Attune Documentation

ServerTribe

Mar 08, 2024

DOCUMENTATION CONTENTS

1 Why Attune **3**

2 First steps **5**

3 Getting Help **7**

4 Engage the community **9**

5 How the documentation is organised **11**

5.1 Getting Started 11
5.2 Using Attune 20
5.3 "How-to" guides 30
5.4 Reference guides 89
5.5 FAQ 98
5.6 Glossary 101
5.7 Release Notes 101

Everything you need to know about Attune.

CHAPTER

ONE

WHY ATTUNE

Attune by ServerTribe streamlines and orchestration digital systems with:

- Intuitive script, command, and process automation, mirroring manual inputs.
- Easy development of custom automation solutions.
- Compatibility with popular scripting languages.
- Quick automation process construction.
- IP portability and sharing capabilities.
- Comprehensive multi-server orchestration.
- Centralised scheduling system.
- Automated documentation generation.

Designed for flexibility, Attune serves as a centralised, self-documenting platform for reusable processes, files, and knowledge. It's versatile enough for enterprise-grade infrastructure and home networks alike.

CHAPTER

TWO

FIRST STEPS

Are you new to Attune or to scripting? This is the place to start!

- **From scratch:** *Getting Started*
- **Install Attune for free:** Attune Community Edition
- **Tutorial:** *Linux or macOS*

CHAPTER

THREE

GETTING HELP

Having trouble? We'd like to help!

- Try the *FAQ* - it's got answers to many common questions.
- Connect to our Discord Community.
- Report bugs with Attune in our ticket tracker.

CHAPTER

FOUR

ENGAGE THE COMMUNITY

- Discord
- YouTube
- Github

CHAPTER

FIVE

HOW THE DOCUMENTATION IS ORGANISED

Attune has a lot of documentation. A high-level overview of how it's organised will help you know where to look for certain things:

- *Tutorials* take you by the hand through a series of steps to create jobs. Start here if you're new to Attune or scripting. Also look at the "*First steps*".
- *Topic guides* discuss key topics and concepts at a fairly high level and provide useful background information and explanation.
- *Reference guides* contain technical reference of Attune. They describe how it works and how to use it but assume that you have a basic understanding of key concepts.
- *How-to guides* are recipes. They guide you through the steps involved in addressing key problems and use-cases. They are more advanced than tutorials and assume some knowledge of how Attune works.

5.1 Getting Started

New to Attune? Or to scripting in general? Well, you came to the right place: read this material to quickly get up and running.

Explore Attune with our brief 7-minute demonstration video:

Video Demonstration of creating an automated process in Attune.

If you'd like a live online demonstration of Attune and the opportunity to ask questions, please contact ServerTribe.

Continue to learn about Attune's *Design, Plan, and Run Framework*

or jump straight into a tutorial: *Linux or macOS*

5.1.1 Design, Plan, and Run Framework

Attune is divided into 3 workspaces:

- Design,
- Plan, and
- Run.

Design

The *Design Workspace* is where you create your automated and reusable building blocks.

Plan

The *Plan Workspace* is where you'll plan a job to run a blueprint and which values are connected to the parameters in the blueprint. A single blueprint can be used for many jobs with different values.

Schedules can be planned to orchestrate multiple Jobs.

Run

The *Run Workspace* is where you control your Jobs, Debug your jobs and view your historical logs.

Take a moment to view our quick 2-minute overview of the Design, Plan, and Run Framework:

> **Video** Intro to Attune in 2 Minutes

5.1.2 Tutorials

The following tutorials take you by the hand through a series of steps to create jobs.

Creating your first Linux or macOS Project

Let's learn by example.

Throughout this tutorial, we'll walk you through the creation and running of an automated process.

It'll consist of three parts:

1. Design
2. Plan
3. Run

We'll assume you have Attune installed already. This tutorial is written for Attune v23.7.3 targeting a Linux or macOS machine.

Note: If you're having trouble going through this tutorial, please head over to the *Getting Help* section.

Part 1 - Design

Create a project

In the Attune application, navigate to the Design workspace and create a new project.

1. Open the **Project** dropdown
2. Select **New Project** at the bottom of the list
3. Choose **Create blank project**
4. Enter a **Project Name** such as: `Linux Tutorial`
5. Select **Create**

6. Select **Switch Project**

This will create and select the new Project. Verify that the new project is selected in the Project dropdown.

Create parameters

Open the **Parameters** drawer in the Design workspace and create the parameters listed in the table below.

Table 1: Parameters

Type	Name
Linux/Unix Node	`Tutorial Node`
Linux/Unix Credential	`Tutorial User`
Basic Node	`Tcp Ping Node`
Text	`Continue Tutorial`

Close the **Parameters** drawer when you're finished.

Create files

Open the **Files** drawer in the Design workspace and create the two files as per below.

Firstly, create a version controlled file:

1. Select **Create File Archive**
2. Set the **Type**: `Version Controlled Files`
3. Enter the **Name**: `Tutorial Config`
4. Select **Create**

Create a file within the file you created:

1. Select the root folder `./`
2. Select **Create File**
3. Enter the **Name**: `config.cfg`
4. Select **Create**
5. Select the `config.cfg` you created
6. Enter the code below

```
Tutorial Host: {node.hostname}
Tutorial User: {credentials.user}
```

7. Select **Save**

Secondly, create a large file archive:

1. Select **Create File Archive**
2. Set the **Type**: `Version Controlled Files`

3. Enter the **Name**: `Tutorial Large File`
4. Select **Create**
5. Drag a file and drop it on the **Replace Contents** section.

Close the **Files** drawer when you're finished.

Create a blueprint

Create a new blueprint.

1. Select **Create Blueprint**
2. Set the Blueprint **Name**: `Linux Tutorial Blueprint`
3. Select **Create**

Create the steps

Execute Linux script

Select the blueprint you created. On the step select the **Create Step** Option.

Note: The following script will intentionally fail to later demonstrate fixing a running job.

Script:

```
echo "The operation failed."

exit 1
```

Select **Save**

Note: At this point you can skip to *Part 2*.

Execute Linux script with responses

Select the previous step you created. On the step select the **Create Step** Option.

Table 2: Execute Linux Script With Responses

Type	`Execute Linux Script With Responses`
Name	`Linux Tutorial execute linux step with responses`
Target Node	`Tutorial Node`
Credential	`Tutorial User`
Prompt Response	`Continue?==={continueTutorial}`

Script:

```
echo "Continue?"
read response

lowercase_string=$(echo "$response" | tr '[:upper:]' '[:lower:]')

if [ "$lowercase_string" == "yes" ] || [ "$lowercase_string" == "y" ]; then
    echo "You responded: {continueTutorial}!"
else
    echo "You responded: {continueTutorial}! This step will stop!"
    exit 1
fi
```

Select **Save**

Tcp ping

Select the previous step you created. On the step select the **Create Step** Option.

Table 3: Tcp Ping

Type	`Tcp Ping`
Name	`Linux Tutorial tcp ping`
Target Node	`Tcp Ping Node`

1. Select **Save**
2. Drag the step to the beginning of the Blueprint

Group step

Select the last step in the Blueprint. On the step select the **Create Step** Option.

Table 4: Group Step

Type	`Group Step`
Name	`Linux Tutorial group step`

Select **Save**

Push files

Select the group step in the Blueprint. On the step select the **Create Step** Option. Select **> Under** to create the step in the group step.

Table 5: Push Files

Type	`Push Files`
Name	`Linux Tutorial push files`
Target Node	`Tutorial Node`
Credential	`Tutorial User`
Files	`Tutorial Large File`
Remote Path	`~/attuneTutorial/`

Select **Save**

Push compiled files

Select the previous step you created. On the step select the **Create Step** Option.

Table 6: Push Compiled Files

Type	`Push Compiled Files`
Name	`Linux Tutorial push compiled files`
Target Node	`Tutorial Node`
Credential	`Tutorial User`
Files	`Tutorial Config`
Remote Path	`~/attuneTutorial/`

Check the **config.cfg** file.

Table 7: config.cfg

Mako Parameter	Type	Attune Parameter
`node`	`Linux/Unix Node`	`Tutorial Node`
`credentials`	`Linux/Unix Credential`	`Tutorial User`

1. Select **Test Dynamic File Generation** and verify there are no errors in the Mako script
2. Select **Save**

Cleanup the files

Select the previous step you created. On the step select the **Create Step** Option.

Script:

```
directory="~/attuneTutorial/"

if [ -d $directory ]; then
    cd $directory
    echo "Listing all files in $directory:"
    ls -ltrh
    echo "Displaying the contents of config.cfg:"
    cat config.cfg
    echo "Moving out of $directory to prepare for deletion."
    cd ~
    echo "Removing the directory $directory and its contents."
    rm -rf $directory
else
    echo "$directory doesn't exist!"
fi
```

Select **Save**

Link existing step

Select the group step in the Blueprint. On the step select the **Link Existing Step** Option. Select **V After** to link a step after the group step.

1. Set **Step**: `Linux Tutorial execute linux step`
2. Select **Link**

Commit your project changes

1. Select **Commit**
2. Set the **Commit Message**: `Created Linux Tutorial Blueprint`
3. Select **Commit**

Congratulations, you have created your Blueprint.

Part 2 - Plan

In the Attune application, navigate to the Plan workspace.

Create values

Open the **Values** drawer in the Design workspace and create the three values required for the job.

