Technology Alliance Program (TAP):

Quick Start Guide
for N-central

N-ABLE™

Table of Contents

Welcome

This guide is designed to assist TAP members interested in integrating with N-able™ N-central®.

In this document, we provide links to important resources and training, instructions on getting started with N-central, and highlighting the most common features used by other TAP members.

Our goal is to empower your success, starting with a strong foundation. Should you have any questions, please don't hesitate to contact us at TAP@n-able.com.

Helpful links

- N-central Server: integration3.n-able.com

- Username: ‹your email address›

- Password: ‹will be emailed to you›

- Product Doc: http://documentation.n-able.com/N-central/documentation/Content/Introduction-GettingStarted/GetStarted_Home.htm

- API Doc: http://documentation.N-able.com/N-central/documentation/Content/Further_Reading/API_Level_Integration/API_Integration_Introduction.html

- Product Training: https://success.n-able.com/ and select **MSP Institute**

  - You can access N-central training content inside the Institute

Getting started

- Your login provides you the same access as an MSP or IT pro

- The dashboard hierarchy is Service Organization (SO), Customer, Sites

- Upon the first login, the wizard will guide you to Create a Customer
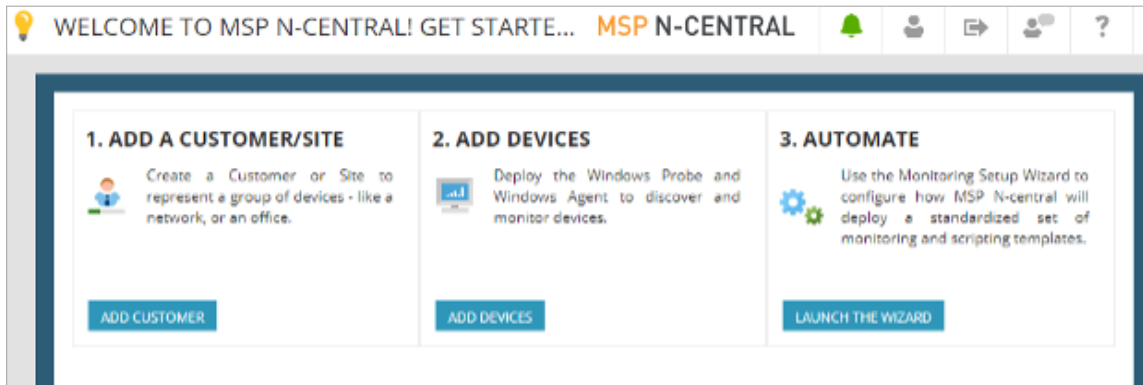
- Leave everything default, including blank network credentials

- Now you want to install an agent on a single device

    - Select **More Options**

    - Select **Download the Agent Software for Your OS**

- Probes are used in domain environments to discover the network and install N-central agents.

- Now that the agent is installed, you can investigate the typical features used by TAP members including:

    - Software deployment

    - Software management using scripts

    - Monitoring

    - Running automation policies (PowerShell® scripts created using automation manager application)

Software deployment

- Software packages can be stored on the N-central server, kept on a local network share, or downloaded from a URL

- To install software on a single device:

    - Select the **All Devices View**

    - Put a **check** beside the device

    - Select **Push Third Party Software**
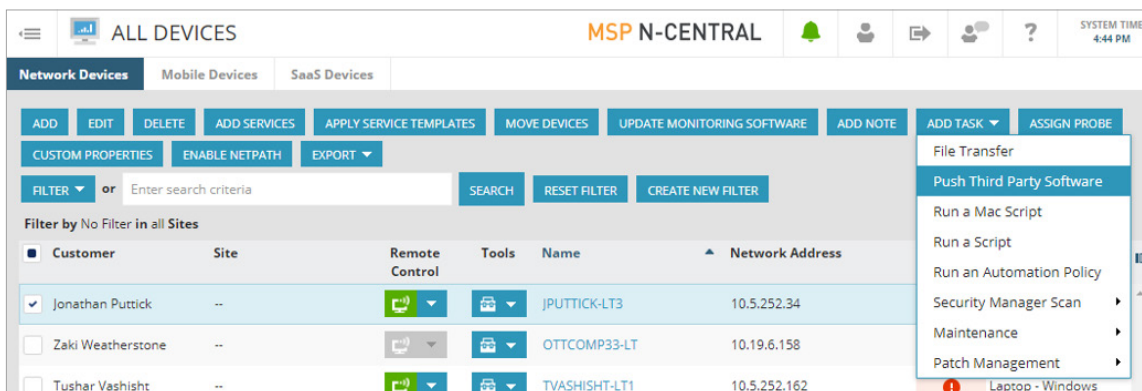
- You can create custom scripts to install software on devices. Supported script types:

    - PowerShell, Visual Basic, Batch, AppleScript®, BASH, AMP

- The amp script type is created using our automation manager, which is a drag-and-drop GUI to create PowerShell scripts.

