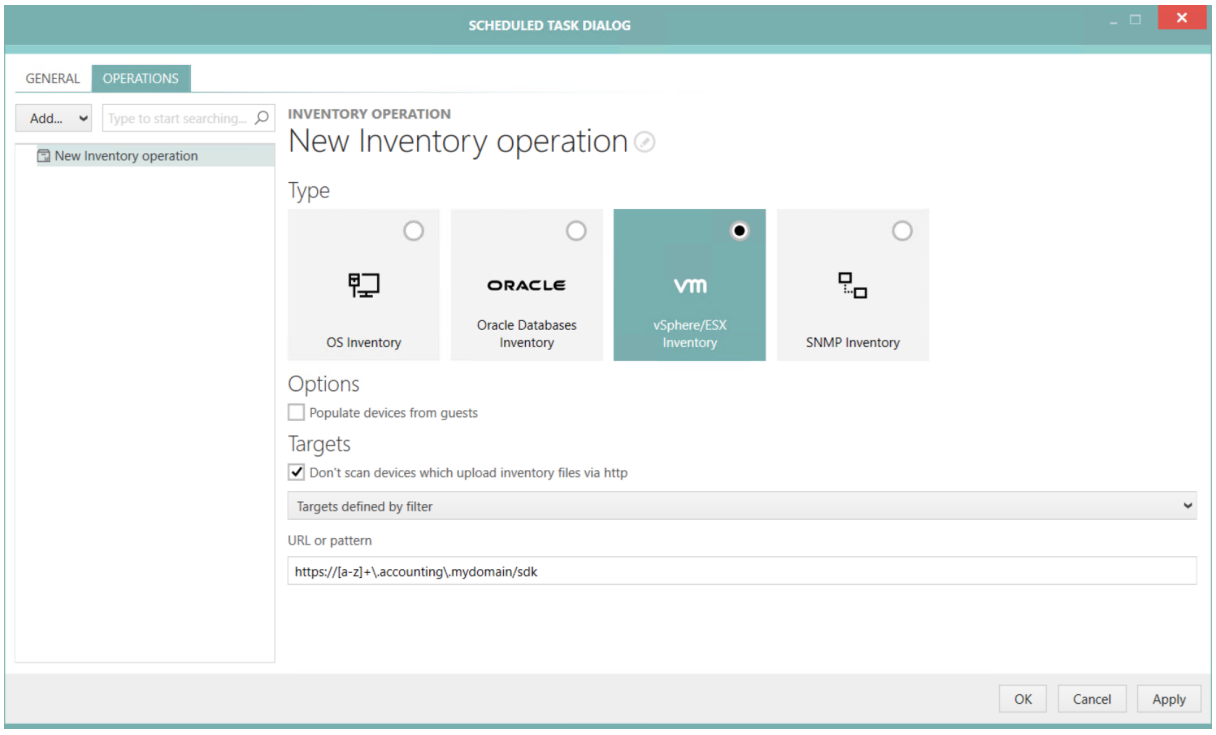
### vSphere/ESX Inventory

A vSphere/ESX inventory is performed by the vSphere/ESX action. The target host for this operation is defined by one of the following options that can be configured using the dropdown menu underneath the **Don't scan devices which upload inventory files via http checkbox**:

- Targets defined by filter
- Targets defined by expression
- Targets defined by list
- Targets filtered by file

Additionally, there is also the option to populate the devices from guests. This option can be activated by selecting the **Populate devices from guests** checkbox.

## Targets Defined by Filter




For the **Targets defined by filter** option, the following settings are available:

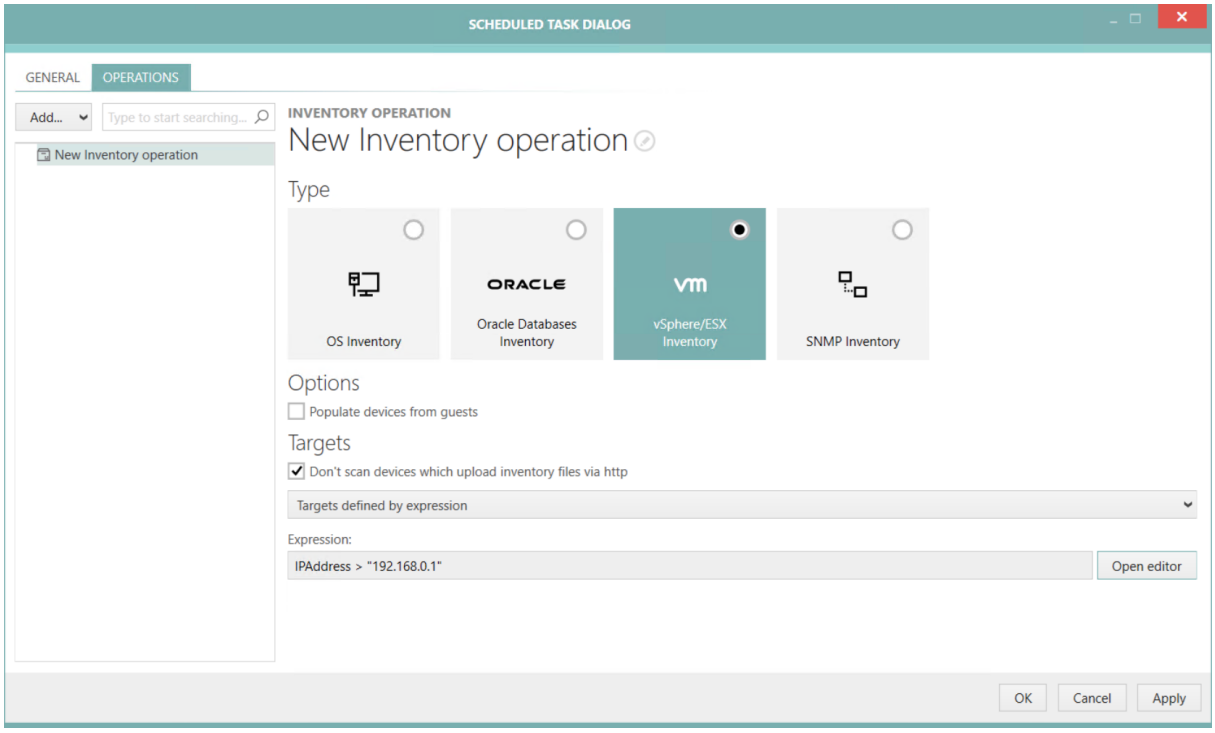
- **URL or pattern**: This field contains a URL or a regular expression. Information about how to use regular expression refer to the Microsoft Documentation.

**Example:**
[a-z]+\.sitea\.mydomain

- [a-z]: all lowercase letters from "a" to "z".
- +: the preceding element can be matched one or more times.
- \.: matches a "."

Therefore this regular expression can be used to filter for everything that starts with letters only and ends with ".sitea.mydomain".

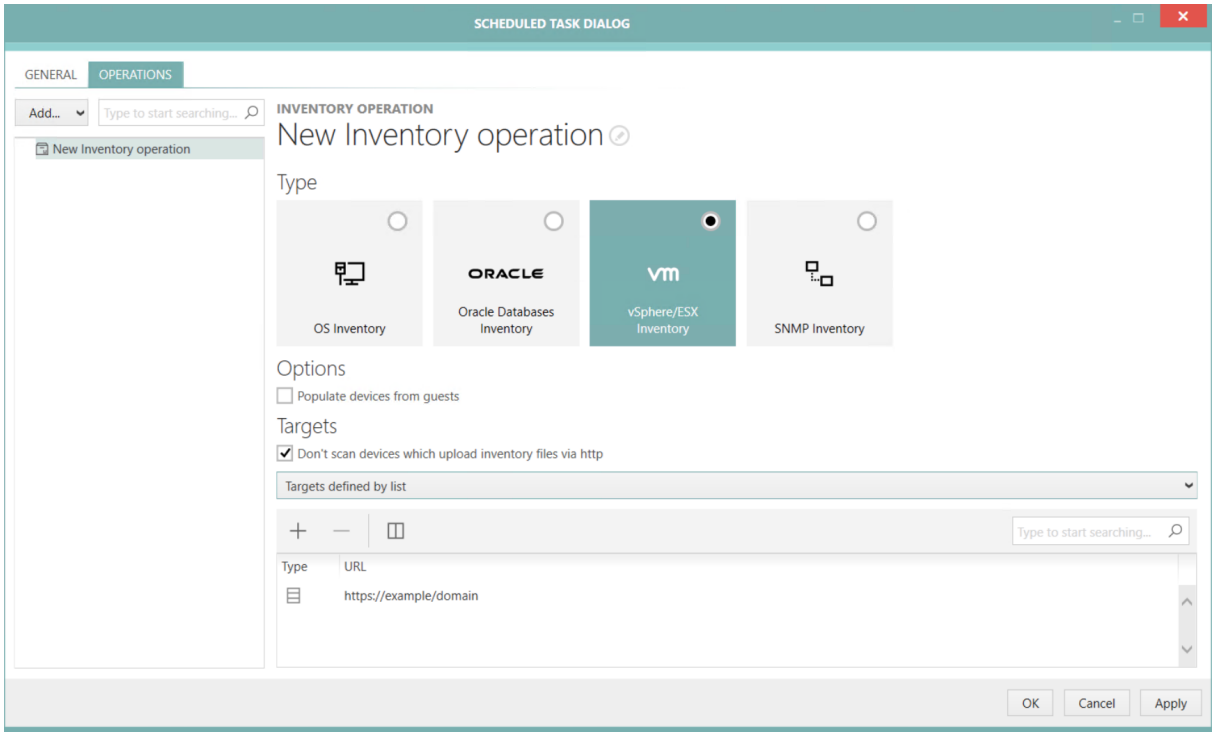
## Targets Defined by Expression




For the **Targets defined by expression** option, the following settings are available:

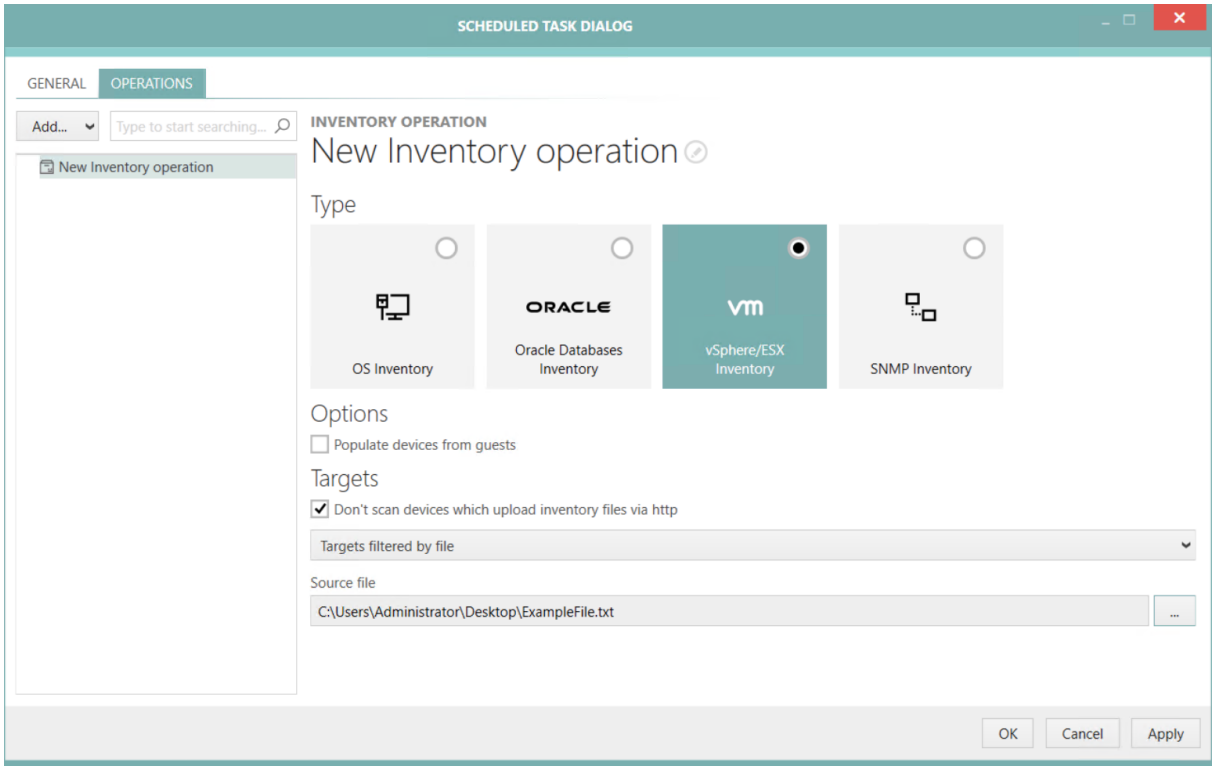
- **Expression**: Enter the expression that is used as filter into this field. The expression can be created by using the **Advanced Filtering** editor. To use the editor, click on the **Open editor** button located behind the **Expression** field.

## Targets Defined by List




The **Targets defined by list** option allows to pick one or multiple hosts from the **ESX/vSphere** list. In order to add a new host, click on the **+** button located at the top of the list. Hosts can be removed by selecting on host and clicking on the **-** button.

## Targets Filtered by File




For the **Targets defined by file** option, the following settings are available:

- **Source file**: Select a file containing the URLs by clicking on the **Browse** [...] button and selecting the target file (either `.txt` or `.csv`).

A simple text file is sufficient as source file. The values need to be separated by either a semicolon or a (NewLineCharacter). The following examples are valid entries for the file:

**Example 1:**

```
https://192.168.170.1/sdk;https://192.168.170.2/sdk
```

**Example 2:**

```
https://192.168.170.1/sdk
https://192.168.170.2/sdk
```

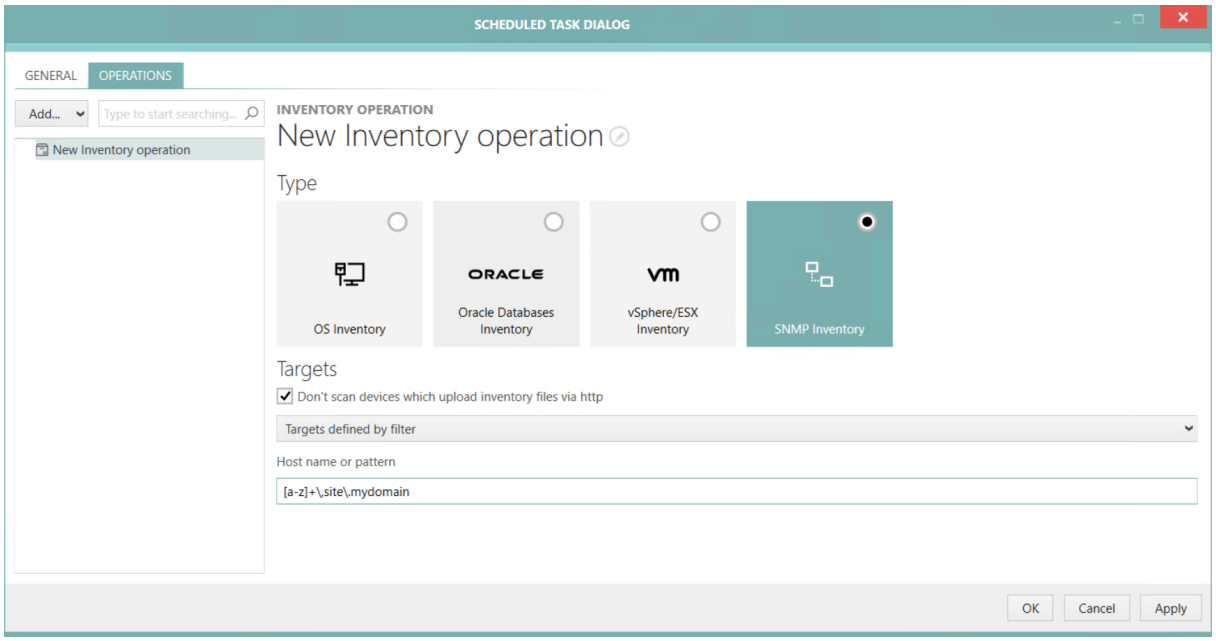
### SNMP Inventory

An SNMP Inventory is performed by the SNMP action. The target hosts for this operation are either defined by a filter (**Targets defined by filter** option) or by a list of service endpoints (**Targets defined by list** option).

A remote OS inventory is performed by the OS Inventory action. The target host for this operation is defined by one of the following options that can be configured using the dropdown menu underneath the **Don't scan devices which upload inventory files via http checkbox**:

- Targets defined by filter
- Targets defined by expression
- Targets defined by list
- Targets filtered by file

## Targets Defined by Filter



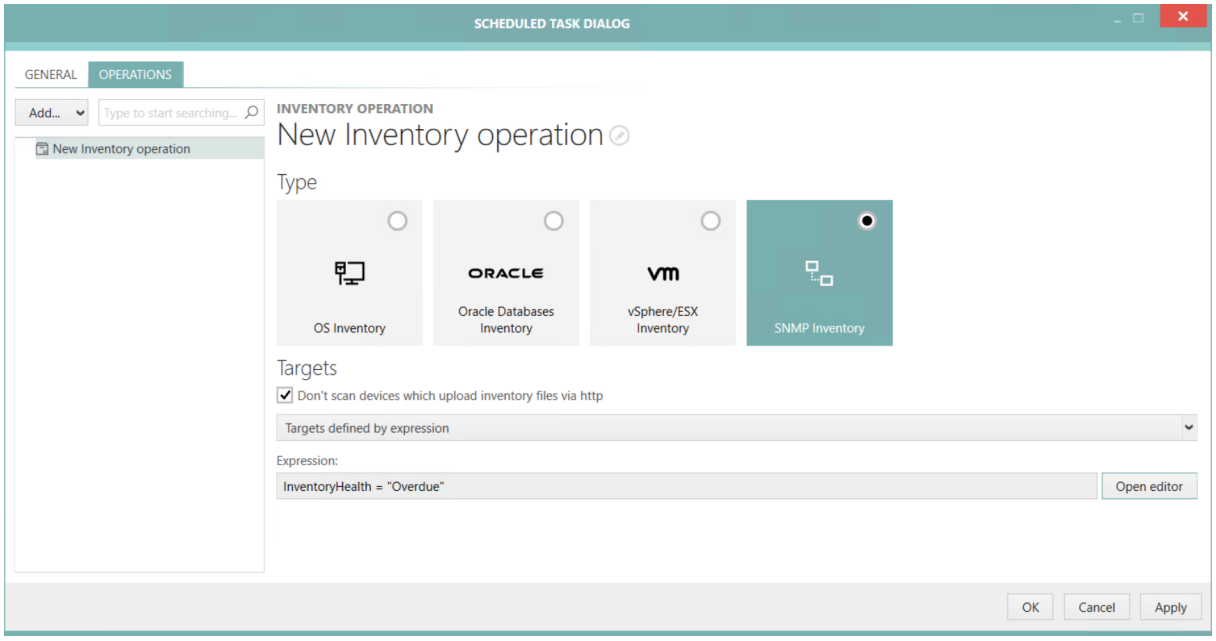

For the **Targets defined by filter** option, the following settings are available:

- **Host name or pattern**: This field can contain a concrete host name or a regular expression. Information about how to use regular expression refer to the Microsoft Documentation.

**Example:**
[a-z]+\.sitea\.mydomain
- [a-z]: all lowercase letters from "a" to "z".
- +: the preceding element can be matched one or more times.
- \.: matches a "."

Therefore this regular expression can be used to filter for everything that starts with letters only and ends with ".sitea.mydomain".

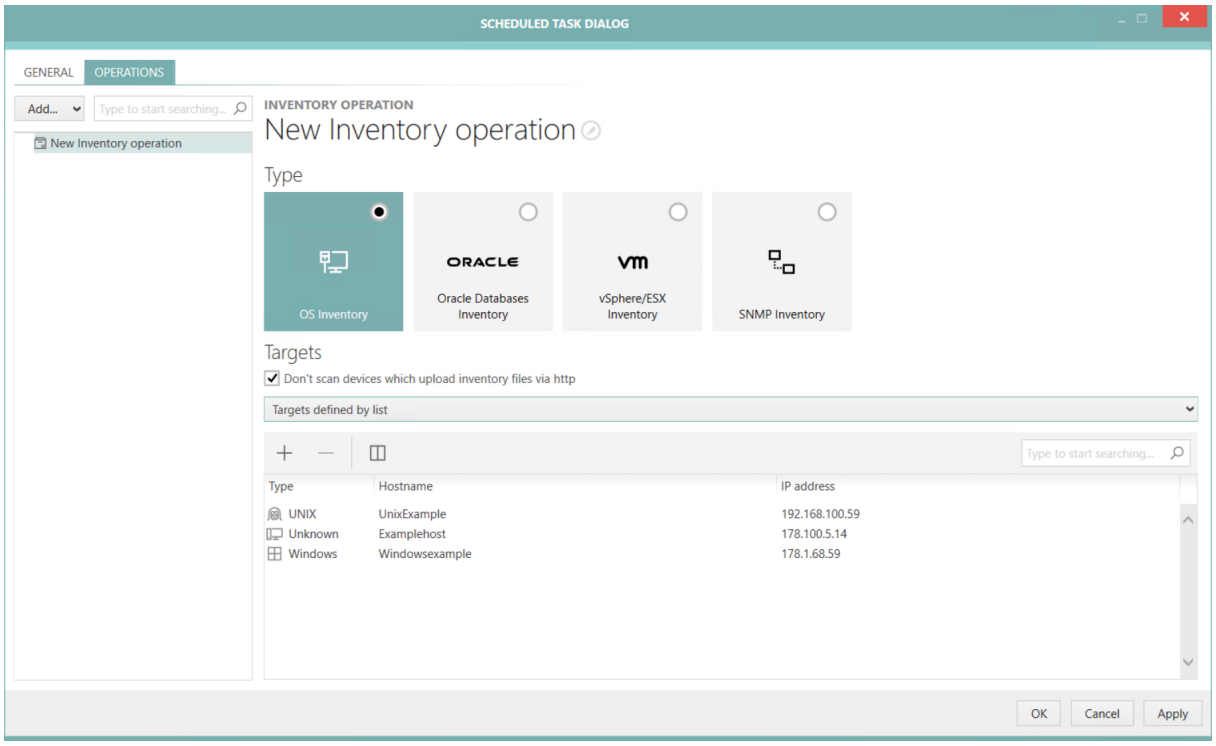
## Targets Defined by Expression




For the **Targets defined by expression** option, the following settings are available:

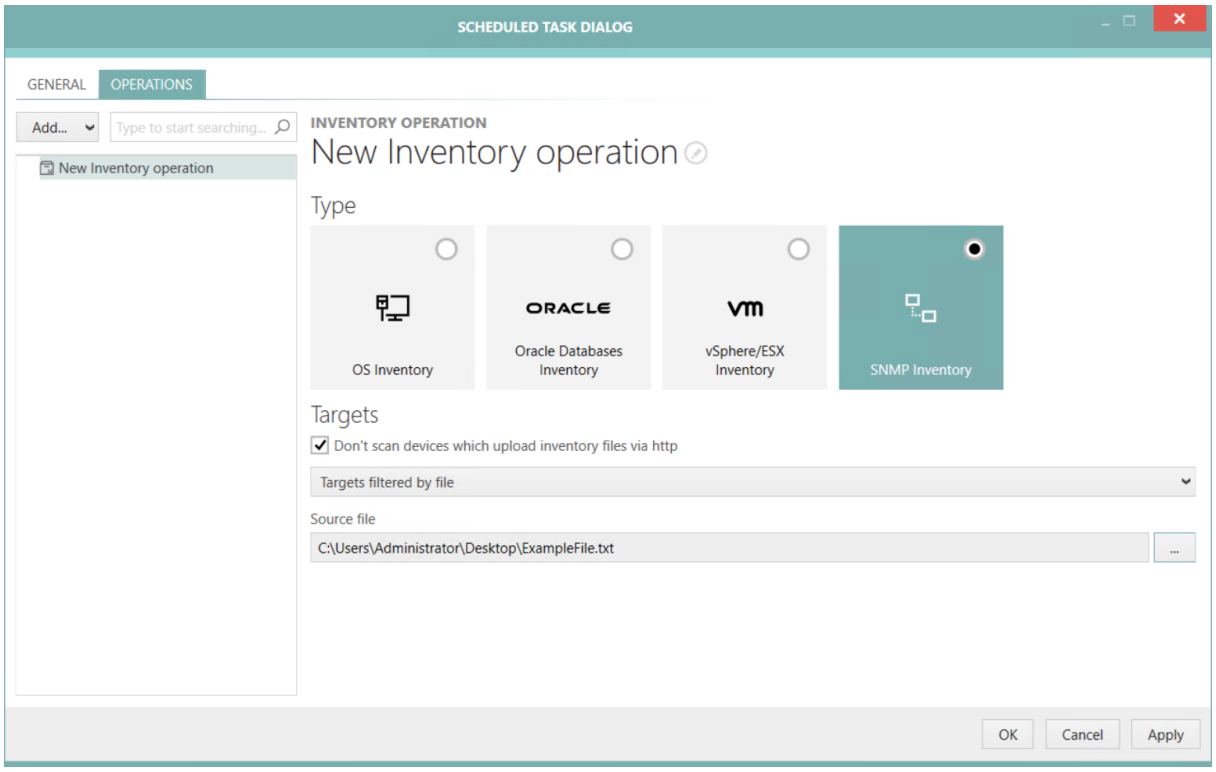
- **Expression**: Enter the expression that is used as filter into this field. The expression can be created by using the **Advanced Filtering** editor. To use the editor, click on the **Open editor** button located behind the **Expression** field.

## Targets Defined by List



The **Targets defined by list** option allows to pick one or multiple hosts from the **SNMP** list. In order to add a new host, click on the **+** button located at the top of the list. Hosts can be removed by selecting on host and clicking on the **-** button.

## Targets Filtered by File




For the **Targets defined by file** option, the following settings are available:

- **Source file**: Select a file containing the hostnames by clicking on the **Browse** [**...**] button and selecting the target file (either `.txt` or `.csv`).

A simple text file is sufficient as source file. The values need to be separated by either a semicolon or a (NewLineCharacter). The following examples are valid entries for the file:

**Example 1:**

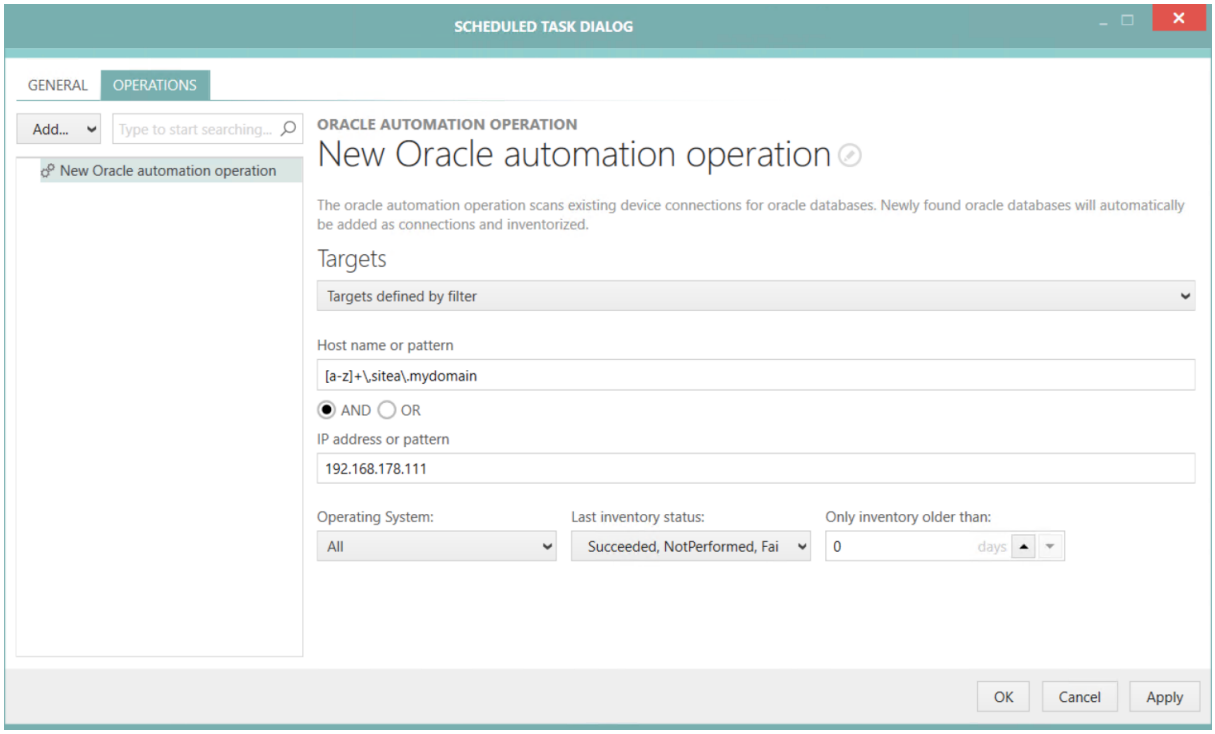
```
Hostname1;Hostname2
```

**Example 3:**

```
Hostname1
Hostname2
```

## Oracle Automation Operation

The **Oracle automation operation** is used to scan existing devices for oracle databases.




For the **Targets defined by filter** option, the following settings are available:

- **Host name or pattern**: This field can contain a concrete host name, address, or regular expression. Information about how to use regular expression refer to the Microsoft Documentation.

**Example:**
[a-z]+\.sitea\.mydomain
- [a-z]: all lowercase letters from "a" to "z".
- +: the preceding element can be matched one or more times.
- \.: matches a "."

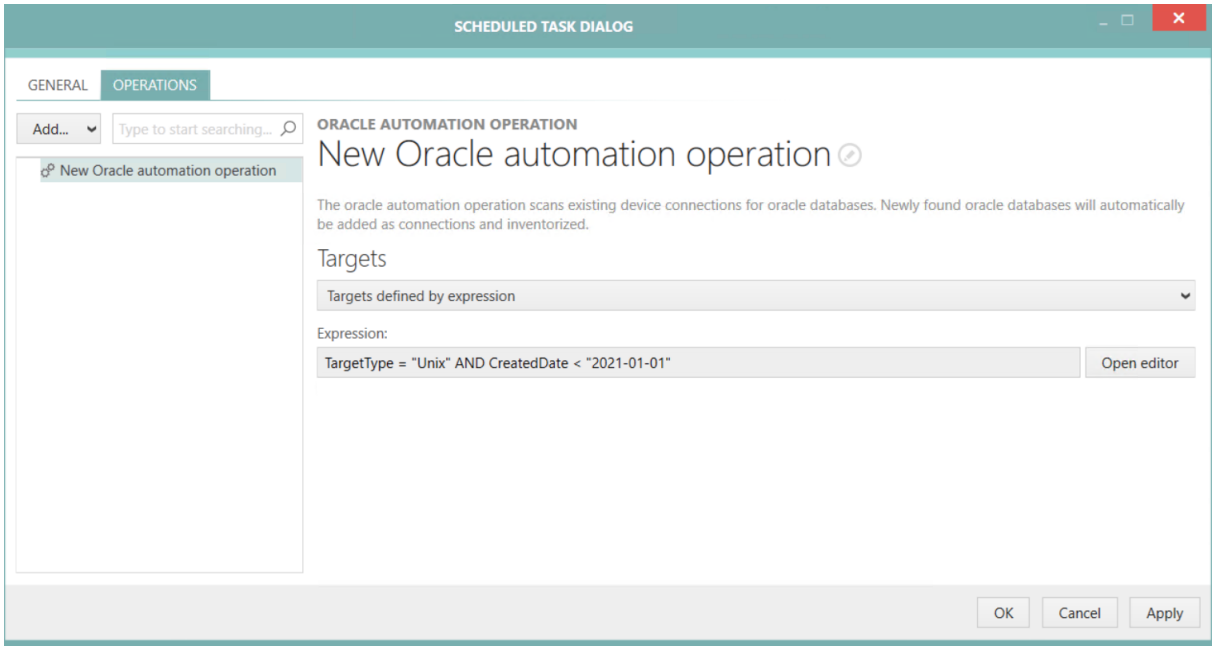
Therefore this regular expression can be used to filter for everything that starts with letters only and ends with ".sitea.mydomain".

The **AND** and the **OR** radio button can be used to defined if the Host name or pattern and the IP address or pattern fields are used together or separately.

- **IP address or pattern**: This field can contain a concrete host name, address, or regular expression.

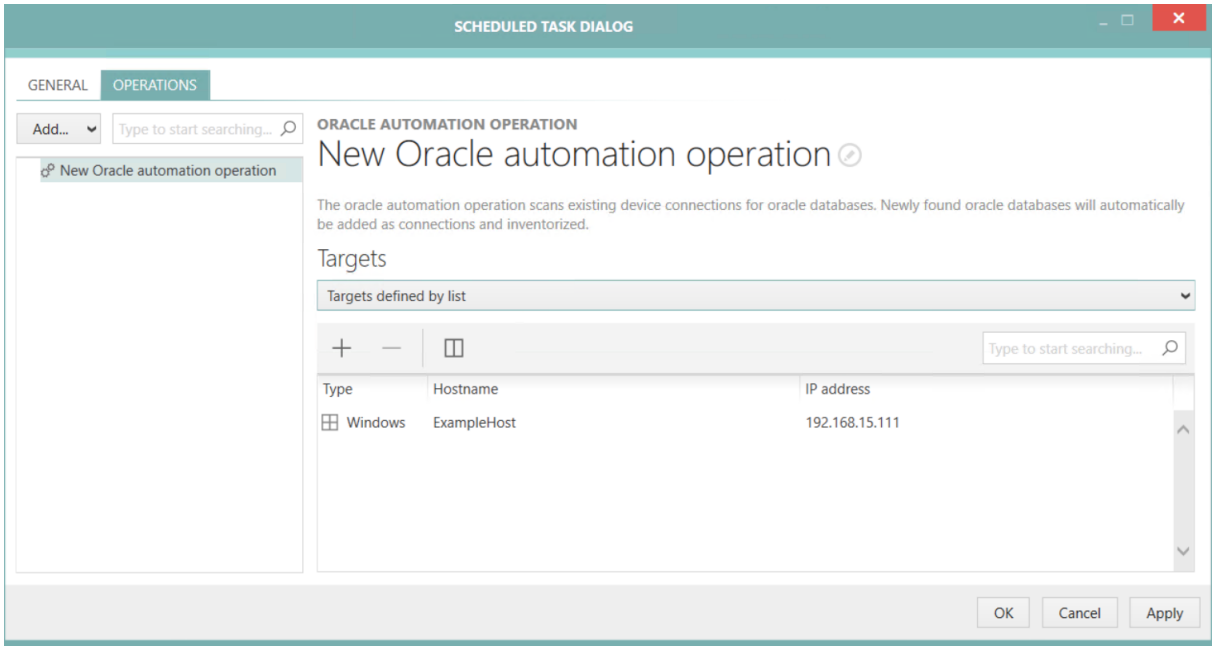
Furthermore, the following additional filter options are available.

- **Operating System**: This filter is used to filter after the operating system. It can be set to **All**, **Windows**, **UNIX***, and **Unknown**.
- **Last inventory status**: This filter is used to filter after the status of last inventory. It can be set to any combination of **Succeeded**, **NotPerformed**, or **Failed**.
- **Only inventory older than**: This filter is used to filter according to the age of the inventory. Configure the minimum age in days of the last inventory.




For the **Targets defined by expression** option, the following settings are available:

- **Expression**: Enter the expression that is used as filter into this field. The expression can be created by using the **Advanced Filtering** editor. To use the editor, click on the **Open editor** button located behind the **Expression** field.

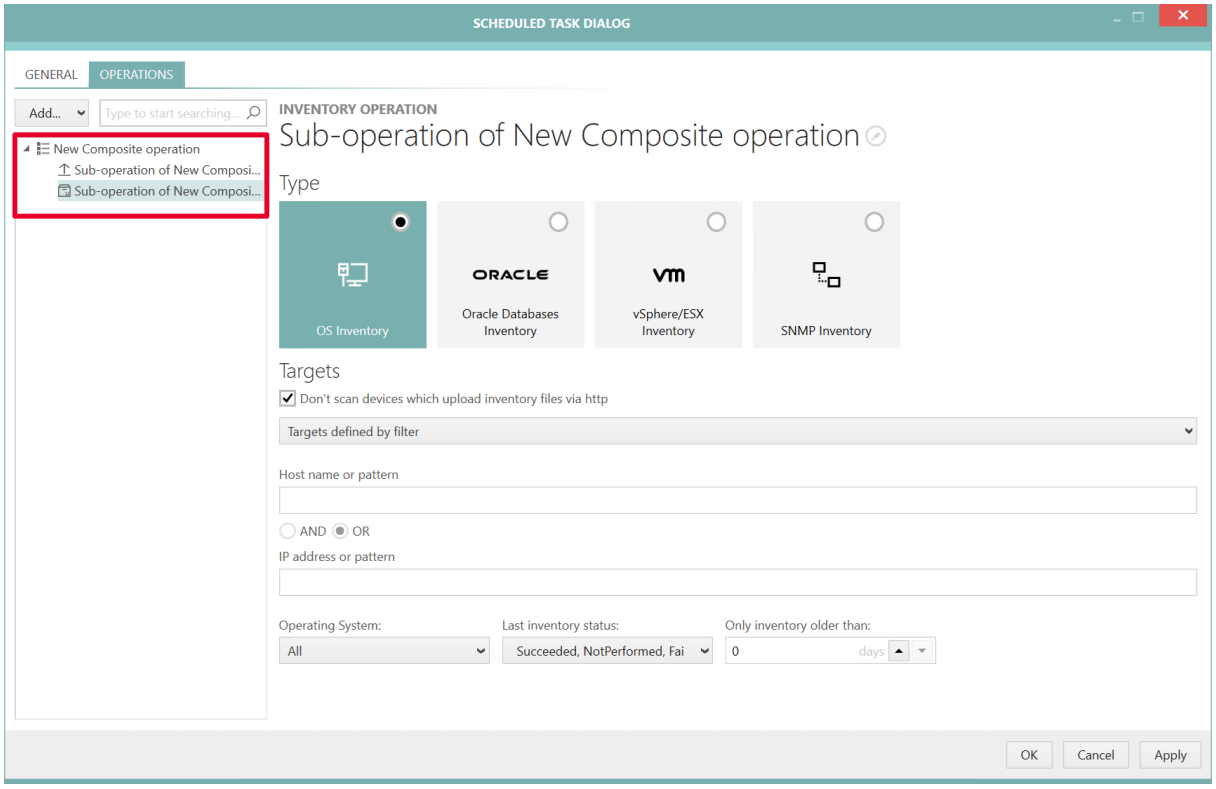



The **Targets defined by list** option allows to pick one or multiple hosts from the **OS connections** list. In order to add a new host, click on the **+** button located at the top of the list. Hosts can be removed by selecting on host and clicking on the **-** button.

## Composite Operation

The composite operation / action encompasses one or more actions / operations in one step. The actions / operations in a composite action / operation will be executed in parallel. When all operations of the composite operation are finished, the next action in the list will be started.

The schedule editor does not allow for the nesting of composite operations within composite operations.




## Chaining Operations

It is possible to execute one or multiple actions / operations with a task. The operations will be executed one after another using the same order in which they appear in the action list starting at the top. The position of an operation can be shifted using the arrow buttons located in the action bar on top of the list. A checkbox is available for each operation in order to ignore errors that occur during the execution of the previous operation. If the checkbox is not activated, the whole task will be canceled if an error occurs during any of the previous operations. No further operations of this task will be executed until the task is triggered once more.

# Inventory Agent

The **RayVentory Inventory Agent (RVIA)** is designed to continuously deliver hard- and software inventory and usage data from computer systems running Windows, Linux or Unix.

Inventory Agent has the following features:

- Inventory of computer hardware and software (Operating System inventory)
- Discovery and inventory of Oracle databases
- Upload of discovery/inventory results
- Download of configuration data
- Secure transfer of credentials for upload/download
- Scheduling of inventory and discovery operations, automatic download of configuration, uploading of results and execution of custom commands

Additionally, Windows devices running the Inventory Agent can use the following:

- Application usage metering (runs continuously as a service)
- SaaS discovery (for (web-)browser-based applications)

# Installation

## On Windows

To install the agent, utilize Windows Installer (`msiexec`) and the available configurations from the MSI package.

The following MSI properties are most commonly used:

| Property | Description |
|---|---|
| `INSTALLDIR` | The installation folder<br>Default value: C:\Program Files (x86)\RayVentory\ |
| `CONFIGDOWNLOADSOURCE` | The URL from which the configuration will be downloaded.<br>**This is a required property if installed in a non-interactive mode.** |
| `RESULTUPLOADDESTINATION` | The URL for the upload location to which the NDI files will be uploaded. |
| `SCHEDULEGETCONFIG` | The default schedule for getting the configuration<br>Default value: `25 0 * * *`<br>Default value represents downloading a new configuration file every day at 00:25 |
| `SAASDISCODAYSBACK` | Configures the SaaS discovery<br>Default: 30 days of browser history |
| `USAGEAGENTDISABLE` | Disabled usage agent (one of the following: `true` or `false`)<br>Default value: `true` |

For a detailed description for each available config command, please, refer to the corresponding chapters.

## Example

```
msiexec RVIA.msi INSTALLDIR="C:\RayVentory Agent"
SCHEDULEGETCONFIG="25 0 * * *"
CONFIGDOWNLOADSOURCE=http://192.168.123.123:951/rviaconfig/
special.cfg RESULTUPLOADDESTINATION=http://192.168.123.123:951/
Inventories/
```

This command installs the agent and configures it to download the configuration from the defined location every day at `00:25`.

**Be aware:**
Changes in the configuration file may be later overridden by incoming configuration files pulled from the download location.

**Be aware:**
Ensure that the value for resultUploadDestination ends on a slash (/) as RVIA simply concatenates this value with the destination filename to generate the URL for the individual file upload. This behavior is intentional as it allows to prefix each uploaded file.
**Example:**
`resultUploadDestination=http://myhost:591/`
**Example with prefix:**
`resultUploadDestination=http://myhost:591/myprefix`
This will produce files on the RVSE host that start with "myprefix".

### On Non-Windows

The command used for the installation on Linux/Unix machines depends on the operating system.

- **AIX:** `sudo installp -aYF -d <package> rvia.rte`
- **DEB:** `sudo dpkg -i <package>`
- **MacOS:** `sudo installer -pkg <package> -target /`
- **RPM:** `sudo rpm -ivh <package>`
- **Solaris:** `sudo pkgadd -d <path/to/package> RVIA`

The configuration of the agent is a separate step after the installation (`cd /opt/rvia; sudo ./rvia getconfig <url>`).

## Installation with an Initial Configuration

The inventory agent can be installed using an initial configuration by executing the following steps:

1. Take a config file from a RayVentory Scan Engine installation and edit it as needed. The config file can usually be found at `C:\ProgramData\Raynet\RayVentoryPortal\Results\rviaconfig`.
2. Copy the config file to the target machine (currently AIX, Linux, macOS, Solaris, and Windows are supported).
3. Open a terminal/command prompt on the target machine.

On AIX, Linux, macOS, or Solaris:

a. Set the environment variable `RVIACONFIGFILE` and set the absolute path to the config file as value (for example: `export RVIACONFIGFILE=/home/user/default.cfg`).
b. Proceed to install the Inventory Agent package (the command differs depending on the operating system used on the target machine).

On Windows:

a. Set the msi parameter `RVIACONFIGFILE` during the installation of the Inventory Agent and set the path to the config file as value (for example: `msiexec /i RayVentory_Inventory_Agent.msi /quiet /qn /norestart /L*v C:\Users\myuser\install.log RVIACONFIGFILE=C:\Users\myuser`

\default.cfg).

4. Now, the `rvia.cfg` file in the installation path should be identical to the `default.cfg`.

Detailed information for the installation of RVIA on a Windows Client can be found in the chapter Installation and Configuration of RVIA for Windows.
Detailed information for the installation of RVIA on a non-Windows Client can be found in the chapter Installation and Configuration of RVIA for Non-Windows.

# Configuration

After installing the agent, it operates using a configuration file called `rvia.cfg`, which can be found below the `ProgramData` directory. This file is created during the first run of the Inventory Agent and takes over the settings from `template.cfg`, which can be found in the Inventory Agent installation directory.

Configuration files can also be distributed remotely by the RayVentory Scan Engine (see `configDownloadSource` setting).

## Default configuration

The following snippet shows the default configuration served by RayVentory Scan Engine:

```
configDownloadSource=http<s>://
<RVSE_hostname>:<RVSE_HTTP_service_port>/rviaconfig/default.cfg
configDownloadUser=<encrypted HTTP basic authentication username>
configDownloadPassword=<encrypted HTTP basic authentication
password>
#configDownloadProxyURL=
#configDownloadProxyUser=
#configDownloadProxyPassword=
#configDownloadCurlArgs=
#configDownloadMaxDelay=
resultUploadDestination=http<s>://
<RVSE_hostname>:<RVSE_HTTP_service_port>/
#resultDirectory=
resultUploadUser=<encrypted HTTP basic authentication username>
resultUploadPassword=<encrypted HTTP basic authentication password>
#resultUploadProxyURL=
#resultUploadProxyUser=
#resultUploadProxyPassword=
#resultUploadCurlArgs=
#resultUploadMaxDelay=
generalTaskMaxDelay=120
#oracleUser=
#oraclePass=
#oracleSysDba=
#saasDiscoDaysBack=30
#usageWhitelist=\\winword\.exe
#usageWhitelist=\\powerpnt\.exe
#usageWhitelist=\\teams\.exe
```

```
#usageWhitelist=\\outlook\.exe
usageDisabled=true
#usageLogFileSize=
#usageEnableSessionLogging=
#usageSessionBackupPeriod=
#usageUploadPeriod=86400
#usageStartupDelay=
#usageMinRunTime=
#usageProcessUpdatePeriod=
#encryptionKey=
logLevel=default
#logFileSizeLimit=
#schedule:command:echo Hello world! > /tmp/message.txt:0 0 * * *
schedule:horizon::logon
schedule:horizon::logoff
schedule:getconfig::25 0 * * *
#schedule:saas::26 0 * * *
schedule:oracle::27 0 * * *
schedule:inventory::30 0 * * *
#schedule:inventory:-o IncludeDirectory=/opt -o ExcludeDirectory=/
proc:30 0 * * *
schedule:upload::45 0 * * *
#schedule:command:curl.exe --output "C\:\Program Files (x86)
\RayVentory\InventoryAgent\whitelisted.xml" -f http\://rayventory
\:591/rviaconfig/whitelisted.xml:45 0 * * *
#schedule:command:curl.exe --output "C\:\ProgramData\Raynet
\RayVentoryInventoryAgent\SaaSwhitelisted.xml" -f http\://
rayventory\:591/rviaconfig/SaaSwhitelisted.xml:45 0 * * *
```

Removing/Adding a hash (#) will enable or disable the entries.

The `rvia.cfg` default values that are used by RVIA if there is no `template.cfg` file.

```
oracleSysDba=false
usageDisabled=true
usageLogFileSize=4000000
usageEnableSessionLogging=false
usageSessionBackupPeriod=600
usageUploadPeriod=86400
usageStartupDelay=300
usageMinRunTime=10
usageProcessUpdatePeriod=60
saasDiscoDaysBack=30
logLevel=default (currently synonymous to 'info')
logFileSizeLimit=1024000
```

The `rvia.cfg` default values provided to RVIA by the `template.cfg`.

```
#configDownloadSource=
#configDownloadUser=
```

```
#configDownloadPassword=
#configDownloadProxyURL=
#configDownloadCurlArgs=
#configDownloadMaxDelay=
#resultUploadDestination=
#resultDirectory=
#resultUploadUser=
#resultUploadPassword=
#resultUploadProxyURL=
#resultUploadCurlArgs=
#resultUploadMaxDelay=
#oracleUser=
#oraclePass=
#oracleSysDba=
#saasDiscoDaysBack=
#usageWhitelist=\\winword\.exe
#usageWhitelist=\\powerpnt\.exe
#usageWhitelist=\\teams\.exe
#usageWhitelist=\\outlook\.exe
#usageDisabled=false;
#usageLogFileSize=4000000;
#usageEnableSessionLogging=false;
#usageSessionBackupPeriod=600;
#usageStartupDelay=300;
#usageMinRunTime=10;
#usageProcessUpdatePeriod=60
#logLevel=
logFileSizeLimit=1024000
#encryptionKey=
#schedule:command:echo Hello world! > /tmp/message.txt:0 0 * * *
schedule:getconfig::1/10 * * * *
#schedule:saas::26 0 * * *
#schedule:inventory:-o IncludeDirectory=/opt:30 0 * * *
#schedule:oracle:-o user=smith -o pass=tiger -o asSysDBA=true:33 0
* * *
#schedule:upload::45 0 * * *
#schedule:command:curl.exe --output "C\:\Program Files (x86)
\RayVentory\InventoryAgent\whitelisted.xml" -f http\://rayventory
\:591/rviaconfig/whitelisted.xml:45 0 * * *
#schedule:command:curl.exe --output "C\:\ProgramData\Raynet
\RayVentoryInventoryAgent\SaaSwhitelisted.xml" -f http\://
rayventory\:591/rviaconfig/SaaSwhitelisted.xml:45 0 * * *
```

## SaaS Discovery

By default the entire history from Chromium Edge and Google Chrome web browsers for the given number of days (by default 30 days) will be taken and the following tables will be populated with it:

- `ComputerUsage`
- `SoftwareFile`

- SoftwareFileName
- SoftwareFileUsage
- SoftwareUsagePerWeek
- SoftwareVersion

It is possible to reduce the amount of data by using a whitelist. To use a whitelist, create a file named SaaSwhitelist.xml. Then the schedule line to download the .xml in the default.cfg needs to be configured and enabled. If a whitelist has been defined, only those URLs listed in the whitelist will be taken into account when running the SaaS discovery.