Linux/Unix Node

1. Select **Create Value**
2. Populate the form

Table 8: Linux/Unix Node

Type	Linux/Unix Node
Name	{name of your device}
IP Address	{device IP address}
Hostname	{hostname}
Domain Name	{domain name}

3. Select **Create**

Linux/Unix Credential

1. Select **Create Value**
2. Populate the form

Table 9: Linux/Unix Credential

Type	Linux/Unix Credential
Name	{name of your user}
User	{username}

3. Select **Create**
4. Find the value and update the **Password**

Basic Node

1. Select **Create Value**
2. Populate the form

Table 10: Linux/Unix Node

Type	Basic Node
Name	`search.brave.com`
IP Address	`search.brave.com`
Hostname	`search`
Domain Name	`brave.com`

3. Select **Create**

Text

1. Select **Create Value**
2. Populate the form

Table 11: Text

Type	Text
Name	`yes`
Variable	`yes`

3. Select **Create**

Close the **Values** drawer when you're finished.

Create the tree

Create the tree of your environment.

1. Select **Create Tree**
2. Set the **Name**: {environment name}
3. Select **Create**
4. Expand the dropdown on the first node in your tree and select **Create Node**
5. Set the **Name**: {device hostname}
6. Select **Create**

Create the plan

Create the plan for your tutorial job.

1. Expand the dropdown on the device node and select **Create Plan**
2. Set the **Name**: `{hostname} Linux Tutorial Job`
3. Select **Create**
4. Select the plan you created
5. **Add Blueprint**
6. Select **Save**
7. Populate the Values

Table 12: Plan Inputs

Parameter	Value
Tutorial Node	select the device
Tutorial User	select the credentials
Tcp Ping Node	select the ping node `search.brave.com`
Continue Tutorial	select `yes`

8. Select **Save**

Congratulations, you have created your Plan.

Part 3 - Run

In the Attune application, navigate to the Run workspace.

1. Search for and select you Job
2. Select **Reset Job**
3. Select play

Fix the error

Fix the script on the failed step and resume the job.

1. Select the failed step **Linux Tutorial execute linux step**
2. Select **Edit Step**
3. Replace the **Script** with the following

```
echo "This step works!"
```

4. Select **Save**
5. Select **Run only this step**
6. Verify the step run successfully
7. Select **Run job from select step**

Reset the Job and archive the current logs

Review the logs for each steps then archive.

1. Inspect each step
2. Select **Reset the job and archive the current logs**
3. Select the **History** tab
4. Inspect the most recent log

Tutorial completed

Congratulations, you have completed the tutorial.

5.2 Using Attune

High-level topics and concept to help you create and deploy your best work:

5.2.1 Designing Automation Projects

Projects

Creating Projects

Ideally, you aim to initiate a Project with a defined purpose, crafting it as a modular building block for seamless integration into end-to-end automated workflows.

Use a name that describes the purpose of the Project. If the Project will be shared publicly, consider Search Engine Optimisation (SEO) when naming the Project. Attune generates the files for GitHub Pages to generate a website for your Project.

The following examples are Projects created to be modular building blocks:

- Create Windows ISO with autounattend built on Windows
- Create Windows ISO with autounattend built on macOS or Linux
- Create Red Hat Enterprise Linux RHEL with kickstart built on macOS or Linux
- Intel NUC Drivers
- MSI Drivers
- PiKVM APIs from Windows
- PiKVM APIs from macOS or Linux
- VMWare ESXi APIs from macOS or Linux
- oVirt APIs from macOS or Linux
- PiKVM Windows Installer from a Windows Machine

The following examples link Blueprints from the above Projects to automate an end-to-end process:

- PiKVM Windows Installer from a Windows Machine
- VMware ESXi Red Hat Enterprise RHEL Installer from a macOS or Linux Machine

This promotes re-usability and makes it easier to update and maintain specific pieces of functionality.

Project Descriptions

Use the Project description to describe the purpose of the Project. A quality Project description encourages collaboration, maintainability, and knowledge transfer.

Project Management

You can version control your Projects, ensure you frequently commit your changes. Publish and push your commits to Git repositories. Leverage publicly shared Projects such as Attune Automation on GitHub.

Blueprints

Creating Blueprints

Blueprints complete a required process. Make best effort to use Parameters so that you minimise the number of Blueprints to maintain. Use a name that describes the process being performed.

Blueprint Descriptions

Use the Blueprint description to describe the process performed by the Blueprint. A quality Blueprint description encourages collaboration, maintainability, and knowledge transfer.

Steps

Break down your script into small, discrete steps, each dedicated to performing a specific action. Additionally, ensure that your scripts are designed to be idempotent, meaning they can be run multiple times without causing unexpected side effects. This approach not only enhances the clarity of your code but also simplifies the debugging process, making it more efficient and less error-prone.

Parameters

Using Parameters in Steps and Files greatly increases the flexibility of the Blueprints. Ensure that names are intuitive and that comments clearly describe the expected value to help users when populating the Plans.

5.2.2 Automate Operating System Installation

Attune provides a solution to streamline the OS installation process. It works across Windows and Linux on many physical or virtual controllers, such as:

- iDrac
- PiKVM
- oVirt
- HyperV
- VirtualBox
- Proxmox
- ESXi

Or you can create bootable flash drives if you don't have access to a controller.

Getting Started

Setup the Worker

A worker is key in creating ISO files. Workers handle tasks. These tasks include unpacking ISO files. They also involve adding drivers and settings.

You will send files to the controller and call APIs.

The worker can be on macOS, Linux, or Windows. Attune Projects work with Windows, macOS, and Linux workers.

Here is a list of the setup projects to clone:

- Setup Windows to Automate Operating System Installations
- Setup macOS or Linux to Automate Operating System Installation

Follow the instructions within the project to setup your worker.

Understanding the process

Two parts make up the OS installation automation.

1. *Creating the ISO files* and
2. *Building the Machine*.

The two parts merged into Glue Projects, performing a unique process.

Creating the ISO(s)

To make an ISO image, set up separate Attune Projects. Each OS and worker type needs Blueprints for ISO creation.

The ISO creation process is controller-independent. No controller-specific details needed.

Tasks needing controller info go in Controller APIs Project.

- Create Windows ISO with autounattend built on Windows
- Create Windows ISO with autounattend built on macOS or Linux
- Create Red Hat Enterprise Linux RHEL with kickstart built on macOS or Linux

Project Naming

The following way names the projects to create the ISO's.

`Create {os_name} ISO with {method} built on {worker_type}.`

`os_name` examples:

- Red Hat Enterprise Linux RHEL
- Windows
- Ubuntu
- Debian

`method` examples:

- Kickstart
- Preseed
- autoinstall
- autounattend

`worker_type` examples:

- Windows
- macOS or Linux

Parameters

This Project will need a worker to build the ISO image. This worker can be the device Attune is running on or another device. We will name this device:

`Automation Worker {type}`

We will name the parameters for the new operating system that we are creating:

`New OS {type}`

`type` examples:

- Node
- User: {account such as 'root', 'Administrator', or 'non privileged'}
- Base Directory

Blueprints

We will name the Blueprint:

`Create {os_distribution} {installation_method} ISO.`

`os_distribution` examples:

- Red Hat Enterprise Linux 8 (RHEL8)
- Windows 10 (Win10)
- Windows Server 2019 (Win2019)

`installation_method` (for Windows only) examples:

- Single
- Dual
- WinPE

Attaching Drivers

The device drivers and ISO creation projects work independently. Drivers go into a drop directory before ISO creation. When creating an ISO, it copies drivers into it.

- Intel NUC Drivers
- MSI Drivers

Create new projects for your devices drivers and add to this growing list.

Test and Cleanup

The Project will include a Blueprint to test the installation. It will also include a Blueprint to clean the build files on the worker.

Building the Machine

In this section, we'll set up distinct Attune Projects for each controller. To ensure Blueprints work on any operating system, we can also make a bootable flash disk.

We'll name these projects and specify details. These include the device running scripts, the controller, host machine, and build machine. Additionally, we'll include information on CPU count and boot ISO directory.

Our goal is to make Blueprints in the Controller APIs Project compatible with any OS. Yet, Linux and Windows differ. So, complete OS agnosticism may not always be possible.

- PiKVM APIs from Windows
- PiKVM APIs from macOS or Linux
- VMWare ESXi APIs from macOS or Linux
- oVirt APIs from macOS or Linux
- Windows Storage cmdlets

Project Naming

The Projects perform processes on machine controllers and virtual hosts. Here are their names:

`{controller} APIs from {worker_type}`

`controller` examples:

- iDrac
- PiKVM
- VMWare ESXi

`worker_type` examples:

- Windows
- macOS or Linux

Or, the Project Name for creating bootable USB disks will differ.

Parameters

This Project may need a device to run scripts and send commands to the controller. This device can be the Attune Server or another device. We will name this device:

`Automation Worker {type}`

Name the controller or host machine:

`Controller {type}`

We will name the machine that we are going to build or rebuild:

`Build Machine {type}`

`type` examples:

- Node
- ESXi Host
- vCenter Node
- User: {account such as 'root' or 'non privileged'}
- CPU Count
- Boot ISO Directory

Putting it all Together (Glue Projects)

This last part is about glue projects. They help make Plans for building machines easier. The key is setting parameters and removing duplicates.