- Automation Manager Scripting Allows:

    - Your agent deployment to managed devices (Windows®/macOS®)

    - Agent/device status monitoring

- To upload your scripts and applications to the N-central server:
  - Select the **Service Organization**
  - Select **Configuration › Scheduled Tasks › Script/Software Repository**
- Download the Automation Manager Designer from the N-central dashboard under **Service Organization level › Actions › Start Automation Manager.** Note the designer only runs on Windows.
- Sample automation policy to install software from URL: https://success.N-ablemsp.com/kb/N-able_n-central/Download-and-Install-Google-Chrome

Monitoring software applications

- N-central provides monitoring services used to monitor devices/applications:
  - Log file (appended), log file (batch)
  - Windows event logs, windows application logs
  - Windows service
  - Windows process
- You can also create custom monitors to query the device using SNMP, WMI, PowerShell script, or TCP port availability. (Training videos are available in the MSP Institute; search for N-able N-central: Custom Services)

Grouping the monitoring services

- The services above can be grouped in a service template
- Create the template under SO › Configuration › Monitoring › Service Templates

Automatic software deployment and monitoring

- The software deployment and monitoring can be automatically applied to new devices using N-central rules.
- Rules target groups of devices using filters and can install software, run scripts, and apply the monitoring to target specific customers—or every customer.
- Rules are created under **SO › Configurations › Monitoring › Rules**
- A great example of using all these features is defined in our AV Status monitoring service.
- success.n-able.com/kb/n-able_n-central/AV-Status-Monitoring

API integration

- You can use the API to query N-central database for customer and device data.

- Here's an example of a PSA integration using this API:

  - documentation.n-able.com/N-central/documentation/Content/Administration/PSA_Integration/Custom%20PSA/PSAG_ServiceNow_API_Script.htm?Highlight=scripts

- Here's a sample PowerShell script to extract data from N-central

```
Clear-Host
Clear

#
# Determine where the N-Central server is
# Uncomment this next line if you want the N-central URL to be a command line parameter
# $serverHost = Read-Host "Enter the fqdn of the N-Central Server "
 $serverHost = "MYNCENTRALSERVER.com"

#
# Generate a pseudo-unique namespace to use with the New-WebServiceProxy and
# associated types.
#
# By controlling the namespace, the script becomes portable and is not
# dependent upon the endpoint url the webservice is connecting.  However, this
# introduces another complexity because once the namespace is defined within a
# powershell session, it cannot be reused, nor can it be undefined.  As long as
# all the calls are made to the existing webserviceproxy, then everything would be
# OK. But, if you try to rerun the script without closing and reopening the
# powershell session, you will get an error.
#
# One way around this is to create a unique namespace each time the script is run.
# We do this by using the last 'word' of a GUID appended to our base namespace 'NAble'.
# This means our type names for parameters (such as T_KeyPair) now have a dynamic
# type.  We could pass types to each new-object call using "$NWSNameSpace.T_KeyPair",
# and I find it more readable to define our 'dynamic' types here and use the typenames
# in variables when calling New-Object.
#
$NWSNameSpace = "NAble" + ([guid]::NewGuid()).ToString().Substring(25)
$KeyPairType = "$NWSNameSpace.T_KeyPair"
$KeyValueType = "$NWSNameSpace.T_KeyValue"


#
# Create DeviceData and DeviceIdentifiers data types to hold info
#
Add-Type -TypeDefinition @"
public class DeviceIdentifiers {
        public string ID;
        }
"@


Add-Type -TypeDefinition @"
public class DeviceData {
        public string Name;
    public string Model;
    public string Manufacturer;
    public string SerialNumber;
    public string OS;
    public string Processor;
    public string TotalPhysicalMemory;
    public string HardDriveCapacity;
```

About N-able

N-able empowers managed services providers (MSPs) to help small and medium enterprises navigate the digital evolution. With a flexible technology platform and powerful integrations, we make it easy for MSPs to monitor, manage, and protect their end customer systems, data, and networks. Our growing portfolio of security, automation, and backup and recovery solutions is built for IT services management professionals. N-able simplifies complex ecosystems and enables customers to solve their most pressing challenges. We provide extensive, proactive support—through enriching partner programs, hands-on training, and growth resources—to help MSPs deliver exceptional value and achieve success at scale.

n-able.com

Contact us

If you have any questions, please contact us at TAP@n-able.com.