**Example:**

```
schedule:command:curl.exe --output "C\:\ProgramData\Raynet
\RayVentoryInventoryAgent\SaaSwhitelisted.xml" -f http\://
rayventory\:591/rviaconfig/SaaSwhitelisted.xml:45 0 * * *
```

After RayVentory Scan Engine has been installed, an example file can be found at C:\Program Files (x86)\RayVentoryScanEngine\Contrib \SaaSwhitelistedexample.xml.

In order to add a URL to the whitelist, the following format needs to be used:

```
<Saas Url="{URL}" Regex="{BOOLEAN}" />
```

- **Saas Url**: Should contain the URL (either as string or as regular expression) that should be added to the whitelist.
- **Regex**: Can be either true or false.
  - If Regex is set to false, all URL containing the exact string will be taken into account.
    **Example**:
    <Saas Url="microsoft.com" Regex="false" />
    All URLs that contain the string microsoft.com will be taken into account.
  - If Regex is set to true, regular expressions will be used.
    **Example**:
    <SaaS Url="raynet\.(de|com)" Regex="true" />
    All URLs that contain either raynet.de or raynet.com will be taken into account.
    More information about regular expressions can be found in the Microsoft Documentation.

**SaaSwhitelistedexample.xml File**

```
<?xml version="1.0"?>
<Whitelist>
      <Saas Url="microsoft.com" Regex="false" />
      <Saas Url=".raynet." Regex="true" />
</Whitelist>
```

# Usage

The Agent either operates based on the schedules called from the configuration or when being called manually on the command-line.

## Command-Line

To start the agent, execute the following command:

```
rvia <command>
```

where <command> is the name of one of the supported commands:

| Command | Description |
|---|---|
| help | Shows help and available commands |
| inventory [<options>] | Runs OS inventory. This command may be followed by optional command-line parameters for NDTRACK. |
| oracle [<options>] | Runs the Oracle database discovery and inventory. This command may be followed by optional command-line parameters for ORATRACK. |
| saas [<number-of-days>] | Runs the SaaS discovery. The command may be followed by the number of days to look back in historic usage data. |
| getconfig [<source-host>] | Downloads the configuration. |
| schedule | Applies the current schedule. |
| upload [<destination-host>] | Uploads the results. |
| encrypt [<key>] | Encrypts all usernames and password marked as blank. |

### Example command:

```
rvia inventory
```

### Example command with additional command-line options:

```
rvia inventory -o MachineName=testbox
```

## Usage agent

If enabled the Usage Agent will meter every application (OS related services will not be included) that is being used on the system.

To enable the usage agent set `usageDisabled / USAGEAGENTDISABLE` to `false`.

The usage agent can be configured with the following configurations which are only available for usage in a config file:

| Options | Description |
|---|---|
| `usageLogFileSize` | Max size (in bytes) of the usage log written by the agent.<br>Default value: `4000000` |
| `usageEnableSessionLogging` | If set to `true` a usage log is written which contains all metering information. This should be used for debugging purposes only.<br>Default value: `false` |
| `usageSessionBackupPeriod` | Time span (in seconds) specifying how often the information is dumped from memory into a cache file.<br>Default value: `3600` |
| `usageStartupDelay` | Time span (in seconds) specifying when usage agent should start operating after boot.ing<br>Default value: `300` |
| `usageMinRunTime` | Time span (in seconds) specifying how long application runs until the usage gets picked up by the agent.<br>Default value: `60` |
| `usageProcessUpdatePeriod` | Time span (in seconds) specifying how often running processes should be pulled.<br>Default value: `60` |

## Whitelisting

To restrict the metering to a list of applications it is possible to add a whitelist to the Agent.
To whitelist certain processes, add lines for the setting `usageWhitelist` to the configuration file.

On successful download, these settings will be automatically applied by creating from them a file of the name `whitelisted.xml` in the Inventory Agent installation directory.
This way your process whitelists can be distributed along the other configuration settings.

**Be aware:**
The values for whitelisted processes must represent valid Regular Expressions. This is why in the following examples special characters " . " (dot) and "\" (backslash) are escaped with another backslash.

```
usageWhitelist=\\winword\.exe
usageWhitelist=\\powerpnt\.exe
usageWhitelist=\\teams\.exe
usageWhitelist=\\outlook\.exe
```

You can also manually create a file containing required definitions for whitelisting. The file should have an XML syntax, for example:

```
<Whitelist>
   <Process Path="\\winword\.exe" Regex="true"/>
   <Process Path="\\powerpnt\.exe" Regex="true"/>
   <Process Path="\\teams\.exe" Regex="true"/>
   <Process Path="\\outlook\.exe" Regex="true"/>
</Whitelist>
```

Save it in the root directory where the agent was installed, under the name `whitelisted.xml`.

Alternatively, you may create such a file on your RayVentory Scan Engine host, in the directory for Inventory Agent configuration files.
In this case, the whitelist could also be downloaded using Inventory Agent scheduled arbitrary command line feature, by adding a command to download into the configuration file.

Example:

```
schedule:command:curl.exe --output "C\:\Program Files (x86)
\RayVentory\InventoryAgent\whitelisted.xml" -f http\://rayventory
\:591/rviaconfig/whitelisted.xml
```

## Scheduling

The following operations can be automated by setting up scheduling:

- OS inventory
- Oracle Database Discovery and Inventory
- Upload of discovery/inventory results
- Download of configuration data
- SaaS Discovery
- Horizon connection metering
- Arbitrary commands

Inventory Agent for Windows uses the Windows Task Scheduler to execute its scheduled tasks. On UNIX/Linux, cron is used instead.

A scheduled task is defined in the configuration file by a single line, which needs to follow a specific set of rules:

```
<command>:<options>:<schedule-pattern>
```

Valid values for command are:

| Command | Description |
|---|---|
| `command` | Executes an arbitrary command. |
| `encrypt` | Encrypts all credentials which are not marked as encrypted by the `$$` prefix. |
| `getconfig` | Download a configuration file. |
| `horizon` | Gather Horizon connection information. |
| `oracle` | Runs Oracle database inventory. |
| `saas` | Runs SaaS discovery. |
| `settings` | Applies the registry settings and usage metering whitelist from the Inventory Agent configuration file to the system scheduler resp. registry and whitelist file. |
| `upload` | Uploads the results. |

The triggering-pattern is inspired by cron. On Windows, this pattern is interpreted to translate it to the Windows Task Scheduler triggering scheme. Therefore, only a small subset of cron triggering patterns, including the keywords `logon` and `logoff` can be used for Windows.

**Be aware:**
The schedule trigger `startup` and `logon` are not available for non-windows and can only be used for windows devices.

Overview of possible schedule patterns:
`<Minute> <Hour> <DayOfMonth> <Month> <DayOfWeek>`

| Character | Description |
|---|---|
| (1) `<Minute>` | Minute.<br>Either a range (`0-59`) or `*` for unspecified value. |
| (2) `<Hour>` | Hour.<br>Either a range (`0-23`) or `*` for unspecified value. |
| (3) `<DayOfMonth>` | The day of the month.<br>Either a range (`1-31`) or `*` for unspecified value. |
| (4) `<Month>` | The month.<br>Either a range (`1-12`) or `*` for unspecified value. |
| (5) `<DayOfWeek>` | The day of the week.<br>Either a range (`0-7`) or `*` for unspecified value. Use `0` for Sunday, `1` for Monday, `2` for Tuesday etc. The value of `7` means Sunday,. |

## Examples:

`schedule:getconfig::25 0 * * *`
Download a new configuration file every day at 00:25.

`schedule:saas::26 0 * * *`
Run a SaaS discovery every day at 00:26.

`schedule:inventory:-o IncludeDirectory=/opt:30 0 * * *`
Run an inventory with an additional command-line option for NDTRACK, every day at 00:30.

`schedule:oracle:-o user=smith -o pass=tiger -o asSysDBA=true:33 0 * * *`
Run an Oracle discovery and inventory with an additional command-line option for ORATRACK, every day at 00:33.

`schedule:upload::45 0 * * *`
Upload the discovery, metering and inventory results, every day at 00:45.

`schedule:saas::logon`
Run a SaaS discovery on logon.

# Logging

- **`logLevel`**
  This setting sets the detail level for the Inventory Agent log file. The default log level is `info`. Valid values are (from less to more verbose):
  - `off`
  - `fatal`
  - `error`
  - `warn`
  - `info`
  - `debug`
  - `trace`
  - `all`

- **`logFileSizeLimit`**
  This setting limits the size of the Inventory Agent log file. By default, the size is not limited. The log file is reset/deleted whenever Inventory Agent finds the file to exceed the maximum size before writing a new log entry.

# Commands

## encrypt

This command will encrypt all credentials which are not marked as encrypted by the `$$` prefix.

The encryption scheme used is AES128 CBC and the encrypted credentials are encoded as base64. Inventory Agent will use a default value for the encryption key unless you specify an encryption key in the configuration file by setting `encryptionKey` or provide an encryption key with the encrypt command.

**Note:**
This command is not available for scheduling.

## getconfig

This command will attempt to download a configuration file from a defined host.

Inventory Agent will attempt to download from one of the source URLs in the configuration file unless you include an URL after the command. If an URL was specified, then Inventory Agent will not try any other URLs on error.

After a successful download, Inventory Agent will attempt to apply its schedule to the systems scheduler (delete previously defined scheduled tasks and create new scheduled tasks). Further, usage metering related registry settings will also be applied and the file `whitelisted.xml` for the usage metering service is written if needed.

You can force Inventory Agent to try to apply the schedule in the current configuration file (see command schedule). You need to add a source URL by adding a line for the setting `configDownloadSource` to the configuration file.

Example of downloading a configuration file from a location specified in the configuration file (from command-line):
```
rvia getconfig
```

Example of downloading a configuration file from a specific URL:
```
rvia getconfig http://192.168.123.123:951/rviaconfig/special.cfg
```

Example of two download sources in the configuration file:
```
configDownloadSource=http://192.168.123.123:951/rviaconfig/
special.cfg
configDownloadSource=http://192.168.123.123:951/rviaconfig/
default.cfg
```

Bear in mind that RayVentory Scan Engine will always host a basic configuration file (which sets basic auth credentials and upload- and download locations, according to the RVSE configuration) at
`http://<yourhostname>:<configured-port>/rviaconfig/default.cfg`
or
`https://<yourhostname>:<configured-port>/rviaconfig/default.cfg`

The configuration can be managed from the Settings screen.

Mind that you can use the settings configDownloadUser and configDownloadPassword to specify credentials for basic authentication during download. See command encrypt for details.

You can specify a proxy (including credentials) by the setting `resultUploadProxyURL` for

the upload.

Example:
`configDownloadProxyURL=https://proxyUser:proxyUserPassword@172.16.1.1:8080/`

To pass additional arguments to curl for download, use the setting `configDownloadCurlArgs`.

**Note:**
For large numbers of Inventory Agent instances, you should use `configDownloadMaxDelay` to delay individual downloads to a maximum of the specified number of seconds. That will spread the network load over the time domain.

## horizon

This enhances the usage agent to also gather information about Horizon connections. If enabled, by default the horizon feature is scheduled to run during logon and logoff.

## oracle

The **oracle** command discovers local Oracle databases and gathers license relevant inventory data from these databases. This command accepts further optional command-line arguments for ORATRACK, the tool that RVIA uses to discovery and inventory Oracle databases.

Example command for command-line:
```
rvia oracle
```

Example for command-line with additional command-line options:
```
rvia oracle -o user=sys -o pass=5ecr3T -o asSysDba=true
```

Bear in mind that you do not have to pass credentials this way; RVIA has got settings to accept encrypted credentials in its configuration file. The settings are `oracleUser`, `oraclePass` and `oracleSysDba`.

## saas

This command runs the SaaS Discovery for browser-based applications. By default, the SaaS Discovery will consider the browser data regarding the past 30 days. You can override that by the setting `saasDiscoDaysBack` or provide a number on the command-line.

## schedule

This command will simply try to apply the schedule from Inventory Agent configuration file to the system scheduler. After editing a configuration file in-place, you can use this command to apply that schedule.

**Note:**
This command is only available from a CLI interface.

## settings

This command will try to apply the registry settings and usage metering whitelist from the Inventory Agent configuration file to the system scheduler resp. registry and whitelist file. After editing a configuration file in-place, you can use this command to apply these settings.

**Note:**
This command is not available for scheduling.

## upload

The upload command will attempt to upload all files from Inventory Agent results directory to one of the result destinations.

Add a destination URL by adding a line for the setting `resultUploadDestination` to the configuration file. You can use the settings `resultUploadUser` and `resultUploadPassword` to specify credentials for basic authentication during upload. See command encrypt for more details.

**Be aware:**
Ensure that the value for resultUploadDestination ends on a slash (/) as RVIA simply concatenates this value with the destination filename to generate the URL for the individual file upload. This behavior is intentional as it allows to prefix each uploaded file.
**Example:**
`resultUploadDestination=http://myhost:591/`
**Example with prefix:**
`resultUploadDestination=http://myhost:591/myprefix`
This will produce files on the RVSE host that start with "myprefix".

You can specify a proxy (including credentials) by the setting `resultUploadProxyURL` for the upload.

Example:
```
resultUploadProxyURL=https://
proxyUser:proxyUserPassword@172.16.1.1:8080/
```

To pass additional arguments to curl for upload, use the setting `resultUploadCurlArgs`.

**Note:**
For large numbers of Inventory Agent instances, you should use `configDownloadMaxDelay` to delay individual downloads to a maximum of the

specified number of seconds. That will spread the network load over the time domain.

# Parameters

In the following the parameters that are available for usage in the configuration file are listed. They are ordered in the same way, as they are ordered in the **PROPERTIES VIEW** of the **CONFIGURATION EDITOR** dialog. Furthermore, the property values and the schedules that are by default included in the configuration file are listed here.

## General

| Name and Property | Format | Description |
|---|---|---|
| Encryption Key<br>`encryptionKey` | String | The key that is used as the base for the `encrypt` command for usernames and passwords in the configuration file.<br>**Example:** `Key123!%$` |
| General task maximum delay<br>`generalTaskMaxDelay` | Minutes | A random number of minutes between `0` and `VALUE` by which all schedules are delayed.<br>**Example:** `60`<br>Default value = `120` |

## Download

| Name and Property | Format | Description |
|---|---|---|
| Source<br>`configDownloadSource` | URL | The URL from which to download the RVIA configuration file.<br>**Example:** `http://RVSEhost:591/rviaconfig/default.cfg`<br>Default value = `http://{ComputerName}:{Port}/rviaconfig/default.cfg` |
| User<br>`configDownloadUser` | Username | The username that is used by RVIA to authenticate against the RVSE HTTP service (may be encrypted).<br>**Example:** administrator |
| Password<br>`configDownloadPassword` | Password | The password for the user that is used by RVIA to authenticate against the RVSE HTTP service (may be encrypted).<br>**Example:** Password123 |
| Proxy URL<br>`configDownloadProxyURL` | URL | The URL to the proxy with which to download the RVIA configuration file.<br>**Example:** `http://proxydevice:3128` |
| Proxy User<br>`configDownloadProxyUser` | Username | The username used by RVIA to authenticate against the proxy (may be encrypted).<br>**Example:** administrator |
| Proxy Password<br>`configDownloadProxyPassword` | Password | The password for the user that is used by RVIA to authenticate against the proxy (may be encrypted).<br>**Example:** Password123 |
| Curl args<br>`configDownloadCurlArgs` | Arguments | Additional CURL arguments that will be used by RVIA when downloading the configuration file.<br>**Example:** `--insecure --tlsv1.2` |
| Max delay<br>`configDownloadMaxDelay` | Minutes | A random number of minutes between 0 and `VALUE` by which the `getconfig` schedules are delayed.<br>**Example:** 120 |

## Upload

| Name and Property | Format | Description |
|---|---|---|
| Results directory<br>`resultDirectory` | Path | The path to where RVIA saves the result files to.<br>**Example:** `C:\ProgramData\Raynet\RayVentoryInventoryAgent\results` |
| Destination<br>`resultUploadDestination` | URL | The URL of the RVSE HTTP service to which RVIA uploads its files to.<br>**Example:** `http://RVSEhost:591/`<br>Default value = `http://{ComputerName}:{Port}/` |
| User<br>`resultUploadUser` | Username | The username of the user used by RVIA to authenticate against the RVSE HTTP service (may be encrypted).<br>**Example:** administrator |
| Password<br>`resultUploadPassword` | Password | The password for the user used by RVIA to authenticate against the RVSE HTTP service (may be encrypted).<br>**Example:** Password123 |
| Proxy URL<br>`resultUploadProxyURL` | URL | The URL of the proxy used to upload files with.<br>**Example:** `http://proxydevice:3128`<br>Default value = `https://proxyUser:proxyUserPassword@172.16.12.10:8080/` |
| Proxy user<br>`resultUploadProxyUser` | Username | The username of the user used by RVIA to authenticate against the proxy (may be encrypted).<br>**Example:** administrator |
| Proxy password<br>`resultUploadProxyPassword` | Password | The password for the user used by RVIA to authenticate against the proxy (may be encrypted).<br>**Example:** Password123 |
| Curl args<br>`resultUploadCurlArgs` | Arguments | Additional CURL arguments RVIA will use when uploading files.<br>**Example:** `--insecure --tlsv1.2` |
| Max delay<br>`resultUploadMaxDelay` | Minutes | A random number of minutes between `0` and `VALUE` by which the upload schedules are delayed.<br>**Example:** `120` |

## Oracle

| Name and Property | Format | Description |
|---|---|---|
| User<br>`oracleUser` | Username | The username of the Oracle user that is used for authentication for RVIA Oracle operations.<br>**Example:** sys |
| Password<br>`oraclePass` | Password | The password of the Oracle user that used for authentication for RVIA Oracle operations.<br>**Example:** Password123 |
| User is SYSDBA<br>`oracleSysDba` | Boolean | Set the parameter to `true` if the user is a sysdba. If not, set the parameter to `false`.<br>**Example:** `true` |

## SaaS Discovery

| Name and Property | Format | Description |
|---|---|---|
| Discovery days back<br>`saasDiscoDaysBack` | Days | The number of days that will be considered by the SaaS discovery.<br>**Example:** 45<br>Default value = `30` |

## Usage

| Name and Property | Format | Description |
|---|---|---|
| Whitelist<br>`usageWhitelist` | Filename | The name of executable files that the usage agent will consider (if none are defined, all processes will be considered).<br>**Example:** `\\outlook\.exe`<br>Default value = `\\winword\.exe;#\\powerpnt\.exe;#\\teams\.exe;#\\outlook\.exe` |
| Disabled<br>`usageDisabled` | Boolean | If set to true, the usage metering is disabled. If set to false, usage metering will be actviated.<br>**Example:** `false`<br>Default value = `true` |
| Log file size<br>`usageLogFileSize` | Byte | The maximum logfile size in bytes for the usage metering log.<br>**Example:** `4000000` |
| Enable session loggin<br>`usageEnableSessionLogging` | Boolean | Enables the logging of when the tracked processes are started and when they are ended.<br>**Example:** `true` |
| Session backup period<br>`usageSessionBackupPeriod` | Seconds | The interval in which the sessions are backed up to a cache file.<br>**Example:** `600` |
| Usage upload period<br>`usageUploadPeriod` | Seconds | The interval in which the usage metering creates an `.mmi` file that can be uploaded by RVIA.<br>**Example:** `172800`<br>Default value = `86400` |
| Startup delay<br>`usageStartupDelay` | Seconds | The number of seconds that the tracking of processes will be delayed after the startup of the metering service.<br>**Example:** `300` |
| Minimal run time<br>`usageMinRunTime` | Seconds | The time in seconds that a process must be running before it is considered by the usage metering.<br>**Example:** `60` |
| Process update period<br>`usageProcessUpdatePeriod` | Seconds | How often the metering service will update the processes.<br>**Example:** `60` |

## Log

| Name and Property | Format | Description |
|---|---|---|
| Level<br>`logLevel` | String | The level of logging for the RVIA logfile. The following values are possible:<br>• `off`<br>• `fatal`<br>• `error`<br>• `warn`<br>• `info`<br>• `debug`<br>• `trace`<br>• `all`<br>• `default`<br>**Example:** `error`<br>Default value = `default` |
| File size limit<br>`logFileSizeLimit` | Byte | The maximum logfile size in bytes for the RVIA log.<br>**Example:** 1024000 |

## Schedules

This property can be used to execute scheduled tasks. In the file schedules follow the following pattern:

```
<command>:<options>:<schedule-pattern>
```

As schedule pattern either `logon` or `logoff` can be used or it follows the following pattern:

```
<Minute> <Hour> <DayOfMonth> <Month> <DayOfWeek>
```

It is possible to used multiple schedules in the configuration file.

**Be aware:**
The schedule trigger `startup` and `logon` are not available for non-windows and can only be used for windows devices.

In the following table the default schedule properties are listed. The first three columns show the parameters like they are listed in the **PROPERTIES VIEW** tab. The Property column shows how the actual property looks in the configuration file and the **TEXT VIEW** tab

| Command | Arguments | Trigger | Property |
|---|---|---|---|
| command | echo Hello world! > /tmp/message.txt | 0 0 * * * | schedule:command:echo Hello world! > /tmp/message.text:0 0 * * * |
| horizon | | logon | schedule:horizon::logon |
| horizon | | logoff | schedule:horizon::logoff |
| getconfig | | 25 0 * * * | schedule:getconfig::25 0 * * * |
| saas | | 26 0 * * * | schedule:saas::26 0 * * * |
| oracle | | 27 0 * * * | schedule:oracle::27 0 * * * |
| inventory | | 30 0 * * * | schedule:inventory::30 0 * * * |
| inventory | -o IncludeDirectory=/opt -o ExcludeDirectory=/proc | 30 0 * * * | schedule:inventory:-o IncludeDirectory=/opt -o ExcludeDirectory=/proc:30 0 * * * |
| upload | | 45 0 * * * | schedule:upload::45 0 * * * |
| command | curl.exe --output "C\:\Program Files (x86)\RayVentory\InventoryAgent\whitelisted.xml" -f http\://rayventory\:591/rviaconfig/whitelisted.xml | 45 0 * * * | schedule:command:curl.exe --output "C\:\Program Files (x86)\RayVentory\InventoryAgent\whitelisted.xml" -f http\://rayventory\:591/rviaconfig/whitelisted.xml:45 0 * * * |
| command | curl.exe --output "C\:\ProgramData\Raynet\RayVentoryInventoryAgent\SaaSwhitelisted.xml" -f http\://rayventory\:591/rviaconfig/SaaSwhitelisted.xml | 45 0 * * * | schedule:command:curl.exe --output "C\:\ProgramData\Raynet\RayVentoryInventoryAgent\SaaSwhitelisted.xml" -f http\://rayventory\:591/rviaconfig/SaaSwhitelisted.xml:45 0 * * * |

# Advanced Topics

This chapter covers some advanced topics and refers to technical details for IT professionals and administrators that help to understand how RayVentory Scan Engine works and how to use most of its functionalities.

## Receiving Uploads from Remote Scans

RayVentory Scan Engine exposes a lightweight upload HTTP server, that accepts incoming inventory files (`.ndi`) and legacy discovery files (`.disco`). It uses a zero-touch configuration which by default starts a HTTP server listening on port 8099 without authentication.

The server can be started in one of the following ways:

- If Windows Service for HTTP uploads has been installed, the service is automatically started by Windows.
- Otherwise or if the service is stopped, RayVentory Scan Engine starts a local private HTTP server upon launch and closes it upon closing the main application.

To control firewall access rules, the following executable can be added to firewall rules:
`[INSTALLDIR]\RayVentoryScanEngine.HttpUploadServer.exe`

The default port, authentication, and certificate settings can be changed in the RayVentory Scan Engine options. You can find out more information about the server configuration in the following chapter: Settings > HTTP Services > Server.

Once configured, the HTTP upload service is automatically used for remote-execution methods that collect the results from target devices via HTTP(S) upload (see more on that in chapter Inventory Methods Overview).

You can also use the HTTP upload service to send results from custom implementations of NDTRACK / ORATRACK scans, by using command-line parameters to redirect the output to a specified upload location. The configuration requires a full URL to the local upload server. You can specify it on your own, knowing the configured port and full machine name or let RayVentory Scan Engine help you. Simply press the button **NOTIFICATION CENTER** in the upper-right corner to reveal a pop-up menu. In this flyout, a separate entry is dedicated to the status of the HTTP service, with the possibility to copy its full URL address.

# Uploading Results to Parent Servers

The upload transports the collected inventory files to the HTTP / HTTPS endpoint configured by the setting Upload URL in the UPLOAD section of the settings. Depending on the implementation, it may be any of the following:

- Another RayVentory Scan Engine instance
- RayManageSoft Reporting Location
- RayVentory Reporting Location

RayVentory Scan Engine also supports the following upload locations:

- Local directory
- UNC folder path
- FTP address

An upload URL for the default configuration would look like `http://myRayVentoryServerHostname/ManageSoftRL/inventories` (when reusing the RayVentory / RayManagesoft Reporting Location) or `http://myRayVentoryScanEngineHostname:port/` (when using HTTP Upload Server provided by the parent RayVentory Scan Engine instance).

An upload can be triggered by clicking on the **Upload** tile in the Notification Center. The button is only visible if the upload location is configured and any of the following is true:

- There are some pending uploads.
- The option to upload legacy discovery files is active AND the upload location for legacy discovery files is set.

## Configuring Parent Server URL and Authentication

The upload feature supports authentication by providing a username / password pair that is taken from the Windows type credentials from the credential store or the authentication by a client certificate. You can learn more about parent upload settings in the following chapter: Upload Location.

## Providing Custom HTTP Upload Handlers

The parent upload server can be configured as a custom Microsoft IIS web service that is using anonymous authentication and allows HTTP PUT for compressed (`.ndi.gz`) and uncompressed NDI files (`.ndi`).

# Inventory Methods Overview

This chapter contains detailed descriptions of every supported inventory method together with their impact on the security, the architecture, and general recommendations.

The availability of methods for given jobs may be limited by the following factors:

- Job type (Windows / UNIX / Oracle / vSphere / SNMP etc.)
- Device type (Windows / UNIX)
- Declared device capabilities
- RayVentory Scan Engine settings (HTTP server, Zero-Touch and Remote-Execution settings)
- Current selection of active Inventory methods
- Currently installed software (for example Java runtime)
- Ports opened on the target device
- Other factors

The following guide will help you understand these dependencies and find out which methods are suitable in your environment.

The methods are divided into the following groups:

- Windows Inventory
- Linux / UNIX Inventory
- Oracle Audit
- Oracle Inventory
- SNMP Inventory
- ESX / vSphere Inventory

## Windows Inventory

These inventory methods gather basic hard- and software inventory data on the target device from the Windows OS. The inventory methods operate on targets of the type **Device**.

### Zero-Touch Inventory by WMI / WINAPI on Windows

#### Description

The Zero-Touch OS / platform inventory for Windows hosts using WMI queries. Using WINAPI based Windows-Registry queries as a fallback for earlier Windows versions that do not provide the StdReg-WMI-Provider.

#### Usage and Recommendation

This is the least invasive inventory method for Windows.. The only requirements are sufficient

privileges to run all required WMI queries to RayVentory Scan Engine. This inventory method can be customized to gather additional data, available via WMI or from the Windows registry.

### Technical Details

- **Used credential type:**
  - Windows
- **Required capabilities:**
  - Zero-touch
  - WMI
- **Prerequisites:**
  - None

## Remote-Execution Inventory by Service Manager / SMB Local Files on Windows

### Description

Remote-Execution OS / platform inventory for Windows hosts by a temporary local copy of NDTRACK pushed via SMB, executed by a temporary service, and inventory copied via SMB. RayVentory Scan Engine mounts the target's built-in share `ADMIN$`, copies the NDTRACK files to a temporary subdirectory of `ADMIN$\Temp\`, and starts the NDTRACK via a temporary service. Eventually, the resulting inventory file is copied to the RayVentory Scan Engine host. Later, the temporary service and the temporary directory are deleted.

### Usage and Recommendation

Use this as an alternative to the zero-touch inventory on Windows. This inventory method uses NDTRACK.

### Technical Details

- **Used credential type:**
  - Windows
- **Required capabilities:**
  - Access to the File System
  - Remote execution
  - Windows Service Manager
- **Prerequisites:**
  - None

## Remote-Execution Inventory by WMI / SMB Local Files on Windows

### Description

Remote-Execution OS / platform inventory for Windows hosts by a temporary local copy of NDTRACK pushed via SMB, executed via WMI, and inventory copied via SMB.
RayVentory Scan Engine mounts the target's built-in share `ADMIN$`, copies the NDTRACK files to a temporary subdirectory of `ADMIN$\Temp\`, and starts the NDTRACK by via a temporary service. Eventually, the resulting inventory file is copied to the RayVentory Scan Engine host. Later, the temporary directory is deleted.

### Usage and Recommendation

Use this as an alternative to the zero-touch inventory on Windows. This inventory method uses NDTRACK.

### Technical details

- **Used credential type:**
  - Windows
- **Required capabilities:**
  - Access to the File System
  - Remote execution
  - WMI
- **Prerequisites:**
  - None

## Remote-Execution Inventory by Service Manager Upload HTTP(S) on Windows

### Description

Remote-Execution OS / platform inventory for Windows hosts by NDTRACK, loaded via SMB from the UNC path to the RayVentory Scan Engine utilities, executed by a temporary service, and inventory uploaded via HTTP(S).
RayVentory Scan Engine starts a temporary service that references the service executable, located on the RayVentory Scan Engine utilities share, which starts the NDTRACK, located on the same share. NDTRACK will upload its results to the RayVentory Scan Engine HTTP Service. Later, the temporary service is deleted.

### Usage and Recommendation

Use this as an alternative to the zero-touch inventory on Windows. This inventory method uses NDTRACK.

### Technical Details

- **Used credential type:**
  - Windows
- **Required capabilities:**

  - Remote execution
  - Windows Service Manager
- **Prerequisites:**
  - Configured and reachable RayVentory Scan Engine HTTP Service

## Remote-Execution by Service Manager Upload SMB on Windows

### Description

Remote-Execution OS / platform inventory for Windows hosts by NDTRACK, loaded via SMB from the UNC path to the RayVentory Scan Engine utilities, executed by a temporary service, and inventory uploaded via SMB to the UNC path for RayVentory Scan Engine inventories.
RayVentory Scan Engine starts a temporary service that references the service executable located on the RayVentory Scan Engine utilities share which starts the NDTRACK, located on the same share. NDTRACK will upload its results to the RayVentory Scan Engine inventories share. Later, the temporary service is deleted.

### Usage and Recommendation

Use this as an alternative to the zero-touch ORATRACK inventory. You may need this if remote connections to the database are blocked by the host firewall and local connections are allowed.

### Technical Details

- **Used credential type:**
  - Windows
- **Required capabilities:**
  - Remote execution
  - Upload to the SMB shares
  - Windows Service Manager
- **Prerequisites:**
  - SMB share to receive inventory files
  - The setting **Save inventory results from target devices on the following UNC share** must be configured in the Remote execution section of the **Settings** screen

## Remote-Execution by WMI Upload HTTP(S) on Windows

### Description

Remote-Execution OS / platform inventory for Windows hosts by NDTRACK, loaded via SMB from the UNC path to the RayVentory Scan Engine utilities, executed via WMI, and inventory uploaded via HTTP(S).
RayVentory Scan Engine starts the NDTRACK, located on the RayVentory Scan Engine utilities share via WMI (`Win32_Process`). NDTRACK will upload its results to the RVP HTTP Service.

### Usage and Recommendation

Use this as an alternative to the zero-touch ORATRACK inventory. You may need this if remote connections to the DB are blocked by the host firewall and local connections are allowed.

### Technical Details

- **Used credential type:**
  - Windows
- **Required capabilities:**
  - Remote execution
  - WMI
- **Prerequisites:**
  - The setting **Execute RayVentory Scan Engine Utiliites from the follwing UNC shared path on target device** must be configured in the Remote execution section of the **Settings** screen
  - The scan utilities must be present on the shared location (for example installed via the **Install Scan Utilities** button in the Remote execution section of the **Settings** screen)
  - A configured and reachable RayVentory Scan Engine HTTP Service

## Remote-Execution by WMI Upload SMB on Windows

### Description

Remote-Execution OS / platform inventory for Windows hosts by NDTRACK, loaded via SMB from the UNC path to the RayVentory Scan Engine utilities, executed via WMI, and inventory uploaded via SMB to the UNC path for RayVentory Scan Engine inventories.
RayVentory Scan Engine starts the NDTRACK, located on the RayVentory Scan Engine utilities share, via WMI (`Win32_Process`). NDTRACK will upload its results to the RayVentory Scan Engine inventories share.

### Usage and Recommendation

Use this as an alternative to the zero-touch ORATRACK inventory. You may need this if remote connections to the DB are blocked by the host firewall and local connections are allowed.

### Technical Details

- **Used credential type:**
  - Windows
- **Required capabilities:**
  - Remote execution
  - Upload to the SMB shares
  - WMI
- **Prerequisites:**
  - The SMB share to receive inventory files
  - The setting **Execute RayVentory Scan Engine Utiliites from the follwing UNC shared path on target device** must be configured in the Remote execution section of the **Settings** screen
  - The scan utilities must be present on the shared location (for example installed via the **Install Scan Utilities** button in the Remote execution section of the **Settings** screen)

- The setting **Save inventory results from target devices on the following UNC share** must be configured in the Remote execution section of the **Settings** screen

## Linux / UNIX Inventory

These inventory methods gather basic hard- and software inventory data on the target device from the Linux / UNIX-like Operating Systems. The inventory methods operate on targets of the type **Device**.

### Zero-Touch Inventory by WMI / WINAPI on Linux / UNIX

#### Description

Zero-Touch OS / platform inventory for Linux / UNIX hosts by RIU, using SSH remote commands to query platform specific standard configuration and diagnosis tools and utilities.

#### Usage and Recommendation

This is the least invasive inventory method for Linux / UNIX-like Operating Systems. The only requirements are sufficient privileges to run all the needed commands and utilities or use privilege elevation by `sudo`, `pbrun` or `priv` for RayVentory Scan Engine.

#### Technical Details

- **Used credential type:**
  - SSH
- **Required capabilities:**
  - Zero-touch
- **Prerequisites:**
  - None

### Remote-Execution Inventory by SSH / SCP Local Files on Linux / Unix

#### Description

Remote-Execution OS / platform inventory for Linux / Unix by a temporary local copy of NDTRACK pushed via SCP, executed via SSH, and inventory copied via SCP.
RayVentory Scan Engine copies the NDTRACK files via SCP to a temporary subdirectory of the home directory for the user associated with the username of the SSH credentials. Then it runs NDTRACK by issuing a command via SSH. Eventually, the resulting inventory file is copied to the RayVentory Scan Engine host. Later, the temporary directory is deleted.

#### Usage and Recommendation

Use this as an alternative to the zero-touch inventory on Linux / UNIX-like Operating Systems. This inventory method uses NDTRACK.

### Technical Details

- **Used credential type:**
  - SSH
- **Required capabilities:**
  - Access to the File System
  - Remote Execution
  - SSH
- **Prerequisites:**
  - None

## Oracle Audit

This inventory method gathers files produced by the **Review Lite Script** (provided by Oracle to their customers) or similar scripts on Oracle database instances.
This method operates on targets of the type **Oracle**.

### Zero-Touch Audit by SQLPLUS

#### Description

Zero-Touch Oracle audit (executing the 'review lite script') by an SQLPLUS remote session.
This is a tool for running the 'Review Lite Script' that customers receive via a local installation of SQL*plus.

#### Usage and Recommendation

Use this to help you run the 'Review Lite Script' that you received from the DB vendor on many DB installations at once.

#### Technical Details

- **Used credential type:**
  - Oracle DB
- **Required capabilities:**
  - Zero-touch
- **Prerequisites:**
  - Path to SQL Plus executable has to be configured in the Oracle section of the **Settings** screen.

# Oracle Inventory

These inventory methods gather data on enabled options and usage on Oracle database instances.
These methods operate on targets of the type Oracle.

## Zero-Touch Inventory by ORATRACK

### Description

Zero-Touch Oracle remote inventory by ORATRACK.

### Usage and Recommendation

Use this to generate an inventory of your database instances. This is the least invasive method to create such an inventory. The only requirements are credentials with sufficient privileges on the target database for RayVentory Scan Engine.

### Technical Details

- **Used credential type:**
  - Oracle DB
- **Required capabilities:**
  - Zero-touch
- **Prerequisites:**
  - The setting **Java executable path** has to be configured in the Oracle section of the **Settings** screen
  - Java has to be installed on the host machine
  - (Optional) The setting **Oracle Database Feature Usage Statistics script path** must be configured in the Oracle section of the **Settings** screen to run the DBFUS script together with the inventory

## Remote-Execution by Service Manager / SMB Local Files on Windows

### Description

Zero-Touch Oracle remote inventory by ORATRACK. Remote-Execution Oracle inventory for Windows hosts by a temporary local copy of ORATRACK pushed via SMB, executed by a temporary service, and inventory copied via SMB. RayVentory Scan Engine mounts the target's built-in share `ADMIN$`, copies at least one version of the ORATRACK Java executable and the encrypted queries file to a temporary subdirectory of `ADMIN$\Temp\`, and starts ORATRACK by one of the local Java runtimes via a temporary service. Eventually, the resulting inventory file is copied to the RayVentory Scan Engine host. Later, the temporary service and the temporary directory are deleted.

### Usage and Recommendation

Use this as an alternative to the zero-touch ORATRACK inventory. You may need this if remote connections to the DB are blocked by the host firewall and local connections are allowed. This is the most invasive ORATRACK inventory method as ORATRACK is (temporarily) copied to the target and a temporary service is installed.

### Technical Details

- **Used credential type:**
  - Oracle DB Windows
- **Required capabilities:**
  - Access to the File System
  - Remote execution
  - Windows Service Manager
- **Prerequisites:**
  - None

## Remote-Execution by SSH / SCP Local Files on Linux / Unix

### Description

Remote-Execution Oracle inventory for Linux / Unix hosted by a temporary, local copy of ORATRACK, pushed via SCP, executed via SSH, and inventory copied via SCP. RayVentory Scan Engine copies at least one version of ORATRACK and the encrypted queries file via SCP to a temporary subdirectory of the home directory for the user associated with the username of the SSH credentials. Then it runs ORATRACK via a local Java runtime by issuing a command via SSH. Eventually, the resulting inventory file is copied to the RayVentory Scan Engine host. Later, the temporary directory is deleted.

### Usage and Recommendation

Use this as an alternative for the zero-touch ORATRACK inventory. You may need this if remote connections to the DB are blocked by the host firewall and local connections are allowed.

### Technical Details

- **Used credential type:**
  - Oracle DB SSH
- **Required capabilities:**
  - Access to to File System
  - Remote execution
  - SSH
- **Prerequisites:**
  - None

# Remote-Execution by WMI / SMB Local Files on Windows

## Description

Remote-Execution Oracle inventory for Windows hosts by a temporary, local copy of ORATRACK pushed via SMB, executed via WMI, and inventory copied via SMB. RayVentory Scan Engine mounts the target's built-in share `ADMIN$`, copies at least one version of the ORATRACK Java executable and the encrypted queries file to a temporary subdirectory of `ADMIN$\Temp\`, and starts ORATRACK by one of the local Java runtimes via a temporary service. Eventually, the resulting inventory file is copied to the RayVentory Scan Engine host. Later, the temporary directory is deleted.

## Usage and Recommendation

Use this as an alternative to the zero-touch ORATRACK inventory. You may need this if remote connections to the DB are blocked by the host firewall and local connections are allowed.

## Technical Details

- **Used credential type:**
  - Oracle DB Windows
- **Required capabilities:**
  - Access to the File System
  - Remote execution
  - WMI
- **Prerequisites:**
  - None

# Remote-Execution by Service Manager with HTTP(S) Upload on Windows

## Description

Remote-Execution Oracle inventory for Windows hosts by ORATRACK, loaded via SMB from the UNC path to the RayVentory Scan Engine utilities, executed by a temporary service, and inventory uploaded via HTTP(S). RayVentory Scan Engine starts a temporary service that references the service executable located on the RayVentory Scan Engine utilities share which starts ORATRACK by a Java runtime expected on the target located on the same share. ORATRACK will upload its results to the RayVentory Scan Engine HTTP Service. Later, the temporary service is deleted

## Usage and Recommendation

Use this as an alternative to the zero-touch ORATRACK inventory. You may need this if remote connections to the DB are blocked by the host firewall and local connections are allowed.

### Technical Details

- **Used credential type:**
  - Oracle DB Windows
- **Required capabilities:**
  - Remote execution
  - Windows Service Manager
- **Prerequisites:**
  - Configured and reachable RayVentory Scan Engine HTTP Service

## Remote-Execution by Service Manager Upload SMB on Windows

### Description

Remote-Execution Oracle inventory for Windows hosts by ORATRACK, loaded via SMB from the UNC path to the RayVentory Scan Engine utilities, executed by a temporary service and inventory uploaded via SMB to the UNC path for RayVentory Scan Engine inventories.
RayVentory Scan Engine starts a temporary service that references the service executable located on the RayVentory Scan Engine utilities share which starts ORATRACK, located on the same share by a Java runtime expected on the target. ORATRACK will upload its results to the RayVentory Scan Engine inventories share. Later, the temporary service is deleted.

### Usage and Recommendation

Use this as an alternative to the zero-touch ORATRACK inventory. You may need this if remote connections to the DB are blocked by the host firewall and local connections are allowed.

### Technical Details

- **Used credential type:**
  - Oracle DB Windows
- **Required capabilities:**
  - Remote execution
  - Upload to the SMB shares
  - Windows Service Manager
- **Prerequisites:**
  - SMB share to receive inventory files
  - The setting **Save inventory results from target devices on the following UNC share** must be configured in the Remote Execution section of the **Settings** screen

## Remote-Execution by WMI Upload HTTP(S) on Windows

### Description

Remote-Execution Oracle inventory for Windows hosts by ORATRACK, loaded via SMB from the UNC path to the RVP utilities, and executed via WMI and inventory uploaded via HTTP(S).
RayVentory Scan Engine starts ORATRACK, located on the RayVentory Scan Engine utilities share

by a Java runtime expected on the target via WMI (`Win32_Process`). ORATRACK will upload its results to the RayVentory Scan Engine HTTP Service.

### Usage and Recommendation

This is the second least invasive method together with **Remote-Execution by WMI upload SMB on Windows** to execute an Oracle inventory by ORATRACK. No files are copied to the target, no temporary service is installed. RayVentory Scan Engine will just create a process that starts a local Java runtime to run ORATRACK.

### Technical Details

- **Used credential type:**
  - Oracle DB Windows
- **Required capabilities:**
  - Remote execution
  - WMI
- **Prerequisites:**
  - The setting **Execute RayVentory Scan Engine Utiliites from the follwing UNC shared path on target device** must be configured in the Remote Execution section of the **Settings** screen
  - The scan utilities must be present on the shared location (for example installed via the **Install Scan Utilities** button in the Remote Execution section of the **Settings** screen)

## Remote-Execution by WMI Upload SMB on Windows

### Description

Remote-Execution Oracle inventory for Windows hosts by ORATRACK, loaded via SMB from the UNC path to the RayVentory Scan Engine utilities, executed via WMI, and inventory uploaded via SMB to the UNC path for RayVentory Scan Engine inventories.

RayVentory Scan Engine starts ORATRACK, located on the RayVentory Scan Engine utilities share by a Java runtime expected on the target via WMI (`Win32_Process`). ORATRACK will upload its results to the RayVentory Scan Engine inventories share.

### Usage and Recommendation

This is the second least invasive method together with **Remote-Execution by WMI upload HTTP(S) on Windows** to execute an Oracle inventory by ORATRACK. No files are copied to the target, no temporary service is installed. RayVentory Scan Engine will just create a process that starts a local Java runtime to run ORATRACK.

### Technical Details

- **Used credential type:**
  - Oracle DB Windows
- **Required capabilities:**

- Remote execution
- Upload to the SMB shares
- WMI

- **Prerequisites:**
  - SMB share to receive inventory files
  - The setting **Execute RayVentory Scan Engine Utiliites from the follwing UNC shared path on target device** must be configured in the Remote Execution section of the **Settings** screen
  - The scan utilities must be present on the shared location (for example installed via the **Install Scan Utilities** button in the Remote Execution section of the **Settings** screen)
  - The setting **Save inventory results from target devices on the following UNC share** must be configured in the Remote Execution section of the **Settings** screen

## SNMP Inventory

This inventory method gathers basic data on network devices that support the SNMP protocol. This is intended for devices like printers, UPSs, and network equipment that do not expose Operating Systems or a standardized interfaces besides SNMP. This method operates on targets of the type **SNMP**.

### Zero-Touch by SNMP

#### Description

Zero-Touch SNMP based inventory by the SNMP Tracker for SNMP enabled devices, supporting SNMP versions 1, 2/2c, and 3. This is intended for network devices like printers and UPSs.

#### Usage and Recommendation

Use this to gather basic configuration and inventory data from SNMP enabled network devices. We currently support retrieving data on model, manufacturer, and serial number from a range of different printers, switches, routers, and UPSs. This inventory method can be customized to gather vendor or device specific data.

#### Technical Details

- **Used credential type:**
  - SNMP
- **Required capabilities:**
  - Zero-touch
- **Prerequisites:**
  - None

## ESX / vSphere Inventory

This inventory method gathers data on virtual guests and host configurations from a VMware vSphere instance or single VMware ESX hosts. RayVentory Scan Engine has experimental support for VMware Workstation instances as well, but their data is not taken into account in all of the related reports.
This method operates on targets of type ESX / vSphere.

### Zero-Touch Inventory by VMware API via HTTP(S)

#### Description

Zero-Touch VSphere / ESX inventory using the VMware Management API via HTTP(S).

#### Technical details

- **Used credential type:**
  - ESX / vSphere
- **Required capabilities:**
  - Zero-touch
- **Prerequisites:**
  - None

# Application of Credentials

RayVentory Scan Engine differentiates between five different credential types:

- Windows
- SSH
- Oracle DB
- SNMP
- vSphere

### Windows and SSH Credentials

The types Windows and SSH are used for OS / Device inventory and during Discovery for discovering / probing Oracle databases. The remote execution based inventory methods for Oracle use these credentials too, in order to find evidence for Oracle instances in the host file system and services and for authentication to the host system.

### Oracle DB Credentials

The type Oracle DB is used for authentication against Oracle databases for creating the Oracle inventory.

### SNMP Credentials

The type SNMP is used for selecting an SNMP community or authentication by SNMP v3.

### vSphere Credentials

The type vSphere is used for authentication against the VMware management API for the creation of inventories of vSphere / ESX virtualization hosts.

In general, the application of credentials to a target system / service happens in the order they appear in the credentials list. If certain credentials have been set for a device / service then these are tried first, before other credentials are tried. On a successful inventory, RayVentory Scan Engine remembers which credentials have been used for a device or service and tries these first the next time an inventory is run.

Credentials may be restricted to be applied to certain devices. To restrict credentials to a specific hostname enter that hostname or IP address into the field **Target name pattern**. This field supports Regular Expressions.

**WARNING**

Whilst the idea of creating a list of credentials to try on each host may sound handy (especially if you are not certain which credentials actually belong to a certain device / service), keep in mind that such brute-forcing of credentials may trigger alarms in your security and network monitoring solution. In case of an Oracle database you may even lock a user as by default, as three failed authentication attempts in a row will lock the user / login that has been used. It is highly recommended to use as few generic credentials as possible, by either selecting concrete credentials for devices or setting up filters to limit their application.

# Custom Windows Scans

Zero-Touch Inventory Scan for Windows can be customized by providing a specially prepared `.xml` containing instructions how to query the target device via WMI, as well as file system and the Windows registry.

## To Create a Custom Scan Definition...

1. Go to the installation directory of RayVentory Scan Engine.
2. Start the executable file `RemoteWmiInventory.exe` with a single argument `example`.
3. Save the created file as scan template (by default it will be saved in your current working directory under the name `example.xml`).
4. Customize the file to include custom scanned content.

**Be aware:**
The generated content does not reflect the default settings of Windows Inventory scans.

After the file is created and customized, you can point to it by configuring the path under Settings > Inventory > Windows > Custom configuration.

**WARNING**
This is a critical piece of the Windows Inventory functionality which can be easily broken by incomplete or incorrect configurations. We recommend to have it configured by one of our RayVentory Scan Engine consultants. Failing to correctly customize the Windows scan may result in broken scans or a huge load on the target systems and the networks.