Project Naming

The Projects perform the whole workflow. They install an OS on a machine. The following names are:

`{controller} {os_name} Installer from a {worker_type} Machine`

`controller` examples:

- iDrac
- PiKVM
- VMWare ESXi

`os_name` examples:

- Red Hat Enterprise Linux RHEL
- Windows
- Ubuntu
- Debian

`worker_type` examples:

- Windows
- macOS or Linux

Parameters

When linking Blueprints into the Glue Project, set Parameters in advance. Also, combine any duplicate Parameters.

For an example of a Glue Project: PiKVM Windows Installer from a Windows Machine

Wrapping Up: Streamlining OS Installation

This framework automates installing different OS with Attune Projects. It covers creating ISO images, setting up machines, and integrating components. It focuses on clear naming, scalability, and system-agnostic design. This approach simplifies OS installations, making them efficient and reliable. Users get automated installations on various hardware and virtual setups.

5.2.3 Boolean Best Practices

It is best to use **booleans** in your scripts rather than using an integer to check for the truth of a statement. The reason for this is that it cleans up the code and makes it more intuitive when mapping values to parameter.

For example, if a parameter is called **Create Full Backup**, `true` makes more sense than `1`

Bash

`true` and `false` are native to Mac and Linux operating systems, this means variable can be evaluated without assigning integers.

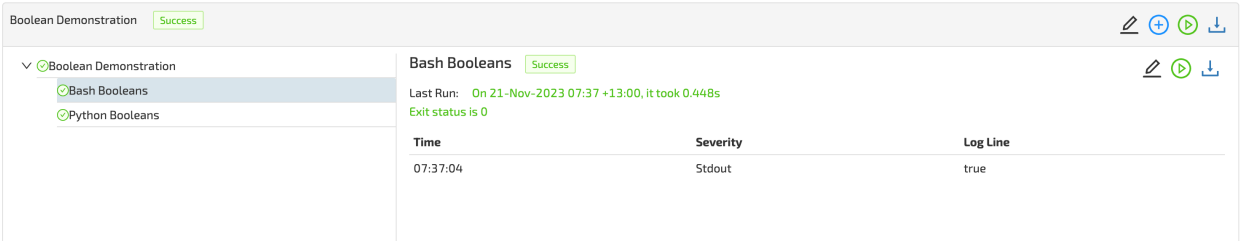
For example:

```
if {attuneVar}
then
    echo "true"
fi
```

Attune substitutes the a value into the `{attuneVar}` parameter.

If the value is `true` :

```
if true
then
    echo "true"
fi
```



If you want to evaluate a variable as false, you can use the NOT operator `!`

```
if ! false
then
    echo 'This is false'
fi
```

Multiple values can be evaluated with the AND operator `&&`

```
if true && ! false
then
    echo 'The first value is true and the second value is false'
fi
```

Powershell

The same is true for Powershell except Powershell use `$true` and `$false`.

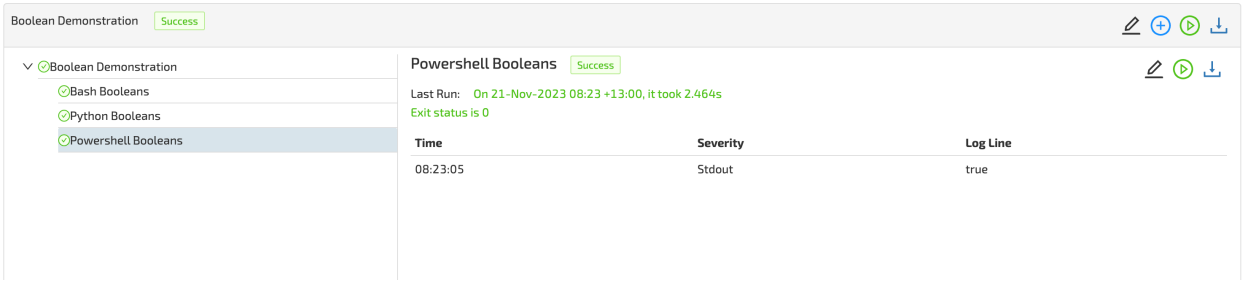
For example:

```
if ($${{attuneVal}}) {
    Write-Host "true"
}
```

Attune substitutes the value into the `{attuneVar}` parameter.

If the value is `true` :

```
if ($true) {
    Write-Host "true"
}
```



If you want to evaluate a variable as false, you can use the NOT operator `-not`

```
if -not ($false) {
    Write-Host "Not False"
}
```

Python

Python uses `True` and `False`. To keep the use of booleans consistent throughout a project, you are able to assign the booleans to variables at the beginning of a Python script.

For example:

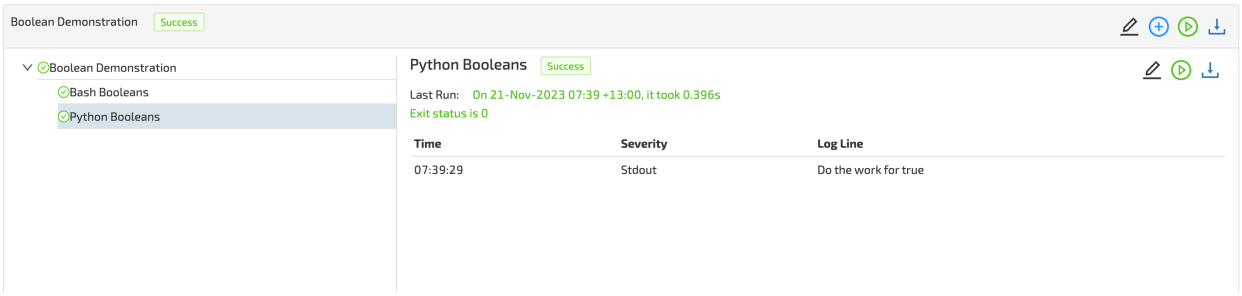
```
true=True
false=False

if {attuneVal}:
    print("Do the work for true")
else
    print("Do the work for else")
```

If the Parameter is mapped to `true`, Attune substitutes `true` into the `{attuneVal}` parameter which is assigned to `True`.

```
true=True
false=False

if true:
    print("Do the work for true")
else
    print("Do the work for else")
```



If you want to evaluate a variable as false, you can use the NOT operator `not`

```
true=True
false=False

if not false:
    print("Do the work for true")
else
    print("Do the work for else")
```

Multiple values can be evaluated with the AND operator `and`

```
true=True
false=False

if not false and true:
    print("Do the work for true")
else
    print("Do the work for else")
```

5.2.4 Documentation Best Practices

Effective project communication is paramount, and Attune provides a robust platform to facilitate this.

This article outlines best practices for crafting clear and concise documentation.

Clear and Concise project Communication

The significance of clear and concise project communication cannot be overstated. It forms the foundation of successful software development. Documentation serves as a critical tool in aligning all stakeholders ensuring a shared understanding of project objectives and requirements. In its absence, miscommunication and confusion may emerge, resulting in delays, errors, and project setbacks.

Benefits of Clear Communication

Clear and concise project communication creates effective team collaboration. By offering precise instructions, guidelines, and expectations, documentation facilitates seamless teamwork, minimising misunderstandings and enhancing productivity and efficiency. Moreover, clear documentation expedites the onboarding of new team members, as they can swiftly grasp the project's structure, codebase, and requirements.

Attune Documentation

The Attune platform empowers all users to add and modify comments and descriptions on all items from projects and Blueprints through to Values and Schedules. The comments and descriptions are easily accessed in the platform, however are also used to populate a repository README.md file and webpage documentation. The webpage documentation is rendered in GitHub Pages.

Attune's support for Markdown makes it an ideal option for crafting clear and well-formatted documentation. Markdown is an intuitive lightweight markup language. Utilising Markdown can significantly improve the readability and visual aesthetics of the documentation.

Writing the project Description

A project description is a crucial component of any Attune project. It serves as the entry point for the project, providing an overview of its purpose, features, and more. Creating a comprehensive project description allows you to effectively communicate the project's value and encourage collaboration with other developers.

1. Start with a project objective

Begin the project description with a brief objective of the project. Clearly state the purpose, goals, and intended audience of the project. This helps readers quickly understand the project and whether it aligns with their needs.

2. Include prerequisite instructions

Provide detailed instructions on prerequisite steps for the project. Include any dependencies, system requirements, and step-by-step instructions. This helps other developers get started with the project quickly and reduces potential issues related to installation. In this step, use numbered lists instead of bullet points.

3. Highlight key features

Showcase the key features and functionalities of the project. This lets readers quickly grasp what the project offers and its potential benefits. Use bullet points or concise descriptions to highlight the main features.

4. Provide usage examples

Include usage examples to help readers understand how to use the project effectively. This can include code snippets, command-line examples, or screenshots demonstrating the project in action. Real-world examples make it easier for readers to relate to the project and envision how it can solve their problems.

5. Document API references

If the project exposes an API, document the API endpoints, request/response formats, and authentication requirements. This helps developers integrate the project with their applications and ensures a smooth integration process.

Writing the Blueprint Description

Similar to the project description, a Blueprint description is a crucial component of any Attune Blueprint. It serves as the entry point for the Blueprint, providing an overview of its purpose, features, and more. Creating a comprehensive Blueprint description allows you to effectively communicate the Blueprint's value and encourage collaboration with other developers.

1. Start with a blueprint objective

Begin the Blueprint description with a brief objective of the Blueprint. Clearly state the purpose, goals, and intended audience of the Blueprint. This helps readers quickly understand the Blueprint and whether it aligns with their needs.