## Sample Configuration File

The following shows a typical content of a custom scan definition:

```
<?xml version="1.0" encoding="utf-8"?>
<QueryFile xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <Queries Mandatory="true" IsSoftware="false"
WmiClass="Win32_ComputerSystem" Namespace="\root\cimv2"
Name="Name">
    <Fields WmiPropertyName="Manufacturer" />
    <Fields WmiPropertyName="Model" />
    <Fields WmiPropertyName="Domain" />
    <Fields WmiPropertyName="DomainRole" />
    <Fields WmiPropertyName="NumberOfProcessors" />
    <Fields WmiPropertyName="NumberOfLogicalProcessors" />
    <Fields WmiPropertyName="TotalPhysicalMemory" />
    <Fields WmiPropertyName="Status" />
```

```
    <Fields WmiPropertyName="UserName" />
  </Queries>
  <Queries Mandatory="false" IsSoftware="false"
WmiClass="Win32_ComputerSystemProduct" Namespace="\root\cimv2"
Name="Name">
    <Fields WmiPropertyName="IdentifyingNumber" />
    <Fields WmiPropertyName="Name" />
    <Fields WmiPropertyName="UUID" />
    <Fields WmiPropertyName="Vendor" />
    <Fields WmiPropertyName="Version" />
  </Queries>
  <Queries Mandatory="false" IsSoftware="false"
WmiClass="Win32_OperatingSystem" Namespace="\root\cimv2"
Name="Caption">
    <Fields WmiPropertyName="Name" />
    <Fields WmiPropertyName="Manufacturer" />
    <Fields WmiPropertyName="Version" />
    <Fields WmiPropertyName="ServicePackMajorVersion" />
    <Fields WmiPropertyName="ServicePackMinorVersion" />
    <Fields WmiPropertyName="SerialNumber" />
    <Fields WmiPropertyName="InstallDate" />
    <Fields WmiPropertyName="LastBootUpTime" />
    <Fields WmiPropertyName="OSLanguage" />
    <Fields WmiPropertyName="FreePhysicalMemory" />
    <Fields WmiPropertyName="FreeVirtualMemory" />
    <Fields WmiPropertyName="CountryCode" />
    <Fields WmiPropertyName="WindowsDirectory" />
    <Fields WmiPropertyName="SystemDirectory" />
    <Fields WmiPropertyName="Caption" />
    <Fields WmiPropertyName="CSDVersion" />
    <Fields WmiPropertyName="Status" />
    <Fields WmiPropertyName="CSName" />
    <Fields WmiPropertyName="OSType" />
    <Fields WmiPropertyName="OSArchitecture" />
  </Queries>
  <Queries Mandatory="false" IsSoftware="false"
WmiClass="Win32_BIOS" Namespace="\root\cimv2" Name="Name">
    <Fields WmiPropertyName="Manufacturer" />
    <Fields WmiPropertyName="Version" />
    <Fields WmiPropertyName="ReleaseDate" />
    <Fields WmiPropertyName="SerialNumber" />
    <Fields WmiPropertyName="BiosCharacteristics" />
    <Fields WmiPropertyName="Status" />
  </Queries>
  <Queries Mandatory="false" IsSoftware="false"
WmiClass="Win32_Processor" Namespace="\root\cimv2" Name="Name">
    <Fields WmiPropertyName="Description" />
    <Fields WmiPropertyName="Manufacturer" />
```

```
      <Fields WmiPropertyName="Version" />
      <Fields WmiPropertyName="ProcessorId" />
      <Fields WmiPropertyName="CurrentClockSpeed" />
      <Fields WmiPropertyName="CurrentVoltage" />
      <Fields WmiPropertyName="L2CacheSize" />
      <Fields WmiPropertyName="Status" />
      <Fields WmiPropertyName="MaxClockSpeed" />
      <Fields WmiPropertyName="Name" />
      <Fields WmiPropertyName="ProcessorType" />
      <Fields WmiPropertyName="NumberOfLogicalProcessors" />
      <Fields WmiPropertyName="NumberOfCores" />
      <Fields WmiPropertyName="DeviceID" />
    </Queries>
    <Queries Mandatory="false" IsSoftware="false"
  WmiClass="Win32_DiskDrive" Namespace="\root\cimv2" Name="Name">
      <Fields WmiPropertyName="Description" />
      <Fields WmiPropertyName="InterfaceType" />
      <Fields WmiPropertyName="Manufacturer" />
      <Fields WmiPropertyName="Model" />
      <Fields WmiPropertyName="Partitions" />
      <Fields WmiPropertyName="Size" />
      <Fields WmiPropertyName="Status" />
    </Queries>
    <Queries Mandatory="false" IsSoftware="false"
  WmiClass="Win32_LogicalDisk" Namespace="\root\cimv2" Name="Name">
      <Fields WmiPropertyName="Description" />
      <Fields WmiPropertyName="VolumeName" />
      <Fields WmiPropertyName="FileSystem" />
      <Fields WmiPropertyName="FreeSpace" />
      <Fields WmiPropertyName="Size" />
      <Fields WmiPropertyName="VolumeSerialNumber" />
      <Fields WmiPropertyName="DriveType" />
      <Fields WmiPropertyName="MediaType" />
      <Fields WmiPropertyName="Status" />
      <Fields WmiPropertyName="ProviderName" />
    </Queries>
    <Queries Mandatory="false" IsSoftware="false"
  WmiClass="Win32_CDROMDrive" Namespace="\root\cimv2" Name="Name">
      <Fields WmiPropertyName="Description" />
      <Fields WmiPropertyName="Manufacturer" />
      <Fields WmiPropertyName="Drive" />
      <Fields WmiPropertyName="Status" />
      <Fields WmiPropertyName="Capabilities" />
    </Queries>
    <Queries Mandatory="false" IsSoftware="false"
  WmiClass="Win32_NetworkAdapter" Namespace="\root\cimv2"
  Name="Name">
      <Fields WmiPropertyName="Manufacturer" />
```

```
    <Fields WmiPropertyName="MACAddress" />
    <Fields WmiPropertyName="MaxSpeed" />
    <Fields WmiPropertyName="Speed" />
    <Fields WmiPropertyName="Status" />
  </Queries>
  <Queries Mandatory="false" IsSoftware="false"
WmiClass="Win32_NetworkAdapterConfiguration" Namespace="\root
\cimv2" Name="Caption">
    <Fields WmiPropertyName="Caption" />
    <Fields WmiPropertyName="Description" />
    <Fields WmiPropertyName="Index" />
    <Fields WmiPropertyName="MACAddress" />
    <Fields WmiPropertyName="IPEnabled" />
    <Fields WmiPropertyName="DHCPEnabled" />
    <Fields WmiPropertyName="IPAddress" />
    <Fields WmiPropertyName="DHCPServer" />
    <Fields WmiPropertyName="DNSHostName" />
    <Fields WmiPropertyName="DNSDomain" />
    <Fields WmiPropertyName="DNSServerSearchOrder" />
    <Fields WmiPropertyName="DefaultIPGateway" />
    <Fields WmiPropertyName="IPSubnet" />
  </Queries>
  <Queries Mandatory="false" IsSoftware="false"
WmiClass="Win32_PhysicalMemory" Namespace="\root\cimv2"
Name="Name">
    <Fields WmiPropertyName="Capacity" />
    <Fields WmiPropertyName="MemoryType" />
    <Fields WmiPropertyName="PositionInRow" />
    <Fields WmiPropertyName="Speed" />
    <Fields WmiPropertyName="Status" />
  </Queries>
  <Queries Mandatory="false" IsSoftware="false"
WmiClass="Win32_SoundDevice" Namespace="\root\cimv2" Name="Name">
    <Fields WmiPropertyName="Name" />
    <Fields WmiPropertyName="Manufacturer" />
  </Queries>
  <Queries Mandatory="false" IsSoftware="false"
WmiClass="Win32_VideoController" Namespace="\root\cimv2"
Name="Name">
    <Fields WmiPropertyName="Name" />
    <Fields WmiPropertyName="VideoProcessor" />
    <Fields WmiPropertyName="DriverVersion" />
    <Fields WmiPropertyName="DriverDate" />
    <Fields WmiPropertyName="InstalledDisplayDrivers" />
    <Fields WmiPropertyName="AdapterRAM" />
  </Queries>
```

```
  <Queries Mandatory="false" IsSoftware="false"
WmiClass="Win32_VideoConfiguration" Namespace="\root\cimv2"
Name="Name">
    <Fields WmiPropertyName="AdapterRAM" />
    <Fields WmiPropertyName="AdapterType" />
    <Fields WmiPropertyName="Description" />
    <Fields WmiPropertyName="HorizontalResolution" />
    <Fields WmiPropertyName="MonitorManufacturer" />
    <Fields WmiPropertyName="MonitorType" />
    <Fields WmiPropertyName="Name" />
    <Fields WmiPropertyName="VerticalResolution" />
  </Queries>
  <Queries Mandatory="false" IsSoftware="false"
WmiClass="Win32_SystemEnclosure" Namespace="\root\cimv2"
Name="Name" />
  <Queries Mandatory="false" IsSoftware="false"
WmiClass="SoftwareLicensingProduct" Namespace="\root\cimv2"
Name="Name">
    <Fields WmiPropertyName="ApplicationID" />
    <Fields WmiPropertyName="Description" />
    <Fields WmiPropertyName="EvaluationEndDate" />
    <Fields WmiPropertyName="GracePeriodRemaining" />
    <Fields WmiPropertyName="LicenseStatus" />
    <Fields WmiPropertyName="MachineURL" />
    <Fields WmiPropertyName="Name" />
    <Fields WmiPropertyName="OfflineInstallationId" />
    <Fields WmiPropertyName="PartialProductKey" />
    <Fields WmiPropertyName="ProcessorURL" />
    <Fields WmiPropertyName="ProductKeyID" />
    <Fields WmiPropertyName="ProductKeyURL" />
    <Fields WmiPropertyName="UseLicenseURL" />
  </Queries>
  <Queries Mandatory="false" IsSoftware="false"
WmiClass="SoftwareLicensingService" Namespace="\root\cimv2"
Name="KeyManagementServiceProductKeyID">
    <Fields WmiPropertyName="ClientMachineID" />
    <Fields WmiPropertyName="IsKeyManagementServiceMachine" />
    <Fields WmiPropertyName="KeyManagementServiceCurrentCount" />
    <Fields WmiPropertyName="KeyManagementServiceMachine" />
    <Fields WmiPropertyName="KeyManagementServiceProductKeyID" />
    <Fields WmiPropertyName="PolicyCacheRefreshRequired" />
    <Fields WmiPropertyName="RequiredClientCount" />
    <Fields WmiPropertyName="Version" />
    <Fields WmiPropertyName="VLActivationInterval" />
    <Fields WmiPropertyName="VLRenewalInterval" />
  </Queries>
  <Queries Mandatory="false" IsSoftware="true"
WmiClass="Win32_Product" Namespace="\root\cimv2" Name="Name" />
```

```
  <Queries Mandatory="false" IsSoftware="false"
WmiClass="MGS_MSSQL2000" Namespace="\root\cimv2" Name="Name" />
  <Queries Mandatory="false" IsSoftware="false"
WmiClass="MGS_MSSQL20058" Namespace="\root\cimv2" Name="Name" />
  <Queries Mandatory="false" IsSoftware="false"
WmiClass="Msvm_ComputerSystem" Namespace="\root\virtualization"
Name="Name" Evidence="virtualization" />
  <Queries Mandatory="false" IsSoftware="false"
WmiClass="Msvm_VirtualSystemSettingData" Namespace="\root
\virtualization" Name="Name" Evidence="virtualization" />
  <Queries Mandatory="false" IsSoftware="false"
WmiClass="Msvm_MemorySettingData" Namespace="\root\virtualization"
Name="Name" Evidence="virtualization" />
  <Queries Mandatory="false" IsSoftware="false"
WmiClass="Msvm_ProcessorSettingData" Namespace="\root
\virtualization" Name="Name" Evidence="virtualization" />
  <Queries Mandatory="false" IsSoftware="false"
WmiClass="Msvm_KvpExchangeComponent" Namespace="\root
\virtualization" Name="Name" Evidence="virtualization" />
  <Queries Mandatory="false" IsSoftware="false"
WmiClass="Msvm_ComputerSystem" Namespace="\root\virtualization\v1"
Name="Name" Evidence="virtualization\v1" />
  <Queries Mandatory="false" IsSoftware="false"
WmiClass="Msvm_VirtualSystemSettingData" Namespace="\root
\virtualization\v1" Name="Name" Evidence="virtualization\v1" />
  <Queries Mandatory="false" IsSoftware="false"
WmiClass="Msvm_MemorySettingData" Namespace="\root\virtualization
\v1" Name="Name" Evidence="virtualization\v1" />
  <Queries Mandatory="false" IsSoftware="false"
WmiClass="Msvm_ProcessorSettingData" Namespace="\root
\virtualization\v1" Name="Name" Evidence="virtualization\v1" />
  <Queries Mandatory="false" IsSoftware="false"
WmiClass="Msvm_KvpExchangeComponent" Namespace="\root
\virtualization\v1" Name="Name" Evidence="virtualization\v1" />
  <Queries Mandatory="false" IsSoftware="false"
WmiClass="Msvm_ComputerSystem" Namespace="\root\virtualization\v2"
Name="Name" Evidence="virtualization\v2" />
  <Queries Mandatory="false" IsSoftware="false"
WmiClass="Msvm_VirtualSystemSettingData" Namespace="\root
\virtualization\v2" Name="Name" Evidence="virtualization\v2" />
  <Queries Mandatory="false" IsSoftware="false"
WmiClass="Msvm_MemorySettingData" Namespace="\root\virtualization
\v2" Name="Name" Evidence="virtualization\v2" />
  <Queries Mandatory="false" IsSoftware="false"
WmiClass="Msvm_ProcessorSettingData" Namespace="\root
\virtualization\v2" Name="Name" Evidence="virtualization\v2" />
```

```
  <Queries Mandatory="false" IsSoftware="false"
WmiClass="Msvm_KvpExchangeComponent" Namespace="\root
\virtualization\v2" Name="Name" Evidence="virtualization\v2" />
  <Queries Mandatory="false" IsSoftware="false"
WmiClass="Win32_serverFeature" Namespace="\root\cimv2"
Name="Name" />
  <Queries Mandatory="false" IsSoftware="false"
WmiClass="MSCluster_Cluster" Namespace="\root\MSCluster"
Name="Name" />
  <Queries Mandatory="false" IsSoftware="false"
WmiClass="Win32_Service" Namespace="\root\cimv2" Name="Name" />
  <Keys Namespace="\root\cimv2" Hive="HKEY_LOCAL_MACHINE"
Path="SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall"
View="64" />
  <Keys Namespace="\root\cimv2" Hive="HKEY_LOCAL_MACHINE"
Path="SOFTWARE\Wow6432Node\Microsoft\Windows\CurrentVersion
\Uninstall" View="32" />
  <Keys Namespace="\root\cimv2" Hive="HKEY_LOCAL_MACHINE"
Path="SOFTWARE\Microsoft\Virtual Machine\Guest\Parameters"
View="32">
    <Values>
      <RegistryValue Name="PhysicalHostNameFullyQualified"
Type="string" />
      <RegistryValue Name="VirtualMachineId" Type="string" />
      <RegistryValue Name="VirtualMachineName" Type="string" />
    </Values>
  </Keys>
  <Keys Namespace="\root\cimv2" Hive="HKEY_LOCAL_MACHINE"
Path="SOFTWARE\Microsoft\Virtual Machine\Guest\Parameters"
View="64">
    <Values>
      <RegistryValue Name="PhysicalHostNameFullyQualified"
Type="string" />
      <RegistryValue Name="VirtualMachineId" Type="string" />
      <RegistryValue Name="VirtualMachineName" Type="string" />
    </Values>
  </Keys>
  <Files GetContent="true" SearchSubdirs="false" Path="temp*\*"
DriveLetter="c" FileName="*.bat" />
</QueryFile>
```

# Custom Scripts for Non-Windows Scans

The custom script feature for RemoteInventoryForUnix (RIU) and ndtrack basically is a new xml based language used to perform custom commands on a Unix or Linux system and to process the results of these commands. It adds more flexibility to ndtrack and RIU by enabling the execution of custom commands and storing the results in the inventory file during its creation.

Custom scripts can be used for the following operating systems:

- HP-UX
- Solaris
- macOS
- AIX
- Linux

A custom script can be execute by using either RIU or ndtrack with the following parameter:

- **RIU:** `scriptFile=<pathToScript>` (the script needs to reside on the host machine on which RIU is going to be executed)
- **ndtrack:** `-o scriptFile=<pathToScript>` (the script needs to reside on the same machine as the `ndtrack.sh`)

## Elements

The following elements are used for custom scripts.

### CustomScriptSet

The root element in the XML structure in which all the custom scripts are placed.

| Attributes | Elements |
|---|---|
| • **Name**: The name of the `CustomScriptSet`.<br>• **Version**: The version that this `CustomScriptSet` and all its scripts, commands, etc. are compatible to. | • `CustomScript` |

**Example:**

```
<CustomScriptSet Name="ExampleName" Version="0.1.1">
      <CustomScript Name[...]>
</CustomScriptSet>
```

### CustomScript

A container for a list of commands that are to be executed. A foreachline loop and/or the item which is written into the inventory file.

| Attributes | Elements |
| --- | --- |
| • **Name**: The name of the `CustomScript` that is executed.<br>• **TargetPlatform** (optional): Used to specify the target platform on which the custom script shoudl be executed. By default, a custom script will be executed on all supported platforms.<br>  ○ **HPUX**: The script will only be executed on HP-UX.<br>  ○ **AIX**: The script will only be executed on AIX.<br>  ○ **macOS**: The script will only be executed on macOS.<br>  ○ **Solaris**: The script will only be executed on Solaris.<br>  ○ **Linux**: The script will only be executed on Linux.<br>  ○ **All**: The script will be executed on all supported platforms. | • `Command`<br>• `Item`<br>• `ForeachLine` |

**Example:**

```
<CustomScript Name="SimpelName">
      <Command Name[...]>
      <Item Name[...]>
      <ForeachLine Values[...]>
</CustomScript>
```

## Command

The command which is going to be executed on the target machine. The output of the executed command will be stored in the variable defined in the **Name** parameter.

| Attributes | Elements |
| --- | --- |
| • **Name**: The name of the command. It can be used as a varialbe in `ForeachLine` sections, the `Item` section and in Properties.<br>• **OnError** (optional): Is used to define how to treat errors. By default, the `Ignore` value will be used.<br>  ○ **Fail**: If the command fails, the whole script will fail and not be processed anymore (if there are more than one script, the next script will be executed).<br>  ○ **Warn**: If the command fails, there is a notification in the log that the script has failed, but the next command/step will be executed.<br>  ○ **Ignore**: If the command fails, there will not be any log entry and the script will continue with the next command/step.<br>• **Text**: the command to be executed on the target machine (needs to be formatted according to XML standard). | • none |

**Example:**

```
<Command Name="CMD1" OnError="Warn">hostname</Command>
```

## ForeachLine

All commands and items defined in a `ForeachLine` will be executed multiple times depending on the number of lines in the input variable set in the parameter `Values`. The `ForeachLine` section needs to contain exactly one item section.

| Attributes | Elements |
|---|---|
| • **Values**: The input variable. This variable has to be defined by a command previously in the script. The number of times the ForeachLine will be executed depends on the number of lines in this variable.<br>• **Name**: During the processing of a ForeachLine script, each line from the input variable will be stored in the Name variable. One after each iteration. The sub commands and items will only see the currently parsed line. The variable and the defined value are exposed to the sub commands and sub item. | • Command<br>• Item |

**Example:**

```
<ForeachLine Values="$(CMD1)" Name="loop1">
      <Command Name[...]>
      <Item Name[...]>
</ForeachLine>
```

## Item

The actual inventory item that will be written into the inventory file. Each CustomScript can only have **one** item! If a CustomScript contains a ForeachLine, the CustomScript cannot contain an item section, as the ForeachLine section will already have one. Each inventory item that is added to the inventory file is stored as a Hardware item.

| Attributes | Elements |
|---|---|
| • **Name**: The internally used name for thiks item.<br>• **Class**: The class which will appear in the inventory file.<br>• **ItemName**: The item name that will be stored as **Name** in the inventory item of the inventory file. | • Property |

**Example:**

```
<Item Name="$(CMD1)" Class="MGS_ComputerSystem_Custom1"
ItemName="$(CMD2)>
      <Property Name[...]>
</Item>
```

### Property

This represents a property in the inventory file.

| Attributes | Elements |
| --- | --- |
| • **Name**: The property name which will be set as name in the inventory file (does not accept variables).<br>• **Value**: The value which will be set a s value in the inventory file. | • none |

**Example:**

```
<Property Name="Hostname" Value="$(CMD1)"/>
```

## Variables

The results of commands are stored in variables which can be accessed via the following syntax:
**`$(MyVariable)`**

It is further possible to treat a variable as an array (splitted by lines only) or as a two dimensional array (splitted by lines and whitespaces) - each array starts with zero.

Syntax to access one specific line of an array (will return the second line of the array):
**`$(MyVariable):1`**

Syntax to access one specific word of a two dimensional array (will return the third word of the second line of a variable):
**`$(MyVariable):1[2]`**

## Example Scripts

A single script for straight execution and the storage of multiple `Commands`:

```
<?xml version="1.0"?>
<CustomScriptSet Name="someName" Version="0.1.1">
      <CustomScript Name="aSimpleExample">
            <Command Name="cmd1" OnError="fail">uname -a</Command>
<!-- If this fails, then the whole inventory fails. -->
            <Command Name="cmd2" OnError="warn">hostname</Command>
<!-- If this fails, a warning is logged (including
CustomScriptSet.Name, ~Version, CustomScript.Name, Command content,
output on stdout, on stderr and exitcode). -->
            <Command Name="cmd3" OnError="ignore">dnsdomain</
Command>
            <Item Name="$(cmd2)" Class="MGS_ComputerSystem_Custom1"
ItemName="$(cmd2)">
                  <Property Name="Hostname" Value="$(cmd2)"/> <!--
Just take the whole output of cmd2 for the value -->
                  <Property Name="Domain" Value="$(cmd3)"/>
                  <Property Name="Description" Value="$(cmd1)"/>
            </Item>
      </CustomScript>
</CustomScriptSet>
```

A single script accessing specific parts of the value of an variable (array):

```
<?xml version="1.0"?>
<CustomScriptSet Name="someName" Version="0.1.1">
      <CustomScript Name="aMoreComplexExample"
TargetPlatform="All">
            <Command Name="cmd1" OnError="fail">uname -a</Command>
            <Command Name="cmd2" OnError="warn">hostname</Command>
            <Command Name="cmd3" OnError="ignore">dnsdomain</
Command>
            <Command Name="cmd4" OnError="warn">ls -la</Command>
            <Item Name="$(cmd2)" Class="MGS_ComputerSystem_Custom2"
ItemName="$(cmd2)">
                  <Property Name="Hostname" Value="$(cmd2)"/>
                  <Property Name="Domain" Value="$(cmd3)"/>
                  <Property Name="Description" Value="$(cmd1)"/>
                  <Property Name="ThirdLineOfOutput"
Value="$(cmd4):2" OnError="ignore"/> <!-- zero-based index for
lines, take the third line of cmd4 for Value -->
                  <Property Name="SecondWordInFourthLineOfOutput"
Value="$(cmd4):2[1]" OnError="ignore"/> <!-- zero-based index for
words, take the second word in the third line of output from cmd4
for Value -->
```

```
            </Item>
        </CustomScript>
    </CustomScriptSet>
```

A single script using `ForeachLine`s to process multiple line results of a `Command`:

```
<?xml version="1.0"?>
<CustomScriptSet Name="someName" Version="0.1.1">
      <CustomScript Name="aLoopExample" TargetPlatform="Linux">
            <Command Name="cmd1" OnError="fail">ip addr | egrep
"^[0-9]+\:" | cut -d: -f2</Command>
            <ForeachLine Values="$(cmd1)" Name="loop1">
                  <Command Name="cmd2" OnError="warn">ip addr show
$(loop1)</Command>
                  <Item Name="$(loop1)"
Class="MGS_NetworkAdapter_Custom2" ItemName="$(loop1)"> <!--
$(loop1) is just the current line passed in from Foreach for this
line -->
                        <Property Name="IPAddress"
Value="$(cmd2):2[1]"/>
                        <Property Name="MACAddress"
Value="$(cmd2):1[1]"/>
                  </Item>
            </ForeachLine>
      </CustomScript>
</CustomScriptSet>
```

# Using Remote Execution Plugins

RayVentory Scan Engine supports advanced Remote-Execution scanning of Windows devices with the help of custom plugins. The following checklist summarizes the actions required to set up custom plugins:

1. Ensure that a valid UNC path is configured in the Settings > Inventory > Remote Execution screen. Press **Install scan utiltities** to install the tools for the first time.
2. Put the custom plugins into the `/plugins` subfolder, which should be located in the root UNC path configured in the first step.
3. Make sure your devices are scanned with Remote-Execution scanning methods (by default, Zero-Touch methods take precedence). This can be achieved in one of the following ways:
   a. Disable Zero-Touch support for particular devices that you want to reach with plugin-based Remote-Execution scan.
   b. Disable Zero-Touch globally (see Inventory Methods chapter for more information on that).
4. Scan the affected Windows devices or rescan already existing devices.
5. Additional information and logs documenting the execution of a plugin can be found on the target devices in the `Tracker.log` file. The level of details can be changed in the RayVentory Scan Engine settings.

# Kubernetes Scan Configuration

Depending on the configuration the following prerequisites are necessary for the scan of a Kubernetes cluster:

- A machine that is either part of a Kubernetes Cluster or a single node cluster (otherwise the machine does not contain any Kubernetes inventory data).
- A user that has root privileges.
- A password for the user or alternatively an SSH-Key.
- The SSH port (only if the used port differs from the default port).

In order to access the infrastructure a `.kubeconfig` file on the scanned system is needed.

The file location can either be defined using the environment variable `KUBECONFIG` or by using a specific file name and a specific location.

**Example:**
When using OpenShift the `.kubeconfig` file can generally be found at `/etc/kubernetes/static-pod-resources/kube-apiserver-certs/secrests/node-kubeconfigs`. Usually, this file will be named `localhost.kubeconfig`.

If this file cannot be found, the folder will be searched for further files ending with `.kubeconfig` and choosing one of those. If no such file can be found in the folder or if the folder does not exist (standard Kubernetes), then each user folder located underneath /home will be searched for a folder named "kube" containing a file named "config".

Usually it is not possible to perform a Kubernetes scan without such a configuration file.

# Command-Line Tools

Certain components of RayVentory Scan Engine can be operated using the command-line. These components are:

- **RIW**: Remote Inventory for Windows
- **RIU**: Remote Inventory for Unix
- **NDTRACK**: OS Inventory Tracker
- **ORATRACK**: Oracle Tracker
- **VMTRACK**: RayVentory VM Console
- **SNMPTRACKER**: SNMP Tracker
- **RVSEMaintenance**: RayVentory Scan Engine Maintenance Tool

A RayVentory consultant may use the binaries of the RayVentory Scan Engine components to create scripts and scheduled tasks for a RayVentory implementation. With the exception of `NDTRACK`, all these command-line tools have got a built-in help that checks command-line arguments and shows a list of command-line arguments with their descriptions. If illegal arguments or the argument help, -help, --help, or /? are being entered, then the programs show their respective usage example and argument list.

The command-line tools do not make use of the settings and credentials from RayVentory Scan Engine.

## Remote Inventory for Windows (RIW)

Use the program  to run a remote OS inventory on Windows targets. See the online help of the program for arguments that allow for customization. The program may use credentials passed to it via the command-line or the current session login for authentication against the target hosts.

Remote Inventory for Windows (`RemoteWmiInventory.exe`) is a command-line oriented tool for generating inventories from remote machines running Windows with WMI or at least SMB and DCOM capability.

RIW uses Windows login, authentication, and impersonation to connect to the target machines in order to perform WMI queries, WMI method invocation, and registry scans. Further it uses SMB for file scans.

The inventory is customizable via a file (called control file). It allows you to perform WMI queries, WMI method invocation, registry, and file system scans.

The command-line arguments are passed to RIW as name-value pairs. Flags are set by their name.

## Usage:

```
RemoteWmiInventory.exe [<argument-name>=<value>|<flag-name>]...
```

## Table of Arguments and Flags

| Name | Type | Description |
|---|---|---|
| batch | TEXT | A list of hosts with optional credentials.<br>Format: <host> [<user> <password>]\|<br><whitespace>\|#<line-comment><br>[[<CR>]<LF>...] |
| classesFile | TEXT | WMI, registry and file queries as XML file. |
| conFailureLog | TEXT | Enables output of the failure log to the file specified. |
| conSuccessLog | TEXT | Enables output of the success log to the file specified. |
| DeviceID | TEXT | RMS / RV network device ID for inventory binding. |
| forceW32registryAccess | FLAG | Force access to the registry by W32 API instead of WMI. |
| help | FLAG | Shows a usage hint and lists the command-line options. Ignores all other arguments except pause and quits. |
| host | TEXT | Host to scan. |
| job | TEXT | Job tag / ID (this is supposed to be used by Raynet's inventory / discovery agents, only). |
| outputpath | TEXT | Path for output Excludes: upload |
| pass | TEXT | Password |
| pause | FLAG | Pauses before exit and waits for a single key press. |
| scanTimeout | INTEGER | Defaults to 1200: Timeout in seconds for scanning a single target. Set this this to 0 in order to disable the timeout. |
| talkingFileNames | FLAG | Use certain bits from the inventory for the inventory file name instead of a generic number. |
| testDelay | INTEGER | Delay between connection tests (3 connection attempts) in milliseconds. |
| testTimeout | INTEGER | Connection test timeout in milliseconds, set to 0 to disable the connection test. |
| upload | TEXT | Address for upload Excludes: outputpath. |
| user | TEXT | Username |
| verbose | FLAG | Log the remote commands. |

| Name | Type | Description |
| --- | --- | --- |
| workers | INTEGER | Defaults to 10: Maximum worker threads running in parallel. |

### Privileges

The user that operates RIW needs privileges to run WMI queries. Some queries and require elevated privileges as a member of the local or domain administrator group has got. Also, file scans require such permissions to access the administrative shares which allow access to the file systems of the target machines.

**Tip:**
In case of Windows XP, the remote Win32 API calls are the only option to get inventory relevant information from the target machines registry.

### Ports in Use

- `135` for running WMI queries or using Win32 API calls as an alternative method for gathering inventory relevant data from the registry.
- `445` for running software inventory relevant file scans
- `80`(`443`) for HTTP(S) upload to the data sink or `445` for SMB upload to the data sink

## Remote Inventory for Unix / Linux (RIU)

Use the program to run a remote OS inventory on Linux and unix-like target platforms. See the online help of the program for arguments that allow customization. The program may use credentials passed to it via the command-line for authentication against the target hosts. This program comes with a built-in credential store, a target database, and job management.

Remote Inventory for Unix / Linux (`RemoteInventory4Unix.exe`) is a command-line oriented tool for generating inventories from remote machines running different unixoid plattforms including different flavors of Linux.

The RIU tool can be operated in two modes:

- **Batch mode**
  Allows you to manually create job files, target hosts, or automatically create job files from the content of target files while specifying simple credential store files with optional encryption. During startup RIU looks for job files, checks their status, and continues them until all job items (target machines) reached one of the end-states (Success, AuthenticationFailure).
- **Single mode**
  Allows you to specify a singe host as a target machine

RIU generates inventories by connection to a target machine via SSH and performing different built-in shell commands and system utilities to determine the platform and architecture characteristics. According to those characteristics, it gathers hard- and software inventory relevant data by running further commands and by parsing and evaluating their results to store them in an inventory file for each target machine. Then the resulting NDI file is uploaded to an inventory data sink like RayVentory Scan Engine, RayVentory Server, and RayManageSoft or it is stored in the local file system.

The command-line arguments are passed to RIU as name-value pairs. Flags are set by their

name.

## Usage

```
RemoteInventory4Unix.exe [<argument-name>=<value>|<flag-name>]...
```

## Table of Arguments and Flags

| Name | Type | Description |
|---|---|---|
| addhosts | TEXT | Add hosts from a .csv file to the targets. Each line of the file must be in the format: hostname[,port][,hostkey]<br>Mind that port and hostkey are optional. |
| authKey | TEXT | Fallback authentication key in the OpenSSH format. This is used if no authentication key is given for a target. |
| authKey.pass | TEXT | Passphrase for the fallback authentication key. |
| authKeyFile | TEXT | Fallback authentication key file in the OpenSSH format. |
| conFailureAuthLog | TEXT | Enables output of the authentication failure log to the file specified. |
| conFailureLog | TEXT | Enables output of the connection failure log to the file specified. |
| config | TEXT | Path to the configuration file. The optional configuration file is an XML file root element Config and sets the values ConsoleRefreshInterval, UploadAddress, and Workers which equal the command-line arguments refresh, upload, and workers. |
| conSuccessLog | TEXT | Enables output of the connection success log to the file specified. |
| credentials.file | TEXT | Defaults to credentials.xml: Credentials store file. |
| credentials.key | TEXT | Encryption key for the passwords in the credentials store file. |
| decryptLog | TEXT | Decrypting the partially encrpyted log file, given, and exit. |
| decryptLog.key | TEXT | The key for decrypting the file given by the argument decryptLog. |
| DeviceID | TEXT | RMS / RV network device ID for inventory binding. |
| forceFileName | TEXT | Forces the use of supplied file name for the inventory file. |

| Name | Type | Description |
|---|---|---|
| help | FLAG | Shows a usage hint and lists the command-line options. Ignores all other arguments except pause and quits. |
| host | TEXT | Host to scan. |
| host.add | FLAG | Adds to the store and exit.<br>Requires: host |
| host.key | TEXT | Expected host key fingerprint for host authentication.<br>Requires: host |
| job | TEXT | Job tag / ID (this is supposed to be used by Raynet's inventory / discovery agents, only). |
| jobs | TEXT | Path to directory for job files. |
| mssqlts.connection | TEXT | Connection string to a MSSQL DB as source for targets<br>Requires: mssqlts.query. |
| mssqlts.query | TEXT | Query string for a MSSQL DB as source for targets.<br>Requires: mssqlts.connection |
| outputpath | TEXT | Path for output.<br>Excludes: upload |
| pass | TEXT | Fallback password. |
| pass.key | TEXT | Encryption key for fallback password. |
| pause | FLAG | Pauses before exit and waits for a single key press. |
| port | INTEGER | defaults to 22: SSH port on the target host. |
| recordHostKeys | FLAG | Add the host-keys that were reported by the hosts for hosts which do not have got a host-key, yet. |
| refresh | INTEGER | Defaults to 5000: Interval for refresh of console and saving of job state in milliseconds. |
| rerunAllJobs | FLAG | Retry all jobs no matter what their status is.<br>Excludes: retryAllJobs, single, retryAuthFailure. |
| retryAllJobs | FLAG | Retry all jobs no matter what their status is.<br>Excludes: single |
| retryAuthFailure | FLAG | Retry targets in jobs which failed due to authentication errors.<br>Excludes: single. |
| scanTimeout | INTEGER | Timeout in seconds for scanning a single target. Set this this to 0 in order to disable the timeout. |

| Name | Type | Description |
|---|---|---|
| single | FLAG | Run for a single host given on command-line, ignoring all job files. |
| successfulHosts | TEXT | During job creation, set the targets in the specified list to status Done.<br>Excludes: single |
| talkingFileNames | FLAG | Use certain bits from the inventory for the inventory file name instead of a generic number. |
| targetsfile | TEXT | Path to a file with targets. |
| upload | TEXT | Address for upload.<br>Excludes: outputpath |
| user | TEXT | Fallback username. |
| user.add | FLAG | Add the fallback user with password or authentication key and authentication key passphrase to the credentials store and exit.<br>Requires: user |
| user.elevpass | TEXT | The password that is passed to sudo.<br>Requires: user |
| user.sudo | FLAG | Indicates that the sudo command is enabled for the fallback user.<br>Requires: user |
| user.super | FLAG | Indicates that the fallback user is a super user Requires: user |
| user.target | TEXT | Regular expression to match the targets (by hostname) that the fallback credentials apply to.<br>Requires: user |
| useTargetCache | FLAG | Loads targets not from the given sources but from a cache when present or creates the cache. |
| verbose | FLAG | Log the remote commands. |
| workers | INTEGER | Maximum number of workers to spawn. |

## Privileges

What privileges are needed depends on what data is needed.
You will need a user with root privileges or that is allowed to sudo, prbrun, or priv to run programs with elevated privileges with or without a keyboard interactive password query.

For example: Without such privileges you will not be able to read inventory and dependency mapping relevant data as for example the BIOS serial number of a VM (to determine which

virtual guest system reflects the target machine) or a system enclosure serial number.

## Ports in Use

- `22` for connecting to the target machine.
- `80(443)` for HTTP(S) upload to the data sink or `445` for SMB upload to the data sink.

# VMware Inventory

RV-VM-Console (`RayVentory.VM.Console.exe`) is a command-line oriented tool for generating inventories from vSphere / ESX hosts using VMware's restful management API. You may specify a single target on the command-line. If no target has been specified on the command-line then it will target all vSphere / ESX hosts that RVP knows.

If the the target is a vSphere cluster then VM-Console will create an NDI file for each ESX host in the cluster.

The command-line arguments are passed to RIW as name-value pairs. Flags are set by their name.

## Usage

```
RayVentory.VM.Console.exe [-o <argument-name>=<value>]...
```

## Table of Arguments and Flags

| Name | Type | Description |
|---|---|---|
| `auth` | `BOOLEAN` | Its entered connection data will be saved. |
| `basepath` | `TEXT` | Path to store configuration files and results unless `vsphereoutput` was specified. |
| `DeviceID` | `TEXT` | RMS / RV network device ID for inventory binding. |
| `encryptionKey` | `TEXT` | Key for encryption and decryption of credentials. |
| `help` | `FLAG` | RMS / RV network device ID for inventory binding. |
| `ignoreSSL` | `BOOLEAN` | Ignore SSL / TLS connection warnings. |
| `job` | `TEXT` | Job tag / ID (this is supposed to be used by Raynet's inventory / discovery agents, only). |
| `password` | `TEXT` | Password of the user for authentication. |
| `upload` | `TEXT` | URL for uploading the result. |
| `url` | `TEXT` | The targets seb service URL. Usualy it has the form of `https://myTargetHost/sdk`. |
| `user` | `TEXT` | Name of the user for authentication. |
| `vspherereconnections` | `TEXT` | Filename for the vSphere connections file. |

| Name | Type | Description |
| --- | --- | --- |
| vsphereoutput | TEXT | Path for the output of the vSphere inventory. |

### Privileges

The RV VM Console needs a user with read privileges on the whole object tree of the vSphere infrastructure or ESX host. Depending on the permissions set, the user may need special permissions to extract sensitive details like the serial number / license key.

### Ports in Use

- `80(443)` for communication with VMware's management API via HTTP(S) and for HTTP(S) upload to the data sink.

## ORATRACK

ORATRACK (`oratrack.jar`) is a command-line oriented tool for generating inventories for Oracle database instances. You may target a single database by command-line arguments, target many databases by a connections file, or let it discover and inventory local and remote databases on its own.

The command-line arguments are passed to ORATRACK as name-value pairs. Flags are set by their name.

### Usage

```
java –jar oratrack.jar [-o <argument-name>=<value>]...
```

### Table of Arguments and Flags

| Name | Type | Description |
|---|---|---|
| `asSysDba` | `BOOLEAN` | Connect as SYSDBA.<br>`true` The connection is established with SYS role.<br>`false` The connection is established without giving the SYS role. |
| `auth` | `BOOLEAN` | Specifies whether the set of targets collected is written to `OracleConnections.xml`. This argument defaults to `true`.<br><br>`true` Write to `OracleConnections.xml`.<br>`false` Do not write. |
| `authPath` | `TEXT` | Specify the targets filename. By default a the current working directory is searched for a file called `OracleConnections.xml`. This file will be created when it is not present and argument `auth` is set to `true`. |
| `auto` | `OPTION` | Turns on discovery of local configuration files in order to discover targets known to the local machine. |

| Name | Type | Description |
|---|---|---|
| | | `local` Ignore remote targets.<br>`all` Use all targets found. |
| `checkCertificates` | `BOOLEAN` | Enable / disable checking the host certificate when uploading using HTTPS. Enabled by default. |
| `csv` | `BOOLEAN` | Set output mode.<br>`true` Write results as CSV files.<br>`false` Write results as NDI file. |
| `dbhost` | `TEXT` | Specifies a target address or a pattern to match target addresses, not specified by command-line.<br><br>`<hostname>` A concrete hostname.<br>`localhost` Special hostname, indicates using the local machine.<br>`127.0.0.1` Special IP address, indicates using the local machine.<br>`<ip-address>` A concrete IP address.<br>`<pattern>` Use targets with host addresses that match this expression, only. |
| `debug` | `BOOLEAN` | Controls the log level.<br>`true` Log debug messages.<br>`false` Suppress debug messages. |
| `deviceId` | `INTEGER` | Sets the device ID. This is supposed to be used by Raynet's inventory / discovery agents, only. |
| `discoverySource` | `OPTION` | This argument is used in conjunction with argument `auto` and controls where to look for the home directory of a database.<br>The following values for the argument `discoverySource` may be may be combined in a comma seperated list. For example:<br>`discoverySource=oratab,files`<br><br>`environment` Scan the environment variables.<br>`registry` Scan the windows registry or oratab file.<br>`oratab` Look for and process the oratab file.<br>`files` Scan scan the filesystem for configuration files. |
| `domain` | `TEXT` | Gives a value to put in the inventory file as domain. |
| `encryption` | `OPTION` | Specifies the encryption algorithm(s) to use as comma |

| Name | Type | Description |
|---|---|---|
| | | separated list of algorithms enclosed in brackets when specifying a target by command-line. Mind that different versions of ORATRACK support different sets of encryption algorithms. For example: RC4_40, AES128, AES192, 3DES112 etc. |
| encryption.level | OPTION | Specifies the encryption level, controlling the negotiation between client and server together with the parameter encryption. Valid values are REQUIRED, REQUESTED, REJECTED, or ACCEPTED. |
| encryptionKey | TEXT | Sets the encryption key that is used for encrypting and decrypting all credentials. |
| dfusScript | TEXT | Sets the optional Oracle DFUS script filename for running it by sqlplus.<br><br>If you set this argument and the file is not present, then this will cause the query to fail with an error.<br><br>If you do not set this argument then oratrack looks for a file called oracle.sql and runs it. |
| failedconnectionbehavior | OPTION | Sets the condition for reporting a failed connection by exit code.<br><br>ignore Ignore failed connections.<br>all Returns an error if all connections failed.<br>any Returns an error if any connection fails. |
| help | FLAG | Show the usage hint and command-line argument list. |
| ignorenames | BOOLEAN | Ignore all service names or SIDs matching the semi-colon-separated regular expressions listed, here, during discovery. Default is CLRExtProc;EXTPROC;PLSExtProc. |
| integrity | OPTION | Specifies the integrity algorithm(s) to use as comma separated list of algorithms enclosed in brackets when specifying a target by command-line. Currently supported: SHA1 or MD5. |
| integrity.level | OPTION | Specifies the integrity level controlling the negotiation between client and server together with the parameter integrity. Valid values are REQUIRED, REQUESTED, REJECTED, or ACCEPTED. |
| job | TEXT | Sets the job tag / ID. This is supposed to be used by Raynet's inventory / discovery agents only. |

| Name | Type | Description |
|---|---|---|
| listenerControl | BOOLEAN | Enable or disable querying the listener control status during discovery. Enabled by default. |
| mode | OPTION | Sets the program mode. This argument defaults to query.<br><br>encryptquery Read query.xml from the current working directory and write an encrypted copy named query.xml.enc to the current working directory.<br>test Tries to connect to a target specified by command-line and returns the result as exit code.<br>query Run the queries on the targets. |
| logfile | TEXT | Sets the filename for the log file. |
| orahome | TEXT | This argument is used in conjunction with the argument auto. When given oratrack will not search for the Oracle instance home directories but scan for configuration files in this directory only.<br>This argument also overrides the argument discoverySource. |
| oratab | TEXT | Specifies an oratab filename and overrides the discovery of that file. This argument conflicts with the argument tnsnames. |
| pass | TEXT | Specifies the password for connecting to a database. |
| path | TEXT | Sets the output directory. |
| plainQueries | TEXT | Format: <filename>[,<query name>][;...]<br>Specify additional text files with queries and an optional name for the result, where the last result is written to the output file.<br><br><filename> Filename for the text file.<br><query name> Optional name for the result (default is PlainQuery) |
| port | INTEGER | Specifies a target port number or a pattern to match target port numbers, which were not specified by command-line. |
| queryPath | TEXT | Specify the queries filename. By default the current working directory is searched for a file called query.xml.enc with fallback to query.xml |

| Name | Type | Description |
|---|---|---|
| reportQueryError | BOOLEAN | Enabled to create an error report instead of a regular inventory on a failed query that is considered an error. Disabled by default.<br><br>true Enable error report.<br>false Disable error report. |
| reportStatus | BOOLEAN | Controls the connection status reporting.<br><br>true On a failure / error creates an NDI file that reports this failure / error.<br>false Does not output an NDI for a failed connection. |
| reportStandbyDB | BOOLEAN | Controls the standby database reporting enabled by default.<br><br>true Looking for indicators for a standby database for the target database.<br>false Do not look for indicators for standby databases. |
| silent | BOOLEAN | Controls the output to the standard output channel.<br><br>true Suppress logging to the standard output channel.<br>false Allow logging to the standard output channel. |
| sname | TEXT | Format: <service name>\|<SID>\|<pattern><br><br>Specifies a target service name or SID or a pattern to match target services, not specified by command-line.<br><br><service name> A concrete service name.<br><SID> A concrete SID.<br><pattern> Use targets with service names that match this expression only. |
| sqlnet | TEXT | Specifies a sqlnet configuration filename and overrides discovery of that file. |
| tnsnames | TEXT | Specifies a tnsnames filename and overrides discovery of that file. This argument conflicts with argument oratab. |
| tnslistener | TEXT | Specifies a tnslistener configuration filename and overrides the discovery of that file. Discovery or explicit specification of this file is implied when argument auto is set to local. |
| upload | TEXT | Specifies the upload location's URL for the output files. |

| Name | Type | Description |
|---|---|---|
| | | This contradicts argument `csv` with value `true` and argument path. |
| `user` | `TEXT` | Specifies the username for connecting to a database.<br><br>`<username>` The username<br>`?` Indicates that the username is read from standard input during execution |

**Tip:**
You cannot use the arguments `path` and `upload` together.The arguments `tnsnames`, `tnslistener`, `sqlnet`, and `oratab` are allowed in conjuction with argument `auto`, only.The argument `csv=true` will only produce output for query files with queries with attribute `LMS="true"`.

## Examples

### 1. Query single target by RVP

This is how the RayVentory Scan Engine UserUI will call oratrack in order to scan a single target.

```
-o dbhost=<host address> -o sname=<service name> -o port=<port
number> -o authPath=<RVP UserUI path to OracleConnections.xml> -o
user=<username> -o pass=<password> -o path=<RVP UserUI path to
Oracle query results folder>
```

### 2. Test single target by RVP

This is how the RayVentory Scan Engine UserUI will call oratrack in order to test connection to a single target.

```
-o dbhost=<host address> -o sname=<service name> -o port=<port
number> -o authPath=<RVP UserUI path to OracleConnections.xml> -o
user=<username> -o pass=<password> -o mode=test
```

### 3. Query all target from a targets file given by RVP

This is how the RayVentory Scan Engine UserUI will call oratrack in order to scan all targets from RVP's targets file.

```
-o authPath=<RVP UserUI path to OracleConnections.xml> -o path=<RVP
UserUI path to Oracle query results folder>
```

### 4. Scan all local targets found in the local configuration files using OS Authentication

Oratrack is installed on every machine that hosts Oracle databases and setup to be frequently run by a scheduler.
The user that runs oratrack is setup for OS Authentication with the targets. Instead of the argument path you can provide the argument upload

```
-o path=<output path> -o auto=local -o auth=false
```

### 5. Scan all targets found in the local configuration files using OS Authentication

This is an alternative to use case 4.
Oratrack is installed on a machine that knows several Oracle DBs and setups to be frequently run by a scheduler.
The user that runs oratrack is setup for OS Authentication with the targets.

```
-o path=<output path> -o auto=all -o auth=false
```

**6. Scan all targets specified in a given tnsnames file with encryption and integrity options specified in a given sqlnet configuration file.**

This is an alternative to use case 4.

Oratrack is installed on a machine that holds a tnsnames file and a sqlnet configuration file.

The user that runs oratrack is setup for OS Authentication with the targets.

```
-o path=<output path> -o auto=all -o auth=false -o tnsnames=<path
to tnsnames file> -o sqlnet=<path to sqlnet configuration file>
```

### Privileges

What privileges are needed depends on how ORATRACK is operated. For running queries against a database instance, the user that is passed to ORATRACK needs permissions to read certain system tables and views. What exactly is needed may change. The Raynet consultants can explain what permissions are needed and give you scripts for creating a user with the needed permissions. You may always use a privileged user like sys.

To make use of ORATRACK's database discovery capabilities, you need to run it as a user with permission to read certain configuration files like `oratab`, `tnsnames.ora`, `listener.ora`, and `sqlnet.ora`. Further, to run `lsnrctl` and read the Oracle databases' home directories.

### Ports in Use

- `1521` (default) for communication with an Oracle database via a TNS listener.

## Java/ORATRACK Compatibility Matrix

RayVentory Scan Engine uses the `oratrack.jar` file included in the installation resources. Depending on the OpenJDK or Oracle Java version used, it might be necessary to manually replace the `oratrack.jar` used by a copy of another `oratrack.jar`. The following table shows which `oratrack.jar` works for the different versions of OpenJDK and Oracle Java.

| Java-Version | Oratrack-Version | Status |
|---|---|---|
| Oracle Java 17 | Oratrack 10 | **supported** |
| Oracle Java 17 | Oratrack 8 | **supported** |
| Oracle Java 17 | Oratrack 7 | **not supported** |
| Oracle Java 17 | Oratrack 5 | **not supported** |
| Oracle Java 17 | Oratrack 4 | **not supported** |
| Oracle Java 11 | Oratrack 10 | **supported** |
| Oracle Java 11 | Oratrack 8 | **supported** |
| Oracle Java 11 | Oratrack 7 | **not supported** |
| Oracle Java 11 | Oratrack 5 | **not supported** |

| Java-Version | Oratrack-Version | Status |
|---|---|---|
| Oracle Java 11 | Oratrack 4 | not supported |
| Oracle Java 8 | Oratrack 10 | not supported |
| Oracle Java 8 | Oratrack 8 | supported |
| Oracle Java 8 | Oratrack 7 | supported |
| Oracle Java 8 | Oratrack 5 | supported |
| Oracle Java 8 | Oratrack 4 | not supported |
| Oracle Java 7 | Oratrack 10 | not supported |
| Oracle Java 7 | Oratrack 8 | not supported |
| Oracle Java 7 | Oratrack 7 | supported |
| Oracle Java 7 | Oratrack 6 | supported |
| Oracle Java 7 | Oratrack 5 | supported |
| Oracle Java 7 | Oratrack 4 | supported |
| Oracle Java 6 | Oratrack 10 | not supported |
| Oracle Java 6 | Oratrack 8 | not supported |
| Oracle Java 6 | Oratrack 7 | not supported |
| Oracle Java 6 | Oratrack 6 | supported |
| Oracle Java 6 | Oratrack 5 | supported |
| Oracle Java 6 | Oratrack 4 | supported |
| Sun Java 5 | Oratrack 10 | not supported |
| Sun Java 5 | Oratrack 8 | not supported |
| Sun Java 5 | Oratrack 7 | not supported |
| Sun Java 5 | Oratrack 6 | not supported |
| Sun Java 5 | Oratrack 5 | supported |
| Sun Java 5 | Oratrack 4 | supported |
| Sun Java 4 | Oratrack 10 | not supported |
| Sun Java 4 | Oratrack 8 | not supported |
| Sun Java 4 | Oratrack 7 | not supported |
| Sun Java 4 | Oratrack 6 | not supported |

| Java-Version | Oratrack-Version | Status |
|---|---|---|
| Sun Java 4 | Oratrack 5 | not supported |
| Sun Java 4 | Oratrack 4 | supported |
| Microsoft JDK 17 (OpenJDK) | Oratrack 10 | supported |
| Microsoft JDK 17 (OpenJDK) | Oratrack 8 | supported |
| Microsoft JDK 17 (OpenJDK) | Oratrack 7 | not supported |
| Microsoft JDK 17 (OpenJDK) | Oratrack 5 | not supported |
| Microsoft JDK 17 (OpenJDK) | Oratrack 4 | not supported |
| Microsoft JDK 16 (OpenJDK) | Oratrack 10 | supported |
| Microsoft JDK 16 (OpenJDK) | Oratrack 8 | supported |
| Microsoft JDK 16 (OpenJDK) | Oratrack 7 | not supported |
| Microsoft JDK 16 (OpenJDK) | Oratrack 5 | not supported |
| Microsoft JDK 16 (OpenJDK) | Oratrack 4 | not supported |
| Microsoft JDK 11 (OpenJDK) | Oratrack 10 | supported |
| Microsoft JDK 11 (OpenJDK) | Oratrack 8 | supported |
| Microsoft JDK 11 (OpenJDK) | Oratrack 7 | not supported |
| Microsoft JDK 11 (OpenJDK) | Oratrack 5 | not supported |
| Microsoft JDK 11 (OpenJDK) | Oratrack 4 | not supported |
| Amazon Coretto 17 (OpenJDK) | Oratrack 10 | supported |
| Amazon Coretto 17 (OpenJDK) | Oratrack 8 | supported |
| Amazon Coretto 17 (OpenJDK) | Oratrack 7 | not supported |
| Amazon Coretto 17 (OpenJDK) | Oratrack 5 | not supported |
| Amazon Coretto 17 (OpenJDK) | Oratrack 4 | not supported |
| Amazon Coretto 11 (OpenJDK) | Oratrack 10 | supported |
| Amazon Coretto 11 (OpenJDK) | Oratrack 8 | supported |
| Amazon Coretto 11 (OpenJDK) | Oratrack 7 | not supported |
| Amazon Coretto 11 (OpenJDK) | Oratrack 5 | not supported |
| Amazon Coretto 11 (OpenJDK) | Oratrack 4 | not supported |
| Amazon Coretto 8 (OpenJDK) | Oratrack 10 | not supported |

| Java-Version | Oratrack-Version | Status |
|---|---|---|
| Amazon Coretto 8 (OpenJDK) | Oratrack 8 | **supported** |
| Amazon Coretto 8 (OpenJDK) | Oratrack 7 | **supported** |
| Amazon Coretto 8 (OpenJDK) | Oratrack 5 | **supported** |
| Amazon Coretto 8 (OpenJDK) | Oratrack 4 | **not supported** |

# NDTRACK

Use the program to run a local OS inventory on Windows or unixoid platforms. The correct scope for the NDTRACK inventory needs to be specified.

NDTRACK (`ndtrack.exe`) is a command-line oriented tool for generating inventories from a Windows or unixoid platform including different flavors of Linux. It is locally executed on the target host and produces an NDI file that may be uploaded to an inventory data sink such as RVSE, RV Server, or RMS AS.

The command-line arguments are passed to NDTRACK as name-value pairs. Flags are set by their name.

## Usage:

```
ndtrack.exe [-o <argument-name>=<value>]...
```

or

```
./ndtrack.sh [-o <argument-name>=<value>]...
```

## Table of Arguments and Flags

| Name | Description | Values/Range |
|---|---|---|
| `BenchmarkCPU` | Enables benchmarking the CPU. This measures the current CPU speed in MHz. | Boolean<br>**Default value**: `True` |
| `CheckCertificateRevocation` | Determines whether it checks the certificate revocation list when accepting web server signatures from an HTTPS server. | Boolean<br>**Default value**: `True` |
| `Compress` | Determines whether inventory files are compressed for upload. | Boolean<br>**Default value**: `True` |
| `CompressedExtension` | File extension for the compressed inventory file. | Valid file extensions.<br>**Default value**: `gz` |

| Name | Description | Values/Range |
|---|---|---|
| ComputerDomain | Contains the name of the domain of the local device from which the inventory is being gathered.<br>One reason for setting this value is to cause UNIX like devices to report in a particular domain in listings and reports. | Valid domain name.<br>**Default value**: On Windows the current local domain. For UNIX-like devices, no value.<br>**Example value**: MyDomain.com |
| Configuration | Configuration name to be added to the name of the NDI file. | String<br>**Default value**: {empty}<br>**Example value**: TestConfiguration |
| DateTime | Overwrites date and time of the inventory file. | DateTime string in DateTimeFormat format.<br>**Default value**: {currentSystemTime}<br>**Example value**: 20200525T103055 |
| DateTimeFormat | DateTime format string for the inventory file. | DateTimeFormat string based on the ANSI C function strftime().<br>**Default value**: %Y%m%dT%H%M%S |
| DETECTCmdLine | Command-line to run the detect utility. | Any valid command-line to execute detect.com that results in the output being written to stand stand output.<br>**Default value**: conspawn detect.com |
| DetectMultiCore | Detect if the targets CPU has a multi-core architecture. | Boolean<br>**Default value**: True |
| DetectSerial | Enables or disables detection of BIOS serial number. | Boolean<br>**Default value**: True |
| DeviceID | RMS / RV network device ID for inventory binding. | Valid network device ID.<br>**Default value**: {empty}<br>**Example value**: 15 |
| Difference | Determines whether differential inventories are performed. | Boolean<br>**Default value**: False |

| Name | Description | Values/Range |
|---|---|---|
| EmbedFileContentDirectory | File content from files in these directories is considered in the filescan. | Any valid folder path. Multiple paths may be specified with a semicolon separator (;) between them. **Default value**: "%ProgramData%,%ProgramFiles%;%ProgramFiles(x86)%" |
| EmbedFileContentExtension | File content from files with these extensions is considered in the filescan. | Any valid file extension. Multiple file extensions can be separated with a semicolon separator (;) between them. **Default value**: swidtag |
| EmbedFileContentMaxSize | File content from files with this maximum file size is considered in the filescan. | Any valid integer which is interpreted as bytes. **Default value**: 1000000 |
| EnableWinOldAppPolicySetting | Allows ndtrack to temporarily disable group policy settings to gain access to BIOS information. | Boolean **Default value**: false |
| ExcludePermissionsMask | Files with these permissions are not considered in the filescan. | An octal value in the format used for chmod. **Default value**: {empty} **Example value**: 0777 |
| ExcludeDirectory | Files from these directories are not considered in the filescan. | Any valid folder path. Multiple paths should be separated by the semicolon (;) character. **Default value**: {empty} **Example value**: $(WinDirectory) |
| ExcludeEmbedFileContentDirectory | File content from these directories is not considered in the filescan. | Any valid folder path. Multiple paths should be separated by the semicolon (;) character. **Default value**: {empty} **Example value**: "C:\Program Files (x86)\Adobe" |
| ExcludeExtension | Files with these extensions are not | Valid file extensions. |

| Name | Description | Values/Range |
|---|---|---|
| | considered in the `filescan`. | Multiple extensions should be separated by the semicolon (;) character.<br>**Default value**: {empty}<br>**Example value**: `dll` |
| `ExcludeFile` | Files with these file names are not considered in the `filescan`. | Valid file names.<br>**Default value**: {empty}<br>**Example value**: `myfile.txt` |
| `ExcludeFileSystemType` | For UNIX-like devices, `ExcludeFileSystemType` allows for the exclusion of specific types of file systems that may otherwise be included. This is not applicable on Microsoft Windows. | Comma separated list of standard file system type names as recognized by the UNIX mount command. Either white spaces need to be omitted or the list needs to be enclosed in double quotation marks.<br>**Default value**: {empty}<br>**Example value**: `zfs` |
| `ExcludeMD5` | Files matching these MD5 checksums are not considered in the `filescan`. | Valid MD5 values. Multiple checksums should be separated by the semicolon (`;`) character.<br>**Default value**: {empty}<br>**Example value**: `7d9d2440656fdb3645f6734465678c6` |
| `ExpectHardware` | Used to set the `Hardware` property. | Boolean<br>**Default value**: `True` |
| `ExpectSoftware` | Used to set the `Software` property. | Boolean<br>**Default value**: `True` |
| `ExpectTrackFilesInUserInventory` | Used to set the `TrackFilesInUserInventory` property. | Boolean<br>**Default value**: `True` |
| `Full` | Sets the type of the inventory. | Valid inventory type (`Full` or `Difference`).<br>**Default value**: `Full` |
| `FullQualifiedDomainName` | Overrides the FQDN that is taken for the inventory `Scope` value. | Valid FQDN.<br>**Default value**: `$(MachineName).$(Domain)` |

| Name | Description | Values/Range |
|---|---|---|
| | | **Example value**: `Hostname1.domain.com` |
| `GenerateMD5` | Generate an MD5 checksum for every file being tracked. | Boolean<br>**Default value**: `False` |
| `Generation` | Sets generation counter to provided value. | Integer between 1 and `GenerationMax`.<br>**Default value**: {`empty`} |
| `GenerationMax` | The number of differential inventories performed between full inventories. | Integer between 1 and 1,000,000,000<br>**Default value**: `9` |
| `Hardware` | Determines whether to track hardware (in the machine context). | Boolean<br>**Default value**: `True` |
| `HyperV` | Not used yet. | Boolean<br>**Default value**: `True` |
| `IncludeDirectory` | Files from these directories are considered in the `filescan`. | Any valid folder path. Multiple paths should be separated by the semicolon (`;`) character.<br>**Default value**: {`empty`}<br>**Example value**: `C:\Temp` |
| `IncludeExecutables` | Determines which files the `filescan` considers executables.<br>If set to `false`, only `.exe` files are considered.<br>If set to `true`, on unixoid platforms files with no filename extension and an executable bit set (in any local, group, or universal scope in the file permission bits) are considered as well. | Boolean<br>**Default value**: `True` |
| `IncludeExtension` | Files with these extensions are considered in the `filescan`. | Valid file extensions. Multiple extensions should be separated by the semicolon (;) character.<br>**Default value**: `exe; dll` |
| `IncludeFile` | Files with these file names are considered in the `filescan`. | Valid file names. Multiple file names should be separated by the semicolon (`;`) character.<br>**Default value**: {`empty`} |

| Name | Description | Values/Range |
|---|---|---|
| | | **Example value**: `xcopy.exe` |
| `IncludeFileSystemType` | For UNIX like devices, IncludeFileSystemType allows for the inclusion of specific types of file systems that may otherwise be excluded. This is not applicable on Microsoft Windows. | Comma-separated list of standard file system type names, as recognized by the UNIX mount command. Either omit white space or enclose the list in double quotation marks. **Default value**: There is no default value in the Registry. If there is no value given, the default behavior is `.ufs, .zfs, .lofs` **Example value**: `ufs, lofs` |
| `IncludeMachineInventory` | Specifies whether or not to conduct a computer inventory of hardware and all user packages. | Boolean **Default value**: `True` (if running as `LocalSystem` or running a machine inventory `-t Machine` on the command line) |
| `IncludeMD5` | Files matching these MD5 checksums are considered in the `filescan`. | Multiple checksums should be separated by the semicolon (;) character. **Default value**: {`empty`} **Example value**: `7d9d2440656fdb3645f6734465678c60` |
| `IncludeNetworkDrives` | Specifies if network drives should be scanned by the `filescan`. | Boolean **Default value**: `False` |
| `IncludePermissionsMask` | Files with these permissions are considered in the `filescan`. | An octal value in a format used for `chmod` **Default value**: `0111` |
| `IncludeRegistryKey` | Registry keys or values to include in the inventory. | Valid registry key or value. **Default value**: If no value is specified, the follwoing value will be used `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft` |

| Name | Description | Values/Range |
|---|---|---|
| | | `\Windows \CurrentVersion \Uninstall\` |
| `IncludeUserInventory` | Specifies whether or not to conduct a user inventory. | Boolean<br>**Default value**: `True` (if running as user or running a user inventory `-t User` on the command line) |
| `IncrementalDiff` | When differential inventories are performed this determines what differences will be collected. | Boolean<br>**Default value**: `False` |
| `Inventory` | Determines the file type of the `UploadRule` that is used. | One of the following:<br>• `Inventory`<br>• `Log`<br>• `PolicyComplianceLog`<br>• `SMSStatusMessage`<br>**Default value**: `Inventory` |
| `InventoryDirectory` | Directory for saving the inventory file. | Valid location<br>**Default value**: {empty}<br>**Example value**: `C:\Inventory` |
| `InventoryExtension` | File extension for saving the inventory file. | Valid file extension<br>**Default value**: `ndi` |
| `InventoryFile` | File name of a local copy of the inventory file. | `*.ndi`<br>**Default value**: `$(UserName) on $(MachineId).ndi`<br>**Example value**: `myComputer.ndi` |
| `InventoryScriptsDir` | (Applies to Windows targets, only): If `RunInventoryScripts` is `True`, this specifies the location of scripts to be run after the inventory scanning completes. | Valid folder path<br>**Default value**: `$(CommonAppDataFolder)\ManageSoft Corp\ManageSoft`<br>**Example value**: `C:\IT-Admin` |
| `InventoryType` | Determines whether a machine-based or a user-based inventory scan | Either `User` or `Machine` |

| Name | Description | Values/Range |
|---|---|---|
| | should be performed. | **Default value**: `User` (unless the scan is permformed by `LocalSystem` in which case the default switches to `Machine`) |
| `job` | `JobTag` that is added to the beginning of the inventory file name. | String<br>**Default value**: `{empty}`<br>**Example value**: `testPrefix` |
| `LibsmbiosCmdLine` | Command or executable that is used to get certain system information (Dell system ID, service tag, product name, BIOS version, and system vendor). | Command<br>**Default value**: `getSystemId.exe` |
| `LogMsgCatPath` | Defines the log message catalog file to be used by the inventory scanner for logging. | File name<br>**Default value**: `ndtmsg.cat` |
| `LowProfile` | When set to `True`, the inventory process runs with low CPU priority. For UNIX-like systems, this sets the nice level of the process to 10. On recent Windows platforms, it uses background processing mode (PROCESS_MODE_BACKGROUND_BEGIN). On legacy Windows platforms where this is not supported (such as Windows XP and earlier), it uses a priority of idle (IDLE_PRIORITY_CLASS).<br><br>When set to `False`, the inventory process runs with normal CPU priority. | Boolean<br>**Default value**: `False` |
| `MachineId` | The machine name that is used in the name of the inventory file. | String<br>**Default value**: `$(MachineName)`<br>**Example value**: `MyMachine` |
| `MachineInventory` | Static value that cannot be changed. | String<br>**Default value**: `Machine` |

| Name | Description | Values/Range |
|---|---|---|
| MachineInventoryDirectory | Location for computer inventories. | Valid location<br>**Default value**:<br>$(CommonAppDataFolder)\ManageSoft Corp\ManageSoft\Tracker\Inventories |
| MachineName | Contains the name of the local machine. Unlike MachineId, this preference should not be changed. | String<br>**Default value**:<br>$(MachineName) |
| MachineZeroTouchDirectory | In case of a remote call the location used for the machine inventories. | Valid location<br>**Default value**:<br>$(CommonAppDataFolder)\ManageSoft Corp\ManageSoft\Tracker\ZeroTouch |
| ManageSoftPackages | Determines whether or not software inventory is collected for packages installed by RayManageSoft. | Boolean<br>**Default value**: True |
| MinInventoryInterval | Specifies the minimum interval (in hours) between the collections of inventories. | Any non-negative integer<br>**Default value**: 0 |
| MSI | Determines whether Microsoft Installer (MSI) packages are included in the inventory. | Boolean<br>**Default value**: True |
| MSIOpenProducts | Specify the MSI product codes which are to be inspected fully, which involves calculating the result of all applied transforms and patches before the retrieval of the UpgradeCode and PIDKEY properties.<br>Use asterisk (*) to include all products. | Valid product code values. Multiple values should be separated by the semicolon (;) character.<br>**Default value**: {empty}<br>**Example value**:<br>{2E11EF4E-901F-4B2D-B68E-3DB2A566C857} |
| MSSQL | Enables tracking of Microsoft SQL Server information as part of the WMI scan. | Boolean<br>**Default value**: True |
| NetworkSense | Determines whether network speed checks are performed prior to attempting inventory file upload. | Boolean<br>**Default value**: False |
| PackageDatabaseTypes | Package types that are tracked in the inventory. Possible values are IA, | Valid package types separated by comma (,) |

| Name | Description | Values/Range |
|---|---|---|
| | BEA, RPM, DPKG, OSX, SUNPKG, IPS, LPP, SDUX. | character.<br>**Default value**: Dependent on the operating system<br>**Example value**: `RPM, OSX` |
| `PkgCacheDirectory` | Path from where cached NDP package files are read on a RayManageSoft agent device (on Windows).<br>Relevant when the `ManageSoftPackages` preferences is enabled. | Valid folder path<br>**Default value**: `%ProgramFiles (x86)\ManageSoft\Launcher\PkgCache` |
| `PkgLiveDirectory` | Path from where cached NDP package files are read on an RayManageSoft agent device (on Linux/Unix).<br>Relevant when the `ManageSoftPackages` preference is enabled. | Valid folder path<br>**Default value**: `/var/tmp/managesoft/live` |
| `PlatformSpecificPackages` | Specifies whether software inventory gathering should seek information about non-Windows, platform-specific packages (for example LPP, PKG, RPM, and SD-UX). | Boolean<br>**Default value**: `True` |
| `PNP` | Enables tracking of Plug and Play hardware in the inventory. | Boolean<br>**Default value**: `False` |
| `ProgramFiles` | Points to the Windows program files folder. | Valid folder path<br>**Default value**: `"C:\Program Files"`<br>**Example value**: `"D:\Program Files"` |
| `ProgressDepth` | The number of directory levels to search by the initialization to approximate the number of directories searched during tracking. | Integer between 1-10<br>**Default value**: `3` |
| `Recurse` | Recurse controls whether the inventory agent drills down for inventory collection.<br>• If set to `True`, the tracker includes folders beneath the top level folders specified by | Boolean<br>**Default value**: `True` |

| Name | Description | Values/Range |
|---|---|---|
| | `IncludeDirectory` or `EmbedFileContentDirectory`.<br>• If set to `False`, the tracker does not include folders beneath the top level folders. It only tracks files immediately within the folders specified by `IncludeDirectory` or `EmbedFileContentDirectory`. | |
| `Remote` | Determines if inventory scan is running locally or remotely | Boolean<br>**Default value**: `True` (when the process is running remotely)<br>`False` (when the process is running locally) |
| `RunInventoryScripts` | (Used in conjunction with `InventoryScriptsDir`): Specifies whether or not to run inventory scripts after gathering inventory data. | Boolean<br>**Default value**: `False` |
| `Security` | Determines whether or not to perform security compliance checking. | Boolean<br>**Default value**: `True` |
| `SecurityAnalysis` | Defines the naming conventions for uploaded files containing security analysis details (`SecurityAnalysisRule`). | String<br>**Default value**: `$(ServerLocation)/SecurityAnalysis/$(MachineId) at $(DateTime).msa` |
| `SecurityAnalysisFile` | Defines the naming conventions for local files containing security analysis details. | String<br>**Default value**: `$(MachineId).msa` |
| `ShowIcon` | Controls whether an icon is displayed in the system tray when an inventory scan is running (Windows only). | Boolean<br>**Default value**: `False` |
| `SkipInventory` | Skips the inventory scan. Will be set to True if `MinInventoryInterval` is not yet reached. | Boolean<br>**Default value**: `False` |

| Name | Description | Values/Range |
|---|---|---|
| Software | Determines whether to track software (in the machine context). | Boolean<br>**Default value**: True |
| SMBIOSCmdLine | Command-line to run the smbios utility. | Any valid command line to execute smbios2.exe that results in output being written to stand output. This command line should include the /1 argument.<br>**Default value**: conspawn smbios2.exe /1 /G |
| SwidTagExtension | File extension for files that are to be handled as Software-ID tags. | Valid file extension<br>**Default value**: swidtag |
| SysDirectory | Points to the Windows System folder. | Valid file directory path<br>**Default value**: The path to the system folder on the Windows operating system.<br>**Example value**: C:\Windows\System32 |
| Title | Title of the inventory tracker window (when UserInteractionLevel is set to Full). | String<br>**Default value**: RayManageSoft |
| TLMAgentIniPath | The configuration file for the IBM ILMT standalone scanner.<br><br>This information is used to detect agents and report them in the inventory (in the MGS_ExternalAgent class). | Valid file path<br>**Default value**: /etc/tlmagent.ini (on Unix/Linux)<br>$(WindowsFolder)/itlm/tlmagent.ini (on Windows) |
| TLMStandaloneIniPath | The configuration file for the IMB ILMT standalone scanner.<br><br>This transformation is used to detect agents and report them in the inventory (in the MGS_ExternalAgent Class). | Valid file path<br>**Default value**: /etc/tlmstandalone.ini (on Unix/Linux)<br>$(WindowsFolder)/itlm/tlmstandalone.ini (on Windows) |
| TrackFilesInUserInventory | This setting controls whether file evidence data is collected for user inventories. By default, file evidence cannot be directly linked to particular users. | Boolean<br>**Default value**: False |

| Name | Description | Values/Range |
|---|---|---|
| | If set to `True`, file evidence for user inventories will be collected. | |
| `TrackProductKey` | Determines whether to report product keys from the Microsoft Installer (MSI) packages as part of the software inventory. | Boolean<br>**Default value**: `True` |
| `Upload` | When set to `True`, the inventory and security analysis files are immediately uploaded to the reporting location.<br>When set to `False`, the inventory and security analysis files are not uploaded. | Boolean<br>**Default value**: `True` |
| `UploadFile` | File name of the uploaded copy of the inventory file. | `*.ndi`<br>**Default value**: Same as `InventoryFile`<br>**Example value**: `myComputer.ndi` |
| `UploadLocation` | The URL of the reporting location to upload to (for example `http://myrvserver/ManageSoftRL/`). The file will be uploaded to `http://myrvserver/ManageSoftRL/inventories`.<br>Supported protocols are `http`, `https`, `ftp`, and `file`. | Valid upload location<br>**Default value**: {`empty`} |
| `UploadRule` | Determines the file type to upload. | One of the following:<br>• `Inventory`<br>• `Log`<br>• `PolicyComplianceLog`<br>• `SMSStatusMessage`<br>**Default value**: `Inventory` |
| `UploadUser` | Provides the username of the account required to access the location to which files are to be uploaded. | Valid username<br>**Default value**: {`empty`}<br>**Example value**: `Joe` |
| `UploadPassword` | Provides the password of the specified `UploadUser`. | Valid username<br>**Default value**: {`empty`}<br>**Example value**: `Password321` |

| Name | Description | Values/Range |
|---|---|---|
| `UploadProxy` | Provides the name of the proxy server required for `UploadLocation`.<br>Specifying `direct` tells the uploader to try to directly access the URL if the connection attempts through the proxy server fail. | The name and port number of a proxy server in the format:<br>`Socks:serverName:portNumber,direct`<br>Default value: {`empty`}<br>Example value:<br>`Socks:proxyServer:3128,direct` |
| `UploadRetries` | Determines how often the agent retries to upload a file in case of failure. | Numerical value between 0 and 1,000,000<br>**Default value**: `0` |
| `UploadType` | Determines whether the upload agent uploads machine generated files or user generated files. For example, machine inventory, user inventory, all user installation logs, or current user installation logs. This is a functionality of the upload library that NDTRACK itself does not use but is relevant if it is used as part of the RMS / RV managed device agent. | • `Machine` (if running as the SYSTEM user)<br>• `User` (if running as any other user) |
| `UserHardware` | Determines whether to track hardware (in the user context). | Boolean<br>**Default value**: `False` |
| `UserId` | Name of the user that is used in the name of the inventory file. | String<br>**Default value**:<br>`$(UserName)`<br>**Example value**: `MyUser` |
| `UserInteractionLevel` | Determines the level of user interaction. | • `Full`<br>• `Auto`<br>• `Quiet`<br>• `Status`<br>**Default value**: `Status` |
| `UserInventory` | Static value that cannot be changed. | String<br>**Default value**: User |
| `UserInventoryDirectory` | Location for user inventories on the managed device. | • A valid location<br>**Default value**:<br>`$(AppDataFolder)\ManageSoft Corp\ManageSoft\Tracker\Inventories` |

| Name | Description | Values/Range |
|---|---|---|
| UserName | Contains the name of the local user. Unlike UserId, this preference should not be changed. | String<br>**Default value**: $(UserName) |
| UserZeroTouchDirectory | Specifies the directory where the user inventory files are written temporarily during a remote execution inventory gathering process. If the upload (called as part of the inventory scanning process) proceeds normally, each temporary file is cleaned up after upload. | Valid location<br>**Default value**: $(AppDataFolder)\ManageSoft Corp\ManageSoft\Tracker\ZeroTouch |
| Version | Contains the version number of ndtrack. | String<br>**Default value**: {ndtrack version number} |
| VersionInfo | Determines whether the file version header information is included in inventory or not. | Boolean<br>**Default value**: True |
| WinDirectory | Points to the Windows folder. | Valid directory path<br>**Default value**: C:\Windows |
| WMI | When set to True, the Windows Management Instrumentation (WMI) tracking is specified as the preferred option for tracking hardware. In this case, if WMI is not available (and the Hardware preference is set to True), the tracker (ndtrack executable) attempts to track hardware using a native API.<br><br>When set to False, the tracker uses another tracking mechanism instead of WMI. | Boolean<br>**Default value**:<br>• True (when running in machine context)<br>• False (when running in user context) |
| WMIConfigFile | Defines the location of the Windows Management Instrumentation (WMI) configuration file, used to inform the tracker (ndtrack executable) what hardware components it should track. This is only used if WMI is True. | Valid file path<br>**Default value**: $(ProgramPath)\wmitrack.ini |

The files ndtrack.sh and ndtrack.ini must reside in the current directory or working directory. Otherwise, NDTRACK will not scan the hardware.

**Tip:**

- The optional http-upload does a HTTP-PUT.
- Instead of HTTP you may use FTP. Just change the protocol prefix from `http` to `ftp`.
- If you do not upload to a RayVentory or RMS server then you must make sure that the upload endpoint's URL ends on `/inventories`. For example: The URL is `http://myhost/Uploads/inventories` and command-line argument is `-o UploadLocation=http://myhost/Uploads/`.

## Examples (UNIX / Linux)

Create an inventory (Machine or 'everything' scope) and upload it to an RMS server. A copy will stay in the default inventory location `/var/tmp/managesoft/tracker/inventories/`:

```
./ndtrack.sh -t Machine -o UploadLocation=http://myRVserver/
ManageSoftRL/
```

Create an inventory (Machine or 'everything' scope) and upload it to an RMS server by CURL. A copy will stay in the default inventory location `/var/tmp/managesoft/tracker/inventories/`:

```
./ndtrack.sh -t Machine -o UploadLocation=http://myRVserver/
ManageSoftRL/ -o UploadCurl=True
```

Create an inventory (Machine or 'everything' scope) and upload it to an RMS server by CURL, providing a username and a password for the upload. A copy will stay in the default inventory location `/var/tmp/managesoft/tracker/inventories/`:

```
./ndtrack.sh -t Machine -o UploadLocation=http://myRVserver/
ManageSoftRL/ -o UploadCurl=True -o UploadUser=myuser -o
UploadPassword=mypassword
```

Create an inventory (Machine or 'everything' scope) and upload it to an RMS server by CURL using a username / password from the uploader credential store with a custom credential store key. A copy will stay in the default inventory location `/var/tmp/managesoft/tracker/inventories/`:

```
./ndtrack.sh -t Machine -o UploadLocation=http://myRVserver/
ManageSoftRL/ -o UploadCurl=True -o UploadCredFile=/home/someuser/
mycredentials.cred -o UploadCredFileKey=myCredentialStoreKey
```

Create an inventory (Machine or 'everything' scope) and store it in the directory `/var/tmp/`

`stuff` (generating an uncompressed `.ndi` file and a compressed `.ndi.gz` file):

```
./ndtrack.sh -t Machine -o Upload=False -o
MachineInventoryDirectory=/var/tmp/stuff -o InventoryDirectory=/
var/tmp/stuff -o UploadDirectory=/var/tmp/stuff
```

Create an inventory and store it together with a compressed copy to the default location `/var/tmp/managesoft/tracker/inventories/`:

```
./ndtrack.sh -t Machine -o Upload=False
```

Create an inventory, upload it, and specify a CA-bundle or CA-cert file. Use this if HTTPS fails because openSSL is not able to get the issuer cert.

```
./ndtrack.sh -t Machine -o UploadLocation=https://myRVserver/
ManageSoftRL/ -o SSLCACertificateFile='absolutePathToCAcert.pem'
```

Create an inventory, upload it, and disable the server / peer-certificate verification (issuer-cert and CRL check). Use this if HTTPS fails because server / peer-cert verification fails.

```
./ndtrack.sh -t Machine -o UploadLocation=https://myRVserver/
ManageSoftRL/ -o CheckCertificateRevocation=false -o
CheckServerCertificate=false
```

Create an inventory and store it to `/home/user/inventory/linuxInventory.ndi` without compression:

```
./ndtrack.sh -o  Upload=False -o InventoryDirectory=/home/user/
inventory/ -o InventoryFile=linuxInventory.ndi -o Compress=False
```

Create an inventory (Machine or 'everything' scope) and upload it to an RMS server by CURL using a certificate (the client-certificate and client-key must be PEM format) for client authentication. A copy will stay in the default inventory location `/var/tmp/managesoft/tracker/inventories/`:

```
./ndtrack.sh -t Machine -o UploadLocation=http://myRVserver/
ManageSoftRL/ -o UploadCurl=True -o UploadCurlCommand="curl --cert
<client-certificate>:<optional passphrase> --key <client-key>"
```

Or with a certificate to check the server certificate:

```
./ndtrack.sh -t Machine -o UploadLocation=http://myRVserver/
ManageSoftRL/ -o UploadCurl=True -o UploadCurlCommand="curl --cert
<client-certificate>:<optional passphrase> --key <client-key> --
cacert <certificate-authority-certificate>"
```

Or with a client certificate that combines public and private keys, specifying a certificate format:

```
./ndtrack.sh -t Machine -o UploadLocation=http://myRVserver/
ManageSoftRL/ -o UploadCurl=True -o UploadCurlCommand="curl --cert-
type <DER, PEM or ENG> --cert <client-certificate>:<optional
passphrase>"
```

Or:

```
./ndtrack.sh -t Machine -o UploadLocation=https://myDS/
ManageSoftRL/ -o UploadCurl=True -o UploadCurlCommand="curl --
cert /home/myUser/Desktop/Certificates/myServerAuthPrivateKey.pem -
v"
```

Or:

```
./ndtrack.sh -t Machine -o UploadLocation=https://myDS/
ManageSoftRL/ -o UploadCurl=True -o UploadCurlCommand="curl --
cert /home/<User>/<path>/<clientCert>.pem -v"
```

### Examples (Windows)

Create an inventory and store it to `%temp%\inventory\WindowsInventory.ndi`:

```
ndtrack.exe -o  Upload=False -o InventoryDirectory=%temp%\inventory
-o InventoryFile=WindowsInventory.ndi -o Compress=False
```

### Ports in Use

- `80` (`443`) for uploads of the inventory to an inventory data sink via HTTP(S).
- `445` for uploads of the inventory to an inventory data sink via SMB.
- `20/21` for uploads of the inventory to an inventory data sink via FTP.

## SNMP Tracker

The SNMPTracker (`SNMPTracker.exe`) is a command-line oriented tool for generating inventories by querying a target device using SNMP. This is intended for SNMP enabled network devices like printers, switches, routers, and UPSs. SNMPTracker is highly customizable and supports a wide range of different devices.

The command-line arguments are passed to SNMPTracker as name-value pairs.

## Usage

```
SNMPTracker.exe [-o <argument-name>=<value>]...
```

## Table of Arguments

| Name | Type | Description |
|---|---|---|
| auth | OPTION | The authentication method. Valid values are MD5 and SHA1. Implies authphrase. |
| authphrase | TEXT | The authentication passphrase for SNMP v3. |
| community | TEXT | The community name for SNMP.<br>Default is: public |
| config | TEXT | The configuration file. Alternatively, you can pass in one or more filenames after the options. |
| control | TEXT | The path or name of the control file, which is an XML file of a certain format. |
| deviceID | INTEGER | Sets the device ID (this is supposed to be used by Raynet's inventory / discovery agents only). |
| encsys | OPTION | Encryption system for SNMP v3. Valid values are AES and DES.<br>Default is: DES |
| host | TEXT | Target host, alternative to scanRule. |
| job | TEXT | Sets the job tag / ID (this is supposed to be used by Raynet's inventory /discovery agents only). |
| maxDelay | INTEGER | Maximum seconds of delay for the upload. The value 0 disables delayed upload.<br>Default is: 0 |
| mibpath | TEXT | The path to the MIB files to compile.<br>Default is: mibs |
| mibpattern | TEXT | The filename pattern for the MIB files to compile. Default is: *.txt |
| minDelay | INTEGER | Minimum seconds of delay for the upload.<br>Default is: 0 |
| output | TEXT | Output directory for inventory files. |
| outfile | TEXT | Output file name if the option host is used. |
| port | SHORT | Port number for communication with the target device.<br>Default is: 161 |
| privacy | OPTION | Privacy method for SNMP v3. Implies the argument privphrase. |
| privphrase | TEXT | Privacy phrase for SNMP v3. |

| Name | Type | Description |
|---|---|---|
| protocol | OPTION | Protocol version.<br>Default is: 2 |
| scanRule | TEXT | Supply a list of lists where the inner lists list targets hosts. If the current machine is <host> then target <targets>.<br><br>Example:<br>myMachine1:target1,target2;myMachine2:target2 |
| security | OPTION | The security level for SNMP v3. Valid values are: NOAUTHNOPRIV, AUTHNOPRIV, and AUTHPRIV. |
| threads | INTEGER | Maximum number of threads.<br>Default is: 1 |
| timeout | INTEGER | Timeout for SNMP responses.<br>Default is: 2000 |
| upload | TEXT | URI for inventory upload. |
| useDns | BOOLEAN | Query the DNS for a hostname for an IP address.<br>Default is: True |
| useProcess | BOOLEAN | Creates processes instead of threads for parallel queries.<br>Default is: False |
| user | TEXT | The user for SNMP v3 authentication. |

## Ports in Use

- 80 (443) for upload of the inventory to an inventory data sink via HTTP(S).
- 445 for upload of the inventory to an inventory data sink via SMB.
- 161 for communication with the target devices via SNMP.

# RayVentory Scan Engine Maintenance Tool

The RayVentory Scan Engine Maintenance Tool is an application that is able to identify duplicate connections and to remove these duplicates.

**WARNING**
This tool is only meant to be used by consultants and advanced users!

## Usage

```
RayVentoryScanEngine.Maintenance.exe [--<argument-name>=<value>]...
```

## Table of Arguments

| Name | Type | Description |
|---|---|---|
| `--dryRun` | `Boolean` | If set to `true`, dry run is activated. During a dry run no connections will be deleted by the algorithm. The dry run can be used to display how many connections will be removed by the algorithm without actually removing them. |
| `--exclusionListFile` | `Text` | The path to the file with a list for each BIOSSerial, ComputerUUID, and Hostname to be excluded by the algorithm. |
| `--help` | `Flag` | Displays the help screen which contains a list of the arguments. |
| `--version` | `Flag` | Displays the version information. |

Execute a Dry Run:

```
RayVentoryScanEngine.Maintenance.exe --dryRun=true
```

Disable Dry Run:

```
RayVentoryScanEngine.Maintenance.exe --dryRun=false
```

Use the Exclusion List File:

```
RayVentoryScanEngine.Maintenance.exe --
exclusionListFile=<pathToFile>
```

It is also possible to combine the parameters. For example:

```
RayVentoryScanEngine.Maintenance.exe --dryRun=false --
exclusionListFile=<pathToFile>
```

# PowerShell Automation

RayVentory Scan Engine exposes its core functions via PowerShell commandlets bundled in a module. To automate the operations via PowerShell import the following module to your session:

```
[INSTALLDIR]\Libs\Raynet.RayVentory.ScanEngine.Automation.psd1
```

The following commands are avilable:

- `Get-DeviceConnections`
- `Get-OracleConnections`
- `Get-SnmpConnections`
- `Get-VsphereConnections`
- `Import-DeviceList`
- `Import-LdapDevices`
- `Import-NetworkDevices`
- `Send-Inventory`
- `Start-DeviceInventory`
- `Start-OracleInventory`
- `Start-SnmpInventory`
- `Start-VsphereInventory`

This user describes the usage of the `Get-DeviceConnections` and the `Send-Inventory` commandlets. For more information about other commands, execute the following command in the PowerShell session (after importing the module first):

```
Get-Help <command_name> -Full
```

## Get-DeviceConnections

Returns all devices or a filtered collection of devices as shown in the **Devices + Services** screen. Returns a list of `OsConnection` objects.

**Syntax:**

```
Get-DeviceConnections [-DnsPatternName <wildcard>] [-
IpAddressPattern <wildcard>] [-StatusPattern <wildcard>] [-OsType
<Windows|Unix|Unspecified>] [-IsSingle]
```

**Parameters:**
All parameters are optional. Specifying filter parameters is additive (all specified filters must match before a device is returned).

**`-DnsNamePattern <wildcard>`**

The wildcard pattern (supports * for zero or more characters and ? for exactly a single character) for the DNS names of the returned devices.

**-IpAddressPattern <wildcard>**
The wildcard pattern (supports * for zero or more characters and ? for exactly a single character) for the IP addresses of the returned devices.

**-StatusPattern <wildcard>**
The wildcard pattern (supports * for zero or more characters and ? for exactly a single character) for the states of the returned devices.

**-OsType <Windows|Unix|Unspecified>**
The type of the hosts to return.

**-IsSingle**
If specified, only the first record is returned.

**Returns**
List of `OsConnection` objects that match the given criteria.

**Full Parameter Reference:**

| Parameter name | Type | Is required? | Specific position? | Accepts pipeline input? | Set name | Aliases | Is dynamic? |
|---|---|---|---|---|---|---|---|
| DnsNamePattern | String | No | No | No | (all) | None | No |
| IpAddressPattern | String | No | No | No | (all) | None | No |
| IsSingle | Switch | No | No | No | (all) | None | No |
| OsType | <HostType> | No | No | No | (all) | None | No |
| StatusPattern | String | No | No | No | (all) | None | No |

**Example:**
To get all connections where the status contains the word "authentication":

```
Get-DeviceConnections -StatusPattern "*authen*"
```

To get all connections where the IP address starts with 192.168:

```
Get-DeviceConnections -IpAddressPattern "192.168.*"
```

To get all UNIX devices where the last inventory suceeded:

```
Get-DeviceConnections -OsType Unix -StatusPattern "OK"
```

To get all devices, show them as a table, and group them by the discovery source:

```
Get-DeviceConnections | Format-Table -GroupBy CreatedBy
```

# Send-Inventory

Uploads the inventory files to the parent upload location.

**Syntax:**

```
Send-Inventory -UploadLocation <string> [-Credentials
<PSCredential>]
```

**Parameters:**
Upload location is required, the credentials are optional:

**`-UploadLocation <string>`**
The full upload location of the parent upload server.

**`-Credentials <PSCredential>`**
The optional credentials to authenticate against the upload location server.

**Full Parameter Reference:**

| Parameter name | Type | Is required? | Specific position? | Accepts pipeline input? | Set name | Aliases | Is dynamic? |
|---|---|---|---|---|---|---|---|
| `UploadLocation` | String | Yes | 0 | No | (all) | None | No |
| `Credentials` | String | No | 1 | No | (all) | None | No |

**Example:**
To upload inventory files to the specified parent:

```
Send-Inventory "http://parent:80/Inventories"
```

# OsConnection

The `OsConnection` objects have the following structure (sample data provided for a reference):

```
Property                            | Value
------------------------------------------------------------------------
----------
Uuid                                : be785fbd-ef84-4f9f-87da-
dd073edc9717
CreatedBy                           : Discovery AD-import
CreatedDate                         : 16.10.2018 09:09:44
Credentials                         :
CurrentInventoryFiles               : {}
Host                                : MWS0231.raynet.corp
Hostname                            : MWS0231.raynet.corp
Id                                  : hostname='MWS0231.raynet.corp',
ipaddress= '192.168.120.165'
IPAddress                           : 192.168.120.165
LastInventoryAttempt                : 16.10.2018 09:10:53
LastInventoryDate                   :
SshPort                             : 22
Status                              : Authentication failed: No
credentials allowed us to run ndtrack via mgsreservice.exe.
TargetType                          : Windows
UseWinSessionCreds                  : False
DisabledInventoryMethods            : 0
LastSuccessfulInventoryMethod       :
LastFailedInventoryMethods          :
LastFailedInventoryMethodsDetail    :
UserName                            :
Password                            :
AsSysDba                            :
```

# Appendix I: Prerequisites Inventory Methods

The necessary information to use the different inventory methods available in RayVentory Scan Engine can be found in the following.

## Remote Execution Inventory for Windows

### Required Permission to Run Remote Execution Inventories Against Windows Devices

All Remote Execution methods need administrative rights on the target machine. Furthermore, there are some method specific prerequisites:

#### ServiceManager/SMB local copy

- Rights to create and start a temporary service on the target machine
- SMB access (access to administrative share ADMIN$ on the target machine)
- Disabled Remote UAC on the target device (only when the target device is not in the domain)

#### WMI/SMB local copy

- Rights to execute remote commands via WMI
- SMB access (access to administrative share AMIN$ on the target machine)
- Disabled Remote UAC on target device (only when the target device is not in the domain)

#### ServiceManager/HTTP(S)

- Rights to create and start a temporary service on the target machine
- SMB access (access to a configured utility share)
- Accessible RayVentory Scan Engine HTTP(S) service (default port 591
- The target device needs to have a domain trust relationship with the RayVentory Scan Engine device

#### ServiceManager/SMB

- Rights to create and start a temporary service on the target machine
- SMB access (access to a configured utility share)
- SMB access (access to a configured upload location)
- Target device needs to have a domain trust relationship with the RayVentory Scan Engine device

### WMI/HTTP(S)

- Rights to execute the remote commands via WMI
- SMB access (access to a configured utility share)
- Accessible RayVentory Scan Engine HTTP(S) service (default port 591)
- Disabled Remote UAC on the target device (only when the target device is not in the domain)

### WMI/SMB

- Rights to execute remote commands via WMI
- SMB access (access to a configured utility share)
- SMB access (access to a configured upload location)
- Disabled Remote UAC on the target device (only when the target device is not in the domain)

# Remote Execution Inventory for Unix/Linux

## Linux/Unix Remote Execution Login Methods

### Installation Specifications

We are using the library **SSH.NET** which supports the following private key formats:

- RSA in OpenSSL PEM and ssh.com format
- DSA in OpenSSL PEM and ssh.com format
- ECDSA 256/384/521 in OpenSSL PEM format
- ECDSA 256/384/521, ED25519 and RSA in OpenSSH key format

Private keys can be encrypted using one of the following cipher methods:

- DES-EDE3-CBC
- DES-EDE3-CFB
- DES-CBC
- AES-128-CBC
- AES-192-CBC
- AES-256-CBC

For further information, please read the documentation for the library: https://github.com/sshnet/SSH.NET

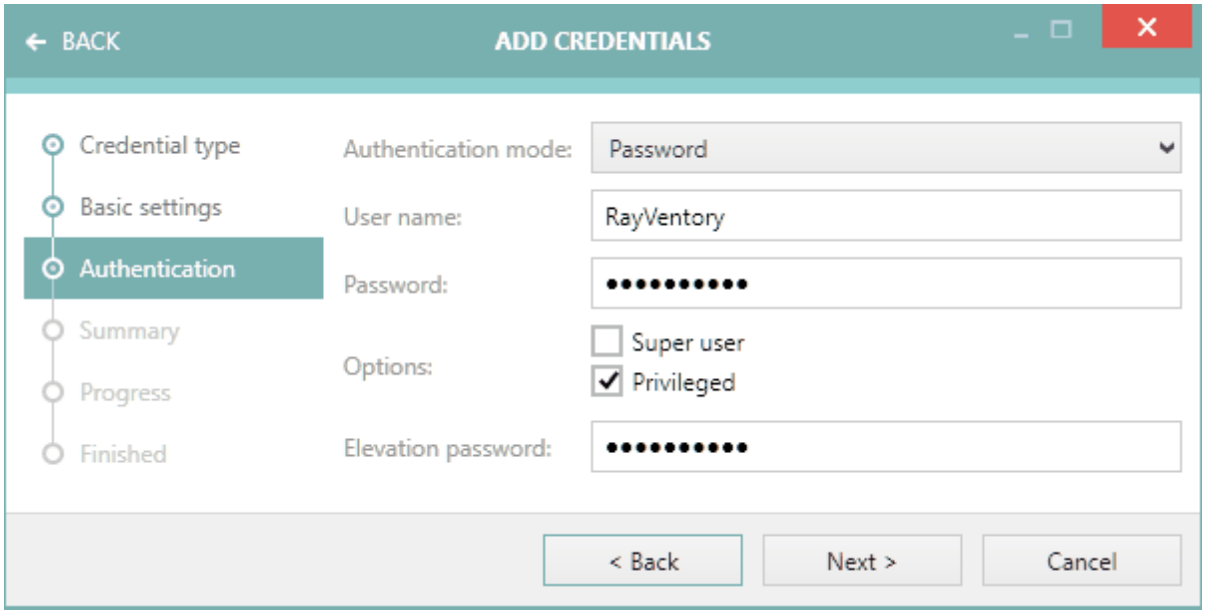
### Username and Password

The simplest way is to create a user and password combination:

1. Create a user (e.g. useradd RayVentory).
2. Set a password for the User (passwd RayVentory).
3. Set permissions like described below.
4. Add User to the Credentials Store of RVSE.

**Tip:**
Prefer the "privileged" option and add the elevation password.
Superusers like "root" are commonly not permitted to execute every command without elevated rights!