2. Include prerequisite instructions

Provide detailed instructions on prerequisite steps for the Blueprint. Include any dependencies, system requirements, and step-by-step instructions. This helps other developers get started with the Blueprint quickly and reduces potential issues related to installation. In this step, use numbered lists instead of bullet points.

3. Highlight key features

Showcase the key features and functionalities of the Blueprint. This lets readers quickly grasp what the Blueprint offers and its potential benefits. Use bullet points or concise descriptions to highlight the main features.

4. Provide usage examples

Include usage examples to help readers understand how to use the Blueprint effectively. This can include code snippets, command-line examples, or screenshots demonstrating the Blueprint in action. Real-world examples make it easier for readers to relate to the Blueprint and envision how it can solve their problems.

5. Document API references

If the Blueprint exposes an API, document the API endpoints, request/response formats, and authentication requirements. This helps developers integrate the Blueprint with their applications and ensures a smooth integration process.

5.3 "How-to" guides

Here you'll find short answers to "How do I. . . ?" types of questions. These how-to guides don't cover topics in depth. However, these guides will help you quickly accomplish common tasks.

5.3.1 How To Clone a GIT Project

Objective

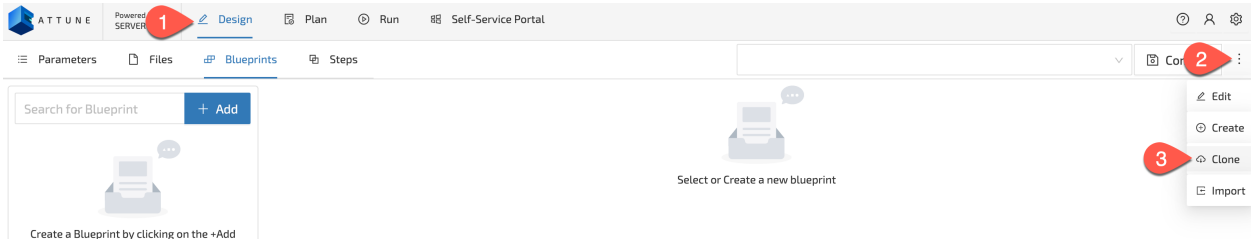
This procedure provides instructions to clone a project into an instance of Attune.

Procedure

Clone a GIT Repository into Attune

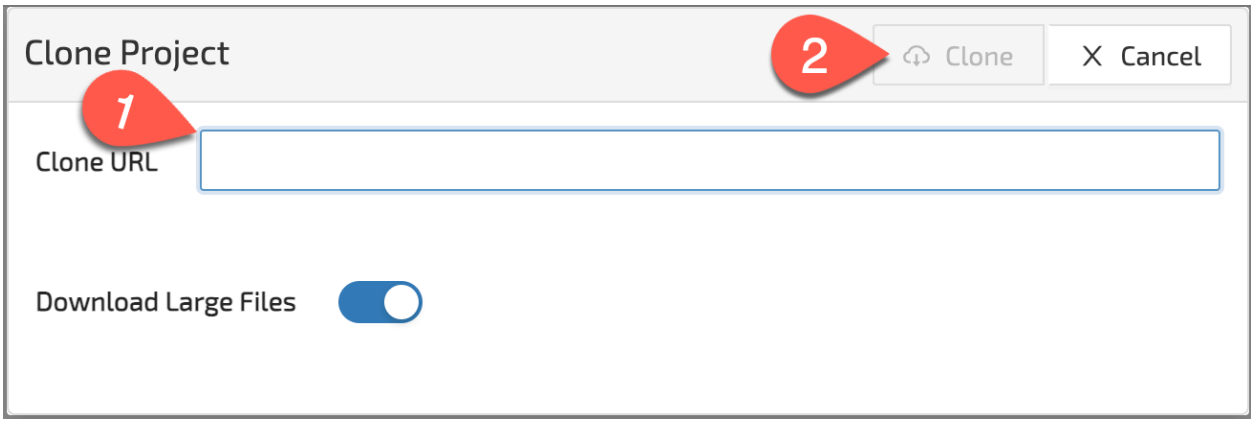
Start the Clone process in Attune.

1. Open the Design Workspace
2. Open the Project options dropdown
3. Select Clone



Paste the GIT repository URL into Attune.

1. Paste the URL
2. Select Clone



Note: If the Download Large Files option is enabled, Attune will attempt to download files that have a URL to their source.

Complete

This procedure is now complete. Now that this project is in your Attune instance you can begin creating Jobs.

5.3.2 How To Create a Job

Objective

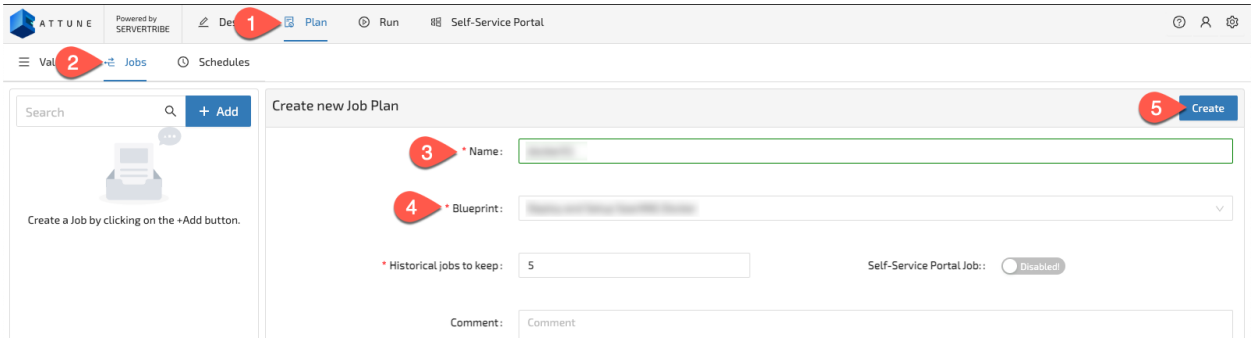
This procedure provides instructions to create adn run a Job in Attune.

Procedure

Create a Job

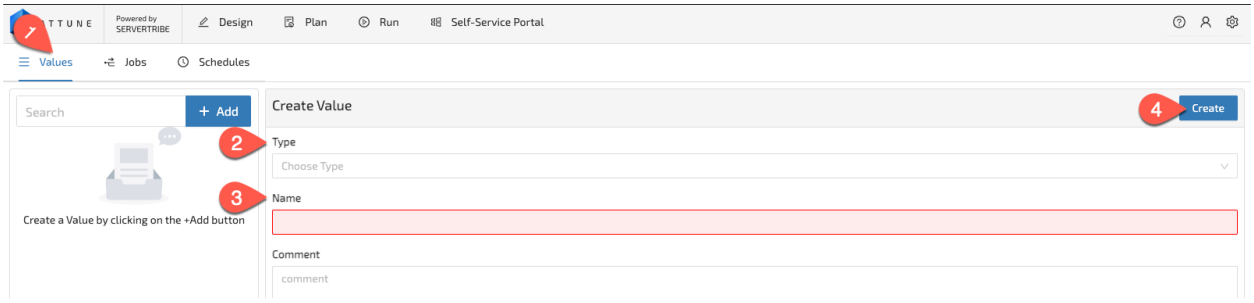
Navigate to the Plan workspace and create a Job from a Blueprint in the Project you cloned.

1. Open the Plan Workspace
2. Navigate to Jobs
3. Enter your Job Name
4. Select the Blueprint to use
5. Select Create



Create the Values required to fill the Parameters for the Job.

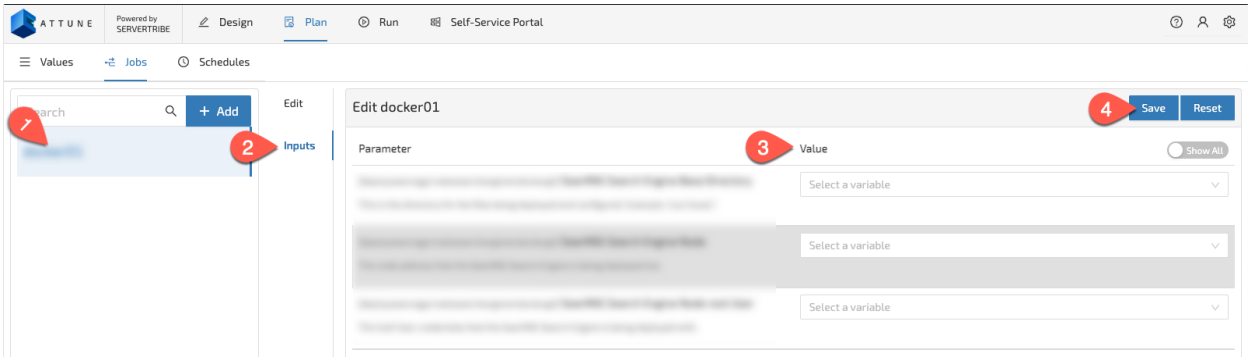
1. Navigate to Values
2. Select the Value Type
3. Enter the Value Name
4. Select Create



Note: After selecting the Type of Value you'll be provided fields to complete.

Configure the Parameters for the Job you created.