RVSE:

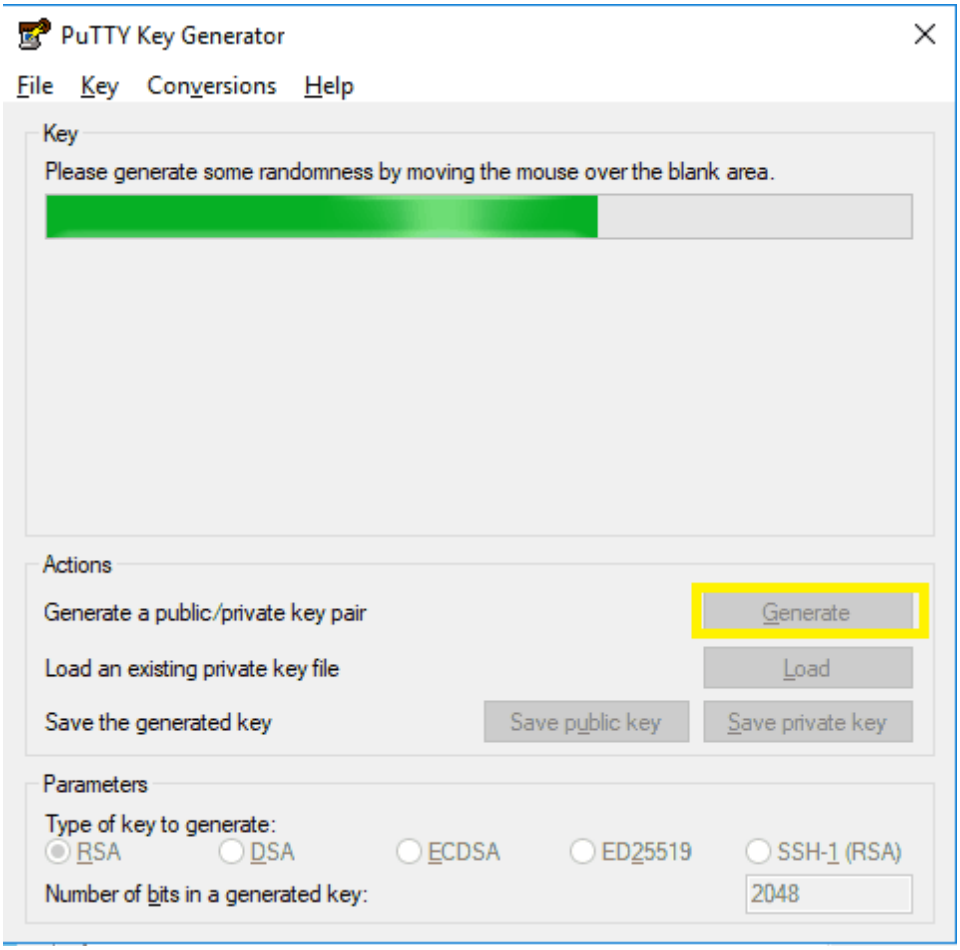



## SSH Key-Based Authentication

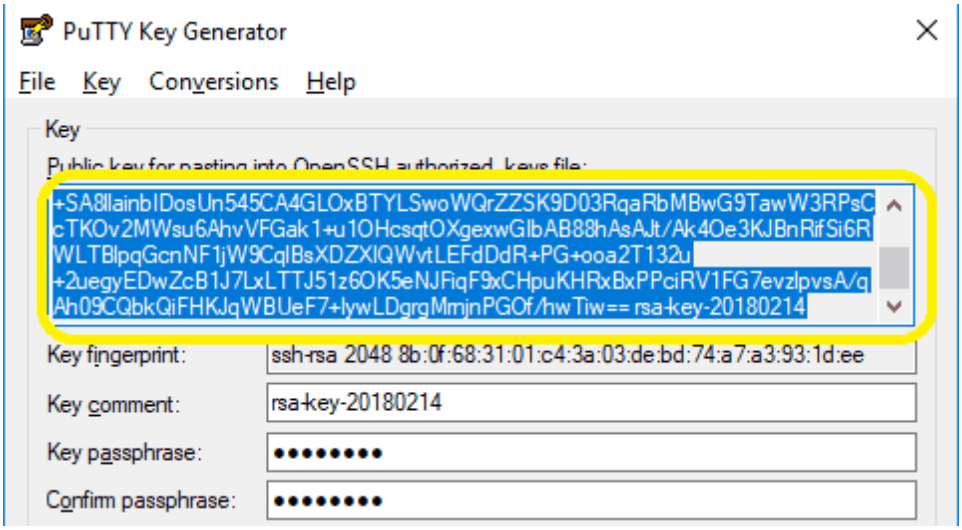
This description might need adoptions, depending on if using Windows or other platforms.

1. Login to Linux/Unix system with the designated user account used for RayVentory.
2. Create a local folder `.ssh` in the users home directory, if such does not exist already
3. If the file `./.ssh/`authorized_keys does not exist, create it.•
   To this file we need to add the public key later.
4. Make sure **sshd** configuration and service is set up properly.
5. Download `puttygen.exe` from https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html.
6. Check the download with an Antivirus tool.
7. Start `puttygen.exe` (screenshot taken from Windows).

8. Start generating a public key by selecting the **Generate** button.

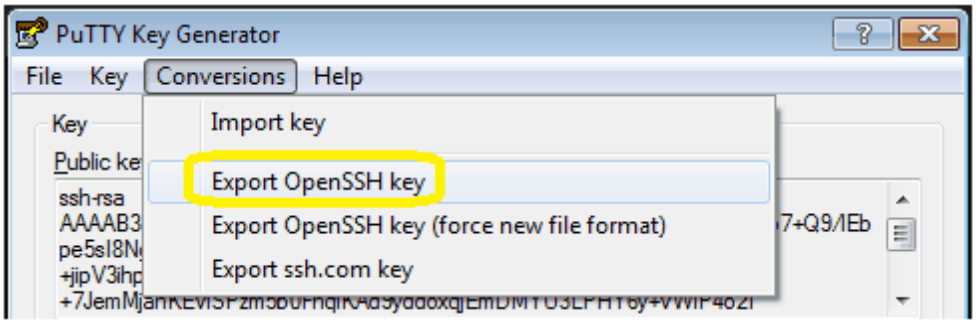



9. Move the mouse around to randomly generate the key.
10. Set a "Key passphrase" for the public key. This password is required later for changing the public key and generating private keys.
11. Save the public key as file to disk.
12. Copy all data within the public key field and add it to the `./ssh/authorized_keys` file within the user directory of the RayVentory user on the Linux or Unix machine. Section highlighted in the following screenshot:




**Tip:**
Prefer the "privileged" option and add the elevation password.
Superusers like "root" are commonly not permitted to execute every command without elevated rights!

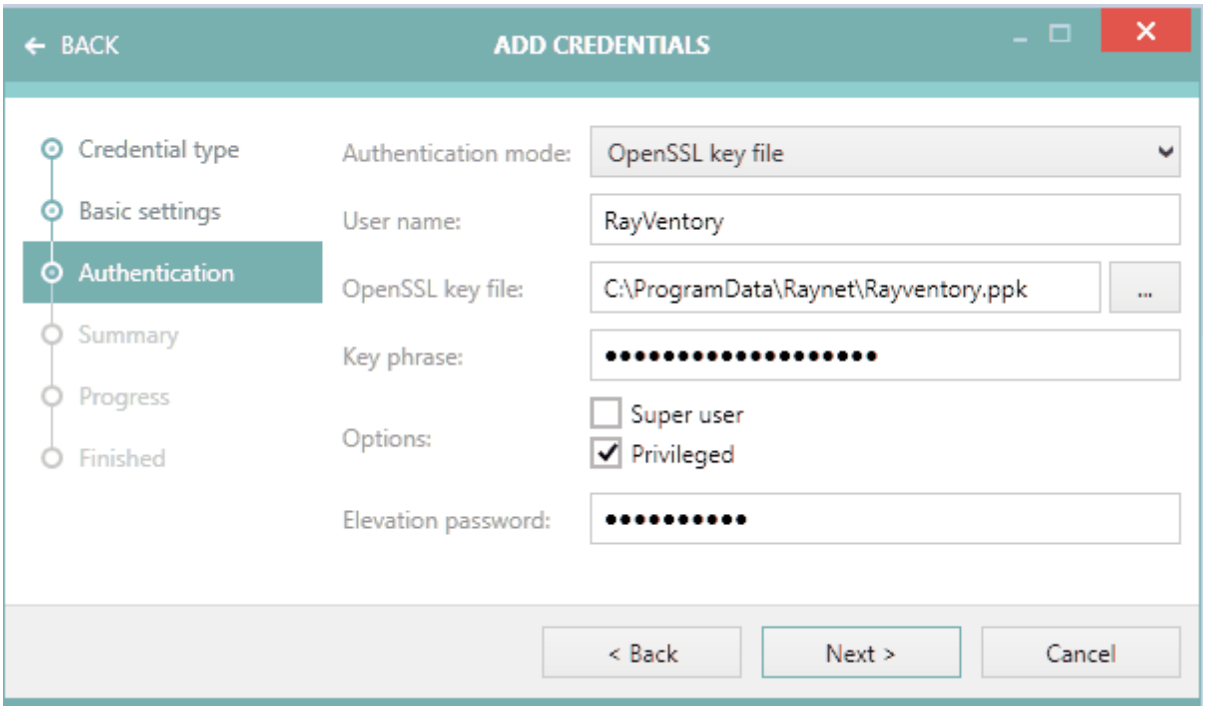
13.Export the private key as an **OpenSSH** key and save it.




14.Copy the file containing the "OpenSSH" private key to the RVSE server.

## Enabling RVSE Using the Private Key File

1. Start RVSE
2. Open the Credential Store and create an SSH•credential
3. Select Authentication method "Key file"
4. Add the Username, path of the "OpenSSH key file" and Key passphrase
5. Select option "Privileged" and enter the elevation password



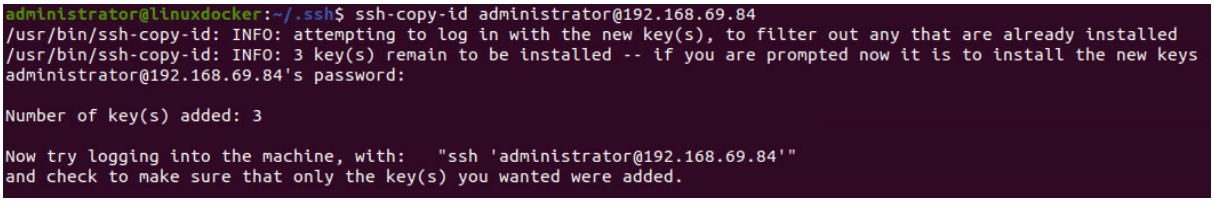

## Alternatively, It Is Possible to Create the SSH-Keys Directly on a Linux/ Unix Machine

1. Create the SSH-Key with the following command:
   ssh-keygen -m PEM -t rsa -b 2048

```
administrator@linuxdocker:~/.ssh$ ssh-keygen -m PEM -t rsa -b 2048
Generating public/private rsa key pair.
Enter file in which to save the key (/home/administrator/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/administrator/.ssh/id_rsa.
Your public key has been saved in /home/administrator/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:ZH7ky9M03EY9GhAu1AxBfQYZleLEEkPtGgRJ9HYpDwM administrator@linuxdocker
The key's randomart image is:
+---[RSA 2048]----+
|      oE=OX=*..  |
|       .+ooXo+ . |
|        +B*++....|
|       +.+B+ oo .|
|        S =.+.o  |
|         + + o   |
|          + .    |
|           .     |
|                 |
+----[SHA256]-----+
```

2. Use the following command to add the public key to the authorized keys:

```
ssh-copy-id <user>@"IP address of the linux machine"
```




3. Copy the Private-Key file (file at `/home/<user>/.ssh`) onto your RVSE server.

### Deploy Public Key File to Linux/Unix Systems

After creating a new Public Key for SSH and completing tests with RVSE, the public key is ready for deployment to all Linux/Unix systems.

## Required Permissions to Run Remote Execution Tasks Against Linux/Unix Devices

### Service Used from Remote

We need rights to the following services from remote machines:

- SSH (Login)
- SCP (Login/use)

### SUDO Rights for Hardware/Software Inventory

Sudo rights without password prompt are needed for the following commands:

- `/sbin/date`
- `/bin/date`
- `/bin/sh ./ndtrack.sh *`
- `/bin/sh ./ndtrack.sh *`
- `/sbin/rm -f ./ndtrack.sh ./ndtrack.ini`
- `/bin/rm -f ./ndtrack.sh ./ndtrack.ini`

### The Reultant SUDO Configuration Line Is as Follows:

```
{UserName} ALL = (root) NOPASSWD: /bin/date,/sbin/date,/bin/sh ./
ndtrack.sh *,/sbin/sh ./ndtrack.sh *,/bin/rm -f ./ndtrack.sh ./
ndtrack.ini,/sbin/rm -f ./ndtrack.sh ./ndtrack.ini
```

**Note:**
Please replace {UserName} with the user for RayVentory.

**Note:**
To edit the sudo configuration use the visudo command.

# Remote Execution Inventory for Oracle

## Linux/UNIX Remote Execution Login Methods

### Installation Specifications

We are using the library **SSH.NET** which supports the following private key formats:

- RSA in OpenSSL PEM and ssh.com format
- DSA in OpenSSL PEM and ssh.com format
- ECDSA 256/384/521 in OpenSSL PEM format
- ECDSA 256/384/521, ED25519 and RSA in OpenSSH key format

Private keys can be encrypted using one of the following cipher methods:

- DES-EDE3-CBC
- DES-EDE3-CFB
- DES-CBC
- AES-128-CBC
- AES-192-CBC
- AES-256-CBC

For further information, please read the documentation for the library: https://github.com/sshnet/SSH.NET

### Username and Password

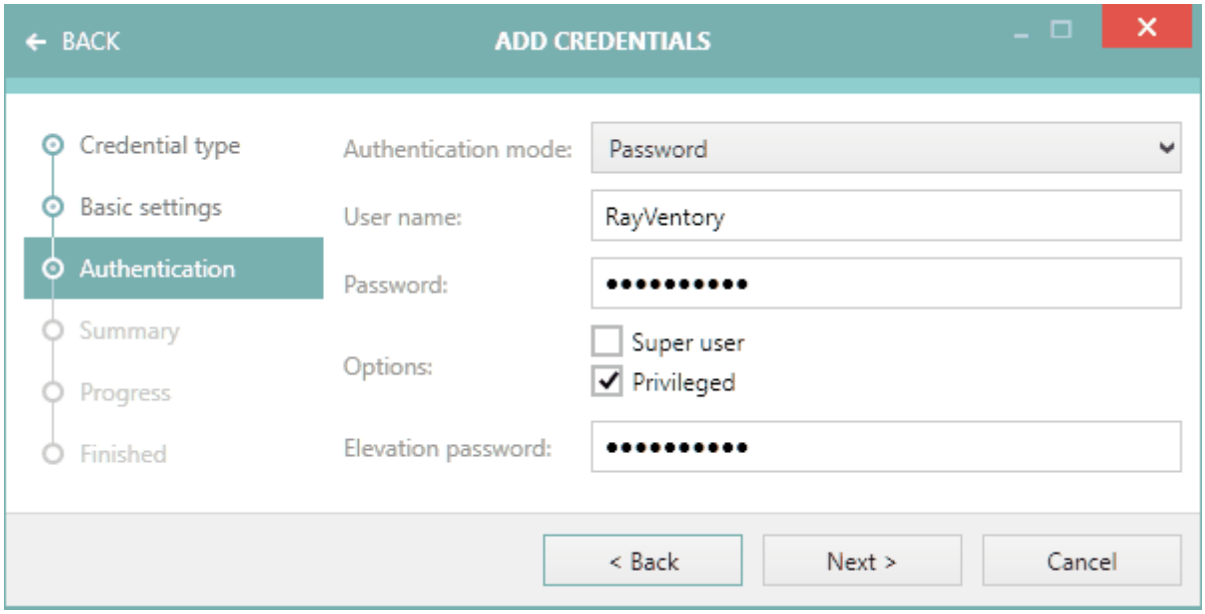
The simplest way is to create a user and password combination:

1. Create a user (e.g. useradd RayVentory).
2. Set a password for the User (passwd RayVentory).
3. Set permissions like described below.
4. Add User to the Credentials Store of RVSE.

**Tip:**
Prefer the "privileged" option and add the elevation password.
Superusers like "root" are commonly not permitted to execute every command without

elevated rights!

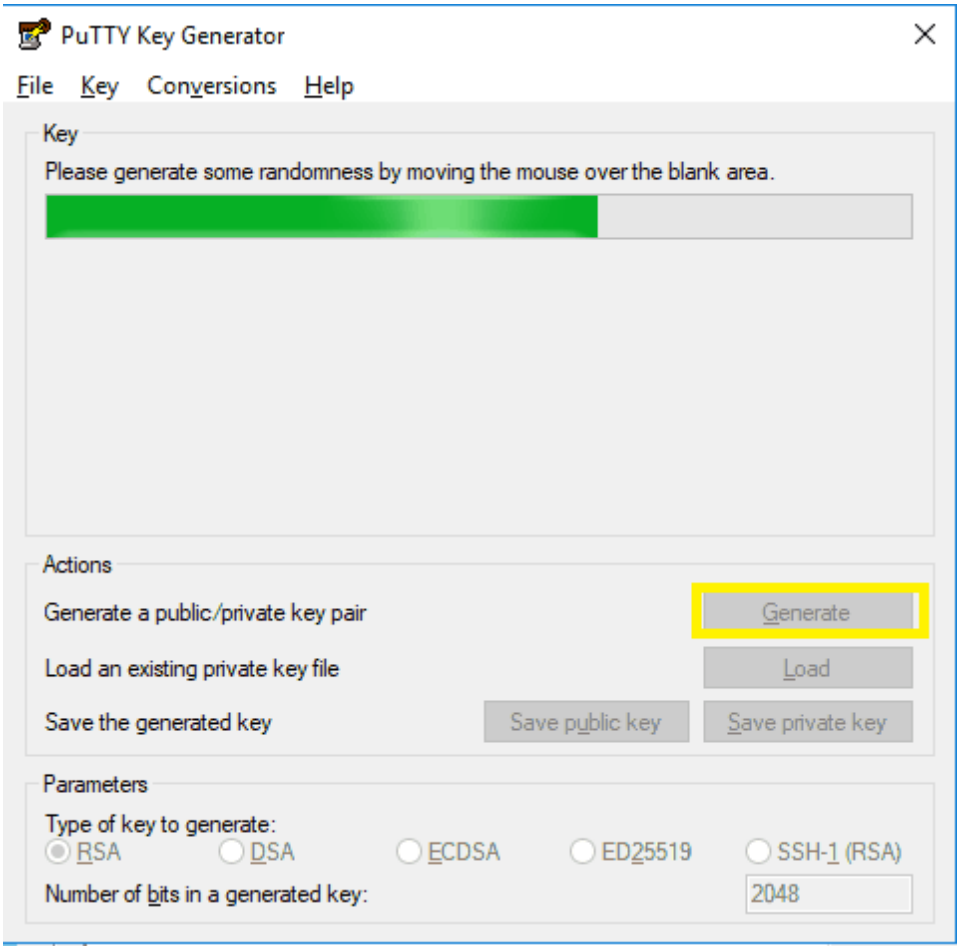
RVSE:



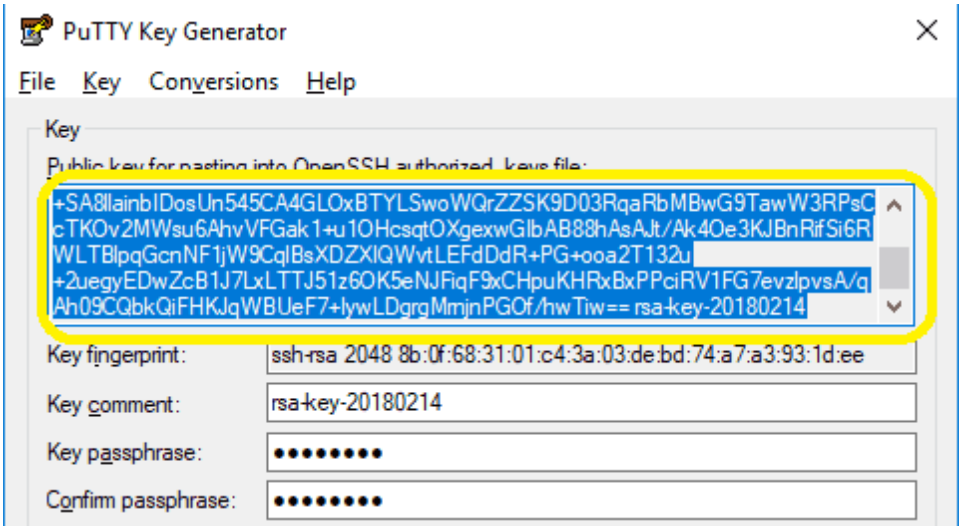

## SSH Key-Based Authentication

This description might need adoptions, depending on if using Windows or other platforms.

1. Login to Linux/Unix system with the designated user account used for RayVentory.
2. Create a local folder `.ssh` in the users home directory, if such does not exist already
3. If the file `./.ssh/`authorized_keys does not exist, create it.•
   To this file we need to add the public key later.
4. Make sure **sshd** configuration and service is set up properly.
5. Download `puttygen.exe` from https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html.
6. Check the download with an Antivirus tool.
7. Start `puttygen.exe` (screenshot taken from Windows).
8. Start generating a public key by selecting the **Generate** button.

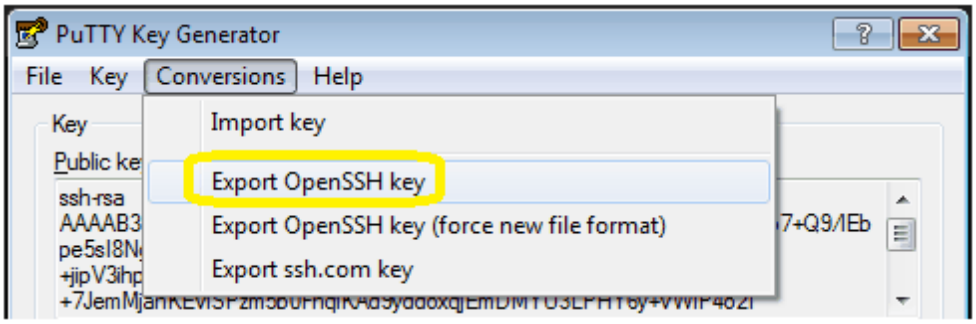



9. Move the mouse around to randomly generate the key.
10. Set a "Key passphrase" for the public key. This password is required later for changing the public key and generating private keys.
11. Save the public key as file to disk.
12. Copy all data within the public key field and add it to the `./ssh/authorized_keys` file within the user directory of the RayVentory user on the Linux or Unix machine. Section highlighted in the following screenshot:




**Tip:**
Prefer the "privileged" option and add the elevation password.
Superusers like "root" are commonly not permitted to execute every command without elevated rights!

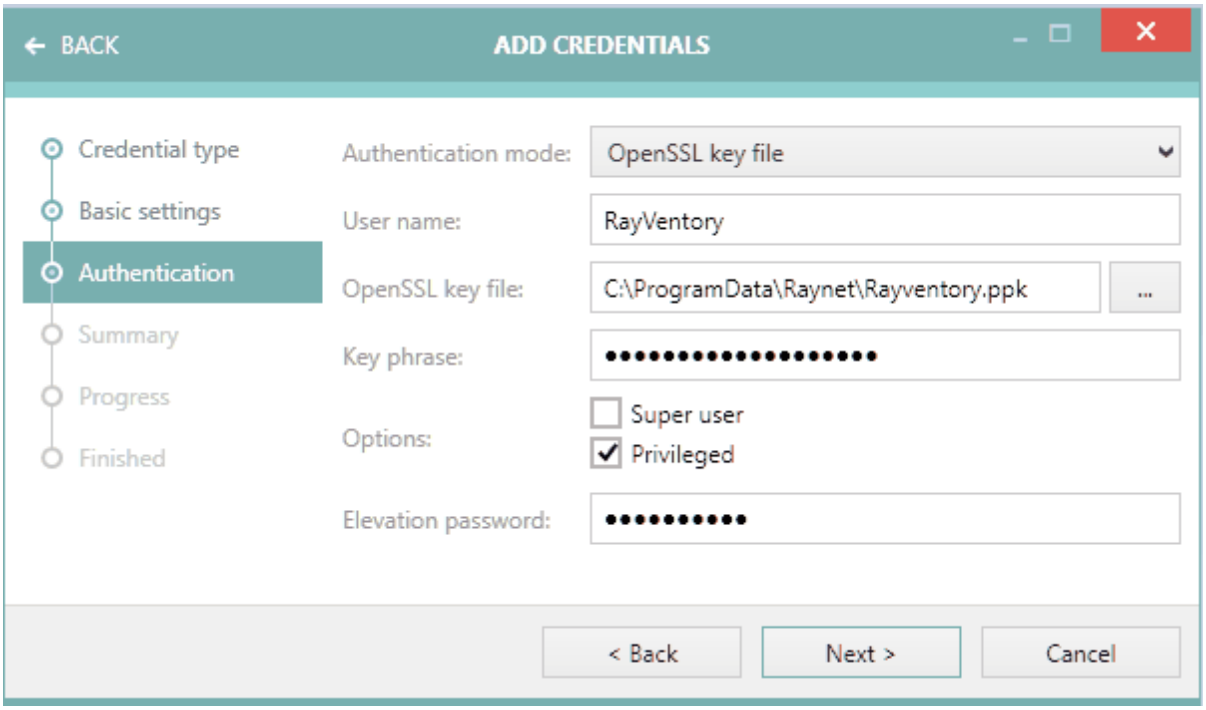
13.Export the private key as an **OpenSSH** key and save it.




14.Copy the file containing the "OpenSSH" private key to the RVSE server.

## Enabling RVSE Using the Private Key File

1. Start RVSE
2. Open the Credential Store and create an SSH•credential
3. Select Authentication method "Key file"
4. Add the Username, path of the "OpenSSH key file" and Key passphrase
5. Select option "Privileged" and enter the elevation password




## Alternatively, It Is Possible to Create the SSH-Keys Directly on a Linux/ Unix Machine

1. Create the SSH-Key with the following command:
   ssh-keygen -m PEM -t rsa -b 2048

```
administrator@linuxdocker:~/.ssh$ ssh-keygen -m PEM -t rsa -b 2048
Generating public/private rsa key pair.
Enter file in which to save the key (/home/administrator/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/administrator/.ssh/id_rsa.
Your public key has been saved in /home/administrator/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:ZH7ky9M03EY9GhAu1AxBfQYZleLEEkPtGgRJ9HYpDwM administrator@linuxdocker
The key's randomart image is:
+---[RSA 2048]----+
|      oE=OX=*..  |
|       .+ooXo+ . |
|        +B*++....|
|       +.+B+ oo .|
|        S =.+.o  |
|         + + o   |
|          + .    |
|           .     |
|                 |
+----[SHA256]-----+
```

2. Use the following command to add the public key to the authorized keys:

```
ssh-copy-id <user>@"IP address of the linux machine"
```

```
administrator@linuxdocker:~/.ssh$ ssh-copy-id administrator@192.168.69.84
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 3 key(s) remain to be installed -- if you are prompted now it is to install the new keys
administrator@192.168.69.84's password:

Number of key(s) added: 3

Now try logging into the machine, with:   "ssh 'administrator@192.168.69.84'"
and check to make sure that only the key(s) you wanted were added.
```

3. Copy the Private-Key file (file at `/home/<user>/.ssh`) onto your RVSE server.

### Deploy Public Key File to Linux/Unix Systems

After creating a new Public Key for SSH and completing tests with RVSE, the public key is ready for deployment to all Linux/Unix systems.

## Required Permissions to Run a Local Oracle Inventory Scan Via a Remote Execution Task Against Lunix/Unix Devices

### Services Used from Remote

We need rights to the following services from remote machines:

- SSH (Login)
- SCP (Login/use)

### Access on the System

- Read access on the whole filesystem*
- Read/Write/Execute on home directory
  - /home/<UserName>

*Permission to the whole filesystem is required in case that the user is not allowed to execute java via sudo. This is also required in case that the user not allowed to execute 'find' via sudo and Java is not available to the scanning user.

### Read Access to All Configuration Files

- `/etc/oratrab`
- `*/listener.ora`
- `*/tnsnames.ora`

### SUDO Rights for Oracle Inventory

Execute Java (in order to run ORATRACK which searches for Oracle configuration files) via sudo, without restrictions to the command line arguments and no password prompt:

- `sudo cmod 755 ./oratrack*.jar`
- `sudo java -jar oratrack*.jar`

The execution of Java is not limited to a single version. If several versions of Java are installed, all Java versions will have to be executed.

Execute 'find' (to search for the available Java Runtime Environments) via sudo, without restrictions to the command line arguments and no password prompt:

- `sudo find`

**Note:**
Please replace {UserName} with the user for RayVentory.

# Zero-Touch/Remote Inventory for Windows (RIW)

## Services Required for a Zero-Touch Windows Inventory

The following services are required (for Windows 7) to run on the target system to allow for zero-touch OS inventory:

- Remote Procedure Call
- Remote Procedure Call locator
- Remote Registry
- RPC endpoint mapper
- Windows management instrumentation
- Windows remote management

## Required Permissions for a Zero-Touch Inventory of Windows Devices

This chapter describes the permission standard for an Inventory Service User account (Domain/local) or a specified Group (domain/local) to have scanning permission for the Zero Touch Windows scanning technology.

### User Specifications

#### Option 1: Use a local Administrator account

This is the highest permission level.

The user account needs to be member of the local Administrators group. Local administrators usually have full permissions to WMI. Such user needs to be permitted and rolled out to every target device in scope.

#### Option 2: Create an Inventory Service user account with dedicated

## permissions

This is the least privilege approach.

For granting dedicated permissions to specified Service Users or Groups the following needs to be configured on every target device in scope:

## Group Membership

The User or Group needs to be member of the following groups:

| | |
|---|---|
| Performance Monitor Users | S-1-5-32-558 |
| Distributed COM Users | S-1-5-32-562 |
| Remote Management Users<br>(Not needed on Windows 7 and its counterparts) | S-1-5-32-580 |

**Note:**
Windows Domain Controllers use Domain Groups only. Therefore, the designated Inventory Service Users needs to be member of the corresponding Domain Groups.

## Required permissions on the target device

The following permissions on the WMI-Namespaces for the specified user or group should be granted:

## Common WMI permissions

| Namespace | Permissions | Inheritance |
|---|---|---|
| `\root` | • Enable Account<br>• Remote Enable | No |
| `\root\cimv2` | • Enable Account<br>• Execute Methods<br>• Remote Enable<br>• Read Security | Yes |

## WMI permissions for MS SQL Servers

| Namespace | Permissions | Inheritance |
|---|---|---|
| `\root\Microsoft\SqlServer` | • Enable Account<br>• Execute Methods | Yes |

| Namespace | Permissions | Inheritance |
|---|---|---|
| | • Remote Enable<br>• Read Security | |

### WMI permissions for MS SQL Server version <= 2000

| Namespace | Permissions | Inheritance |
|---|---|---|
| `\root\Microsoft\SqlServer\MSSQL_Server` | • Enable Account<br>• Execute Methods<br>• Remote Enable<br>• Read Security | Yes |
| `\root\Microsoft\SqlServer\MSSQL_RegistrySetting` | • Enable Account<br>• Execute Methods<br>• Remote Enable<br>• Read Security | Yes |

### WMI permissions for MS SQL Server version >= 2005

| Namespace | Permissions | Inheritance |
|---|---|---|
| `\root\Microsoft\SqlServer\ComputerManagementXX` (XX is the major version number of SQL Server) | • Enable Account<br>• Execute Methods<br>• Remote Enable<br>• Read Security | Yes |

### WMI permissions for Hyper-V

| Namespace | Permissions | Inheritance |
|---|---|---|
| `\root\virtualization` | • Enable Account<br>• Execute Methods<br>• Remote Enable<br>• Read Security | Yes |
| `\root\virtualization\v1` | • Enable Account<br>• Execute Methods<br>• Remote Enable<br>• Read Security | Yes |

| Namespace | Permissions | Inheritance |
|---|---|---|
| \root\virtualization\v2 | • Enable Account<br>• Execute Methods<br>• Remote Enable<br>• Read Security | Yes |
| \root\MSCluster | • Enable Account<br>• Execute Methods<br>• Remote Enable<br>• Read Security | Yes |

## Locale group for Hyper-V

For a full Hyper-V inventory to work, it is necessary that the inventory user is in the group on all Hyper-V hosts:

- Hyper-V Administrators

## Permissions for Windows Services

To get a full inventory including WIndows Services, the user needs to have the following permissions:

- QueryStatus
- QueryConfig
- Interrogate
- EnumerateDependents
- Start
- ReadPermissions

For full SQL Details, it's as well needed to have the described rights for Windows Services.

## Local Firewall rule

WMI connection needs to be allowed on the device that is about to be scanned.

Example command for Windows Firewall:

- `"netsh.exe advfirewall firewall set rule group="windows management instrumentation (wmi)" new enable=yes"`

## WMI Service Restart

The WMI Service (Windows Management Instrumentation) needs to be restarted in order to apply changes of WMI permissions.

# Script Template to Grant Permissions for a Zero-Touch Inventory of Windows Devices

## Description:

The Powershell script template can create a local user or use an existing Domain-User and set all rights and permissions required for Zero-Touch inventory – following the least-privilege-approach supported by RayVentory.
Therefore, the script will perform the following tasks:

- Starting WMI-Service (winmgmt) in case it is not running
- (Optional:) Creates local user account
- The designated domain or local user will be added to the following built-in groups: „Performance Monitor Users“, „Distributed COM Users“, „Remote Management Users“ (depending on Windows version)
- The user will be added to the necessary namespaces of WMI with read-permissions, only
- In addition, user will be enabled to read all existing Windows services properties
- (Optional:) Restart the "Windows Management Instrumentation" service (winmgmt) In addition, all depending services will be checked and restarted as well
- (Optional:) Add local firewall rule for allowing remote WMI access

## Prerequisites:

The following prerequisites are required before the script template can be executed:

- If the target user is an existing AD-User, the following information should be known:
  - Account name and domain of the user
  - (Optional:) The SID of the user (then AD-lookup will be skipped)
- If the target user is a local user, the following information should be known:
  - The name of the user
  - The password of the user (if user doesn’t exist)
  - (Optional:) The SID of the user
- Furthermore, the script must be executed with administrative privileges to overcome UAC limitations; just using an account who is member of the Administrator group is not enough!

## Limitations:

We are aware of incomplete data of the zero-touch inventory when using this script on Windows 2008 R2 and Windows 7. Due to WMI API, the granted permissions are not sufficient for the named operating systems. On these systems, attributes of “Win32_DiskDrive”, “Win32_CDROMDrive” and “Win32_NetworkAdapter” may be missed or limited.
Raynet recommends checking the inventory data from those devices. If some information is required but not available due to limitations, the workaround is to add the user to the local administrator group.

## Preparation:

Before changing and executing the template script, we recommend that you verify and test the final version before deploying it to all Windows computers and servers.

Open the script using the Powershell editor or a text editor (e.g Notepad++) and navigate to the section [CmdletBinding()].

**Be aware:**
The log-file path must exist otherwise errors might occur. The log file with the specific name ("`RIW_CreateUser_<Username>.log`") will be created in "`c:\temp`".

Check if you want to immediately restart of the service **Windows Management Instrumentation**

- If `$RestartService = "True"`, the service "winmgmt" and every dependent service will be restarted.

**Be aware:**
Some permission changes are effective after a restart of the service or after the next Windows restart.

For using an existing Domain-User:

1. Replace `$UserName = "UserName"` with the name of the user and `$Domain = "Domain"` with the AD domain name (Netbios or DNS name).
2. Replace `$SID = $null` and add the SID of the domain user account as string (reduces runtime).
3. Set "$CreateUser"to"False" (not recommended to create a domain user).

For a local user:

1. Replace "`UserName`" with the name of the user.
2. Let domain empty ("") or "." to make sure the user is a local user.
3. If the user should be created, set `"CreateUser"` to `"True"` and enter the password under `"UserPass"`.

Common Settings:

- Verify or change `$LogPath = "c:\Temp\"` for the log-file
- Set `$SetServicePermissions = $false` to `$true` for collecting Windows service attributes
- Set `$SetWmiFirewallRules = $false` to `$true` in case local firewall might block remote WMI access
- Set $RestartService = $false to $true to update WMI permissions immediately

## Execution:

Ensure that you have administrator privilege by starting "Powershell as Administrator" or "Cmd as Administrator" (and start "powershell" from within the cmd shell).

PS: "`<PathToScript>\RIW_CreateUser_LocalExecution_v1.3.ps1`"

## Troubleshooting:

For further information, check the log file and Powershell messages.

**Change History:**

- 1.0 - Original script release
- 1.1 - Domain value support for both DNS and NETBIOS names
- 1.2 - Change NET LOCALGROUP sequence to respect Domain and failover for conflicts with localized group authorities - Start WMI Service including dependent services, if WMI Service is not running (mandatory)
- 1.3 - Multilanguage group names parsing added

The former VB-Script template is not available anymore. If needing VBS support, contact Raynet consulting or your Sales representative.