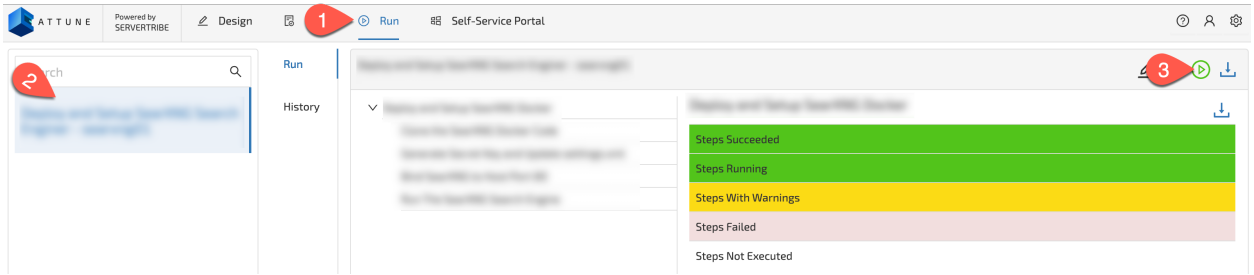
1. Select the Job
2. Navigate to Inputs
3. Populate the Values
4. Select Save



Run a Job

Run your Job.

1. Open the Run Workspace
2. Select the Job
3. Select Run



Complete

This procedure is now complete, you've successfully run a Job in Attune.

5.3.3 How To Setup WinRM, CIFS Manually

Objective

This procedure will setup a Windows desktop or server to allow connections and Automation from Attune. This is done via enabling Windows File Sharing (CIFS) and Windows Remote Management (WinRM).

Procedure

To enable WinRM, run the following command in PowerShell.

```
# Enable WinRM with HTTPS, via PowerShell
# This is for windows deployment step
Enable-PSRemoting -SkipNetworkProfileCheck -Force
Remove-Item -Path WSMan:\Localhost\listener\listener* -Recurse;
New-Item -Path WSMan:\LocalHost\Listener `
      -Transport HTTPS `
      -Address * `
      -CertificateThumbPrint `
            (New-SelfSignedCertificate `
            -CertstoreLocation Cert:\LocalMachine\My `
            -DnsName $env:computername `
            -NotAfter (get-date).AddYears(6)).Thumbprint `
   -Force
Restart-Service -Force WinRM
New-NetFirewallRule -DisplayName 'WinRM HTTPS' `
                  -Name 'WinRM_HTTPS' `
                  -Profile Any `
                  -LocalPort 5986 `
                  -Protocol TCP

# Enable CredSSP on Win11, via PowerShell
$osName = (Get-ComputerInfo).OsName
if ($osName -like "*Windows 11*") {
   Enable-WSManCredSSP -Role Server
}
```

To verify WinRM with HTTPS is enabled, run the following command in PowerShell:

```
Get-WSManInstance WinRM/Config/Listener -Enumerate | Where-Object { $_.transport -eq
↪"HTTPS" }
```

To enable file and print sharing, run the following command in PowerShell:

```
# Enable File and Print sharing, via PowerShell
# This is for windows deployment step
Get-NetFirewallRule -DisplayGroup 'File and Printer Sharing' `
    | Set-NetFirewallRule -Profile Any -Enabled true
```

To verify file and print sharing is enabled, run the following command in PowerShell:

```
Get-NetFirewallRule -DisplayGroup "File And Printer Sharing" | select DisplayName,␣
↪Enabled
```

Complete

This procedure is now complete. Your file and printing sharing is enabled.

5.3.4 How To Setup WinRM Via AD

Setting up WinRM via Active Directory:

Objective

This procedure provides instructions to automatically enable WinRM with HTTPS via Active Directory group policies.

Attune uses WinRM to execute commands on windows desktops and servers. WinRM, combined with improvements in PowerShell Cmdlets is Microsofts emerging solution for scriptable administration of windows servers.

Note: If you don't have a domain and the target computers joined to the domain, then this procedure isn't for you.

Note: This setup is straight forward with defaults, your corporate environment may require alterations to the procedure.

Setup :

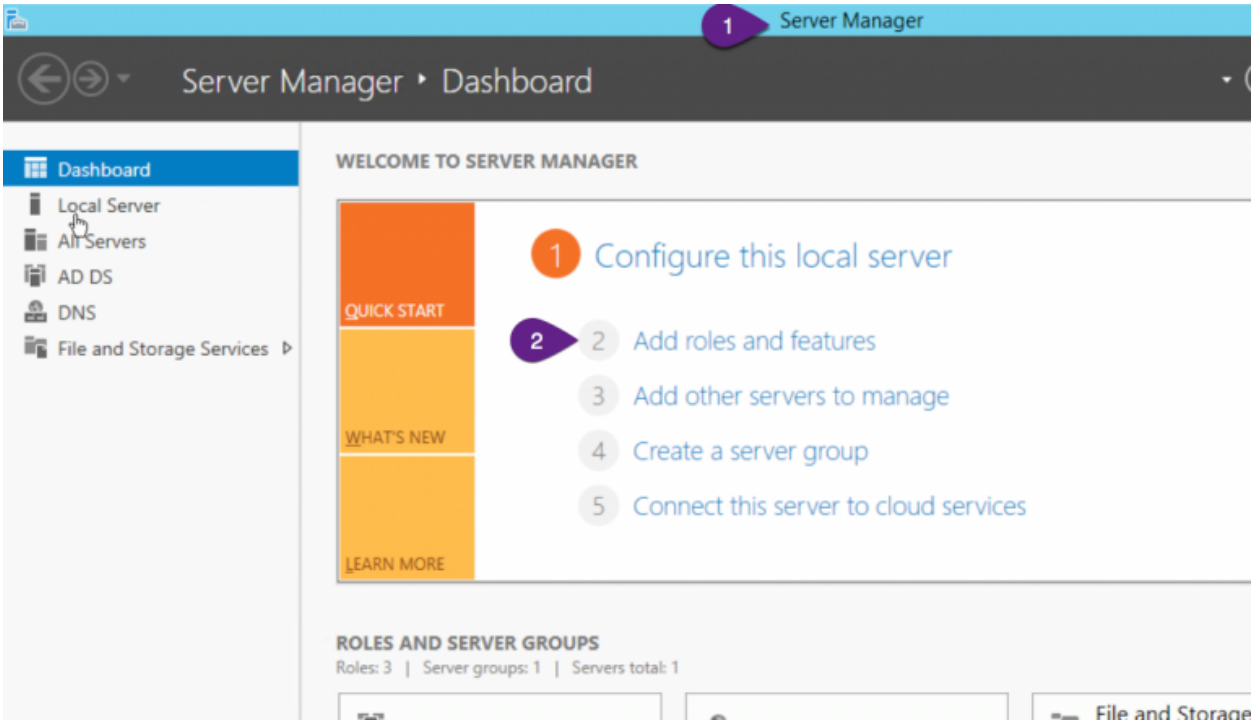
1. Windows 2012 R2 Server, with Active Directory Domain Services configured.
2. Target servers are joined to the domain.

Procedure

The following procedure all performed via a Remote Desktop session to the domain server.

Adding Certificate Server Role

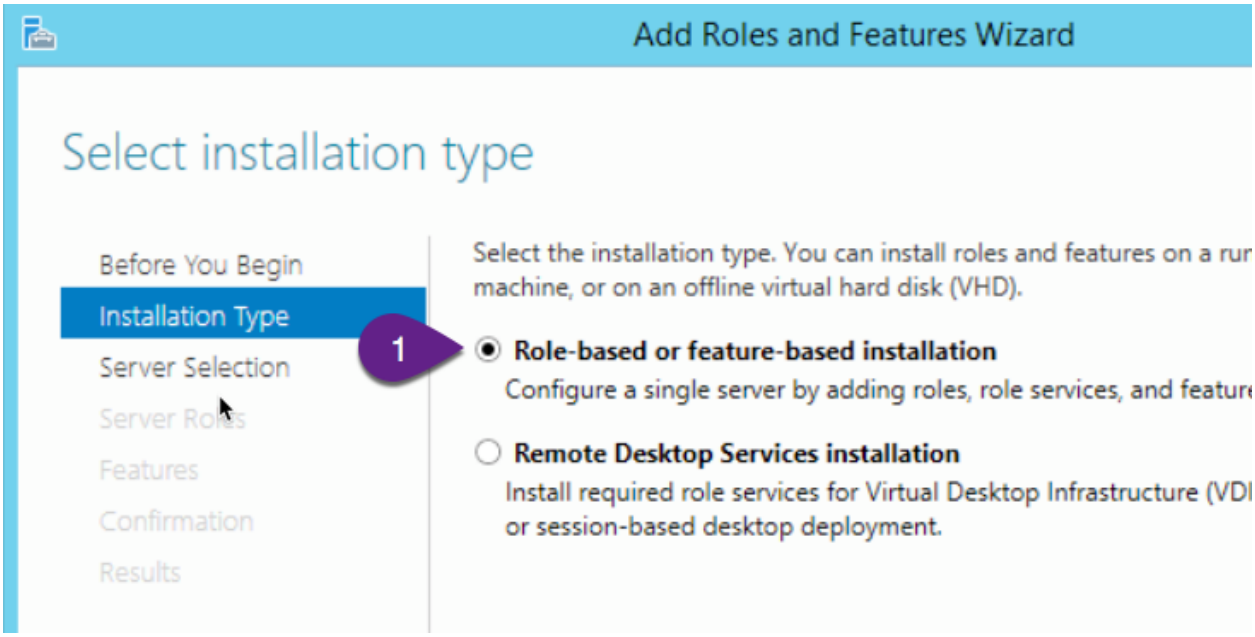
1. Open the Server Manager
2. Select "Add roles and features"



Click through the “Before You Begin” screen

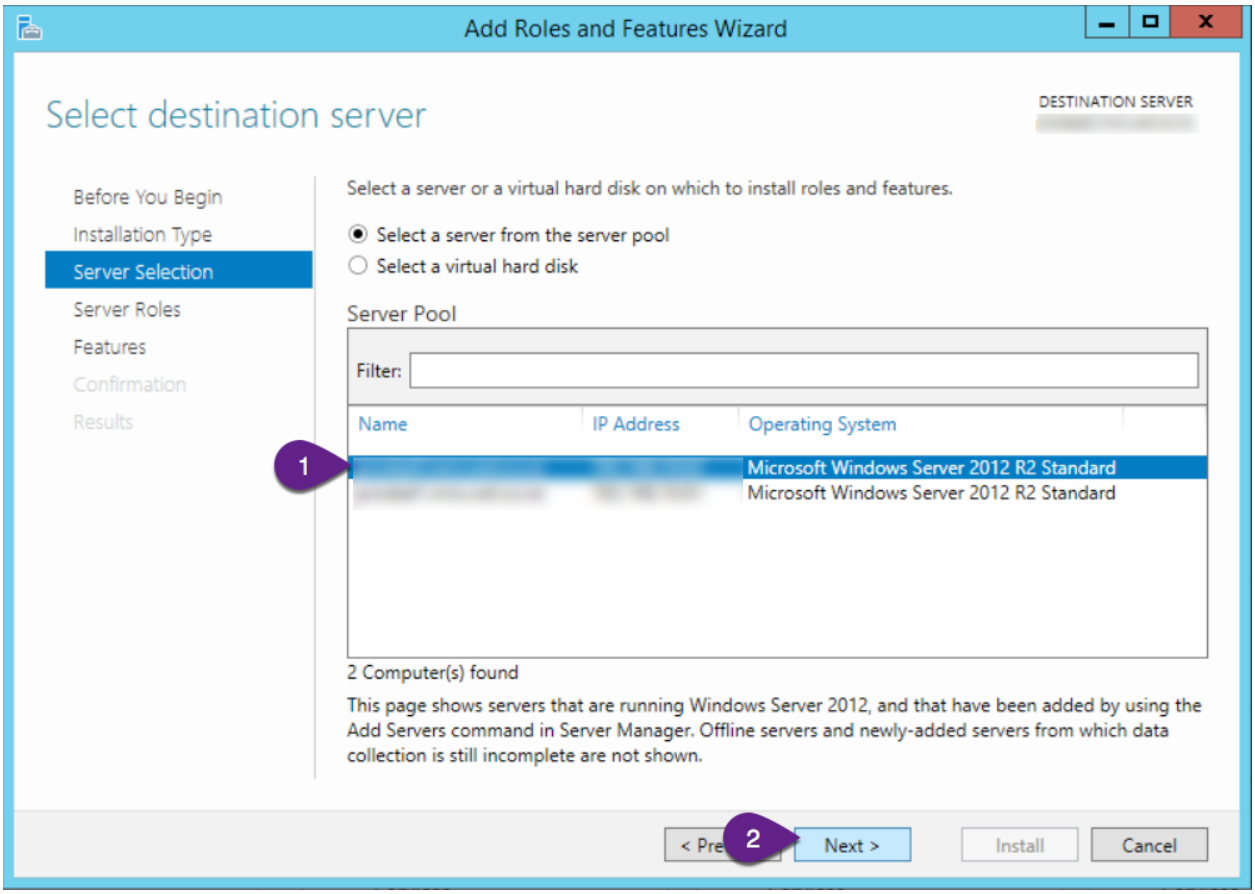
On the “Installation Type screen” :

1. Select “Role-based or feature-based installation
2. Click “Next”



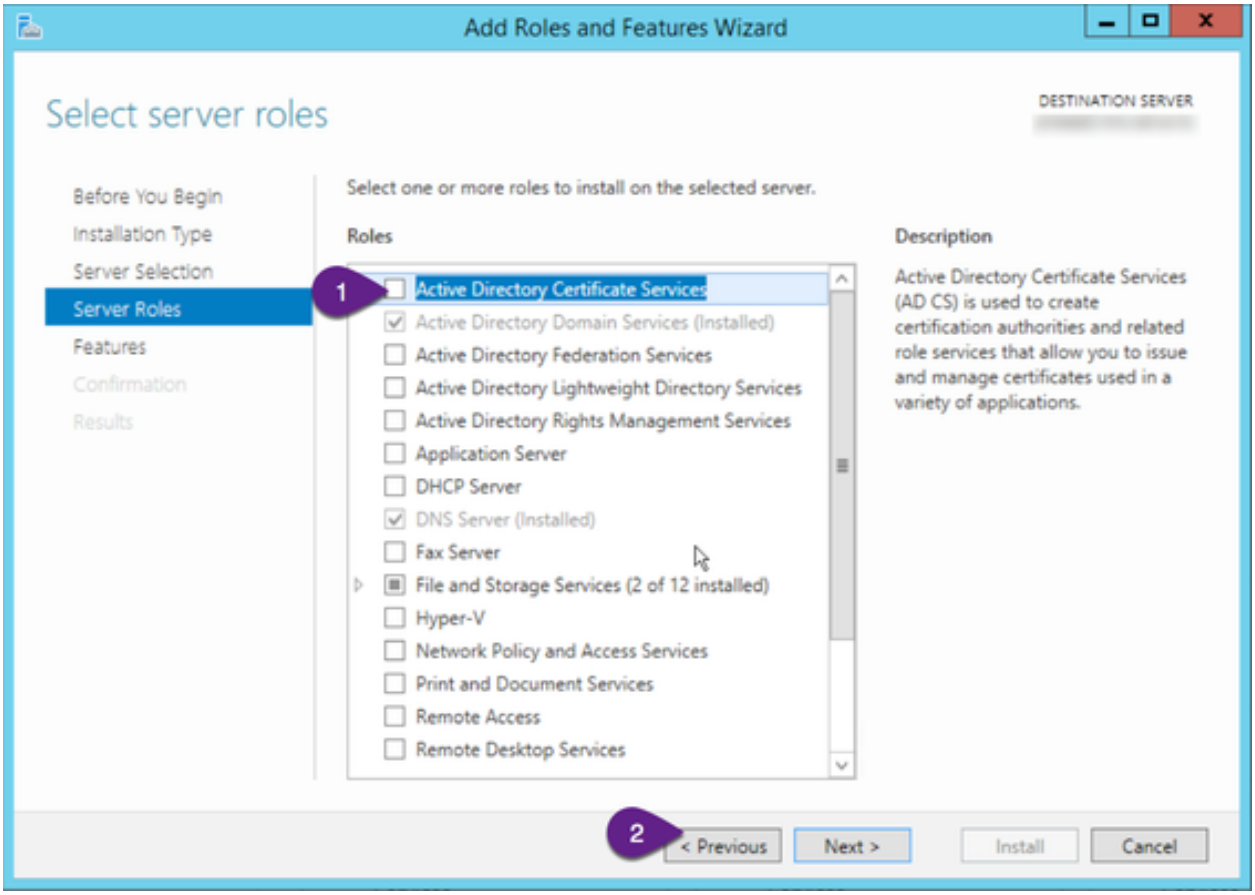
On the “Server Selection” screen :

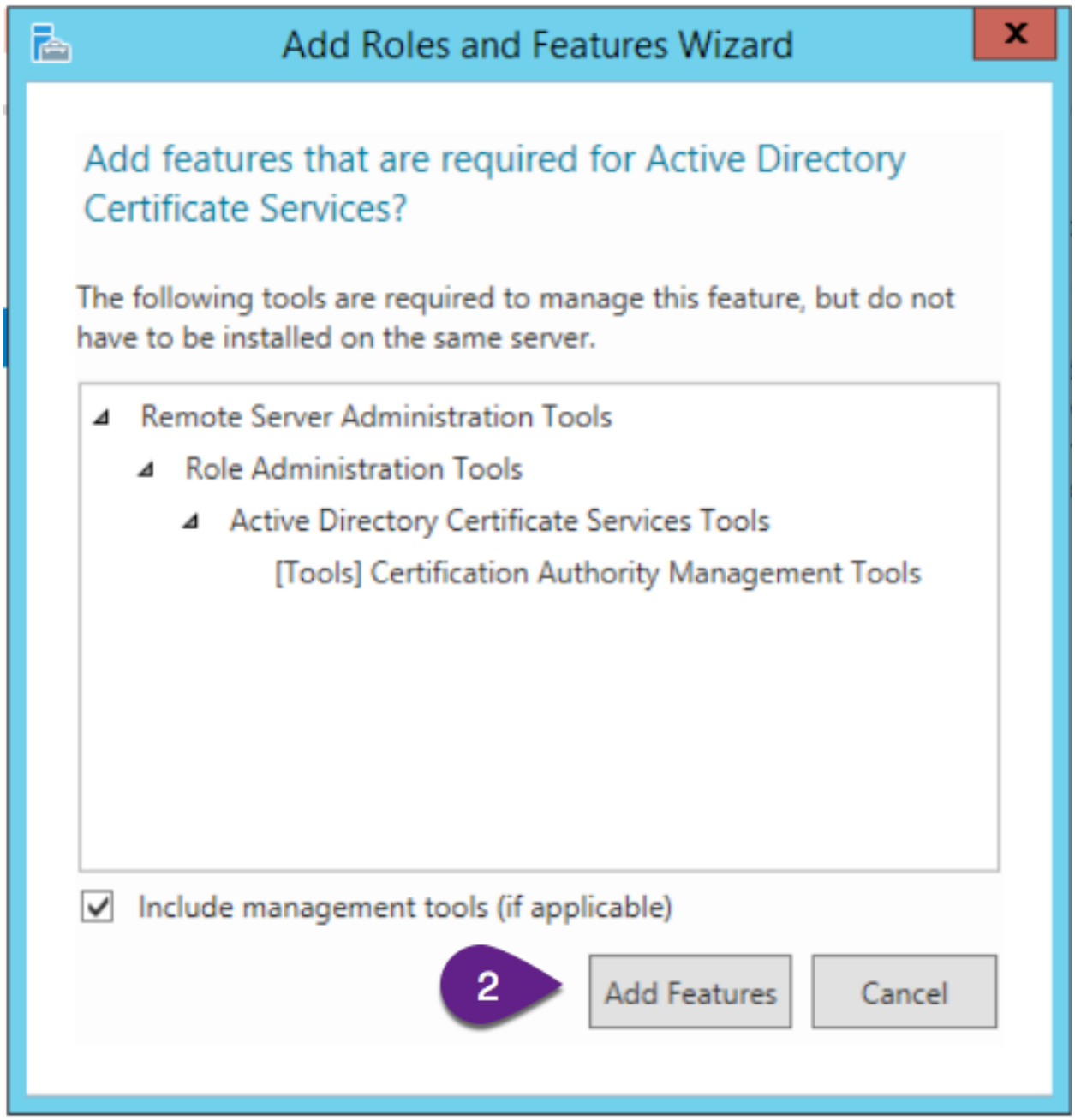
1. Select the server to install the Certificate service on
2. Click “Next”