### RIW_CreateUser_LocalExecution_v1.3.ps1

```
<#
.SYNOPSIS
This script is used to fully configure an User on the target Client
for RayVentory ZeroTouch Inventory Method.
Requires Powershell 2.0 or higher in order to work.
#


.DESCRIPTION
This script performs a set of actions on a target system to make
sure that ZeroTouch inventory method will work for specified
credentials.
Local or AD user credentials can be used during script execution.
Following changes are made to the system by default:
 - User being added to specific local groups if those are existing
on a target platform.
   (Performance Monitor Users, Distributed COM Users, Remote
Management Users)
 - User is being granted read/execute permissions on specific WMI
namespaces.
   (root/cimv2,root/virtualization,root/MSCluster,root/Microsoft/
SqlServer,root/MicrosoftSqlServer)


Following changes are made to the system if specific flag is set:
 - Creation of a local user with provided name/password (flag "-
CreateUser"). Required if provided user credentials are not
existing.
 - Grant user service read permissions (flag "-
```

```
SetServicePermissions"). Required to read a list of services using
ZeroTouch.
 - Set WMI firewall rules (flag "-SetWmiFirewallRules"). Enables
windows integrated firewall rules for WMI.
   (netsh.exe advfirewall firewall set rule group="windows
management instrumentation (wmi)" new enable=yes)
 - Restart required services (flag "-RestartService"). Will restart
WMI service and it's dependencies. Without this flag a reboot will
be required.

.PARAMETER CreateUser
Set to create a new user locally. Will be ignored if user already
exists.

.PARAMETER UserName
Enter Username here.

.PARAMETER UserPassword
Enter user password here. If "-CreateUser" is set then a password
is required.

.PARAMETER Domain
Enter User-Domain here. Leave empty or "." if you want to use local
machine.

.PARAMETER SID
Enter SID of the User here. If empty, the SID will be searched for
provided domain\user combination.

.PARAMETER LogPath
The path to a directory where the log file will be written
(default: "c:\temp\")

.PARAMETER SetServicePermissions
Flag. Defines if service permissions will be set for a user.

.PARAMETER SetWmiFirewallRules
Flag. Defines if firwall rules will be enabled for a user.
```

```
.PARAMETER RestartService
Flag. Defines if firwall rules will be enabled for a user

.EXAMPLE
Example with local user creation:
.\RIW_CreateUser_LocalExecution.ps1 -UserName "riwUser" -
UserPassword "Password123" -Domain "." -CreateUser -
SetServicePermissions -SetWmiFirewallRules -RestartService

Example for existing AD user:
RIW_CreateUser_LocalExecution.ps1 -UserName "m.mustermann" -Domain
"ORGANIZATION" -SetServicePermissions

.NOTES
General notes

1.0 - Original script release
1.1 - Domain value support for both DNS and NETBIOS names
1.2 - Change NET LOCALGROUP sequence to respect Domain and failover
for conflicts with localized group authorities
    - Start WMI Service including dependend services, if WMI
Service is not running (mandatory)
1.3 - Multilanguage group names parsing added

.LIMITATIONS
- If user and computer domain are different, then a domain trust
must exist, otherwise permissions cannot be set
#>

[CmdletBinding()]
param (
    [string]
    $UserName = "UserName",
    [string]
    $Domain = "Domain",
    [string]
```

```
    $SID = $null,
    [switch]
    $CreateUser = $false,
    [string]
    $UserPassword = $null,
    [string]
    $LogPath = "c:\Temp\",
    [switch]
    $SetServicePermissions = $false,
    [switch]
    $SetWmiFirewallRules = $false,
    [switch]
    $RestartService = $false
)

### Global variables

# The version of a script
[string]$Global:ScriptVersion = "1.3"

# Supported log levels: "ERROR","WARN","INFO","DEBUG","VERBOSE"
# This level defines the serity message need to have in order to be
written into a log file
[string]$Global:LogLevel = "INFO"

# Script global variables. Set during execution of the script.
# Please don't change values here.
[System.Collections.Generic.List[string]]$Global:UserGroups
[System.Collections.Generic.List[string]]$Global:WmiNamespaces
[string]$Global:ComputerName
[string]$Global:WorkingDomain
[string]$Global:UserSID
[string]$Global:LogFile

# Logging configuration. Please change "LogLevel" to manipulate log
output.
$currentLogLevel = $null
```

```
switch ($Global:LogLevel)
{
    "INFO" { $currentLogLevel = @("ERROR","WARN","INFO") }
    "WARN" { $currentLogLevel = @("ERROR","WARN") }
    "DEBUG" { $currentLogLevel = @("ERROR","WARN","INFO","DEBUG") }
    "ERROR" { $currentLogLevel = @("ERROR") }
    "VERBOSE" { $currentLogLevel =
@("ERROR","WARN","INFO","DEBUG","VERBOSE") }
    Default {}
}

#################################################################
# Writing messages to console and file with different severities.
# As it is visible from tha validation parameter, following log
levels are supported:
# 'INFO', 'WARN', 'DEBUG', 'ERROR', 'VERBOSE'
# Example: Log -message "Text" -severity INFO
# Default severity: INFO
#################################################################
function Log {
    param (
        [string]$message,
        [ValidateSet('INFO', 'WARN', 'DEBUG', 'ERROR', 'VERBOSE')]
        [string]$severity = 'INFO'
    )

    if($currentLogLevel -contains $severity)
    {
        $dateTime = Get-Date -Format "yyyy.MM.dd HH:mm:ss"

        $output = $dateTime + " [" + $severity + "]" + ": " +
$message

        switch ($severity) {
            "INFO" { Write-Host $output }
            "WARN" { Write-Warning $output }
            "DEBUG" { Write-Debug $output }
```

```
                "ERROR" { Write-Error $output }
                "VERBOSE" { Write-Verbose $output }
                Default { Write-Host $output}
            }

            $output | Out-File -FilePath $Global:LogFile -Encoding utf8
-Append
        }
}

#####################################################################
# Exiting the script should always be done using this function.
# Takes single parameter - exit code that will be returned.
#####################################################################
function ExitScript {
    param ([int]$exitCode = 0)

    $exitMessage = "Exiting script with exit code $exitCode"
    if($exitCode -ne 0)
    {
        Log -severity WARN -message "$exitMessage"
    }
    else {
        Log $exitMessage
    }

    Log "********************* Footer ********************`n"

    exit $exitCode
}

#####################################################################
# Used to execute script steps with nice output wrapping.
#####################################################################
function WrapStepBlock {
    param (
```

```
        [Parameter(Mandatory=$true)]
        [scriptblock]$stepBody,
        [Parameter(Mandatory=$true)]
        [string]$stepName)

        Log "------------------------------------------------"

        $stepLength = $stepName.Length
        [int]$leftLength = (48 - $stepLength) / 2

        if($stepLength%2 -eq 1)
        {
            $leftLength -=1
            $sufix = $('#' * $leftLength) + "#"
        }
        else {
            $sufix = $('#' * $leftLength)
        }
        $prefix = $('#' * $leftLength)

        Log "$prefix $stepName $sufix"
        try {
            if($currentLogLevel -contains 'DEBUG')
            {
                $out = Invoke-Command $stepBody
                log -severity DEBUG -message $out
            }
            else
            {
                $stepBody.Invoke()
            }
        }
        finally {
            Log "###################### Done
######################"
        }
```

```
}

#################################################################
# Required by "SetWmiPersmission" method. Converts human friendly
# names for permissions into access mask flags.
#################################################################
Function Get-AccessMaskFromPermission($permissions) {
    $WBEM_ENABLE            = 1
            $WBEM_METHOD_EXECUTE = 2
            $WBEM_FULL_WRITE_REP    = 4
            $WBEM_PARTIAL_WRITE_REP              = 8
            $WBEM_WRITE_PROVIDER    = 0x10
            $WBEM_REMOTE_ACCESS     = 0x20
            $WBEM_RIGHT_SUBSCRIBE = 0x40
            $WBEM_RIGHT_PUBLISH      = 0x80
    $READ_CONTROL = 0x20000
    $WRITE_DAC = 0x40000

    $WBEM_RIGHTS_FLAGS =
$WBEM_ENABLE,$WBEM_METHOD_EXECUTE,$WBEM_FULL_WRITE_REP,`
        $WBEM_PARTIAL_WRITE_REP,$WBEM_WRITE_PROVIDER,$WBEM_REMOTE_A
CCESS,`
        $READ_CONTROL,$WRITE_DAC
    $WBEM_RIGHTS_STRINGS =
"Enable","MethodExecute","FullWrite","PartialWrite",`
        "ProviderWrite","RemoteAccess","ReadSecurity","WriteSecurit
y"

    $permissionTable = @{}

    for ($i = 0; $i -lt $WBEM_RIGHTS_FLAGS.Length; $i++) {
        $permissionTable.Add($WBEM_RIGHTS_STRINGS[$i].ToLower(),
$WBEM_RIGHTS_FLAGS[$i])
    }

    $accessMask = 0
```

```
    foreach ($permission in $permissions) {
        if (-not
$permissionTable.ContainsKey($permission.ToLower())) {
            throw "Unknown permission: $permission`nValid
permissions: $($permissionTable.Keys)"
        }
        $accessMask += $permissionTable[$permission.ToLower()]
    }

    $accessMask
}

#################################################################
# Adds or deletes passed WMI permissions from passed WMI namespase.
# If "allowInherit" is set to "true" permissions will be propagated
to subcontainers.
#################################################################
Function SetWmiSecurity
{
    Param ( [parameter(Mandatory=$true,Position=0)][string]
$namespace,
    [ValidateSet("add","delete")]
    [parameter(Mandatory=$true,Position=1)][string] $operation,
    [ValidateSet("Enable","MethodExecute","FullWrite","PartialWrite
","ProviderWrite","RemoteAccess","ReadSecurity","WriteSecurity")]
    [parameter(Position=3)][string[]] $permissions = $null,
    [bool] $allowInherit = $false,
    [bool] $deny = $false)

    #$ErrorActionPreference = "Stop"

    $computerName = "."
    $remoteparams = @{ComputerName=$computerName}
    $invokeparams =
@{Namespace=$namespace;Path="__systemsecurity=@"} + $remoteParams

    $output = Invoke-WmiMethod @invokeparams -Name
GetSecurityDescriptor
```

```
    if ($output.ReturnValue -ne 0) {
        throw "GetSecurityDescriptor failed:
$($output.ReturnValue)"
    }

    $acl = $output.Descriptor
    $OBJECT_INHERIT_ACE_FLAG = 0x1
    $CONTAINER_INHERIT_ACE_FLAG = 0x2

    switch ($operation) {
        "add" {
            if ($null -eq $permissions) {
                throw "-Permissions must be specified for an add
operation"
            }
            $accessMask = Get-
AccessMaskFromPermission($permissions)

            $ace = (New-Object
System.Management.ManagementClass("win32_Ace")).CreateInstance()
            $ace.AccessMask = $accessMask
            if ($allowInherit) {
                $ace.AceFlags = $CONTAINER_INHERIT_ACE_FLAG
            } else {
                $ace.AceFlags = 0
            }

            $trustee = (New-Object
System.Management.ManagementClass("win32_Trustee")).CreateInstance(
)
            $trustee.SidString = $Global:UserSID
            $ace.Trustee = $trustee

            $ACCESS_ALLOWED_ACE_TYPE = 0x0
            $ACCESS_DENIED_ACE_TYPE = 0x1

            if ($deny) {
```

```
                    $ace.AceType = $ACCESS_DENIED_ACE_TYPE
                } else {
                    $ace.AceType = $ACCESS_ALLOWED_ACE_TYPE
                }

                $acl.DACL += $ace.psobject.immediateBaseObject
            }

            "delete" {
                if ($null -ne $permissions) {
                    throw "Permissions cannot be specified for a delete
operation"
                }

                [System.Management.ManagementBaseObject[]]$newDACL =
@()
                foreach ($ace in $acl.DACL) {
                    if ($ace.Trustee.SidString -ne $Global:UserSID) {
                        $newDACL += $ace.psobject.immediateBaseObject
                    }
                }

                $acl.DACL = $newDACL.psobject.immediateBaseObject
            }

            default {
                throw "Unknown operation: $operation`nAllowed
operations: add delete"
            }
        }

        $setparams =
@{Name="SetSecurityDescriptor";ArgumentList=$acl.psobject.immediate
BaseObject} + $invokeParams

        $output = Invoke-WmiMethod @setparams
        if ($output.ReturnValue -ne 0) {
```

```
        throw "SetSecurityDescriptor failed:
$($output.ReturnValue)"
    }
}

#################################################################
# Used by "StartRequiredServices" function. Collects the list
# of all dependent services recursivly, which are not running
#################################################################
function CollectStoppedDependentServices([string]$serviceName)
{
    $resultCollection = New-Object
System.Collections.Generic.HashSet[string]
    $currentServices = Get-Service $serviceName -DependentServices
    foreach($depService in $currentServices)
    {
        if([string]::IsNullOrEmpty($depService.Name))
        {
            continue
        }

        if($depService.Status -ne
[System.ServiceProcess.ServiceControllerStatus]::Running)
        {
            $additional = CollectStoppedDependentServices
$depService.Name
            foreach ($item in $additional) {
                $dummy = $resultCollection.Add($item)
            }
            $dummy = $resultCollection.Add($depService.Name)
        }
    }
    return $resultCollection
}

#################################################################
# Used by "RestartRequiredServices" function. Collects the list
```

```
# of all running dependent services recursively.
#################################################################
function CollectRunningDependentServices([string]$serviceName)
{
    $resultCollection = New-Object
System.Collections.Generic.HashSet[string]
    $currentServices = Get-Service $serviceName -DependentServices
    foreach($depService in $currentServices)
    {
        if([string]::IsNullOrEmpty($depService.Name))
        {
            continue
        }

        if($depService.Status -eq
[System.ServiceProcess.ServiceControllerStatus]::Running)
        {
            $additional = CollectRunningDependentServices
$depService.Name
            foreach ($item in $additional) {
                $dummy = $resultCollection.Add($item)
            }
            $dummy = $resultCollection.Add($depService.Name)
        }
    }
    return $resultCollection
}

# General logic

#################################################################
# Outputting log header with basic script information
#################################################################
function StartScript
{
    $Global:LogFile =
[System.IO.Path]::Combine($LogPath,"RIW_CreateUser_$UserName.log")
```

```
    try {

        if(-not(Test-Path $LogPath))
        {
            $dummy = New-Item -ItemType Directory -Path $LogPath
        }

        $passwordPlaceholder = "<empty>"
        if(-not [string]::IsNullOrEmpty($UserPassword))
        {
            $passwordPlaceholder = "<hidden. not empty>"
        }

        Log "********************* Header *********************"
        Log "##############################################################"
        Log "RIW create user script v.$ScriptVersion"
        Log "##############################################################"

        Log ""
        Log " ____                          __  __                    __
                    "
        Log "/\  _`\                       /\ \/\ \                  /\
\__                      "
        Log "\ \ \L\ \     __      __  __\ \ \ \ \     __    ___\ \
,_\   ___   _ __   __  __     "
        Log " \ \ ,  /   /'__`\  /\ \/\ \\ \ \ \ \  /'__`\/' _ `\ \
\/  / __`\/\`'__\/\ \/\ \    "
        Log "  \ \ \\ \ /\ \L\.\_\ \ \_\ \\ \ \_/ \/\  __//\ \/\ \
\ \_/\ \L\ \ \ \/ \ \ \_\ \   "
        Log "   \ \_\ \_\ \__/.\_\\/`____ \\ `\___/\ \____\ \_\ \_\
\__\ \____/\ \_\  \/`____ \  "
        Log "    \/_/\/ /\/__/\/_/ `/___/> \`\/__/  \/____/\/_/\/_/
\/__/\/___/  \/_/   `/___/> \"
        Log "                           /\___/
                    /\___/"
        Log "                           \/__/
                    \/__/ "
        Log ""
```

```
        Log "Startup arguments:"
        Log "UserName: $UserName"
        Log "UserPassword: $passwordPlaceholder"
        Log "Domain: $Domain"
        Log "SID: $SID"
        Log "LogPath: $LogPath"
        Log "CreateUser: '$CreateUser'"
        Log "SetServicePermissions: '$SetServicePermissions'"
        Log "SetWmiFirewallRules: '$SetWmiFirewallRules'"
        Log "RestartService: '$RestartService'"
        Log "Log level is set to $Global:LogLevel"
    }
    catch {
        $errorMessage = $_.Exception.Message
        Write-Error "Logger failed to initialize with error:
$errorMessage"
        ExitScript 1
    }
}


#-----------------------------------------------------------------
# Logic steps
#-----------------------------------------------------------------

###################################################################
# Additional parameter validation and potentially platform checks.
###################################################################
function CheckPrerequisites {
    Log -message "Checking for conditions to start the execution of
logic." -severity DEBUG
    if([string]::IsNullOrEmpty($UserName))
    {
        Log -message "User name is empty. The name is required by a
script." -severity ERROR
        ExitScript 1
```

```
    }

    if($CreateUser.IsPresent)
    {
        if([string]::IsNullOrEmpty($UserPassword))
        {
            Log -message "User password is empty. The password is
required if 'CreateUser' flag is set." -severity ERROR
            ExitScript 1
        }
    }

    Log -message "Script is ready to be executed." -severity INFO
}

#####################################################################
# Preparing required data for a script execution:
# - resolving localized group names
# - preparing a list of WMI namespaces to be processed
# - resolving environment variables
#####################################################################
function PrepareWorkingParameters {
    # Groups where the user will be added. Default is "Performance
Monitor Users","Distributed COM Users","Remote Management Users"

     # Get the name of the Groups by SID

          # Performance Monitor Users :   S-1-5-32-558
          # Distributed COM Users:        S-1-5-32-562
        # Remote Management Users:        S-1-5-32-580
    $groupSids = @("S-1-5-32-558","S-1-5-32-562","S-1-5-32-580")
    $groupNames = New-Object -TypeName
System.Collections.Generic.List[string]

    Log "Preparing a list of required groups"
    foreach ($item in $groupSids)
    {
```

```
        Log -severity DEBUG -message "Processing SID " + $item
        $objSID = New-Object
System.Security.Principal.SecurityIdentifier($item)
        try {
            $objUser =
$objSID.Translate( [System.Security.Principal.NTAccount])
        }
        catch {
            continue
        }
        $groupName = $objUser.Value
        if($groupName.IndexOf('\') -ge 0)
        {
            $groupName = $groupName.Split('\')[1];
        }

        $dummy = $groupNames.Add($groupName)
        Log -message "Resolved required group SID $item to name
$groupName" -severity DEBUG
    }

    $Global:UserGroups = $groupNames

    Log "Preparing a list of WMI namespaces"
    $Global:WmiNamespaces = New-Object -TypeName
System.Collections.Generic.List[string]
    $Global:WmiNamespaces = @("root/cimv2","root/
virtualization","root/MSCluster","root/Microsoft/SqlServer","root/
MicrosoftSqlServer")

    $Global:ComputerName = $env:COMPUTERNAME
    Log -message "Resolved computer name: ${Global:ComputerName}"

    if([string]::IsNullOrEmpty($Domain) -or "." -eq $Domain)
    {
        $Global:WorkingDomain = $Global:ComputerName
        Log "Domain is empty. Will use ComputerName
'$Global:WorkingDomain' as domain name"
```

```
    }
    else {
        $Global:WorkingDomain = $Domain
        Log "Domain is set to '$Global:WorkingDomain'"
    }
}

#################################################################
# Logic for creating local user account.
# By default this step is skipped.
# '-CreateUser' parameter is required to be set in order to run the
logic.
#################################################################
function CreateLocalUser {
    if (-not $CreateUser.IsPresent -or $CreateUser -eq $false)
    {
        Log "'-CreateUser' flag is not present or is set to to
false."
        Log "Skipping this step."
    }
    elseif ([string]::IsNullOrEmpty($Global:WorkingDomain) -or "."
-eq $Global:WorkingDomain -or $Global:ComputerName -eq
$Global:WorkingDomain)
    {
        $isUserExists = $false
        Log -severity DEBUG -message "Checking if user with given
name exists"
        $existingUsers = Get-WmiObject -Query "Select * from
Win32_UserAccount where LocalAccount = True AND Name = '$UserName'"

        $existingUser = $null
        foreach ($user in $existingUsers) {
            if($UserName -eq $user.Name)
            {
                $existingUser = $user
                break
            }
```

```
        }

        if($null -eq $existingUser)
        {
            Log "User is not found. Will be created with following
name : $UserName"
            $output = & NET @("USER", "$UserName","$UserPassword",
"/ADD", "/fullname:$UserName", "/expires:never", "/Y")
            $exitCode = $LASTEXITCODE
            Log -severity VERBOSE -message $output
            if($exitCode -ne 0)
            {
                throw New-Object System.ApplicationException
"Adding group has failed with exit code: $exitCode"
            }
        }
        else
        {
            Log "User is found. It will be reused."
            $isUserExists = $true
        }
    }
    else
    {
        Log -message "CreateUser flag is set but the domain name is
not local. Skipping creating local user."
    }
}

#################################################################
# Logic for resolving the SID for provided user name/domain
paramters.
#################################################################
function SetUserSID
{
    if([string]::IsNullOrEmpty($SID))
    {
```

```
        Log "The SID is empty. SID lookup will be executed."
        if($Global:ComputerName -eq $Global:WorkingDomain)
        {
            Log -severity DEBUG -message "Local user lookup
execution"
            Log -severity DEBUG -message "Domain:
$Global:WorkingDomain User: $UserName"
            $query = "Select * from Win32_UserAccount where
LocalAccount = True AND Caption = '${Global:WorkingDomain}\
\$UserName'"
            $existingUsers = Get-WmiObject -Query $query
            if (-not ($existingUsers -is [array]))
            {
                $existingUsers = @($existingUsers)
            }

            if($existingUsers.Count -gt 0)
            {
                $Global:UserSID = $existingUsers[0].SID

                if ($Global:UserSID.StartsWith("S-1-5-21"))
                {
                    Log "Found local user $UserName with SID
$Global:UserSID"
                }
                else {
                    Log -severity WARN -message "User and/or SID
could not be found on the local machine. Exiting."
                    ExitScript -exitCode 2
                }
            }
            else
            {
                Log -severity WARN -message "User and/or SID could
not be found on the local machine. Exiting."
                ExitScript -exitCode 2
            }
```

```
        }
        else
        {
            Log -severity DEBUG -message "AD user lookup execution"
            $existingUsers = Get-WmiObject -Query "Select * from
Win32_UserAccount where Caption = '${Global:WorkingDomain}\
\$UserName'"
            if($null -eq $existingUsers)
            {
                Log -severity DEBUG -message "User/Domain
combination was not found. Attemping search by UserName"
                $existingUsers = Get-WmiObject -Query "Select *
from Win32_UserAccount where Name = '$UserName'"
                if($null -ne $userCandidates)
                {
                    Log -severity DEBUG -message "User candidates
were found"
                }
            }

            if (-not ($existingUsers -is [array]))
            {
                $existingUsers = @($existingUsers)
            }

            if($existingUsers.Count -gt 0)
            {
                $Global:UserSID = $existingUsers[0].SID
                $Global:WorkingDomain = $existingUsers[0].Domain
                Log "Found  user $Global:WorkingDomain\\$UserName
with SID $Global:UserSID"
            }
            else {
                Log -severity WARN -message "User and/or SID could
not be found. Exiting."
                ExitScript -exitCode 2
            }
```

```
        }

    }
    else {
        Log "The SID is set from command line. Provided value will
be used:"
        $Global:UserSID = $SID
        Log "$Global:UserSID"
    }
}

#################################################################
# Adding user to required local groups.
# The set of groups is defined in step "PrepareWorkingParameters"
#################################################################
function AddUserToLocalGroups
{
    foreach ($groupName in $Global:UserGroups)
    {
        $LASTEXITCODE = 0
        Log "Adding user $WorkingDomain\$UserName to group
$groupName"
        $output = & NET @("LOCALGROUP", "$groupName",
"$Global:WorkingDomain\$UserName", "/add")
        $exitCode = $LASTEXITCODE
        Log -severity VERBOSE -message $output
        if($exitCode -ne 0)
        {
            if($exitCode -eq 2)
            {
                Log "The specified account name is already a member
of the group."
            }
            else {
                throw New-Object System.ApplicationException
"Adding group has failed with exit code: $exitCode"
            }
```

```
        }
        # fail over in case of local group name requires just the
name without contect
        $i = $groupname.IndexOf("\")
        if ($i -gt 0)
        {
            $groupName = $groupname.Substring($i+1)
            Log -severity VERBOSE -message $Groupname
            $LASTEXITCODE = 0
            Log "Adding user $WorkingDomain\$UserName to group
$groupName"
            $output = & NET @("LOCALGROUP", "$groupName",
"$WorkingDomain\$UserName", "/add")
            $exitCode = $LASTEXITCODE
            Log -severity VERBOSE -message $output $exitCode
            if($exitCode -ne 0)
            {
                throw New-Object System.ApplicationException
"Adding group has failed with exit code: $exitCode"
            }
        }
    }
}

#################################################################
# Granting WMI permissions for defined namespaces.
# List of namespaces is defined in step "PrepareWorkingParameters".
# As it's visible from the command, the current list of permissions
is:
# - Enable,MethodExecute,RemoteAccess,ReadSecurity
#################################################################
function GrantUserWmiPermissions
{
    $failedNamespaces = 0

    try {
```

```
        Log "Setting 'RemoteAccess' permission on Root without
inheritance."
        SetWmiSecurity -namespace "Root" -operation add -
permissions RemoteAccess
    }
    catch {
        $message = $_.Exception.Message
        Log -message $message -severity WARN
    }

    foreach ($nameSpace in $Global:WmiNamespaces)
    {
        try {
            Log "Setting
'Enable,MethodExecute,RemoteAccess,ReadSecurity' permissions on
$nameSpace with inheritance."
            SetWmiSecurity -namespace "$nameSpace" -operation add -
permissions Enable,MethodExecute,RemoteAccess,ReadSecurity -
allowInherit $true
        }
        catch {
            $message = $_.Exception.Message
            Log -message $message -severity WARN
            $failedNamespaces++
        }
    }

    if($failedNamespaces -eq $Global:WmiNamespaces.Count)
    {
        Log -severity ERROR -message "Granting permissions for WMI
namespaces failed. Not enough permissions or WMI repository is
corrupted on this machine."
        throw "Grant user wmi permissions failed."
    }
}

#################################################################
# Setting user permissions to read (list) services.
```

```
# This step is optional and enabled by using "-
SetServicePermissions" flag
#####################################################################
function GrantUserServicePermissions
{
    if($SetServicePermissions.IsPresent -and $SetServicePermissions
-eq $true)
    {
        # The permissions will be set here
        # Set of permissions: QueryStatus, QueryConfig,
Interrogate, EnumerateDependents, Start, ReadPermissions
        $UserPermissionsSddl = "D:
(A;CIOI;GRRCRPLCLOFRKR;;;$Global:UserSID)"
        $services = Get-WmiObject -Query "Select Name from
Win32_Service"

        if($null -eq $services)
        {
            Log -severity WARN -message "Can't read services using
WMI. Skipping step execution."
        }
        else
        {
            if(-not($services -is [array]))
            {
                $services = @($services)
            }

            $convertedArray = New-Object
System.Collections.ArrayList(,$services)
            $convertedArray.Add(@{Name="scmanager"})

            foreach ($service in $convertedArray)
            {
                $serviceName = $service.Name
                Log "Processing $serviceName"
                [string]$output =  & sc.exe @("sdshow",
"$serviceName")
```

```
                $exitCode = $LASTEXITCODE
                Log -severity VERBOSE -message $output

                if($exitCode -ne 0)
                {
                    Log -severity WARN -message "Service SDDL
string could not be extracted"
                    continue
                }

                $output = $output -replace
[System.Environment]::NewLine, [string]::Empty
                $userSidIndex = $output.IndexOf("$Global:UserSID")
                if($userSidIndex -gt -1)
                {
                    Log "User already assigned to service
$serviceName"
                }
                else
                {
                    $newServiceSddl = $output -replace "D:",
$UserPermissionsSddl
                    Log "Adding user to service $serviceName"
                    $output = & sc.exe @("sdset", "$serviceName",
"$newServiceSddl")
                    $exitCode = $LASTEXITCODE
                    Log -severity VERBOSE -message $output

                    if($exitCode -ne 0)
                    {
                        Log -severity WARN -message "Setting new
permissions failed."
                        continue
                    }
                }
            }
        }
```

```
    }
    else
    {
        Log "Service permissions flag '-SetServicePermissions' is
not set. Skipping this step."
    }
}

#####################################################################
# Setting firewall rules to accept WMI connections.
# This step is optional and enabled by using "-SetWmiFirewallRules"
flag
#####################################################################
function SetWmiFirewallRules
{
    if($SetWmiFirewallRules.IsPresent -and $SetWmiFirewallRules -eq
$true)
    {
        Log "Executing 'netsh' to configure firewall for WMI"
        $output = & netsh.exe @("advfirewall","firewall","set",
"rule", "group=",'"@FirewallAPI.dll,-34251"',"new","enable=yes")
        $exitCode = $LASTEXITCODE
        Log -severity VERBOSE -message $output

        if($exitCode -ne 0)
        {
            Log -severity WARN -message "Setting firewall failed
with exit code: $exitCode"
        }
    }
    else
    {
        Log "Set WMI firewall rules flag '-SetWmiFirewallRules' is
not set. Skipping this step."
    }
}
```

```
#################################################################
# Restart WMI Service
# This step is optional and enabled by using "-RestartService" flag
#################################################################
function RestartRequiredServices
{
    if($RestartService.IsPresent -and $RestartService -eq $true)
    {
        Log -severity DEBUG -message "Getting service object and
it's dependencies"
        $wmiService = Get-Service Winmgmt
        $dependantServices = CollectRunningDependentServices
Winmgmt

        if($wmiService.Status -eq
[System.ServiceProcess.ServiceControllerStatus]::Running)
        {
            Log "WMI service is running. Performing forced stop."
            Stop-Service Winmgmt -Force
        }

        Log "Restarting WMI service."
        Start-Service Winmgmt

        Log "Restarting dependent services."
        foreach ($depService in $dependantServices)
        {
            if([string]::IsNullOrEmpty($depService))
            {
                continue
            }

            $currentServiceState = Get-Service $depService|Select-
Object -ExpandProperty Status

            if($currentServiceState -ne
[System.ServiceProcess.ServiceControllerStatus]::Running)
```

```
                {
                    Log "$depService ..."
                    Start-Service -Name $depService
                }
                else
                {
                    Log "Skipping $depService. It's already running."
                }
            }
        }
        else
        {
            Log "Restart services flag '-RestartService' is not set.
Skipping this step."
        }
}

#################################################################
# Check if WMI Service is running and start it (mandatory)
#################################################################
function StartRequiredServices
{
     Log -severity DEBUG -message "Check if WMI service is running
and start it"
        $wmiService = Get-Service Winmgmt

        if($wmiService.Status -ne
[System.ServiceProcess.ServiceControllerStatus]::Running)
        {
            Log "WMI service is not running. Performing start ..."
            Start-Service Winmgmt

            $dependantServices = CollectStoppedDependentServices
Winmgmt

            foreach ($depService in $dependantServices)
            {
```

```
                if([string]::IsNullOrEmpty($depService))
                {
                    continue
                }

                $currentServiceState = Get-Service $depService|
Select-Object -ExpandProperty Status

                if($currentServiceState -ne
[System.ServiceProcess.ServiceControllerStatus]::Running)
                {
                    Log "$depService ..."
                    Start-Service -Name $depService
                }
                else
                {
                    Log "Skipping $depService. It's already
running."
                }
            }

        }
        else
        {
            Log "WMI service already running. Skipping this step."
        }
}

#################################################################
# MAIN SECTION: Executing steps for setting required permissions
#################################################################

TRY
{
    StartScript
    WrapStepBlock -stepName "Check prerequisites" -stepBody
{ CheckPrerequisites }
```

```
    WrapStepBlock -stepName "Check WMI Service" -stepBody
{ StartRequiredServices }
    WrapStepBlock -stepName "Prepare working parameters" -stepBody
{ PrepareWorkingParameters }
    WrapStepBlock -stepName "Create local user" -stepBody
{ CreateLocalUser }
    WrapStepBlock -stepName "Set user SID" -stepBody { SetUserSID }
    WrapStepBlock -stepName "Add user to local groups" -stepBody
{ AddUserToLocalGroups }
    WrapStepBlock -stepName "Grant user WMI permissions" -stepBody
{ GrantUserWmiPermissions }
    WrapStepBlock -stepName "Grant user service permissions" -
stepBody { GrantUserServicePermissions }
    WrapStepBlock -stepName "Set WMI firewall rules" -stepBody
{ SetWmiFirewallRules }
    WrapStepBlock -stepName "Restart required services" -stepBody
{ RestartRequiredServices }
}
catch {
    $exceptionMessage = $_.Exception.Message
    $line = $_.InvocationInfo.ScriptLineNumber
    $offset = $_.InvocationInfo.OffsetInLine
    Log -severity WARN -message "Line: $line : $offset; Message:
$exceptionMessage"
    ExitScript 1
}

ExitScript 0
```

# Zero-Touch/Remote Inventory for Linux/Unix (RIU)

## Linux/UNIX Remote Execution Login Methods

### Installation Specifications

We are using the library **SSH.NET** which supports the following private key formats:

- RSA in OpenSSL PEM and ssh.com format

- DSA in OpenSSL PEM and ssh.com format
- ECDSA 256/384/521 in OpenSSL PEM format
- ECDSA 256/384/521, ED25519 and RSA in OpenSSH key format

Private keys can be encrypted using one of the following cipher methods:

- DES-EDE3-CBC
- DES-EDE3-CFB
- DES-CBC
- AES-128-CBC
- AES-192-CBC
- AES-256-CBC

For further information, please read the documentation for the library: https://github.com/sshnet/SSH.NET

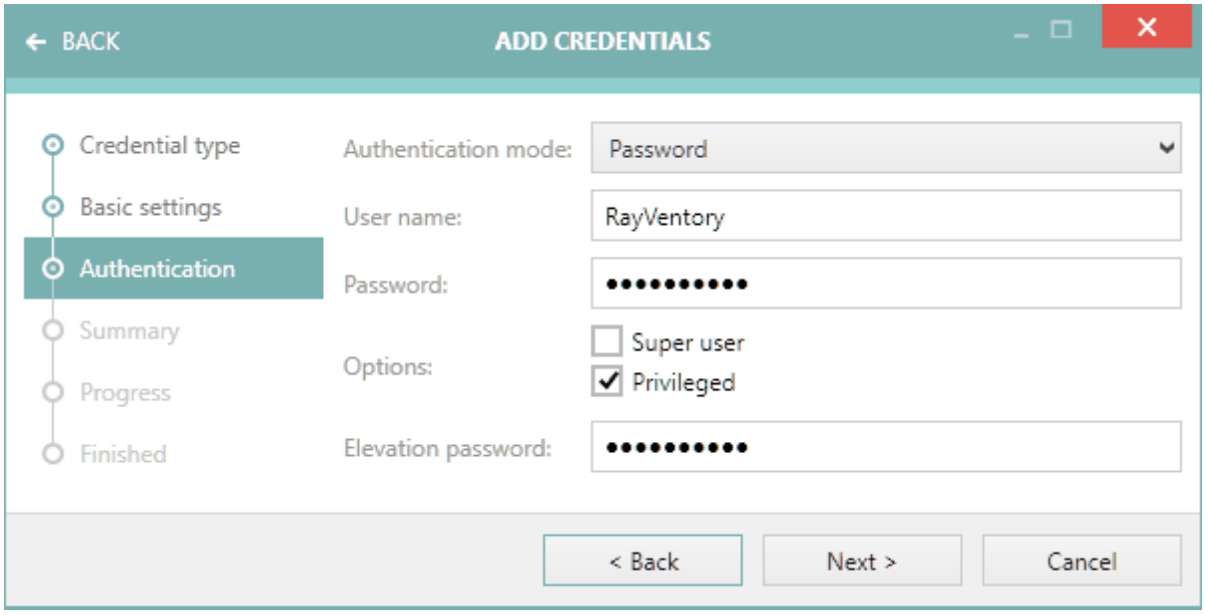
## Username and Password

The simplest way is to create a user and password combination:

1. Create a user (e.g. useradd RayVentory).
2. Set a password for the User (passwd RayVentory).
3. Set permissions like described below.
4. Add User to the Credentials Store of RVSE.

**Tip:**
Prefer the "privileged" option and add the elevation password.
Superusers like "root" are commonly not permitted to execute every command without elevated rights!

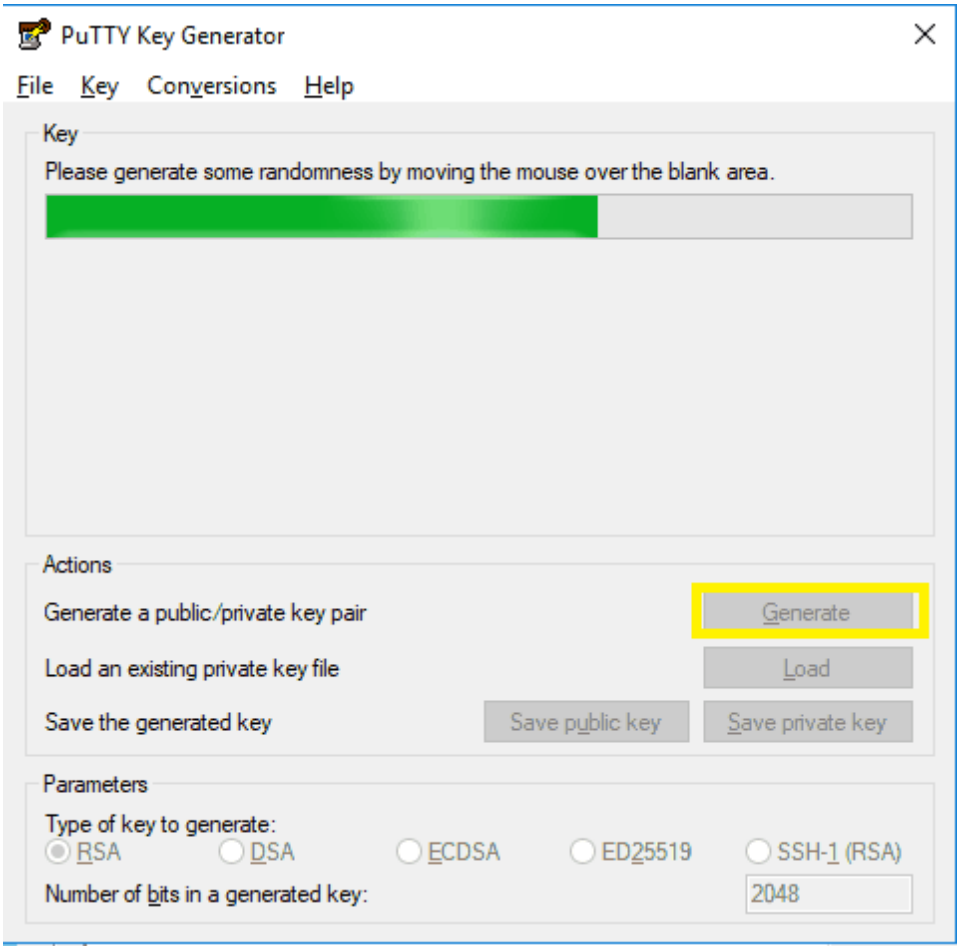
### RVSE:



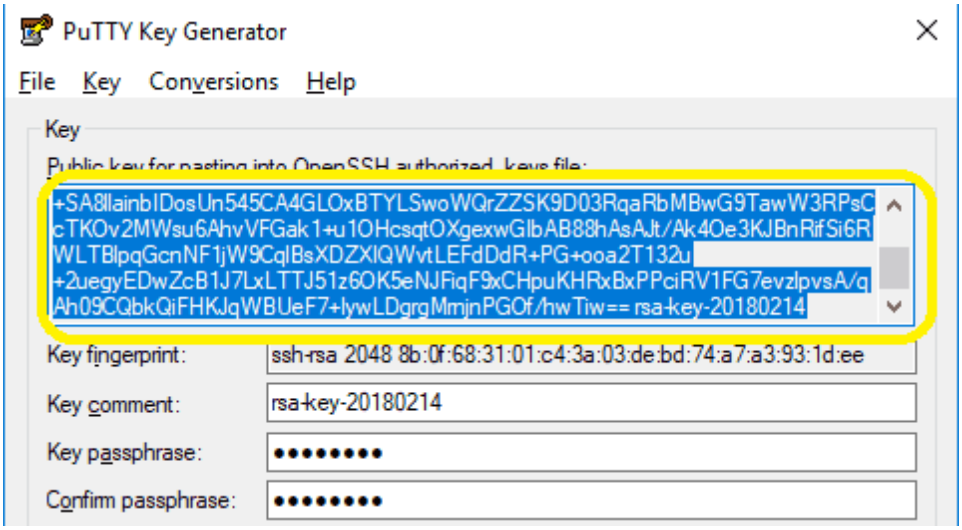

## SSH Key-Based Authentication

This description might need adoptions, depending on if using Windows or other platforms.

1. Login to Linux/Unix system with the designated user account used for RayVentory.
2. Create a local folder `.ssh` in the users home directory, if such does not exist already
3. If the file `./.ssh/`authorized_keys does not exist, create it.•
   To this file we need to add the public key later.
4. Make sure **sshd** configuration and service is set up properly.
5. Download `puttygen.exe` from https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html.
6. Check the download with an Antivirus tool.
7. Start `puttygen.exe` (screenshot taken from Windows).
8. Start generating a public key by selecting the **Generate** button.

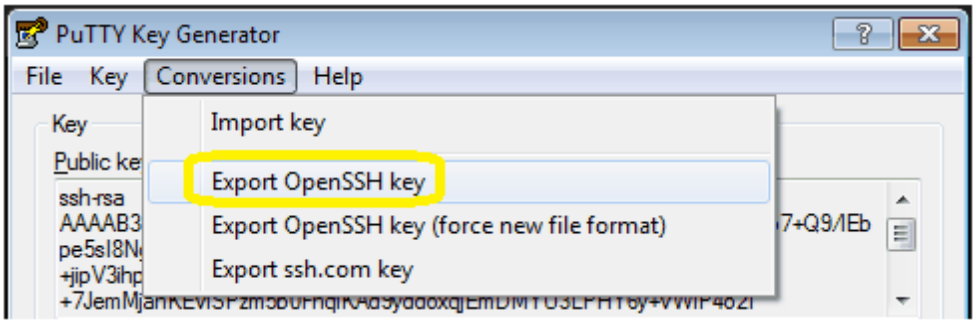



9. Move the mouse around to randomly generate the key.
10. Set a "Key passphrase" for the public key. This password is required later for changing the public key and generating private keys.
11. Save the public key as file to disk.
12. Copy all data within the public key field and add it to the `./ssh/authorized_keys` file within the user directory of the RayVentory user on the Linux or Unix machine. Section highlighted in the following screenshot:




**Tip:**
Prefer the "privileged" option and add the elevation password.
Superusers like "root" are commonly not permitted to execute every command without elevated rights!

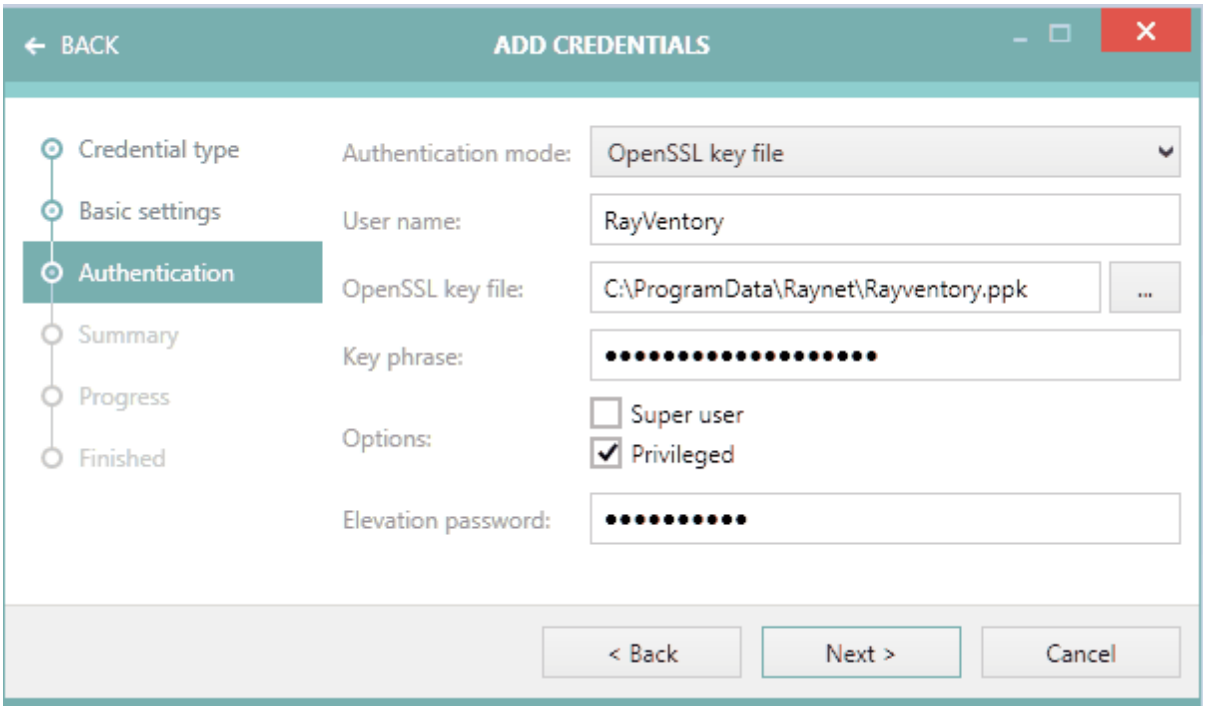
13.Export the private key as an **OpenSSH** key and save it.




14.Copy the file containing the "OpenSSH" private key to the RVSE server.

## Enabling RVSE Using the Private Key File

1. Start RVSE
2. Open the Credential Store and create an SSH•credential
3. Select Authentication method "Key file"
4. Add the Username, path of the "OpenSSH key file" and Key passphrase
5. Select option "Privileged" and enter the elevation password




## Alternatively, It Is Possible to Create the SSH-Keys Directly on a Linux/ Unix Machine

1. Create the SSH-Key with the following command:
   ssh-keygen -m PEM -t rsa -b 2048

```
administrator@linuxdocker:~/.ssh$ ssh-keygen -m PEM -t rsa -b 2048
Generating public/private rsa key pair.
Enter file in which to save the key (/home/administrator/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/administrator/.ssh/id_rsa.
Your public key has been saved in /home/administrator/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:ZH7ky9M03EY9GhAu1AxBfQYZleLEEkPtGgRJ9HYpDwM administrator@linuxdocker
The key's randomart image is:
+---[RSA 2048]----+
|      oE=OX=*..  |
|       .+ooXo+ . |
|        +B*++....|
|       +.+B+ oo .|
|        S =.+.o  |
|         + + o   |
|          + .    |
|           .     |
|                 |
+----[SHA256]-----+
```

2. Use the following command to add the public key to the authorized keys:

```
ssh-copy-id <user>@"IP address of the linux machine"
```

```
administrator@linuxdocker:~/.ssh$ ssh-copy-id administrator@192.168.69.84
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 3 key(s) remain to be installed -- if you are prompted now it is to install the new keys
administrator@192.168.69.84's password:

Number of key(s) added: 3

Now try logging into the machine, with:   "ssh 'administrator@192.168.69.84'"
and check to make sure that only the key(s) you wanted were added.
```

3. Copy the Private-Key file (file at `/home/<user>/.ssh`) onto your RVSE server.

### Deploy Public Key File to Linux/Unix Systems

After creating a new Public Key for SSH and completing tests with RVSE, the public key is ready for deployment to all Linux/Unix systems.

## Required Permissions for a Zero-Touch Inventory of Linux, UNIX, and Mac Devices

In the following, the permissions required for an Inventory Service Account used to perform a Zero-Touch Inventory of Linux, Unix, and Mac devices are needed in order to connect via SSH.

### Option 1: Sudoer

A sudoer without any restrictions on the command-lines is the simplest approach to enable RayVentory to execute all necessary commands and to read some folders and files. Such a service account needs to be added to each device, permitted by the sudoer's list and rolled out to all devices that will be targeted by this user account.

### Option 2: Account with Minimum Permissions

This option realizes a least-privilege approach. Permissions are described in the following tables covering all commands and files required for the Zero-Touch inventory. Such an approach requires named permissions on files and commands granted to the inventory service account which will access the target devices by SSH. Once the permissions have been set for each platform, the credentials and permissions need to be rolled out to all devices in the scope of scanning by Zero-Touch.

### Legend

| Symbol | Description |
|---|---|
| **X** | Command applies / File is read |
| **!** | Command applies / File is read regardless of the platform and is expected to fail or likely not to be present |

## List of Commands Which Do Not Need Privileges

| Subject | Platform | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Basic Support | | | | | | | | | Extended Support |
| | AIX | HP-UX | | Linux | | MacOS | | Solaris | | Linux |
| **Command** | POWER | PA-RISC | Itani-um | x86 | POWER | x86 | M1 | SPARC | x86 | ARM (nm) |
| grep | X | X | X | X | X | X | X | X | X | X |
| awk | X | X | X | X | X | X | X | X | X | X |
| model | | X | X | | | | | | | |
| mdSsum | X | ! | ! | X | X | X | X | ! | ! | X |
| dd | X | X | X | X | X | X | X | X | X | X |
| ls | X | X | X | X | X | X | X | X | X | X |

## List of Files Which Do Not Need Privileges

| Subject | Platform | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Basic Support | | | | | | | | | Extended Support |
| | AIX | HP-UX | | Linux | | MacOS | | Solaris | | Linux |
| **File** | POWER | PA-RISC | Itani-um | x86 | POWER | x86 | M1 | SPARC | x86 | ARM (nm) |
| /sys/class/net/(list directory) | | | | X | X | | | | | X |

## List of Commands Which Do Not Explicitly Require Privileged Rights

| Subject | Platform | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Basic Support | | | | | | | | | Extended Support |
| | AIX | HP-UX | | Linux | | MacOS | | Solaris | | Linux |
| **Command** | POWER | PA-RISC | Itani-um | x86 | POWER | x86 | M1 | SPARC | x86 | ARM (nm) |
| prtconf | X | | | | | | | X | X | |
| lsattr | X | | | | | | | | | |
| system_profiler | | | | | | X | X | | | |
| ioreg | | | | | | X | X | | | |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| smbios | | | | | | | | ! | X | |
| lscfg | X | | | | | | | | | |
| lparstat | X | | | ! | ! | | | | | ! |
| sysctl | | | | | | X | X | | | |
| print_manifest | | X | X | | | | | | | |
| zonename | | | | | | | | X | X | |
| eeprom | ! | | | ! | ! | | | X | X | ! |
| df | X | X | X | X | X | X | X | X | X | X |
| ioscan | | X | X | | | | | | | |
| diskinfo | | X | X | | | | | | | |
| smartctl | ! | | | ! | ! | ! | ! | X | X | ! |
| lsblk | ! | | | X | X | ! | ! | ! | ! | X |
| udevadm | ! | | | ! | ! | ! | ! | ! | ! | ! |
| blockdev | ! | | | X | X | ! | ! | ! | ! | X |
| iostat | | | | | | | | X | X | |
| fcinfo | | | | | | | | X | X | |
| find | X | X | X | X | X | X | X | X | X | X |
| sh | ! | ! | ! | ! | ! | ! | ! | ! | ! | ! |
| dladm | | | | | | | | ! | ! | |
| zoneadm | | | | | | | | X | X | |
| zonecfg | | | | | | | | X | X | |
| prctl | | | | | | | | X | X | |
| docker | ! | ! | ! | ! | ! | ! | ! | ! | ! | ! |
| ifconfig | X | X | X | ! | ! | X | X | X | X | ! |
| lanscan | | X | X | X | X | | | | | X |
| ip (addr) | | | | ! | ! | | | | | ! |
| vmstat | X | | | | | | | | | |
| svmon | X | | | | | | | | | |
| sw_vers | | | | | | X | X | | | |
| dmesg | | X | X | | | | | | | |
| swlist | X | | | | | | | | | |
| odmget | | X | X | | | | | | | |
| lsconf | X | | | | | | | | | |
| free | | | | X | X | ! | ! | | | X |
| cstm | | ! | ! | | | | | | | |
| kstat | | | | | | | | X | X | |

## List of Files Which Do Not Explicitly Require Privileged Rights

| Subject | Platform | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Basic Support | | | | | | | | | Extended Support |
| | AIX | HP-UX | | Linux | | MacOS | | Solaris | | Linux |
| File | POWER | PA-RISC | Itani-um | x86 | POWER | x86 | M1 | SPARC | x86 | ARM (nm) |
| /sys/class/dmi/id/chasis_vendor (accessed by cat | | | | X | | ! | | | ! | |
| /sys/class/dmi/id/bios_version | | | | X | | ! | | | ! | |
| /sys/class/dmi/id/chassis_serial | ! | ! | ! | X | ! | ! | ! | ! | ! | ! |
| /sys/class/dmi/id/bios_version | ! | ! | ! | X | ! | ! | ! | ! | ! | ! |
| /sys/class/dmi/id/chassis_vendor | ! | ! | ! | X | ! | ! | ! | ! | ! | ! |
| /etc/hostname.ce0 (accessed by cat) | | | | ! | ! | | | X | X | ! |
| /sys/class/dmi/id/product_name (accessed by cat) | ! | | | X | ! | | | ! | ! | ! |
| /sys/class/dmi/id/product_uuid (accessed by cat) | ! | | | X | ! | | | ! | ! | ! |
| /etc/passwd (accessed by cat) | | | | | | | | | | |
| beahomelist (accessed by cat) | ! | ! | ! | ! | ! | ! | ! | ! | ! | ! |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| registry.xml (accessed by cat) | ! | ! | ! | ! | ! | ! | ! | ! | ! | ! |
| /sys/class/net/<NICs>/speed (accessed by cat) | ! | | | X | X | | | | | X |
| /sys/class/net/<NICs>/address (listing directory by ls) | | | | X | X | | | | | X |

## List of Commands Which Deliver the Best Results With Privileged Rights

| Subject | Platform | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Basic Support | | | | | | | | | Extended Support |
| | AIX | HP-UX | | Linux | | MacOS | | Solaris | | Linux |
| Command | POWER | PA-RISC | Itani-um | x86 | POWER | x86 | M1 | SPARC | x86 | ARM (nm) |
| lshal | ! | ! | ! | ! | ! | ! | ! | ! | ! | ! |
| smbios | | | | | | | | ! | X | |
| prtdiag | ! | | | ! | ! | | | X | X | ! |
| bootinfo | X | | | | | | | | | |
| dnsdomainname | | ! | ! | X | X | | | ! | ! | X |
| domainname | | X | X | X | X | | | X | X | X |
| hexdump | | | | ! | | ! | | | X | |
| lspci | ! | ! | ! | X | X | ! | ! | ! | ! | X |
| oslevel | X | | | | | | | | | |
| dpkg-query | | | | ! | ! | | | | | ! |
| rpm | ! | ! | ! | ! | ! | | | | | ! |
| pkginfo | | | | | | | | X | X | |
| getent | X | | | X | X | ! | ! | X | X | X |
| id | X | | | X | X | ! | ! | X | X | X |
| db2licm | ! | ! | ! | ! | ! | ! | ! | ! | ! | ! |
| db2ls | ! | ! | ! | ! | ! | ! | ! | ! | ! | ! |
| lscpu | | | | ! | ! | | | | | ! |

## List of Files Which Deliver the Best Results With Privileged Rights

| Subject | Platform | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Basic Support | | | | | | | | | Extended Support |
| | AIX | HP-UX | | Linux | | MacOS | | Solaris | | Linux |
| File | POWER | PA-RISC | Itani-um | x86 | POWER | x86 | M1 | SPARC | x86 | ARM (nm) |
| /sys/class/dmi/id/chassis_vendor (accessed by cat) | | | | X | ! | | | ! | ! | ! |
| /dev/xsvc (accessed by hexdump) | | | | ! | | ! | | | X | |
| /dev/mem (accessed by hexdump) | | | | ! | | ! | | | | |
| /proc/partitions (accessed by cat) | X | | | X | X | ! | ! | ! | ! | X |
| /etc/oracle-release | | | | ! | ! | | | ! | ! | ! |
| /etc/os-release | | | | ! | ! | | | ! | ! | ! |
| /etc/SuSE-release | | | | ! | ! | | | ! | ! | ! |
| /etc/centos-release | | | | ! | ! | | | ! | ! | ! |
| /etc/enterprise-release | | | | ! | ! | | | ! | ! | ! |
| /etc/redhat-release | | | | ! | ! | | | ! | ! | ! |
| /etc/issue.net | | | | ! | ! | | | ! | ! | ! |
| /etc/debian_version | | | | ! | ! | | | ! | ! | ! |
| /etc/issue | | | | ! | ! | | | ! | ! | ! |
| /etc/lsb-release | | | | ! | ! | | | ! | ! | ! |
| Info.plist (of | | | | | | X | X | | | |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| installed packages, accessed by shell and defaults) | | | | | | | | | | |
| Info-macos.plist (off installed packages, accessed by shell defaults) | | | | | | X | X | | | |
| /etc/*release (accessed by echo and cat) | | | | | | | | | | |
| /proc/cpuinfo | ! | | | X | X | ! | ! | ! | ! | X |

List of Commands Which Could Require Privileged Rights Depending on the OS Version

| Subject | Platform | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Basic Support | | | | | | | | | Extended Support |
| | AIX | HP-UX | | Linux | | MacOS | | Solaris | | Linux |
| **Command** | POWER | PA-RISC | Itani-um | x86 | POWER | x86 | M1 | SPARC | x86 | ARM (nm) |
| dmidecode | ! | ! | ! | X | ! | ! | ! | ! | X | ! |
| uname | X | X | X | X | X | X | X | X | X | X |
| cat | X | X | X | X | X | X | X | X | X | X |
| getconf | X | X | X | | | | | | | |
| whoami | X | X | X | X | X | X | X | X | X | X |
| hostname | X | X | X | X | X | X | X | X | X | X |
| hostid | X | | | X | X | | | X | X | X |
| netstat | X | X | X | X | X | X | X | X | X | X |

# Zero-Touch/Remote Inventory for Oracle

# Required Permission to Run a Zero-Touch Oracle Inventory

## Required Permissions

The service account requires the following read permissions:

| Option | Description |
|---|---|
| CONTENT.ODM_DOCUMENT | Feature usage statistics |
| DMSYS.DM$MODEL | Feature usage statistics |
| DMSYS.DM$OBJECT | Feature usage statistics |
| DMSYS.DM$P_MODEL | Feature usage statistics |
| DVSYS.DBA_DV_REALM | Feature usage statistics |
| LBACSYS.LBAC$POLT | Feature usage statistics |
| MDSYS.ALL_SDO_GEOM_METADATA | Feature usage statistics |
| MDSYS.SDO_GEOM_METADATA_TABLE | Feature usage statistics |
| ODM.ODM_MINING_MODEL | Feature usage statistics |
| ODM.ODM_RECORD | Feature usage statistics |
| OLAPSYS.DBA$OLAP_CUBES | Feature usage statistics |
| SYS.CDB_FEATURE_USAGE_STATISTICS | Displays information about database feature usage statistics in case of Container Database |
| SYS.DBA_ADVISOR_TASKS | Displays information about all tasks in the database |
| SYS.DBA_AUDIT_TRAIL | Displays all audit trail entries |
| SYS.DBA_AWS | Feature usage statistic |
| SYS.DBA_CUBES | Describes all OLAP cubes in the database |
| SYS.DBA_ENCRYPTED_COLUMNS | Maintains encryption algorithm information for all encrypted columns in the database |
| SYS.DBA_FEATURE_USAGE_STATISTICS | Displays information about database feature usage statistics |
| SYS.DBA_LOB_PARTITIONS | Feature usage statistics |
| SYS.DBA_LOB_SUBPARTITIONS | Feature usage statistics |

| | |
|---|---|
| SYS.DBA_LOBS | Displays the BLOBs and CLOBs contained in all tables in the database |
| SYS.DBA_MINING_MODELS | Feature usage statistics |
| SYS.DBA_OBJECTS | Describes all objects in the database |
| SYS.DBA_RECYCLEBIN | Container for dropped objects |
| SYS.DBA_REGISTRY | Displays information about the components loaded in the database |
| SYS.DBA_SEGMENTS | Describes the storage allocated for all segments in the database |
| SYS.DBA_SQL_PROFILES | Displays information about SQL profiles currently created for specific SQL statements |
| SYS.DBA_SQLSET | Feature usage statistics |
| SYS.DBA_TAB_PARTITIONS | Feature usage statistics |
| SYS.DBA_TAB_SUBPARTITIONS | Feature usage statistics |
| SYS.DBA_TABLES | Describes all relational tables in the database |
| SYS.DBA_TABLESPACES | Describes all tablespaces in the database |
| SYS.DBA_USERS | Describes all users of the database |
| SYS.DBA_VIEWS | Describes all relational views in the database |
| SYS.DUAL | A table in the data dictionary that Oracle database and user-written programs can reference to guarantee a known result |
| SYS.GV_$DATABASE | Displays information about the database in the global view |
| SYS.GV_$INSTANCE | Displays the state of the current instance |
| SYS.GV_$PARAMETER | Displays information about the initialization parameters |
| SYS.MODEL$ | Feature usage statistics |
| SYS.V_$ARCHIVE_DEST_STATUS | Displays runtime and configuration informatoin for the archived redo log destinations |

| | |
|---|---|
| SYS.V_$BLOCK_CHANGE_TRACKING | Displays the Status of block change tracking for the database |
| SYS.V_$CONTAINERS | Displays information about PDBs and the root associated with the current instance |
| SYS.V_$DATABASE | Displays information about the database from the control file |
| SYS.V_$INSTANCE | Displays the state of the current instance |
| SYS.V_$LICENSE | Displays information about license limits |
| SYS.V_$OPTION | Feature usage statistics |
| SYS.V_$PARAMETER | Displays information about the initialization parameters that are currently in effect for the session |
| SYS.V_$VERSION | Displays version numbers of core library components in the Oracle Database |
| SYSMAN.MGMT_ADMIN_LICENSES | Management pack usage statistics |
| SYSMAN.MGMT_LICENSE_CONFIRMATION | Management pack usage statistics |
| SYSMAN.MGMT_LICENSE_DEFINITIONS | Management pack usage statistics |
| SYSMAN.MGMT_LICENSES | Management pack usage statistics |
| SYSMAN.MGMT_TARGETS | Management pack usage statistics |
| SYS.V_$IM_SEGMENTS | List INMEMORY Segments |

## Only for Oracle ERP System

- applsys.fnd_app_servers
- applsys.fnd_nodes
- applsys.fnd_product_installations
- applsys.fnd_application_tl
- applsys.fnd_responsibility
- applsys.fnd_user
- apps.fnd_user_resp_groups

## Multitenant System

A credential needs to be created which has the required prefix added to its name for the service

account to see the systems containers during any inventory scan.
By default, this prefix is C##, which means that if the Inventory-User (example: RVUser) is being used, then the credential needs to be created as C##RVUser.
If the system has been configured with a common_user_prefix value, then that prefix should be used instead of the default one.

## Script Templates to Grant Permissions for a Zero-Touch Inventory of Oracle Databases

There are two different script templates available that can be used to grant the necessary permissions.

1. `create_user_1.5.4.8_oldformat.sql` - This is a simple script without any error checking. The script simply tries to authorize all tables for the given user. The multitenant prefix user will need to be specified within this script.
2. `create_user_1.5.4.8_.sql` - This script checks the output and is the script which is recommended by Raynet.

- The script is checking if a Container User is needed (C##).
- If it is not (C##), the script reads the container sign for the user.
- If the user does not exist, the script creates the user.
- If the user does exist, the script releases the user and sets the password.
- The script sets the rights for the existing tables, in order to avoid any error messages.

The templates for all scripts can be found as attachments underneath this article.
**Please keep in mind that you will need to check the scripts if they can be applied to your environment! These are templates and modifications may be needed!**

- Create_user_1.5.4.8_oldformat.sql
- Create_user_1.5.4.8_user.sql

## Troubleshooting

In rare cases when the hostname of Oracle database inventory does not match the inventory of the operating system, RayVentory cannot manage the link between both inventories automatically. This situation could be indicated by:

1. The server inventory exists when viewing the Inventory report, but ...
2. Oracle Server overview report does not show a link to the server inventory and the column "Status" shows "No Hardware Inventory"

For these cases the IP-Address used by the Oracle database service could solve the problem.

## Requirements:

- use OraTrack XML version 1.5.9 or later
- apply a change of the OracleDB ACL by using the attached sample "set_user_acl12.sql"; if other OracleDB versions are used, contact Raynet support please.

- set_user_acl12.sql

### Create_user_1.5.4.8_oldformat.sql

```
/* ************************************************
   * Script to Create User with rights for oratrack
   ************************************************
   * Version:      1.5.4.8
   * Last Change:  06.08.2021
   ************************************************
   * Changes:
   * 10.06.2018 fix CDB selection if no CDB DB exists
   *            create userr
   *            grant select v_$CONTAINERS
   * 01.08.2018 fix CDB selection if no lines in table
   *            create RVSUSER
   * 02.08.2018 add check on v$database for CDB
   *            Get Parameter common_user_prefix
   *            fix create user for CDB
   * 18.09.2018 Add more Comments
   * 12.03.2019 fix CDB check
   * 09.04.2019 fix Password and special chars
   * 06.08.2021 add right for V$IM_SEGMENTS
   ************************************************
   * Please Change Variables
   *   sUsername   With the default Username
   *               If the DB is a CDB / Multitenant DB
   *               it at automaticly C## in front of the
   *               Username
   *   sPassword   With the Password what should be used
   ********************************************** */
-- USER SQL
  CREATE USER RVUSER IDENTIFIED BY "RayVentory01";
-- SYSTEM PRIVILEGES
  GRANT CREATE SESSION TO RVUSER ;

  -- Start Base Tables, this should allways exist
```

```
  GRANT SELECT ON SYS.DBA_ADVISOR_TASKS TO RVUSER ;
--  GRANT SELECT ON SYS.DBA_AUDIT_TRAIL TO RVUSER ;
  GRANT SELECT ON SYS.DBA_AWS TO RVUSER ;
  GRANT SELECT ON SYS.DBA_CUBES TO RVUSER ;
  GRANT SELECT ON SYS.DBA_ENCRYPTED_COLUMNS TO RVUSER ;
  GRANT SELECT ON SYS.DBA_FEATURE_USAGE_STATISTICS TO RVUSER ;
  GRANT SELECT ON SYS.DBA_LOB_PARTITIONS TO RVUSER ;
  GRANT SELECT ON SYS.DBA_LOB_SUBPARTITIONS TO RVUSER ;
  GRANT SELECT ON SYS.DBA_LOBS TO RVUSER ;
  GRANT SELECT ON SYS.DBA_MINING_MODELS TO RVUSER ;
  GRANT SELECT ON SYS.DBA_OBJECTS TO RVUSER ;
  GRANT SELECT ON SYS.DBA_RECYCLEBIN TO RVUSER ;
  GRANT SELECT ON SYS.DBA_REGISTRY TO RVUSER ;
  GRANT SELECT ON SYS.DBA_SEGMENTS TO RVUSER ;
  GRANT SELECT ON SYS.DBA_SQL_PROFILES TO RVUSER ;
  GRANT SELECT ON SYS.DBA_SQLSET TO RVUSER ;
  GRANT SELECT ON SYS.DBA_TAB_PARTITIONS TO RVUSER ;
  GRANT SELECT ON SYS.DBA_TAB_SUBPARTITIONS TO RVUSER ;
  GRANT SELECT ON SYS.DBA_VIEWS TO RVUSER ;
  GRANT SELECT ON SYS.DBA_TABLES TO RVUSER ;
--  GRANT SELECT ON SYS.DBA_ALL_TABLES TO RVUSER ;
  GRANT SELECT ON SYS.DBA_TABLESPACES TO RVUSER ;
  GRANT SELECT ON SYS.DBA_USERS TO RVUSER ;
  GRANT SELECT ON SYS.DUAL TO RVUSER ;
  GRANT SELECT ON SYS.MODEL$ TO RVUSER ;
--  GRANT SELECT ON SYS.USER_ROLE_PRIVS TO RVUSER ;
--  GRANT SELECT ON SYS.USER_SYS_PRIVS TO RVUSER ;
  GRANT SELECT ON SYS.V_$BLOCK_CHANGE_TRACKING TO RVUSER ;
  GRANT SELECT ON SYS.V_$ARCHIVE_DEST_STATUS TO RVUSER ;
  GRANT SELECT ON SYS.V_$DATABASE TO RVUSER ;
  GRANT SELECT ON SYS.V_$INSTANCE TO RVUSER ;
  GRANT SELECT ON SYS.V_$LICENSE TO RVUSER ;
  GRANT SELECT ON SYS.V_$OPTION TO RVUSER ;
  GRANT SELECT ON SYS.V_$PARAMETER TO RVUSER ;
--  GRANT SELECT ON SYS.V_$SESSION TO RVUSER ;
--  GRANT SELECT ON SYS.V_$SESSION_CONNECT_INFO TO RVUSER ;
  GRANT SELECT ON SYS.V_$VERSION TO RVUSER ;

-- Add permission for Globale Views
```

```
  GRANT SELECT ON SYS.GV_$DATABASE TO RVUSER ;
  GRANT SELECT ON SYS.GV_$INSTANCE TO RVUSER ;
  GRANT SELECT ON SYS.GV_$PARAMETER TO RVUSER ;

-- Add permission for Global DBFUS
   GRANT SELECT ON SYS.CDB_FEATURE_USAGE_STATISTICS TO RVUSER

-- Add permission for Container DB's
  GRANT SELECT ON SYS.V_$CONTAINERS TO RVUSER ;

-- Add permission for Inmemory
  GRANT SELECT ON SYS.V_$IM_SEGMENTS TO RVUSER ;

-- Add permission for Management Tables
  GRANT SELECT ON SYSMAN.MGMT_ADMIN_LICENSES TO RVUSER ;
  GRANT SELECT ON SYSMAN.MGMT_LICENSE_CONFIRMATION RVUSER ;
  GRANT SELECT ON SYSMAN.MGMT_LICENSE_DEFINITIONS TO RVUSER ;
  GRANT SELECT ON SYSMAN.MGMT_LICENSES TO RVUSER ;
  GRANT SELECT ON SYSMAN.MGMT_TARGETS TO RVUSER ;

-- Add permission for ODM_DOCUMENT
  GRANT SELECT ON CONTENT.ODM_DOCUMENT TO RVUSER ;

-- Add permission for DM Tables
  GRANT SELECT ON DMSYS.DM$MODEL TO RVUSER ;
  GRANT SELECT ON DMSYS.DM$OBJECT TO RVUSER ;
  GRANT SELECT ON DMSYS.DM$P_MODEL TO RVUSER ;

-- Add permission for DBA_DV_REALM
  GRANT SELECT ON DVSYS.DBA_DV_REALM TO RVUSER ;

-- Add permission for Label Security
  GRANT SELECT ON DVSYS.LBACSYS.LBAC$POLT TO RVUSER ;

-- Add permission for GEOM Tables
  GRANT SELECT ON MDSYS.ALL_SDO_GEOM_METADATA TO RVUSER ;
  GRANT SELECT ON MDSYS.SDO_GEOM_METADATA_TABLE TO RVUSER ;

-- Add Permission for ODM Table
  GRANT SELECT ON ODM.ODM_MINING_MODEL TO RVUSER ;
```

```
  GRANT SELECT ON ODM.ODM_RECORD TO RVUSER ;

-- Add Permission for OLAP Views
  GRANT SELECT ON OLAPSYS.DBA$OLAP_CUBES TO RVUSER ;

-- Add Permission for Oracle ERP only
  GRANT SELECT ON applsys.fnd_app_servers TO RVUSER ;
  GRANT SELECT ON applsys.fnd_nodes TO RVUSER ;
  GRANT SELECT ON applsys.fnd_product_installations TO RVUSER ;
  GRANT SELECT ON applsys.fnd_application_tl TO RVUSER ;
  GRANT SELECT ON applsys.fnd_responsibility TO RVUSER ;
  GRANT SELECT ON applsys.fnd_user TO RVUSER ;
  GRANT SELECT ON apps.fnd_user_resp_groups TO RVUSER ;

EXIT;
```

### Create_user_1.5.4.8_user.sql

```
/* ***********************************************
   * Script to Create User with rights for oratrack
   ***********************************************
   * Version:      1.5.4.8
   * Last Change:  06.08.2021
   ***********************************************
   * Changes:
   * 10.06.2018 fix CDB selection if no CDB DB exists
   *            create userr
   *            grant select v_$CONTAINERS
   * 01.08.2018 fix CDB selection if no lines in table
   *            create RVSUSER
   * 02.08.2018 add check on v$database for CDB
   *            Get Parameter common_user_prefix
   *            fix create user for CDB
   * 18.09.2018 Add more Comments
   * 12.03.2019 fix CDB check
   * 09.04.2019 fix Password and special chars
   * 27.04.2020 fix Oracle 9i/10i
   * 05.06.2020 fix CommonUserPrefix if not set
   *            fix reset Password
   * 06.08.2021 add right for V$IM_SEGMENTS
   ***********************************************
```

```
    * Please Change Variables
    *   sUsername   With the default Username
    *               If the DB is a CDB / Multitenant DB
    *               it at automaticly C## in front of the
    *               Username
    *   sPassword   With the Password what should be used
    ********************************************** */
SET SERVEROUTPUT ON
DECLARE
  sUsername         VARCHAR2(50) := 'RVUSER';           -- Basis
Username, if Container, script set C## in Front or the defined
Value
  sPassword         VARCHAR2(50) := 'RayVentory01';     -- Password
for the User
  sGlobalDB         VARCHAR2(50) := '';                 --
Temporary Needed
  sCommonUserPrefix VARCHAR2(50) := '';                 --
Temporary Nedded, used to Store the defined User Prefix if
Container Database
  c INT;
BEGIN
  -- Start Check if CDB
  -- Find Out if this Server is a Container Database, for this it
Execute to Selects and see if Container exists, if not it create an
Exception and end this script part
  BEGIN
    DBMS_OUTPUT.put_line('- Check if CDB Database User needed');
    EXECUTE IMMEDIATE '
      DECLARE
       c INT;
       exCustom EXCEPTION;
       PRAGMA EXCEPTION_INIT(exCustom, -20001);
      BEGIN
       SELECT COUNT(*) INTO c FROM V$DATABASE WHERE CDB=''YES'';
       IF c = 0 THEN
         DBMS_OUTPUT.put_line(''-- CDB Database: NO'');
         raise_application_error(-20001, ''CDB Database: NO'');
       END IF;
       DBMS_OUTPUT.put_line(''-- CDB Database: YES'');
      END;
    ';
```

```
      -- It looks like it is a Server with Container Database.
      -- Get Parameter common_user_prefix if it is set, store it in
sCommonUserPrefix
      SELECT COUNT(*) INTO c FROM V$PARAMETER WHERE
NAME='common_user_prefix';
      -- If Parameter common_user_prefix is not set, set in the
Variable sCommonUserPrefix the Oracle default Value C##
      IF c = 0 THEN
       sCommonUserPrefix := 'C##';
       DBMS_OUTPUT.put_line('-- No common_user_prefix is set, use
default C##');
    ELSE
       SELECT VALUE INTO sCommonUserPrefix FROM V$PARAMETER WHERE
NAME='common_user_prefix';
       DBMS_OUTPUT.put_line('-- Found common_user_prefix with Value:
' || sCommonUserPrefix);
    END IF;
      -- Add in Front of the sUsername the sCommonUserPrefix
    sUsername := sCommonUserPrefix || sUsername;
      -- Set extra Option for Container Databases, this is needed
for create User, so that the user get rights on all Containers
    sGlobalDB := ' CONTAINER=ALL';
  EXCEPTION
    WHEN OTHERS THEN
      DBMS_OUTPUT.put_line('-- No CDB_FEATURE_USAGE_STATISTICS');
  END;
  -- End Check if CDB

  -- Start Check / Create User / Reset Password / Unlock existing
User
  DBMS_OUTPUT.put_line('- Work with Username: ' || sUsername);
  -- Check if Username exist
  SELECT COUNT(*) INTO c FROM DBA_USERS WHERE USERNAME =
UPPER(sUsername);
  IF c > 0 THEN
    -- Check if the existing User is Locked
    SELECT COUNT(*) INTO c FROM DBA_USERS WHERE USERNAME =
UPPER(sUsername) AND ACCOUNT_STATUS LIKE '%LOCKED%';
    IF c > 0 THEN
       -- The User is Locked, Unlock the account
      DBMS_OUTPUT.put_line('-- Unlock Account');
```

```
      EXECUTE IMMEDIATE 'ALTER USER ' || sUsername || ' ACCOUNT
UNLOCK';
    END IF;
     -- Set the Password to the actual Password Value
    EXECUTE IMMEDIATE 'ALTER USER ' || sUsername || ' IDENTIFIED BY
"' || sPassword || '"' ;
    DBMS_OUTPUT.put_line('-- Set actual password');
  ELSE
    -- User not exist, create User
    -- USER SQL
    EXECUTE IMMEDIATE 'CREATE USER ' || sUsername || ' IDENTIFIED
BY "' || sPassword || '" ' || sGlobalDB;
    DBMS_OUTPUT.put_line('-- Create User');
    -- SYSTEM PRIVILEGES
     -- Add Session rights
    EXECUTE IMMEDIATE 'GRANT CREATE SESSION TO ' || sUsername ||
sGlobalDB;
    DBMS_OUTPUT.put_line('-- Add Session permission');
  END IF;
  -- End Check / Create User / Reset Password / Unlock existing
User


  -- Start Base Tables, this should always exist
  EXECUTE IMMEDIATE 'GRANT SELECT ON SYS.DBA_LOB_PARTITIONS TO ' ||
sUsername ;
  EXECUTE IMMEDIATE 'GRANT SELECT ON SYS.DBA_LOB_SUBPARTITIONS TO '
|| sUsername ;
  EXECUTE IMMEDIATE 'GRANT SELECT ON SYS.DBA_LOBS TO ' || sUsername
;
  EXECUTE IMMEDIATE 'GRANT SELECT ON SYS.DBA_OBJECTS TO ' ||
sUsername ;
  EXECUTE IMMEDIATE 'GRANT SELECT ON SYS.DBA_SEGMENTS TO ' ||
sUsername ;
    EXECUTE IMMEDIATE 'GRANT SELECT ON SYS.DBA_TAB_PARTITIONS TO '
|| sUsername ;
  EXECUTE IMMEDIATE 'GRANT SELECT ON SYS.DBA_TAB_SUBPARTITIONS TO '
|| sUsername ;
  EXECUTE IMMEDIATE 'GRANT SELECT ON SYS.DBA_VIEWS TO ' ||
sUsername ;
  EXECUTE IMMEDIATE 'GRANT SELECT ON SYS.DBA_TABLES TO ' ||
sUsername ;
--  EXECUTE IMMEDIATE 'GRANT SELECT ON SYS.DBA_ALL_TABLES TO ' ||
sUsername ;
```

```
  EXECUTE IMMEDIATE 'GRANT SELECT ON SYS.DBA_TABLESPACES TO ' ||
sUsername ;
  EXECUTE IMMEDIATE 'GRANT SELECT ON SYS.DBA_USERS TO ' ||
sUsername ;
  EXECUTE IMMEDIATE 'GRANT SELECT ON SYS.DUAL TO ' || sUsername ;
--  EXECUTE IMMEDIATE 'GRANT SELECT ON SYS.USER_ROLE_PRIVS TO ' ||
sUsername ;
--  EXECUTE IMMEDIATE 'GRANT SELECT ON SYS.USER_SYS_PRIVS TO ' ||
sUsername ;
  EXECUTE IMMEDIATE 'GRANT SELECT ON SYS.V_$ARCHIVE_DEST_STATUS TO
' || sUsername ;
  EXECUTE IMMEDIATE 'GRANT SELECT ON SYS.V_$DATABASE TO ' ||
sUsername ;
  EXECUTE IMMEDIATE 'GRANT SELECT ON SYS.V_$INSTANCE TO ' ||
sUsername ;
  EXECUTE IMMEDIATE 'GRANT SELECT ON SYS.V_$LICENSE TO ' ||
sUsername ;
  EXECUTE IMMEDIATE 'GRANT SELECT ON SYS.V_$OPTION TO ' ||
sUsername ;
  EXECUTE IMMEDIATE 'GRANT SELECT ON SYS.V_$PARAMETER TO ' ||
sUsername ;
--  EXECUTE IMMEDIATE 'GRANT SELECT ON SYS.V_$SESSION TO ' ||
sUsername ;
--  EXECUTE IMMEDIATE 'GRANT SELECT ON SYS.V_$SESSION_CONNECT_INFO
TO ' || sUsername ;
  EXECUTE IMMEDIATE 'GRANT SELECT ON SYS.V_$VERSION TO ' ||
sUsername ;
  DBMS_OUTPUT.put_line('-- Add permission for System Tables and
Views');
  -- End Base Tables, this should always exist

  -- Start the following tables not exists in Oracle 9i
  SELECT COUNT(*) INTO c FROM DBA_VIEWS WHERE VIEW_NAME =
'DBA_ADVISOR_TASKS';
  IF c > 0 THEN
    EXECUTE IMMEDIATE 'GRANT SELECT ON SYS.DBA_ADVISOR_TASKS TO '
|| sUsername ;
    DBMS_OUTPUT.put_line('-- Add permission for
DBA_ADVISOR_TASKS');
  END IF;
--  EXECUTE IMMEDIATE 'GRANT SELECT ON SYS.DBA_AUDIT_TRAIL TO ' ||
sUsername ;
```

```
  SELECT COUNT(*) INTO c FROM DBA_VIEWS WHERE VIEW_NAME =
'DBA_AWS';
  IF c > 0 THEN
    EXECUTE IMMEDIATE 'GRANT SELECT ON SYS.DBA_AWS TO ' ||
sUsername ;
    DBMS_OUTPUT.put_line('-- Add permission for DBA_AWS');
  END IF;

  SELECT COUNT(*) INTO c FROM DBA_VIEWS WHERE VIEW_NAME =
'DBA_ENCRYPTED_COLUMNS';
  IF c > 0 THEN
    EXECUTE IMMEDIATE 'GRANT SELECT ON SYS.DBA_ENCRYPTED_COLUMNS TO
' || sUsername ;
    DBMS_OUTPUT.put_line('-- Add permission for
DBA_ENCRYPTED_COLUMNS');
  END IF;

  SELECT COUNT(*) INTO c FROM DBA_VIEWS WHERE VIEW_NAME =
'DBA_FEATURE_USAGE_STATISTICS';
  IF c > 0 THEN
    EXECUTE IMMEDIATE 'GRANT SELECT ON
SYS.DBA_FEATURE_USAGE_STATISTICS TO ' || sUsername ;
    DBMS_OUTPUT.put_line('-- Add permission for
DBA_FEATURE_USAGE_STATISTICS');
  END IF;

  SELECT COUNT(*) INTO c FROM DBA_VIEWS WHERE VIEW_NAME =
'DBA_RECYCLEBIN';
  IF c > 0 THEN
    EXECUTE IMMEDIATE 'GRANT SELECT ON SYS.DBA_RECYCLEBIN TO ' ||
sUsername ;
    DBMS_OUTPUT.put_line('-- Add permission for DBA_RECYCLEBIN');
  END IF;

  SELECT COUNT(*) INTO c FROM DBA_VIEWS WHERE VIEW_NAME =
'DBA_REGISTRY';
  IF c > 0 THEN
    EXECUTE IMMEDIATE 'GRANT SELECT ON SYS.DBA_REGISTRY TO ' ||
sUsername ;
    DBMS_OUTPUT.put_line('-- Add permission for DBA_REGISTRY');
  END IF;
```

```
  SELECT COUNT(*) INTO c FROM DBA_VIEWS WHERE VIEW_NAME =
'DBA_SQL_PROFILES';
  IF c > 0 THEN
    EXECUTE IMMEDIATE 'GRANT SELECT ON SYS.DBA_SQL_PROFILES TO ' ||
sUsername ;
    DBMS_OUTPUT.put_line('-- Add permission for DBA_SQL_PROFILES');
  END IF;

  SELECT COUNT(*) INTO c FROM DBA_VIEWS WHERE VIEW_NAME =
'DBA_SQLSET';
  IF c > 0 THEN
    EXECUTE IMMEDIATE 'GRANT SELECT ON SYS.DBA_SQLSET TO ' ||
sUsername ;
    DBMS_OUTPUT.put_line('-- Add permission for DBA_SQLSET');
  END IF;

  SELECT COUNT(*) INTO c FROM DBA_VIEWS WHERE VIEW_NAME =
'V_$BLOCK_CHANGE_TRACKING';
  IF c > 0 THEN
    EXECUTE IMMEDIATE 'GRANT SELECT ON SYS.V_$BLOCK_CHANGE_TRACKING
TO ' || sUsername ;
    DBMS_OUTPUT.put_line('-- Add permission for
V_$BLOCK_CHANGE_TRACKING');
  END IF;
  -- End the following tables not exists in Oracle 9i

  -- Start the following tables not exists in Oracle 9i and 10i
  SELECT COUNT(*) INTO c FROM DBA_VIEWS WHERE VIEW_NAME =
'DBA_CUBES';
  IF c > 0 THEN
    EXECUTE IMMEDIATE 'GRANT SELECT ON SYS.DBA_CUBES TO ' ||
sUsername ;
    DBMS_OUTPUT.put_line('-- Add permission for DBA_CUBES');
  END IF;

  SELECT COUNT(*) INTO c FROM DBA_VIEWS WHERE VIEW_NAME =
'DBA_MINING_MODELS';
  IF c > 0 THEN
    EXECUTE IMMEDIATE 'GRANT SELECT ON SYS.DBA_MINING_MODELS TO '
|| sUsername ;
    DBMS_OUTPUT.put_line('-- Add permission for
DBA_MINING_MODELS');
```

```
  END IF;

  SELECT COUNT(*) INTO c FROM DBA_TABLES WHERE TABLE_NAME =
'MODEL$';
  IF c > 0 THEN
    EXECUTE IMMEDIATE 'GRANT SELECT ON SYS.MODEL$ TO ' || sUsername
;
    DBMS_OUTPUT.put_line('-- Add permission for MODEL$');
  END IF;
  -- End the following tables not exists in Oracle 9i and 10i

  -- Start Add Global Views
  -- Globale Views
  -- Check if Global Views exist
  SELECT COUNT(*) INTO c FROM DBA_VIEWS WHERE VIEW_NAME =
'GV_$DATABASE';
  IF c > 0 THEN
    -- Add right to the Global View
    EXECUTE IMMEDIATE 'GRANT SELECT ON SYS.GV_$DATABASE TO ' ||
sUsername ;
    DBMS_OUTPUT.put_line('-- Add permission for Globale View
GV_$DATABASE');
  END IF;

  -- Check if Global Views exist
  SELECT COUNT(*) INTO c FROM DBA_VIEWS WHERE VIEW_NAME =
'GV_$INSTANCE';
  IF c > 0 THEN
    -- Add right to the Global View
    EXECUTE IMMEDIATE 'GRANT SELECT ON SYS.GV_$INSTANCE TO ' ||
sUsername ;
    DBMS_OUTPUT.put_line('-- Add permission for Globale View
GV_$INSTANCE');
  END IF;

  -- Check if Global Views exist
  SELECT COUNT(*) INTO c FROM DBA_VIEWS WHERE VIEW_NAME =
'GV_$PARAMETER';
  IF c > 0 THEN
    -- Add right to the Global View
    EXECUTE IMMEDIATE 'GRANT SELECT ON SYS.GV_$PARAMETER TO ' ||
sUsername ;
```

```
    DBMS_OUTPUT.put_line('-- Add permission for Globale View
GV_$PARAMETER');
  END IF;
  -- End Add Global Views

  -- Start Multitenant / Containers
  BEGIN
    -- Check if Table CDB_FEATURE_USAGE_STATISTICS exist and if CDB
is Flaged in V$DATABASE. IF CDB then add rights on
CDB_FEATURE_USAGE_STATISTICS
    EXECUTE IMMEDIATE '
      DECLARE
        c INT;
       BEGIN
        SELECT COUNT(*) INTO c FROM CDB_FEATURE_USAGE_STATISTICS;
        SELECT COUNT(*) INTO c FROM V$DATABASE WHERE CDB=''YES'';
        IF c = 0 THEN
          DBMS_OUTPUT.put_line(''-- no CDB Database, no permission
for CDB_FEATURE_USAGE_STATISTICS needed'');
        ELSE
          EXECUTE IMMEDIATE ''GRANT SELECT ON
SYS.CDB_FEATURE_USAGE_STATISTICS TO ' || sUsername || ''' ;
          DBMS_OUTPUT.put_line(''-- Add permission for
CDB_FEATURE_USAGE_STATISTICS (need for Multitenant)'');
        END IF;
       END;
    ';
  EXCEPTION
    WHEN OTHERS THEN
      DBMS_OUTPUT.put_line('-- No CDB_FEATURE_USAGE_STATISTICS');
  END;

  -- Check if View v_$CONTAINERS exist, if yes grant rights to the
RayVentory User
  SELECT COUNT(*) INTO c FROM DBA_VIEWS WHERE VIEW_NAME =
'V_$CONTAINERS';
  IF c > 0 THEN
    EXECUTE IMMEDIATE 'GRANT SELECT ON SYS.V_$CONTAINERS TO ' ||
sUsername ;
    DBMS_OUTPUT.put_line('-- Add permission for View
V_$CONTAINERS');
```

```
  ELSE
    DBMS_OUTPUT.put_line('-- No View V_$CONTAINERS');
  END IF;
  -- End Multitenant / Containers

  -- Check if rights for Inmemory exists
  SELECT COUNT(*) INTO c FROM DBA_VIEWS WHERE VIEW_NAME =
'V_$IM_SEGMENTS';
  IF c > 0 THEN
    -- Add rights for Inmemory exists
    EXECUTE IMMEDIATE 'GRANT SELECT ON SYS.V_$IM_SEGMENTS TO ' ||
sUsername ;
    DBMS_OUTPUT.put_line('-- Add permission for Inmemory
V_$IM_SEGMENTS');
  END IF;
  -- End Add rights for Inmemory exists

  -- Management Tabels
  BEGIN
    EXECUTE IMMEDIATE '
      DECLARE
        c INT;
       BEGIN
        SELECT COUNT(*) INTO c FROM MGMT_ADMIN_LICENSES;
        EXECUTE IMMEDIATE ''GRANT SELECT ON
SYSMAN.MGMT_ADMIN_LICENSES TO ' || sUsername || ''' ;
        EXECUTE IMMEDIATE ''GRANT SELECT ON
SYSMAN.MGMT_LICENSE_CONFIRMATION ' || sUsername || ''' ;
        EXECUTE IMMEDIATE ''GRANT SELECT ON
SYSMAN.MGMT_LICENSE_DEFINITIONS TO ' || sUsername || ''' ;
        EXECUTE IMMEDIATE ''GRANT SELECT ON SYSMAN.MGMT_LICENSES TO
' || sUsername || ''' ;
        EXECUTE IMMEDIATE ''GRANT SELECT ON SYSMAN.MGMT_TARGETS TO
' || sUsername || ''';
        DBMS_OUTPUT.put_line(''-- Add permission for Management
Tables'');
       END;
    ';
  EXCEPTION
    WHEN OTHERS THEN
      DBMS_OUTPUT.put_line('-- No Management Tables');
```

```
  END;

  -- ODM_DOCUMENT
  BEGIN
    EXECUTE IMMEDIATE '
      DECLARE
        c INT;
       BEGIN
        SELECT COUNT(*) INTO c FROM DBA_TABLES WHERE TABLE_NAME =
''ODM_DOCUMENT'';
        IF c > 0 THEN
          EXECUTE IMMEDIATE ''GRANT SELECT ON CONTENT.ODM_DOCUMENT
TO ' || sUsername || ''' ;
          DBMS_OUTPUT.put_line(''-- Add permission for
ODM_DOCUMENT'');
        ELSE
          DBMS_OUTPUT.put_line(''-- No ODM_DOCUMENT Table'');
         END IF;
       END;
    ';
  EXCEPTION
    WHEN OTHERS THEN
      DBMS_OUTPUT.put_line('-- No ODM_DOCUMENT Table');
  END;

  -- DM Tables
  BEGIN
    EXECUTE IMMEDIATE '
      DECLARE
        c INT;
       BEGIN
        SELECT COUNT(*) INTO c FROM DMSYS.DM$MODEL;
        EXECUTE IMMEDIATE ''GRANT SELECT ON DMSYS.DM$MODEL TO ' ||
sUsername || ''' ;
        EXECUTE IMMEDIATE ''GRANT SELECT ON DMSYS.DM$OBJECT TO ' ||
sUsername || ''' ;
        EXECUTE IMMEDIATE ''GRANT SELECT ON DMSYS.DM$P_MODEL TO '
|| sUsername || ''' ;
        DBMS_OUTPUT.put_line(''-- Add permission for DM Tables'');
       END;
```

```
    ';
  EXCEPTION
    WHEN OTHERS THEN
      DBMS_OUTPUT.put_line('-- No DM Tables');
  END;

  -- DBA_DV_REALM
  BEGIN
    EXECUTE IMMEDIATE '
      DECLARE
        c INT;
       BEGIN
        SELECT COUNT(*) INTO c FROM DBA_VIEWS WHERE VIEW_NAME =
''DBA_DV_REALM'';
        IF c > 0 THEN
          EXECUTE IMMEDIATE ''GRANT SELECT ON DVSYS.DBA_DV_REALM TO
' || sUsername || ''' ;
          DBMS_OUTPUT.put_line(''-- Add permission for
DBA_DV_REALM'');
        ELSE
          DBMS_OUTPUT.put_line(''-- No Table DBA_DV_REALM'');
         END IF;
       END;
    ';
  EXCEPTION
    WHEN OTHERS THEN
     DBMS_OUTPUT.put_line('-- No Table DBA_DV_REALM');
  END;

  -- Label Secuity
  BEGIN
    EXECUTE IMMEDIATE '
      DECLARE
        c INT;
       BEGIN
        SELECT COUNT(*) INTO c FROM DBA_TABLES WHERE TABLE_NAME =
''LBAC$POLT'';
        IF c > 0 THEN
          EXECUTE IMMEDIATE ''GRANT SELECT ON
DVSYS.LBACSYS.LBAC$POLT TO ' || sUsername || ''' ;
```

```
            DBMS_OUTPUT.put_line(''-- Add permission for
LBACSYS.LBAC$POLT'');
          ELSE
            DBMS_OUTPUT.put_line(''-- No Table LBACSYS.LBAC$POLT'');
          END IF;
        END;
    ';
  EXCEPTION
    WHEN OTHERS THEN
     DBMS_OUTPUT.put_line('-- No Table LBACSYS.LBAC$POLT');
  END;

  -- GEOM Tables
  BEGIN
    EXECUTE IMMEDIATE '
      DECLARE
        c INT;
       BEGIN
        SELECT COUNT(*) INTO c FROM DBA_VIEWS WHERE VIEW_NAME =
''ALL_SDO_GEOM_METADATA'';
        IF c > 0 THEN
          EXECUTE IMMEDIATE ''GRANT SELECT ON
MDSYS.ALL_SDO_GEOM_METADATA TO ' || sUsername || ''' ;
          EXECUTE IMMEDIATE ''GRANT SELECT ON
MDSYS.SDO_GEOM_METADATA_TABLE TO ' || sUsername || ''' ;
          DBMS_OUTPUT.put_line(''-- Add permission for GEOM
Tables'');
        ELSE
          DBMS_OUTPUT.put_line(''-- No GEOM Tables'');
          END IF;
        END;
    ';
  EXCEPTION
    WHEN OTHERS THEN
      DBMS_OUTPUT.put_line('-- No GEOM Tables');
  END;

  -- ODM Table
  BEGIN
    EXECUTE IMMEDIATE '
```

```
      DECLARE
        c INT;
       BEGIN
        SELECT COUNT(*) INTO c FROM DBA_TABLES WHERE TABLE_NAME =
''ODM_MINING_MODEL'';
        IF c > 0 THEN
          EXECUTE IMMEDIATE ''GRANT SELECT ON ODM.ODM_MINING_MODEL
TO ' || sUsername || ''' ;
          EXECUTE IMMEDIATE ''GRANT SELECT ON ODM.ODM_RECORD TO '
|| sUsername || ''' ;
          DBMS_OUTPUT.put_line(''-- Add permission for ODM
Tables'');
        ELSE
          DBMS_OUTPUT.put_line(''-- No ODM Tables'');
         END IF;
       END;
    ';
  EXCEPTION
    WHEN OTHERS THEN
      DBMS_OUTPUT.put_line('-- No ODM Tables');
  END;

  -- OLAP Views
  BEGIN
    EXECUTE IMMEDIATE '
      DECLARE
        c INT;
       BEGIN
        SELECT COUNT(*) INTO c FROM DBA_VIEWS WHERE VIEW_NAME =
''DBA$OLAP_CUBES'';
        IF c > 0 THEN
          EXECUTE IMMEDIATE ''GRANT SELECT ON
OLAPSYS.DBA$OLAP_CUBES TO ' || sUsername || ''' ;
          DBMS_OUTPUT.put_line(''-- Add permission for OLAP
Tables'');
        ELSE
          DBMS_OUTPUT.put_line(''-- No OLAP Tables'');
         END IF;
       END;
    ';
  EXCEPTION
```

```
    WHEN OTHERS THEN
     DBMS_OUTPUT.put_line('-- No OLAP Tables');
  END;

  -- Oracle ERP only
  SELECT COUNT(*) INTO c FROM DBA_TABLES WHERE TABLE_NAME =
'fnd_app_servers';
  IF c > 0 THEN
    EXECUTE IMMEDIATE 'GRANT SELECT ON applsys.fnd_app_servers TO '
|| sUsername ;
    EXECUTE IMMEDIATE 'GRANT SELECT ON applsys.fnd_nodes TO ' ||
sUsername ;
    EXECUTE IMMEDIATE 'GRANT SELECT ON
applsys.fnd_product_installations TO ' || sUsername ;
    EXECUTE IMMEDIATE 'GRANT SELECT ON applsys.fnd_application_tl
TO ' || sUsername ;
    EXECUTE IMMEDIATE 'GRANT SELECT ON applsys.fnd_responsibility
TO ' || sUsername ;
    EXECUTE IMMEDIATE 'GRANT SELECT ON applsys.fnd_user TO ' ||
sUsername ;
    EXECUTE IMMEDIATE 'GRANT SELECT ON apps.fnd_user_resp_groups TO
' || sUsername ;
    DBMS_OUTPUT.put_line('-- Add permission for ERP Tables');
  ELSE
    DBMS_OUTPUT.put_line('-- No ERP Tables');
  END IF;
END;
/

EXIT;
```

### set_user_acl12.sql

```
-- ACL for Oracle DB 12
SET SERVEROUTPUT ON SIZE UNLIMITED
DECLARE
  sUsername         VARCHAR2(50) := 'RVUSER';           -- Basis
Username, if Container, script set C## in Front or the defined
Value
  sCommonUserPrefix VARCHAR2(50) := '';                 --
Temporary Nedded, used to Store the defined User Prefix if
Container Database
  c INT;
  v_count12 NUMBER;
```

```
  compile_error exception;
  pragma exception_init(compile_error, -06550);
BEGIN
  BEGIN
    DBMS_OUTPUT.put_line('- Check if CDB Database User needed');
    EXECUTE IMMEDIATE '
      DECLARE
        c INT;
        exCustom EXCEPTION;
        PRAGMA EXCEPTION_INIT(exCustom, -20001);
       BEGIN
        SELECT COUNT(*) INTO c FROM V$DATABASE WHERE CDB=''YES'';
        IF c = 0 THEN
          DBMS_OUTPUT.put_line(''-- CDB Database: NO'');
          raise_application_error(-20001, ''CDB Database: NO'');
        END IF;
        DBMS_OUTPUT.put_line(''-- CDB Database: YES'');
       END;
    ';
     -- It looks like it is a Server with Container Database.
     -- Get Parameter common_user_prefix if it is set, store it in
sCommonUserPrefix
    SELECT VALUE INTO sCommonUserPrefix FROM V$PARAMETER WHERE
NAME='common_user_prefix';
     -- If Parameter common_user_prefix is not set, set in the
Variable sCommonUserPrefix the Oracle default Value C##
     IF sCommonUserPrefix IS NULL THEN
      sCommonUserPrefix := 'C##';
      DBMS_OUTPUT.put_line('-- No common_user_prefix is set, use
default C##');
    ELSE
      DBMS_OUTPUT.put_line('-- Found common_user_prefix with Value:
' || sCommonUserPrefix);
    END IF;
     -- Add in Front of the sUsername the sCommonUserPrefix
    sUsername := sCommonUserPrefix || sUsername;
  EXCEPTION
    WHEN OTHERS THEN
      DBMS_OUTPUT.put_line('-- No CDB_FEATURE_USAGE_STATISTICS');
  END;
```

```
  DBMS_OUTPUT.put_line('- Check if Oracle DB 12 or higher');
  SELECT COUNT(*) INTO v_count12 FROM V$INSTANCE WHERE
regexp_like(VERSION,'^1[2-9]\.|^[2-9][0-9]\.');
  IF v_count12 >= 1 THEN
    EXECUTE IMMEDIATE q'<
        BEGIN
      DBMS_NETWORK_ACL_ADMIN.append_host_ace (
        host       => '*',
        ace        => xs$ace_type(privilege_list =>
xs$name_list('resolve'),
                                  principal_name => '>' ||
sUsername || q'<',
                                  principal_type =>
xs_acl.ptype_db));
      dbms_output.put_line('  - Added ACL to Oracle DB 12 or
higher');
        END;
    >';
  ELSE
    dbms_output.put_line('  - Can not find Oracle DB 12 or
higher');
  END IF;
EXCEPTION
  WHEN compile_error THEN
    dbms_output.put_line('  - Can not find Oracle DB 12 or
higher');
END;
/
```

# Zero-Touch/Remote Inventory for ESX/vSphere

## Required Permissions to Run a Zero-Touch VMWare Inventory

### Permissions in VMware

The user needs read permissions on the full object tree. In addition to the read permissions, on the host Port 80 / 443 needs to be open for communcation.

The access is controlled by using user / group permissions with the assigned roles.

### Permissions:

A permission consists of a user / group and an assigned role for a specific object like a data center / Cluster / Host.

### The role:

The roles are a set of access rights and privileges which can be created and assigned to each role. Multiple privileges can be assigned to each role.

### The user:

The users and groups are created through the Windows domain, the Active Directory database, or the ESX / ESXi host. The users are part of the assigning privileges process.

The user will be regularly verified against the domain. This causes a denying or a deletion of all permissions if the username is changed or deleted. If a new user with the same name is created before the verification, the new user will get all the rights of the old user.

## Needed Permissions

For a vSphere / ESX scan the system role `'Read only'` that grants the following permissions is required:

- The permission to view the state and details of the object.
- The permission to view all the tab panels in the vSphere Client except the Console tab.
- It cannot perform any actions through the menus and toolbars.
- This role is available on the ESX / ESXi and the vCenter Server.

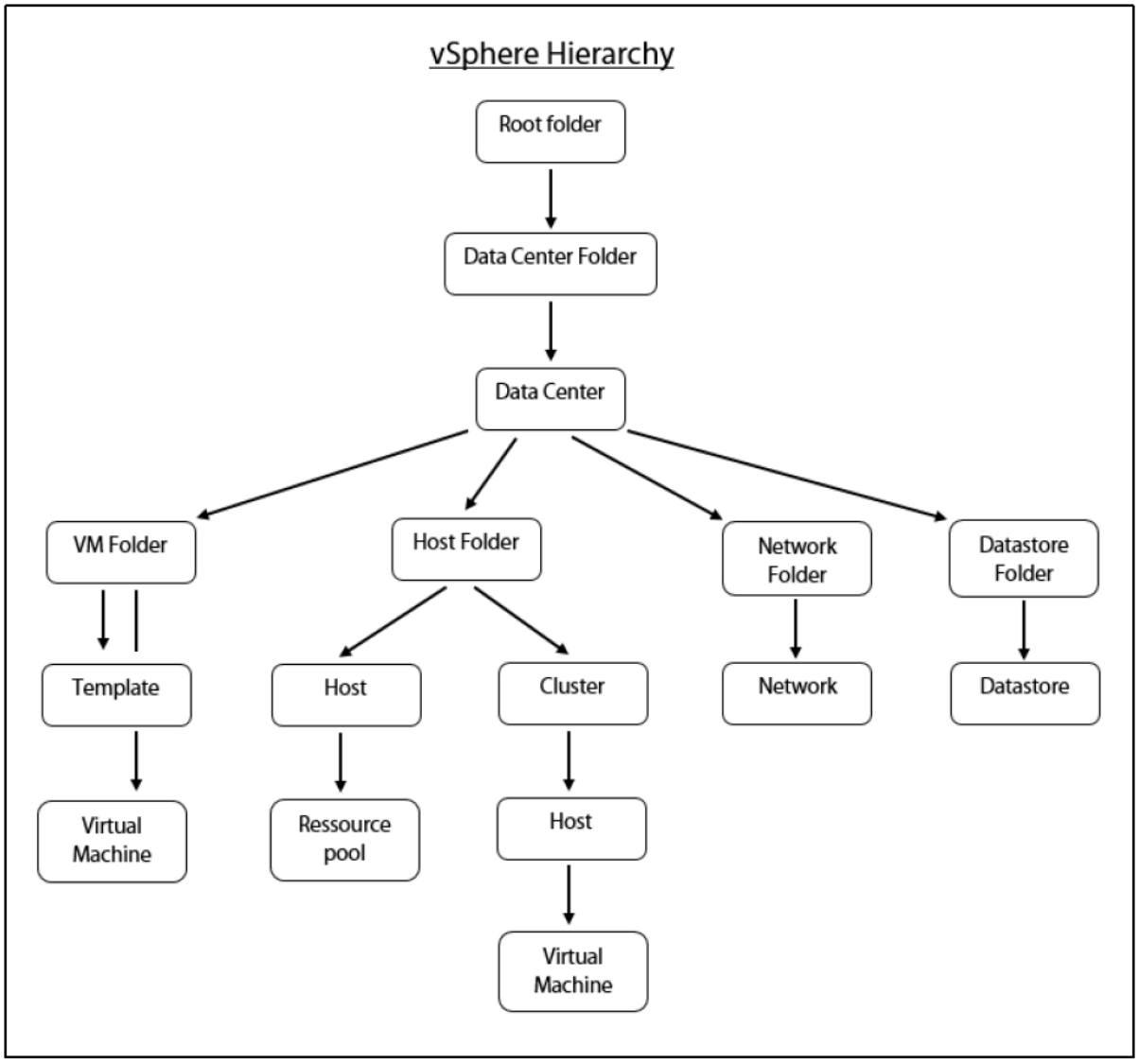
The permissions can be applied by adding the user in the related resource pool or folder.

## Reading the license key

If the full license key should be read out, the user need additional rights. For additional rights, the role 'Read only' can be cloned and edited afterwards. Be aware, that the cloned role will not be applied to the the same users / groups and objects.
The additional right to read the key must be selected under **Roles** > **Global** > **Licenses**.

## Setting the Permissions

The easiest way to set the required permissions is to set the 'Read only' right for the user in the root folder. When the permissions are applied, it is possible to choose if the permissions propagate down the object.




## Executed Queries

RayVentory is using the base version 2.00 vSphere SOAP API for the most part, with an additional call per host and cluster to a later API method RetrievePorepertiesEx to retrieve specific additional properties which are not available from the 2.0 API.
These are the Queries which are collecting the data after logon:

| Number | Description | Query | Query Target |
|---|---|---|---|
| 1. | A single call to retrieve the `ServiceInstance` to enable further queries. | • `GetServiceContent` | |

| Number | Description | Query | Query Target |
|---|---|---|---|
| 2. | A single call, starting from the root folder and returning all `HostSystem`, `Datacenter`, `Folder`, `ComputeResource`, and `ClusterComputeResource` objects (with 2-5 associated property values each) via recursive traversal specifications. | • `RetrieveProperties` | • `ComputeResource.host`<br>• `ClusterComputeResource.host`<br>• `Datacenter.hostFolder`<br>• `Datacenter.vmFolder`<br>• `Folder.childEntity` |
| 3. | For each `HostSystem` returned, an additional `RetrieveProperties` call, starting from the respective `HostSystem` object and returning all related `VirtualMachine` and `ResourcePool` objects (with 4 associated property values each) via recursive traversal specifications | | • `HostSystem.parent`<br>• `ComputeResource.resourcePool`<br>• `ResourcePool.resourcePool`<br>• `ResourcePool.vm` |

# Zero-Touch/Remote Inventory for SNMP

## Required Permissions for Inventory Scans of Network Devices

### General

Devices can be created and/or discovered via one or more of these methods:

- IP address(s), IP range(s), subnet(s) - with & without CIDR
- Hostname
- FQDN
- Optional: Import from Active Directory (AD)

### Network

- ICMP open between RayVentory Portal (RVP) and target systems for ping sweep
- UDP port 161 (SNMP) open between RVP and target systems
- Optional: Read permissions against AD for import of devices

### Network Device

- SNMP must be enabled & configured
- Community string name(s) for SNMP v1, v2c & v3 devices
- Read permission of all information or only the information you want to share with an inventory scan
- RVP server added as an allowed host
- Authentication and/or privacy algorithm's & credentials for SNMP v3 devices

# RayVentory Inventory Agent for Windows

## Installation and Configuration of RVIA for Windows

**Be aware:**
The RayVentory Inventory Agent for Windows needs the Windows Task Scheduler 2.0 (Windows Vista/Server 2008 and newer)!

### Serverside Configuration

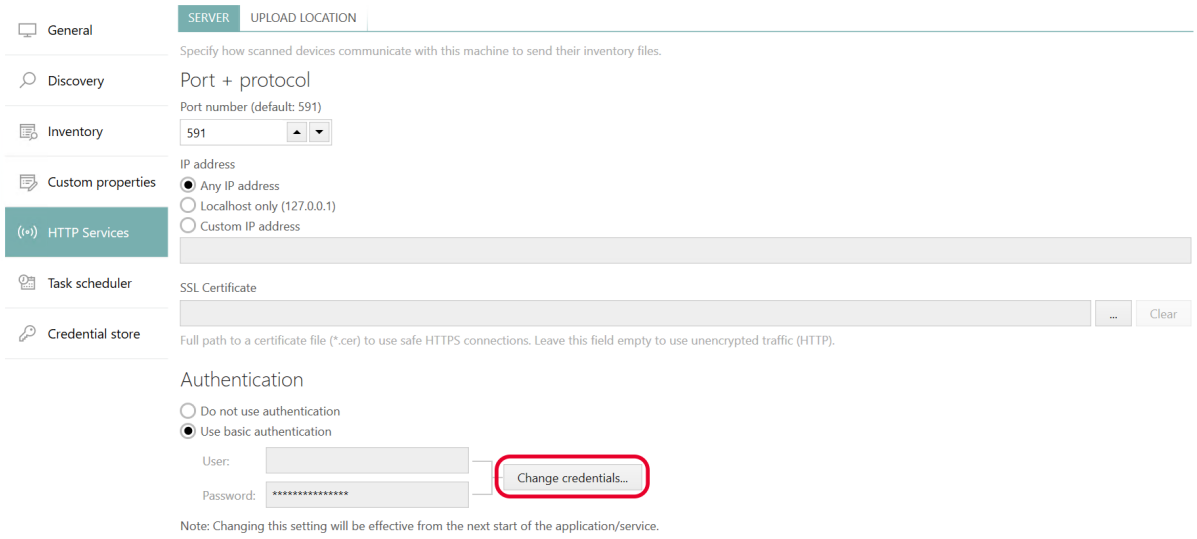
#### Upload and Download

AD Domain users are always able to download files and upload inventory data. If necessary, RayVentory Scan Engine also support Basic authentication. In order to change the credentials for uploads and downloads after the installation it is necessary to change the configuration of the agent. In order to do this, it might be necessary to deploy the software again.

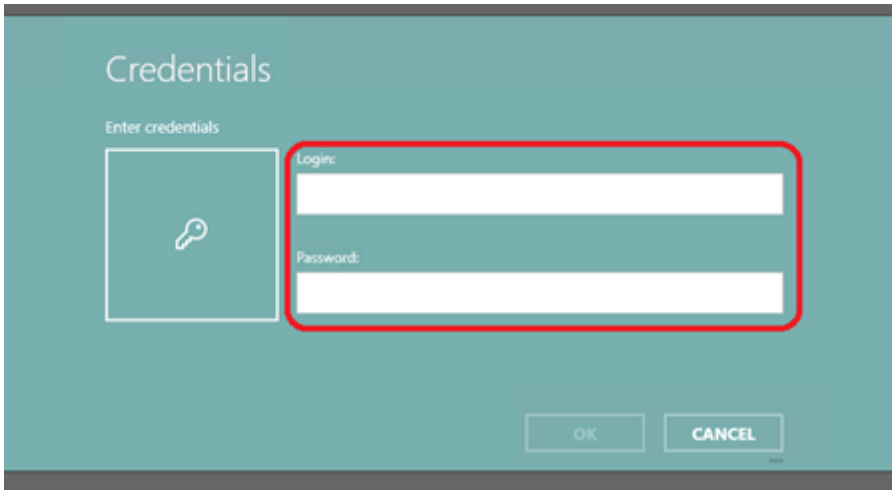
#### Add the Windows Account to RayVentory Scan Engine

Execute the following steps in order to activate Basic authentication and to add the necessary Windows User credentials.

In the **HTTP Services** tab in the **Settings** activate the **Use basic authentication** option by selecting the radio button. Two new fields (**User** and **Password**) and the **Change credentials...** button will now be available in the tab.

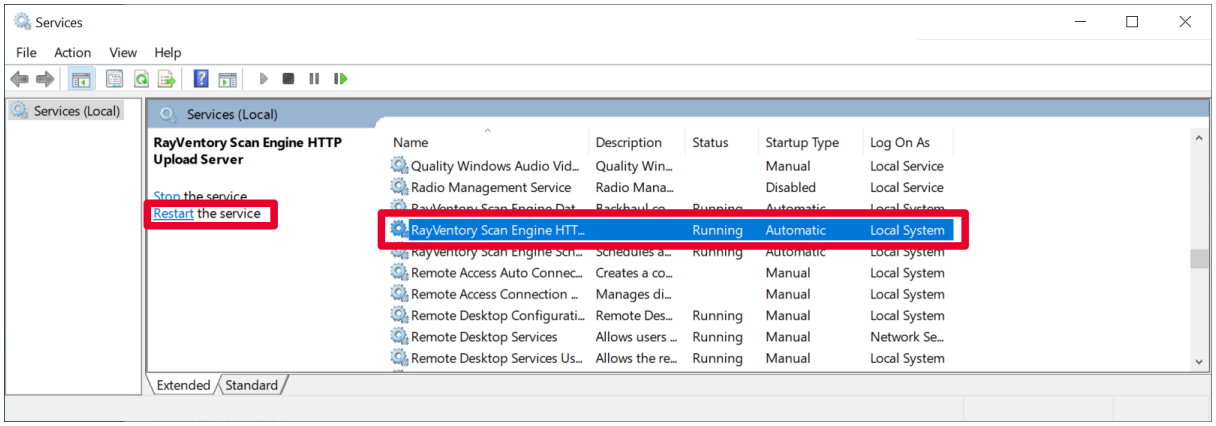



Click on the **Change credentials...** button. The **Credentials** dialog will be opened.




Enter the user into the **Login** field of the dialog and the password into the **Password** field of the dialog and click on the **OK** button.

After the credentials have been added or changed, it is necessary to restart the **RayVentory Scan Engine Scan Engine HTTP Upload Server** service.




After the restart of the server the new settings will be applied.

## Windows User Setup

The user has to be part of a local user group which has by default read access to the `C:\ProgramData\Raynet\RayVentoryPortal\Results\rviaconfig` folder. This folder contains all configuration files for the agent.

**Note:**
The path to the folder may differ from the one given here. In order to find the path for the specific installation start the RayVentory Scan Engine and go to **Settings** > **Inventory** > **Inventory Agent**.

For the upload, the user needs writing permissions for the following folder:

`C:\ProgramData\Raynet\RayVentoryPortal\Results\RemoteExecution`

**Note:**
Directly after the first installation of the RayVentory Scan Engine the RemoteExecution folder is missing. It will be created with the first upload of an agent. It is also possible to create it manually and then grant writing permissions to the user, by either adding the user to the local users group or by explicitly granting writing permissions to this user.

## Configuration of the Windows Agent

It is possible to create multiple configurations before starting a rollout of the Windows agent. It is recommended to create one configuration file for clients and one configuration file for server.

**Best Practice:**
It is recommended to create one configuration file for Windows clients and one configuration file for Windows server.

**Example:**

| Configuration file name | Recommended for | Scope |
|---|---|---|
| `ClientStandard.cfg` | `Windows 7 - Windows 10` | Default inventory executed after the Windows logon and at specific times. |
| `ServerStandard.cfg` | `Windows Server 2012 R - Windows Server 2019` | Inventory during the start of the system and at specific times. It should contain all the default plugins for the extended software inventory (for example: Microsoft Exchange, Microsoft Sharepoint, IIS, and other products). |

By default, the configuration files can be found in the following folder on the server where RayVentory Scan Engine is installed:

```
C:\ProgramData\Raynet\RayVentoryPortal\Results\rviaconfig
```

Experienced users can edit the `.cfg` files using any text editor. It is important that the changed `.cfg` file will be saved in the ANSI format. A description of the different parameter that can be used in the file can be found in the Parameters chapter.

## Installation of the Windows Agent

### Custom Installation Variants

The installation is based on the Microsoft Windows Installer and allows for extensive modifications to roll out specific settings or modified files. For example, it allows for the following:

- Changing the content or the scope of the plugins.
- Changing the content of the `wmitrack.ini`.
- Expanding the settings of the `whitelisted.xml`.

## Installation of the Windows Agent

### Source Folder

After installing RayVentory Scan Engine for the first time or after updating it, the latest setup for the agent can be found in the `%Program Files (x86)%\RayVentoryScanEngine \Contrib\InventoryAgent\Windows` folder.

The `RayVentory _Inventory_Agent.msi` can be copied from there and either be integrated into the software deployment tool or made available for download on a network share.

### Installation with Custom Configuration

It is important to add the correct configuration file when installing. The configuration file can be added by using a parameter which defines the URL used for the download and the user that has been added to RayVentory Scan Engine.

Sample call for an unattended installation:

```
msiexec /i "%~dp0RayVentory_Inventory_Agent.msi" /qn /L*v "%temp%
\RayVentorySetup.log"
 CONFIGDOWNLOADSOURCE=https://RVSE:8099/rviaconfig/
ClientStandard.cfg
 CONFIGDOWNLOADUSER=Agent_UploadUser CONFIGDOWNLOADPASSWORD=XYZ
```

`CONFIGDOWNLOADSOURCE` needs to contain the complete URL of the download path for the `.cfg` that is being used. `CONFIGDOWNLOADUSER` needs to contain the user and `CONFIGDOWNLOADPASSWORD` contains the password of the selected user.

### Preparing the Agent Setup for the Deployment by a Software Deployment Tool

In order to use a software deployment tool, it is necessary to create a custom setup. The following steps describe how to create a custom package and how to integrate an SSL certificate in order to simplify the deployment.

1. Create a folder for the creation of the custom setup (for example: `D:\Source\RV-Agent`).
2. Execute the following command-line:
   `msiexec /a "C:\Program Files (x86)\RayVentoryScanEngine\Contrib \InventoryAgent\Windows\RayVentory_Inventory_Agent.msi`
   Chose the newly created folder (`D:\Source\RV-Agent`) as target folder.
3. Copy the extracted agent setup to a network share or a public share.
4. Modify the `template.cfg` file in the `D:\Source\RV-Agent\ProgramFiles \RayVentory\InventoryAgent` folder as needed.
5. Delete the # from a parameter to activate it. Add # in front of a parameter to deactivate it.

Example:
These changes are used to define the configuration that will be used.
Change `#configDownloadSource=` to `configDownloadSource=https://RVSE:8099/rviaconfig/ClientStandard.cfg`.
Change `#resultUploadDestination=` to `resultUploadDestination=http(s)://RVSE:8099/`
Important: The upload URL needs to end with "/"

Example:
This change is used to define that usage metering will be disabled.
Change `#usageDisabled=` to `usageDisabled=true`.
Important: `true` will deactivate this option, `false` is used in order to activate it.

6. Save the `template.cfg`.
7. Modify the `curl-ca-bundle.crt` as described in https://raynetgmbh.zendesk.com/hc/en-us/articles/4408460779796.

For a direct installation using the preconfiguration execute the `D:\Source\RV-Agent\RayVentory_Inventory_Agent.msi`. During the installation the files in the `ProgramFiles` folder will be used for the installation and need to be taken into consideration for the deployment. The parameters `CONFIGDOWNLOADSOURCE`, `CONFIGDOWNLOADUSER`, and `CONFIGDOWNLOADPASSWORD` are no longer needed.

## Manual Upload of Inventory Data

In case a client is offline (for example if the client has no LAN or internet access) it is possible to copy an *.ndi*file created by NDTrack and upload it to RayVentory Scan Engine using one of the following methods:

Using Powershell (Windows only):

```
Invoke-RestMethod -Uri 'https://RVSE:8099/filename.ndi' -Method Put -InFile '.\filename.ndi' -UseDefaultCredentials
```

Using CURL (available for Linux/Unix or the latest Windows 10 and Windows Server 2019 versions):

```
Curl -T filename.ndi https://RVSE:8099/
```

All of the above examples use the current user context. If another user should be used for the upload additional parameter will be needed depending on the command.

All inventories that have been uploaded either manually or by the agent can be found in the file system of the RVSE server in the `C:\ProgramData\Raynet\RayVentoryPortal\Results\RemoteExecution` folder.

**Be aware:**
Device information and inventory data will only be shown in the Devices view of RayVentory Scan Engine if the web upload has been used.

**Note:**
It is possible to simply copy a file into the folder and it will also be forwarded to the RayVentory Server (using the Upload action in RayVentory Scan Engine), but it will not be shown in the devices view.

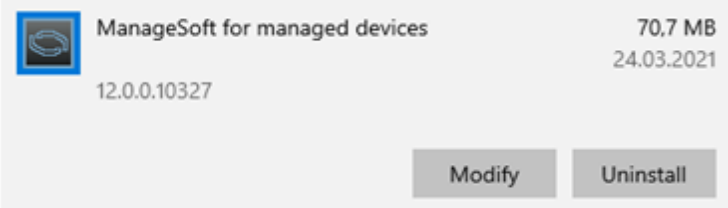
# Uninstallation of the ManageSoft Agent

## Manual Uninstallation

The Managed Device Agent that has been used by RayVentory previous to version 12.2 cannot be updated automatically by the RayVentory Agent setup of version 12.2 and afterward since some fundamental functions have been changed. Therefore the ManageSoft Agent has to be uninstalled manually by executing the following steps:

1. Uninstall the package Default Configuration Schedule for Managed Devices.
   - Furthermore all packages where (via RayManageSoft) is part of the name need to be uninstalled.


2. Uninstall all ManageSoft Agents if the version is lower than 12.2.
   - A restart might be necessary.




3. All previous registry entries for RayManageSoft need to be deleted since they might lead to unpredictable problems when using the RVIA agent.
   - `HKLM\SOFTWARE\WOW6432Node\ManageSoft Corp` (64-Bit)
   - `HKLM\SOFTWARE\ManageSoft Corp` (32-Bit)

**WARNING**
On the RayVentory Server these entries must be kept! They contain critical information about the functionality of the RayVentory Server and without them, the server might no longer be fully functional.

4. Clean the device of all installation files that might still be on the system.
   - Delete the installation folder (by default: `%ProgramFiles(x86)%\ManageSoft`).

## Automated Uninstallation

The individual packages of the ManageSoft agent can be uninstalled using the command-line. For this, administrative rights are needed.

1. The Client Machine Schedule package and the Machine Schedule package can be uninstalled using `ndlaunch`. The schedule depends on the devices being a client or a server.
   - Use the following command to uninstall the old configuration package:
     ```
     "%ProgramFiles(x86)\ManageSoft\Launcher\ndlaunch" -o
     InstallProfile -d "Default Client Machine Schedule"
     ```

**Be aware:**
The correct installation folder for RayManageSoft needs to be used.

   - Use the following command to uninstall the other old configuration package:
     ```
     "%ProgramFiles(x86)\ManageSoft\Launcher\ndlaunch" -o
     InstallProfile -d "Default Machine Schedule"
     ```

**Be aware:**
The correct installation folder for RayManageSoft needs to be used.

2. The uninstallation of the Managed Device agent depends on the installed version.
   - **10.51:** `MsiExec.exe /qn /X{1160EA41-AC3A-4B41-9BA6-1E0210ABBA96}`
   - **11.00:** `MsiExec.exe /qn /X{500B0922-7A7C-4EDC-8797-46F4DCCC75F5}`

3. The registry entries can be deleted using Powershell:
   - `Remove-Item -Path 'HKLM:\SOFTWARE\WOW6432Node\ManageSoft Corp\' -Force`
   - `Remove-Item -Path 'HKLM:\SOFTWARE\ManageSoft Corp\' -Force`

**WARNING**
On the RayVentory Server these entries must be kept! They contain critical information about the functionality of the RayVentory Server and without them, the server might no longer be fully functional.

# Troubleshooting

## The Client or the Windows Agent Does Not Connect to RayVentory Scan Engine

### Check the Availability of the RayVentory Scan Engine URL

1. On the client (Windows) open a web browser.
2. Enter the URL for the server where RayVentory Scan Engine is installed (for example: `https://RVSE:8099/`) and open it.

3. If the connection is successful, the screen will show **OK**.

### If the URL cannot be reached

a) Check if the URL is still correct. Also try using `https` instead of `http` if an SSL certificate is now in use.

4. Check if the name is resolved correctly by the DNS resolver by using the nslookup command (for this example: nslookup RVSE. The IP address for RVSE must be known and given as output. If this is not the case, the Windows administrator needs to check if the correct DNS server has been configured for the device and the network administrator should create a new DNS entry.

**Note:**
In case there is a DNS problem, the URL can also be called using the IP address. This could be useful for the troubleshooting but is not recommended for regular usage. An IP address can be changed at any time and this could lead to further problems, for example if an SSL certificate is used. These are normally issued for a server name / FQDN (due to security reasons) and not for a specific IP address.

b) Check that the port 443 is not blocked by a firewall by using the following command:
`tnc {computer}.{domain} -Port 443`
The result must be:
`TcpTestSucceeded : True`
If the result is `False`, check that the RayVentory Scan Engine settings and the local firewall rules are set for the port 443, before contacting a network administrator.

### The Agent Does Not Update the Configuration File

1. Check the path that has been configured in the `rvia.cfg` that can be found on the device in the `%ProgramData%\Raynet\RayVentoryInventoryAgent folder`.

   The parameter is `configDownloadSource` and an example path would be `https://RVSE:8099/rviaconfig/ClientStandard.cfg`.

2. Check if it is possible to reach the URL. In order to do this, open a web browser and call the URL. If the path is incorrect, the web browser will show an error message.

**Be aware:**
This is case sensitive, therefore It is important that all upper and lower cases are correct.

3. If the path exist, the web browser will ask for a username and a password. These should be the same as those configured in RayVentory Scan Engine in the Basic Authentication.

4. The content of the `.cfg` should be shown directly or after a successful authentication.

5. If there is an error in the path, the parameter `configDownloadSource` in the `rvia.cfg` file needs to be changed. After the change, the **GetConfig** task should be restarted in the Windows Task Scheduler and the result should be checked.

## MSI Parameter

| | SP1 | SP2-4 |
|---|---|---|
| **MSI Parameter** | configDownloadSource<br>logLevel<br>saasDiscoDaysBack<br>usageAgentDisabled | configDownloadSource<br>logLevel<br>saasDiscoDaysBack<br>usageAgentDisabled |
| **Find-and-replace parameters for template.cfg** | configDownloadUser<br>configDownloadPassword<br>resultUploadCurlArgs<br>resultUploadMaxDelay<br>logFileSizeLimit<br>oracleUser<br>oraclePass<br>oracleSysDba<br>encryptionKey<br>configDownloadProxyUrl<br>configDownloadCurlArgs<br>configDownloadMaxDelay<br>resultUploadDestination<br>resultDirectory<br>resultUploadUser<br>resultUploadPassword<br>resultUploadProxyUrl | configDownloadUser<br>configDownloadPassword<br>logFileSizeLimit<br>encryptionKey<br>configDownloadProxyUrl<br>configDownloadCurlArgs<br>configDownloadMaxDelay |

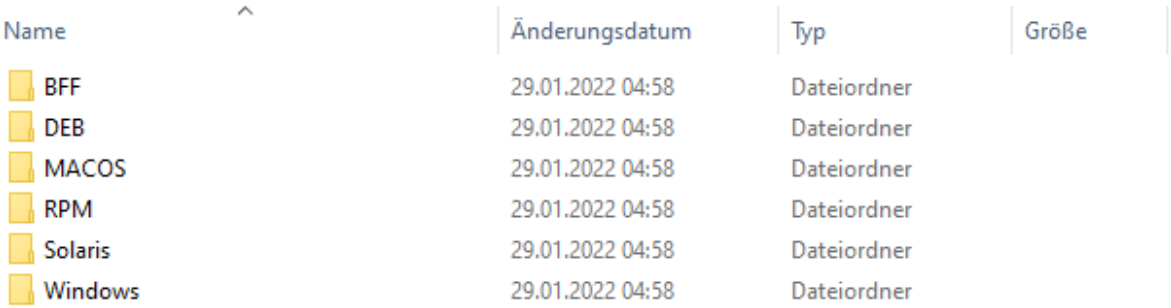
# RayVentory Inventory Agent for Non-Windows

## Installation and Configuration of RVIA for Non-Windows

This chapter describes how to install the RayVentory Inventory Agent (RVIA) on Linux/Unix machines.

### Source Files

After installing RayVentory Scan Engine all agent setups can be found in the installation directory of RayVentory Scan Engine. They are stored in the subfolder `{InstallDir}/Contrib/InventoryAgent`.




The following folder are available:

- BFF for AIX Unix
- DEB for Debian based Linux (x64 & x86)
- RPM for Red Hat based Linux (x64 & x86)
- MACOS for MacOS (x64 only)
- Solaris for Solaris (SPARC & x86)

### Installation

Copy the folder for the operating system on the target device. The command-line used to run the installation depends on the operating system. The following list shows example command-lines for each operating system.

- **AIX:** `sudo installp -aYF -d <package> rvia.rte`
- **DEB:** `sudo dpkg -i <package>`
- **MacOS:** `sudo installer -pkg <package> -target /`
- **RPM:** `sudo rpm -ivh <package>`
- **Solaris:** `sudo pkgadd -d <path/to/package> RVIA`

## Configuration

After installing the package, navigate to `/opt/rvia/`, which is the default installation path of the RayVentory agent. Run the following command with sudo:

```
./rvia getconfig <URL-to-CFG-file>
```

**Example:**

```
administrator@linuxdocker:/opt/rvia$ sudo ./rvia getconfig http://192.168.69.6
7:591/rviaconfig/default.cfg
```

This ensures that the agent immediately downloads the desired configuration from RayVentory Scan Engine. If this step is omitted, the Agent will not work within the environment.

**Note:**
The path to the configuration file in the command needs to be an exact match of the actual path to the configuration file. This is case sensitive! If the `getconfig` command fails with a curl error, check the URL. For example by calling the URL from a web browser on the target device.

Once `getconfig` has been executed successfully, `rvia.cfg` should contain a copy of the desired configuration file managed by RayVentory Scan Engine. Additionally, all agent schedules can be verified with the command `sudo crontab -l`.

## Manual Usage

It is possible to operate the agent manually to verify all the possible commands and navigate to the default install location (`/opt/rvia/`). Execute the command `sudo ./rvia help` to show all possible operation tasks:

```
administrator@linuxdocker:/opt/rvia$ sudo ./rvia help
usage: rvia <command>

Valid commands are:
help                        Shows this usage hint.
getconfig [source-host]     Download configuration.
inventory                   This will run an inventory.
oracle                      This will run an oracle database inventory.
schedule                    Apply the current schedule.
upload [destination-host]   Upload results.
encrypt [key]               Encrypt all usernames and passwords which are marked as blank.
```

It is possible to edit or validate the configuration file rvia.cfg directly on the Linux/Unix device. Open the rvia.cfg with any text editor using sudo permission.

```
  GNU nano 4.3                                rvia.cfg
configDownloadSource=http://win10manuel.mdkb.rms:591/rviaconfig/default.cfg
#configDownloadUser=
#configDownloadPassword=
#configDownloadProxyURL=
#configDownloadCurlArgs=
#configDownloadMaxDelay=
resultUploadDestination=http://win10manuel.mdkb.rms:591/
#resultDirectory=
#resultUploadUser=
#resultUploadPassword=
#resultUploadProxyURL=https://proxyUser:proxyUserPassword@172.16.12.10:8080/
#resultUploadCurlArgs=
#resultUploadMaxDelay=
#oracleUser=
#oraclePass=
#oracleSysDba=
#saasDiscoDaysBack=30dsadadasdsa
#usageWhitelist=\\winword\.exe
#usageWhitelist=\\powerpnt\.exe
#usageWhitelist=\\teams\.exe
#usageWhitelist=\\outlook\.exe
usageDisabled=True
#usageLogFileSize=
#usageEnableSessionLogging=
#usageSessionBackupPeriod=
#usageUploadPeriod=86400
#usageStartupDelay=
#usageMinRunTime=
```

More information about the different parameters usable in the file can be found in the Parameters chapter.

## Non-Default Inventory

The RayVentory agent relies on `ndtrack` for the collection of the inventory. Therefore it is possible to create custom inventory jobs using the ndtrack command-line switches.

Example:

```
schedule:inventory:-o IncludeDirectory=/opt -o ExcludeDirectory=/
opt -o ExcludeDirectory=/proc:30 0 * * *
```

More information on the ndtrack command-line options can be found here.

## Configuration Options Which Are Windows-Only

The following parameters and options are not used by Linux/Unix agents:

- `saasDiscoDaysBack`
- `usageWhitelist`
- `usageDisabled`
- `usageLogFileSize`

- usageEnableSessionLogging
- usageSessionBackupPeriod
- usageUploadPeriod
- usageStartupDelay
- usageMinRunTime
- usageProcessUpdatePeriod
- schedule:horizon::logon
- schedule:horizon::logoff
- schedule:saas:

## How to Execute RVIA as Non-Root

It is possible to execute RVIA as non-root by changing the permissions in retrospect. How this works is described in the following.

After the installation no cronjobs are setup - this is done manually when calling `rvia getconfig <url>`. Before calling RVIA for the first time, it is possible to modify the permission level for the installation directory to grant a specified user (for this example the user will be referred to as `<myuser>`) writing permissions for the `/opt/rvia` directory.

This user is now able to call `./rvia getconfig <url>` using the user context. RVIA will now load the `rvia.cfg` and will write the file to the installation directory with user permissions only. This also applies to the log file and the cronjobs in the Crontab of the user.

Using a custom command for the schedule that should be used by RVIA, it is possible that only ndtrack will be excuted using root permissions.

RVIA is executing the following steps when uploading inventories:

1. RVIA renames the files for the files to conform to Unicode.
2. RVIA reads the files.
3. RVIA uploads the files.
4. Afterwards, RVIA deletes the files.

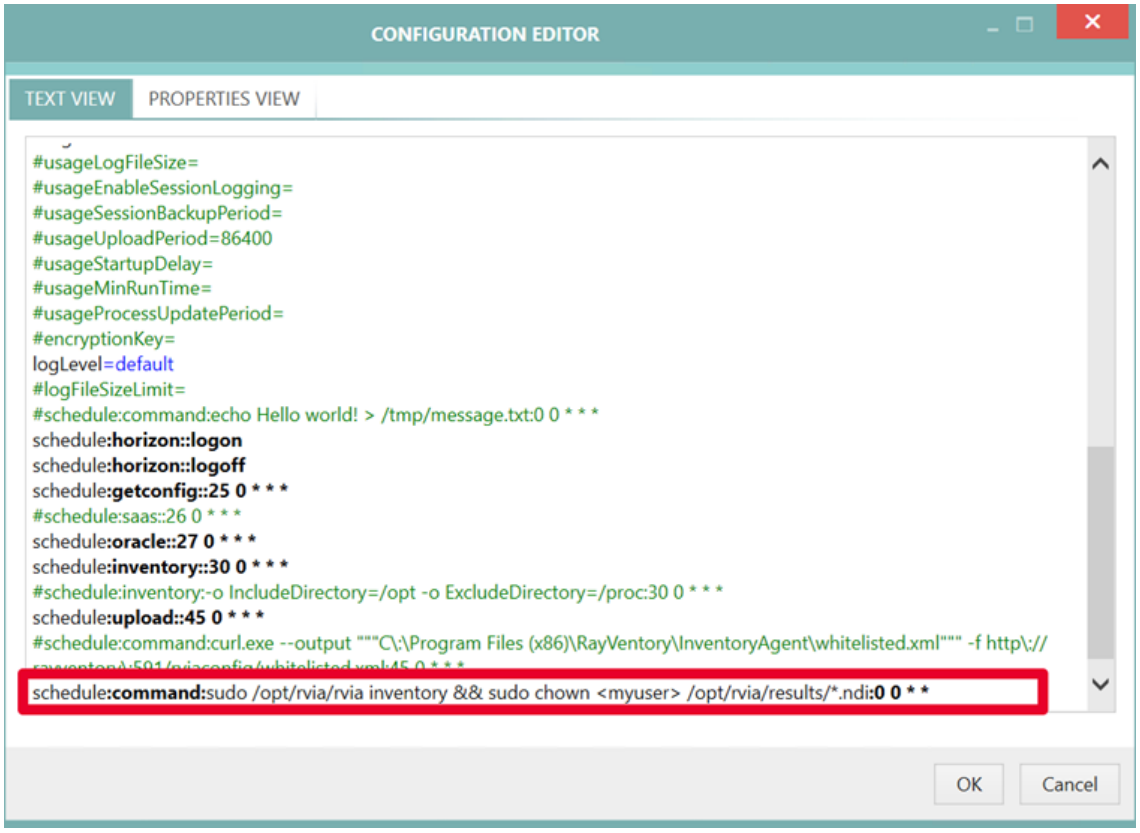
Using the custom command, this behavior will also be rectified to no longer require root permissions.

### Example Custom Command

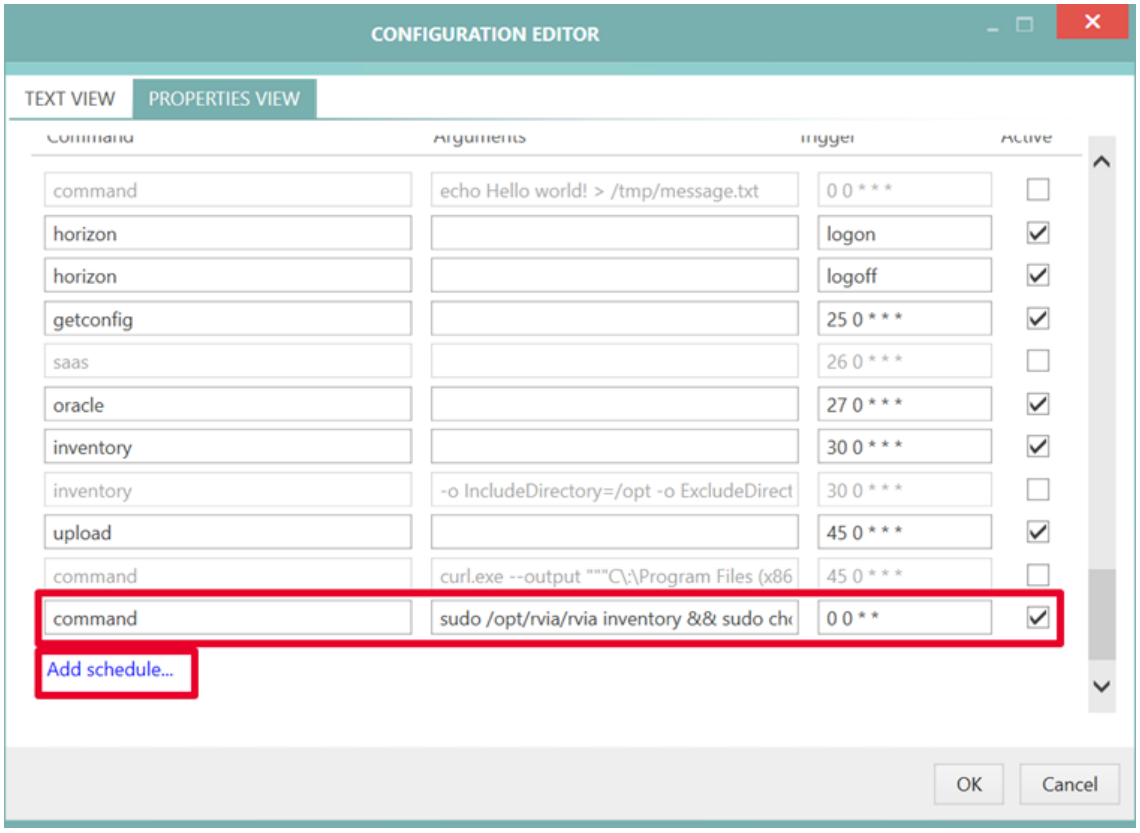
```
Schedule:command:sudo /opt/rvia/rvia inventory && sudo chown
<myuser> /opt/rvia/results/*.ndi:0 0 * *
```

The custom command can be entered using the RVSE interface. Go to **Settings** > **Inventory** > **Inventory Agent**. There select the configuration file and click on **Edit**.

The custom command can now be entered by either using the **TEXT VIEW** and directly entering it (**Screenshot 1**) or by using the **PROPERTIES VIEW** and using the **Add Schedule** option (**Screenshot 2**).




Screenshot 1




Screenshot 2

## Description

First the command `/opt/rvia/rvia` inventory will be executed with root permissions and after the inventory was executed successfully (`&&`) the command `chown` will be executed with root permissions. The user `<myuser>` will now be the owner of `*.ndi` files in the folder `/opt/rvia/results`. Therefore, RVIA will be able to rename and delete the files using user context.

## Important

It is necessary to edit the sudoers file (`/etc/sudoers`) to be able to execute the commands `/opt/rvia/rvia inventory` and `chown <myuser> /opt/rvia/results/*.ndi` without asking for a password.

The following lines need to be added to the sudoers file:

```
<myuser> ALL= (root) NOPASSWD: /opt/rvia/rvia inventory
<myuser> ALL =(root) NOPASSWD: /usr/bin/chown <myuser> /opt/rvia/
results/*.ndi
```

# Ports

The following table lists the ports and protocols used on the target clients:

| Device | Protocols | Ports |
|---|---|---|
| Linux/Unix | SSH | 22 |
| Windows | WMI/DCOM | 135 |
| vSphere | HTTPS/vSphere | 80; 443 |
| SNMP devices | SNMP | 161 |
| Windows | SMB | 139; 445 |
| Oracle DB machines | Zero Touch Oracle DB | 1521 + custom ports |
| Windows | WMI/DCOM | 1024 - 65535 (Can choose any local port to connect or set via group policy values.) |

The following table lists the ports and protocols used on the application server:

| Usage | Ports |
|---|---|
| WMI/DCOM | 1024 - 65535 (Can select either a local port to connect to or a value set via group policy.) |
| HTTP/HTTPS upload | 80; 443 |
| SMB upload | 139; 445 |

# Appendix II: Supported OS for the Different Inventory Methods

The following tables represents the list of the versions of the different operating systems on the different architectures which are supported by the different inventory methods:

## Windows (client)

| Version | Architecture | Portable / standalone / remote-execution | Zero-Touch (using WMI) | Inventory Agent (RVIA) |
|---|---|---|---|---|
| Windows XP ! | x86 | ✓* | ✓ | ⛔ |
| | x86-64 | ✓* | ✓ | ⛔ |
| Windows Vista ! | x86 | ✓* | ✓ | ⛔ |
| | x86-64 | ✓* | ✓ | ⛔ |
| Windows 7 ! | x86 | ✓* | ✓ | ✓* |
| | x86-64 | ✓* | ✓ | ✓* |
| Windows 8 ! | x86 | ✓* | ✓ | ✓* |
| | x86-64 | ✓* | ✓ | ✓* |
| Windows 8.1 ! | x86 | ✓* | ✓ | ✓* |
| | x86-64 | ✓* | ✓ | ✓* |
| Windows 10 | x86 | ✓* | ✓ | ✓* |
| | x86-64 | ✓* | ✓ | ✓* |
| | ARM64 | ✓* | ✓ | ✓* |
| Windows 11 | x86 | ✓* | ✓ | ✓* |
| | x86-64 | ✓* | ✓ | ✓* |
| | ARM64 | ✓* | ✓ | ✓* |

* - requires Microsoft Visual C++ Redistributable for Visual Studio 2015-2022
! - the system is not supported by Microsoft anymore

## Windows Server

| Version | Architecture | Portable / standalone / remote-execution | Zero-Touch (using WMI) | Inventory Agent (RVIA) |
|---|---|---|---|---|
| 2003 ! | x86 | ✓* | ✓* | ⛔ |
| | x86-64 | ✓* | ✓* | ⛔ |
| 2003 R2 ! | x86 | ✓* | ✓* | ⛔ |
| | x86-64 | ✓* | ✓* | ⛔ |
| 2008 ! | x86 | ✓* | ✓* | ✓* |
| | x86-64 | ✓* | ✓* | ✓* |
| 2008 R2 ! | x86-64 | ✓* | ✓* | ✓* |
| 2012 | x86-64 | ✓* | ✓* | ✓* |
| 2012 R2 | x86-64 | ✓* | ✓* | ✓* |
| 2016 | x86-64 | ✓* | ✓* | ✓* |
| 2019 | x86-64 | ✓* | ✓* | ✓* |
| 2022 | x86-64 | ✓* | ✓* | ✓* |

* - requires Microsoft Visual C++ Redistributable for Visual Studio 2015-2022
! - the system is not supported by Microsoft anymore

## RedHat Enterprise Linus (RHEL)

| Version | Architecture | Portable / standalone / remote-execution | Zero-Touch (using WMI) | Inventory Agent (RVIA) |
|---|---|---|---|---|
| 6 ! | x86 | ✓* | ✓* | ✓** |

| Version | Architecture | Portable / standalone / remote-execution | Zero-Touch (using WMI) | Inventory Agent (RVIA) |
|---|---|---|---|---|
| | x86-64 | ✓* | ✓* | ✓** |
| | PPC64 | ⛔ | ✓* | ⛔ |
| | PPC64le | ⛔ | ✓* | ⛔ |
| 7 | x86 | ✓* | ✓* | ✓** |
| | x86-64 | ✓* | ✓* | ✓** |
| | PPC64 | ✓* | ✓* | ⛔ |
| | PPC64le | ⛔ | ✓* | ⛔ |
| | ARM64 | ⛔ | ✓* | ⛔ |
| 8 | x86 | ✓* | ✓* | ✓** |
| | x86-64 | ✓* | ✓* | ✓** |
| | PPC64 | ✓* | ✓* | ⛔ |
| | PPC64le | ✓* | ✓* | ⛔ |
| | ARM64 | ⛔ | ✓* | ⛔ |

* - requires `sudo`
** - requires `sudo` and `curl` >= 7.19
! - the system is not supported by RedHat anymore

## SUSE Professional / Open SUSE

| Version | Architecture | Portable / standalone / remote-execution | Zero-Touch (using WMI) | Inventory Agent (RVIA) |
|---|---|---|---|---|
| 11 ! | x86 | ✓* | ✓* | ✓** |
| | x86-64 | ✓* | ✓* | ✓** |
| | PPC64 | ✓* | ✓* | ⛔ |
| | PPC64le | ✓* | ✓* | ⛔ |

| Version | Architecture | Portable / standalone / remote-execution | Zero-Touch (using WMI) | Inventory Agent (RVIA) |
|---|---|---|---|---|
| | ARM64 | ⛔ | ✅* | ⛔ |
| 12 ! | x86-64 | ✅* | ✅* | ✅** |
| | PPC64le | ✅* | ✅* | ⛔ |
| 15 | x86-64 | ✅* | ✅* | ✅** |
| | PPC64le | ✅* | ✅* | ⛔ |
| | ARM64 | ⛔ | ✅* | ⛔ |

* - requires `sudo`
** - requires `sudo` and `curl` >= 7.19
! - the system is not supported by SUSE anymore

## SUSE Enterprise Server (SLES)

| Version | Architecture | Portable / standalone / remote-execution | Zero-Touch (using WMI) | Inventory Agent (RVIA) |
|---|---|---|---|---|
| 11 ! | x86 | ✅* | ✅* | ✅** |
| | x86-64 | ✅* | ✅* | ✅** |
| | PPC64 | ✅* | ✅* | ⛔ |
| | PPC64le | ✅* | ✅* | ⛔ |
| | ARM64 | ⛔ | ✅* | ⛔ |
| 12 | x86-64 | ✅* | ✅* | ✅** |
| | PPC64le | ✅* | ✅* | ⛔ |
| 15 | x86-64 | ✅* | ✅* | ✅** |
| | PPC64le | ✅* | ✅* | ⛔ |
| | ARM64 | ⛔ | ✅* | ⛔ |

* - requires `sudo`
** - requires `sudo` and `curl` >= 7.19

! - the system is not supported by SUSE anymore

## CentOS

| Version | Architecture | Portable / standalone / remote-execution | Zero-Touch (using WMI) | Inventory Agent (RVIA) |
|---|---|---|---|---|
| 6 ! | x86 | ✓* | ✓* | ⛔ |
| | x86-64 | ✓* | ✓* | ⛔ |
| 7 | x86-x64 | ✓* | ✓* | ⛔ |
| | PPC64 | ✓* | ✓* | ⛔ |
| | PPC64le | ⛔ | ✓* | ⛔ |
| | ARM64 | ⛔ | ✓* | ⛔ |
| 8 ! | x86-x64 | ✓* | ✓* | ⛔ |
| | PPC64le | ✓* | ✓* | ⛔ |
| | ARM64 | ⛔ | ✓* | ⛔ |

* - requires `sudo`
! - the system is not supported by the CentOS Project anymore

## Debian

| Version | Architecture | Portable / standalone / remote-execution | Zero-Touch (using WMI) | Inventory Agent (RVIA) |
|---|---|---|---|---|
| 8 ! | x86 | ✓* | ✓* | ✓** |
| | x86-64 | ✓* | ✓* | ✓** |
| | PPC64le | ✓* | ✓* | ⛔ |
| | ARM64 | ⛔ | ✓* | ⛔ |

| Version | Architecture | Portable / standalone / remote-execution | Zero-Touch (using WMI) | Inventory Agent (RVIA) |
|---|---|---|---|---|
| 9 ! | x86-x64 | ✓* | ✓* | ✓** |
| | PPC64 | ✓* | ✓* | ✓** |
| | PPC64le | ✓* | ✓* | ⛔ |
| | ARM64 | ⛔ | ✓* | ⛔ |
| 10 | x86-x64 | ✓* | ✓* | ✓** |
| | PPC64 | ✓* | ✓* | ✓** |
| | PPC64le | ✓* | ✓* | ⛔ |
| | ARM64 | ⛔ | ✓* | ⛔ |
| 11 | x86-x64 | ✓* | ✓* | ✓** |
| | PPC64 | ✓* | ✓* | ✓** |
| | PPC64le | ✓* | ✓* | ⛔ |
| | ARM64 | ⛔ | ✓* | ⛔ |

* - requires `sudo`
** - requires `sudo` and `curl` >= 7.19
! - the system is not supported by Debian anymore

## Ubuntu

| Version | Architecture | Portable / standalone / remote-execution | Zero-Touch (using WMI) | Inventory Agent (RVIA) |
|---|---|---|---|---|
| 14.04 ! | x86 | ✓* | ✓* | ✓** |
| | x86-x64 | ✓* | ✓* | ✓** |
| | PPC64le | ✓* | ✓* | ⛔ |
| 16.04 ! | x86 | ✓* | ✓* | ✓** |
| | x86-x64 | ✓* | ✓* | ✓** |

| Version | Architecture | Portable / standalone / remote-execution | Zero-Touch (using WMI) | Inventory Agent (RVIA) |
|---|---|---|---|---|
| | PPC64le | ✓* | ✓* | ⛔ |
| 18.04 | x86 | ✓* | ✓* | ✓** |
| | x86-x64 | ✓* | ✓* | ✓** |
| | PPC64le | ✓* | ✓* | ⛔ |
| 20.04 | x86-x64 | ✓* | ✓* | ✓** |
| | PPC64le | ✓* | ✓* | ⛔ |
| | ARM64 | ⛔ | ✓* | ⛔ |
| 21.10 | x86-x64 | ✓* | ✓* | ✓** |
| | PPC64le | ✓* | ✓* | ⛔ |
| | ARM64 | ⛔ | ✓* | ⛔ |
| 22.04 | x86-x64 | ✓* | ✓* | ✓** |
| | PPC64le | ✓* | ✓* | ⛔ |
| | ARM64 | ⛔ | ✓* | ⛔ |

* - requires `sudo`
** - requires `sudo` and `curl` >= 7.19
! - the system is not supported by Canonical anymore

# Fedora

| Version | Architecture | Portable / standalone / remote-execution | Zero-Touch (using WMI) | Inventory Agent (RVIA) |
|---|---|---|---|---|
| 21 ! | x86 | ✓* | ✓* | ⛔ |
| | x86-x64 | ✓* | ✓* | ⛔ |
| | ARM64 | ⛔ | ✓* | ⛔ |
| 22 ! | x86 | ✓* | ✓* | ⛔ |

| Version | Architecture | Portable / standalone / remote-execution | Zero-Touch (using WMI) | Inventory Agent (RVIA) |
|---|---|---|---|---|
| | x86-x64 | ✓* | ✓* | ⛔ |
| | ARM64 | ⛔ | ✓* | ⛔ |
| 23 ! | x86 | ✓* | ✓* | ⛔ |
| | x86-x64 | ✓* | ✓* | ⛔ |
| | ARM64 | ⛔ | ✓* | ⛔ |
| 24 ! | x86 | ✓* | ✓* | ⛔ |
| | x86-x64 | ✓* | ✓* | ⛔ |
| | ARM64 | ⛔ | ✓* | ⛔ |
| 25 ! | x86 | ✓* | ✓* | ⛔ |
| | x86-x64 | ✓* | ✓* | ⛔ |
| | ARM64 | ⛔ | ✓* | ⛔ |
| 26 ! | x86-x64 | ✓* | ✓* | ⛔ |
| | ARM64 | ⛔ | ✓* | ⛔ |
| 27 ! | x86-x64 | ✓* | ✓* | ⛔ |
| | ARM64 | ⛔ | ✓* | ⛔ |
| 28 ! | x86-x64 | ✓* | ✓* | ⛔ |
| | ARM64 | ⛔ | ✓* | ⛔ |
| 29 ! | x86-x64 | ✓* | ✓* | ⛔ |
| | ARM64 | ⛔ | ✓* | ⛔ |
| 30 ! | x86-x64 | ✓* | ✓* | ⛔ |
| | ARM64 | ⛔ | ✓* | ⛔ |
| 31 ! | x86-x64 | ✓* | ✓* | ⛔ |
| | ARM64 | ⛔ | ✓* | ⛔ |
| 32 ! | x86-x64 | ✓* | ✓* | ⛔ |

| Version | Architecture | Portable / standalone / remote-execution | Zero-Touch (using WMI) | Inventory Agent (RVIA) |
|---|---|---|---|---|
| | ARM64 | ⛔ | ✅* | ⛔ |
| 33 ! | x86-x64 | ✅* | ✅* | ⛔ |
| | ARM64 | ⛔ | ✅* | ⛔ |
| 34 ! | x86-x64 | ✅* | ✅* | ⛔ |
| | ARM64 | ⛔ | ✅* | ⛔ |
| 35 | x86-x64 | ✅* | ✅* | ⛔ |
| | ARM64 | ⛔ | ✅* | ⛔ |
| 36 | x86-x64 | ✅* | ✅* | ⛔ |
| | ARM64 | ⛔ | ✅* | ⛔ |

* - requires `sudo`
! - the system is not supported by the Fedora Project anymore

## macOS

| Version | Architecture | Portable / standalone / remote-execution | Zero-Touch (using WMI) | Inventory Agent (RVIA) |
|---|---|---|---|---|
| 10.12 ! | x86-x64 | ✅* | ✅* | ✅** |
| 10.13 ! | x86-x64 | ✅* | ✅* | ✅** |
| 10.14 ! | x86-x64 | ✅* | ✅* | ✅** |
| 10.15 | x86-x64 | ✅* | ✅* | ✅** |
| 11 | x86-x64 | ✅* | ✅* | ✅** |
| | ARM64 (Rosetta 2) M1 | ✅* | ✅* | ✅** |
| 12 | x86-x64 | ✅* | ✅* | ✅** |

| Version | Architecture | Portable / standalone / remote-execution | Zero-Touch (using WMI) | Inventory Agent (RVIA) |
|---|---|---|---|---|
| | ARM64 (Rosetta 2) M1 | ✓* | ✓* | ✓** |

* - requires `sudo`
** - requires `sudo` and `curl` >= 7.19
! - the system is not supported by Apple anymore

## Solaris

| Version | Architecture | Portable / standalone / remote-execution | Zero-Touch (using WMI) | Inventory Agent (RVIA) |
|---|---|---|---|---|
| 10 | SPARC | ✓* | ✓* | ✓** |
| | x86-x64 | ✓* | ✓* | ✓** |
| 11 | SPARC | ✓* | ✓* | ✓** |
| | x86-x64 | ✓* | ✓* | ✓** |

* - requires `sudo`
** - requires `sudo` and `curl` >= 7.19

## AIX

| Version | Architecture | Portable / standalone / remote-execution | Zero-Touch (using WMI) | Inventory Agent (RVIA) |
|---|---|---|---|---|
| 6.1 ! | PPC64 | ✓* | ✓* | ✓** |
| 7.1 | PPC64 | ✓* | ✓* | ✓** |
| 7.2 | PPC64 | ✓* | ✓* | ✓** |
| 7.3 | PPC64 | ✓* | ✓* | ✓** |

* - requires `sudo`
** - requires `sudo` and `curl` >= 7.19
! - the system is not supported by IBM anymore

# HP-UX

HP-UX

| Version | Architecture | Portable / standalone / remote-execution | Zero-Touch (using WMI) | Inventory Agent (RVIA) |
|---|---|---|---|---|
| 11i ! | PARISC | ✓* | ✓* | ⛔ |
| | Itanium | ✓* | ✓* | ⛔ |
| 11i v2 | PARISC | ✓* | ✓* | ⛔ |
| | Itanium | ✓* | ✓* | ⛔ |
| 11i v3 | PARISC | ✓* | ✓* | ⛔ |
| | Itanium | ✓* | ✓* | ⛔ |

* - requires `sudo`
! - the system is not supported by HP anymore

# Supported Configurations for RVIA

The following table represents the configurations that are supported by RVIA.

| Operating System | Minimum Version | Architecture | Additional Dependencies |
|---|---|---|---|
| Microsoft Windows | 7 | x86, x86_64 | Visual C++ Redistributable for Visual Studio 2015-2022 |
| RedHat Enterprise Linux (and compatible distributions) | 6 | x86, x86_64 | sudo, curl >= 7.19 |
| Debian GNU/Linux | 8 | x86, x86_64 | sudo, curl >= 7.19 |
| Ubuntu Linux | 14.04 | x86, x86_64 | sudo, curl >= 7.19 |
| SUSE Linux Enterprise | 11 | x86, x86_64 | sudo, curl >= 7.19 |
| IBM AIX | 6.1 | ppc64 | sudo, curl >= 7.19 |
| Oracle Solaris | 10 | x86, x86_64 | sudo, curl >= 7.19 |
| Apple macOS | 10.12 | x86_64, arm64 | sudo, curl >= 7.19 |

Dependencies on Linux systems are set as package dependencies. The package management systems on UNIX systems do not offer setting package dependencies, but sudo is still required for first time configuration of RVIA and curl is required for uploading inventories and downloading schedules.

RayVentory Scan Engine
is part of the
RaySuite

More information online
www.raynet.de

Raynet GmbH
Technologiepark 22
33100 Paderborn, Germany
T +49 5251 54009-0
F +49 5251 54009-29
info@raynet.de

www.raynet.de