On the “Server Roles” screen:

1. Select “Active Directory Certificate Service”
2. On the popup, click “Add Features”
3. Click Next.


Add Roles and Features Wizard
Select server roles
DESTINATION SERVER
Before You Begin
Installation Type
Server Selection
Server Roles
Features
Confirmation
Results
Select one or more roles to install on the selected server.
Roles
Active Directory Certificate Services
Active Directory Domain Services (Installed)
Active Directory Federation Services
Active Directory Lightweight Directory Services
Active Directory Rights Management Services
Application Server
DHCP Server
DNS Server (Installed)
Fax Server
File and Storage Services (2 of 12 installed)
Hyper-V
Network Policy and Access Services
Print and Document Services
Remote Access
Remote Desktop Services
Description
Active Directory Certificate Services (AD CS) is used to create certification authorities and related role services that allow you to issue and manage certificates used in a variety of applications.
1
2
< Previous
Next >
Install
Cancel



On the “Features” screen:

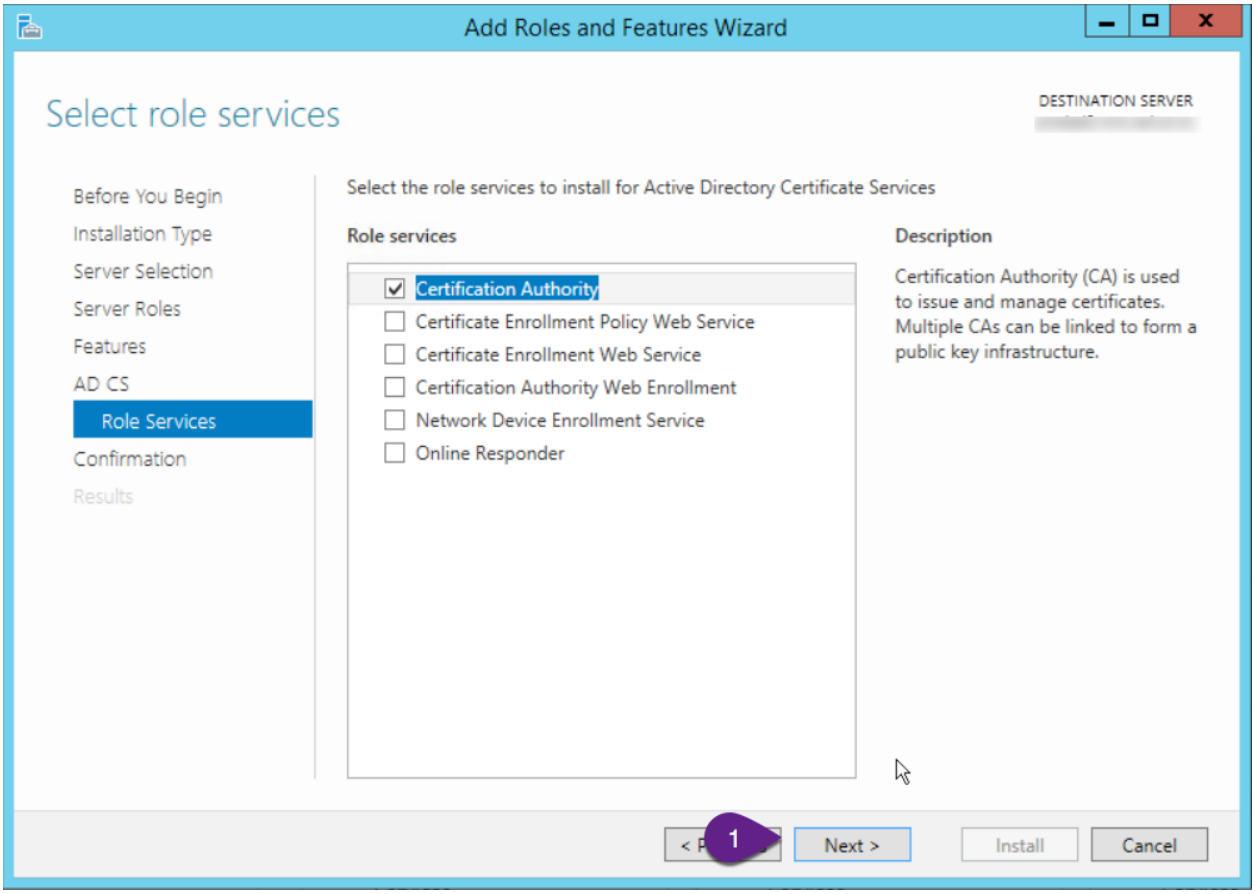
1. Click “Next”

On the “AD CS” screen:

1. Click “Next”

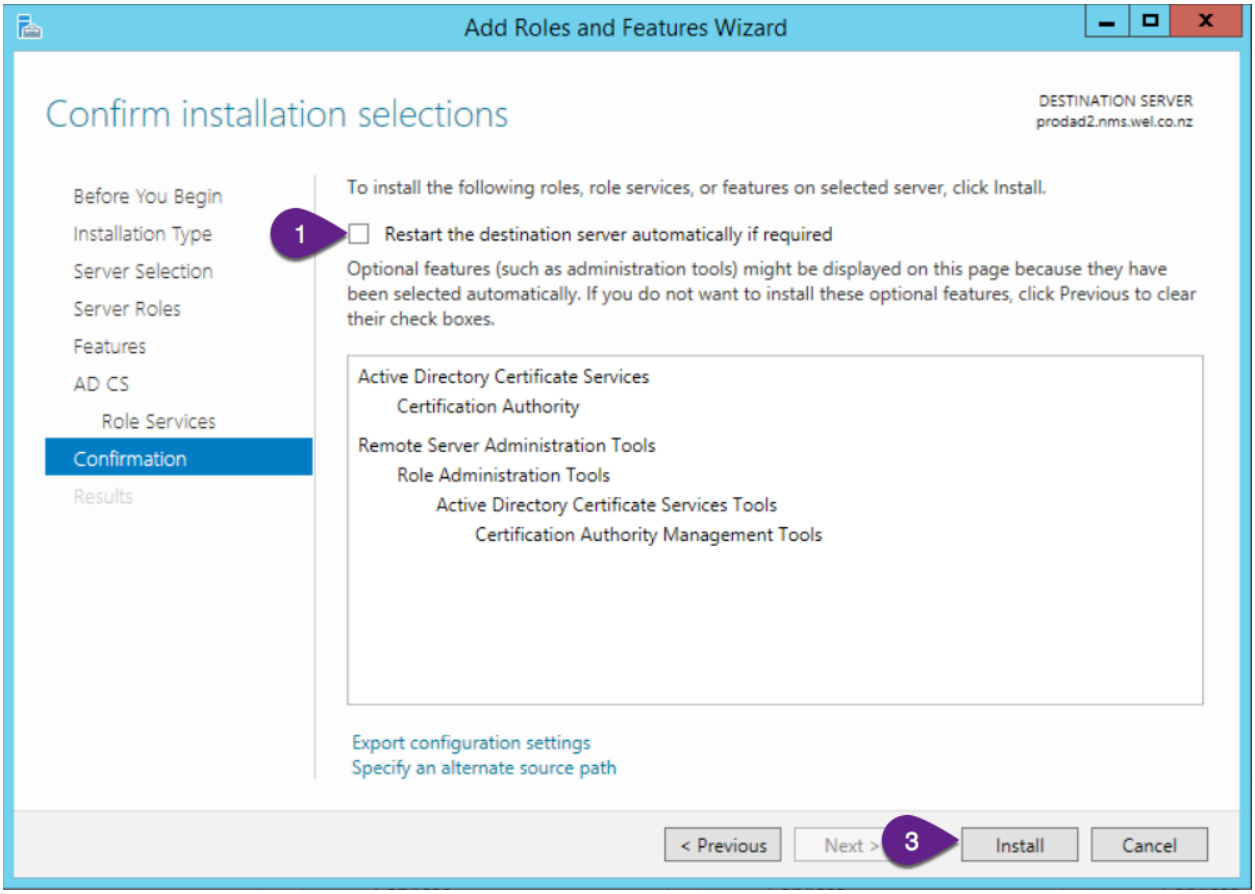
On the “Role Services” screen:

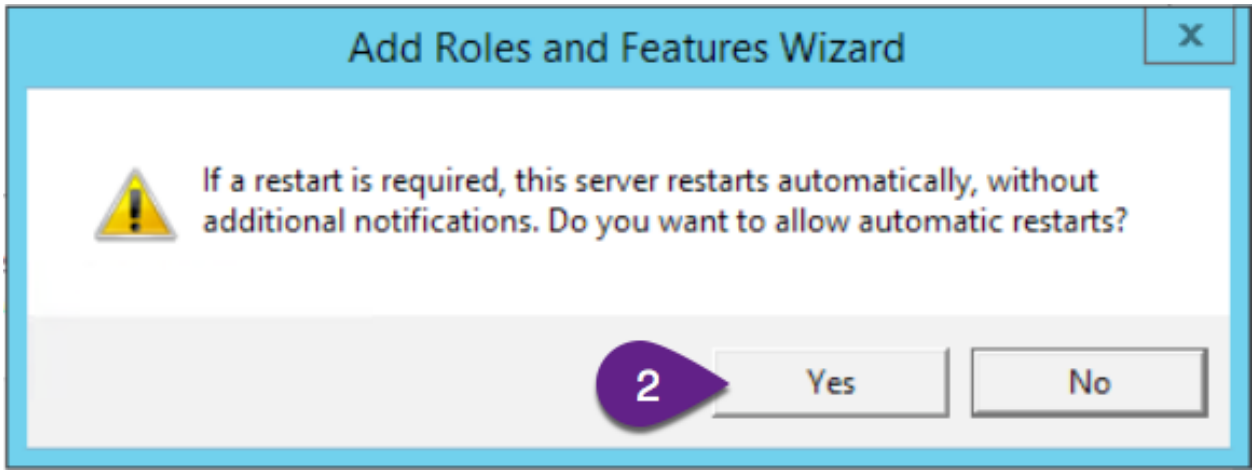
1. Click “Next”



On the “Confirmation” screen:

1. Check the “Restart the destination server automatically if required”
2. Click “Yes” on the confirmation dialog.
3. Click “Install”





The installation will proceed, break time.

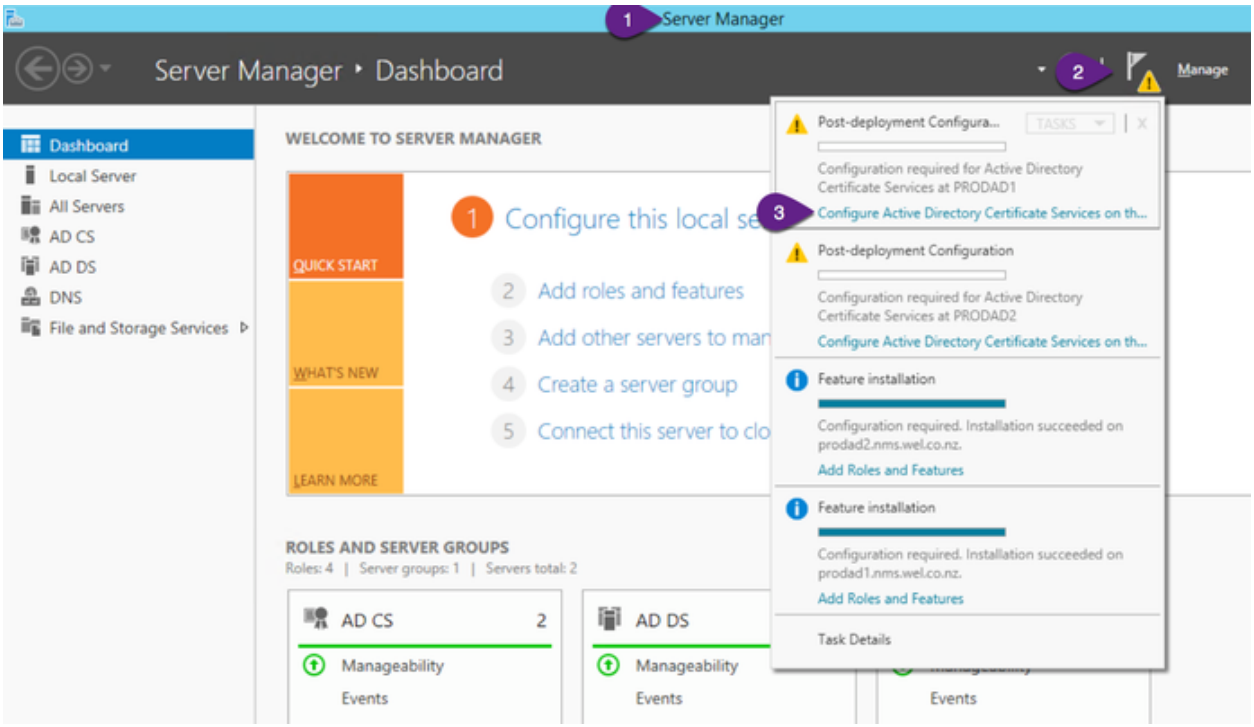
On the "Results" screen:

1. Click "Close"

Repeat the procedure for the other domain controllers in the domain.

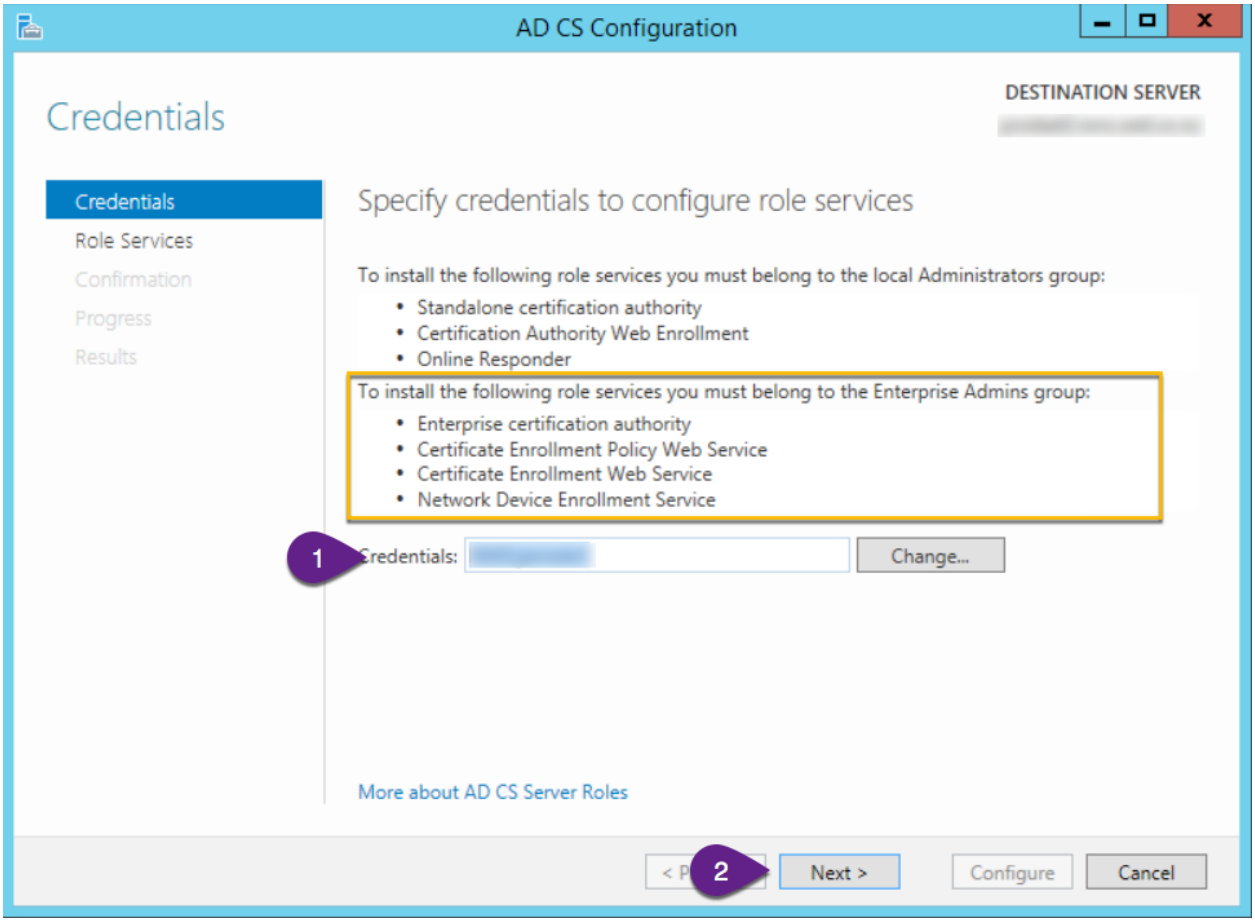
Configuring Certificate Server Role

1. Open the Server Manager
2. Select “Notification” dropdown
3. Click “Configure Active Directory Certificate . . . ”



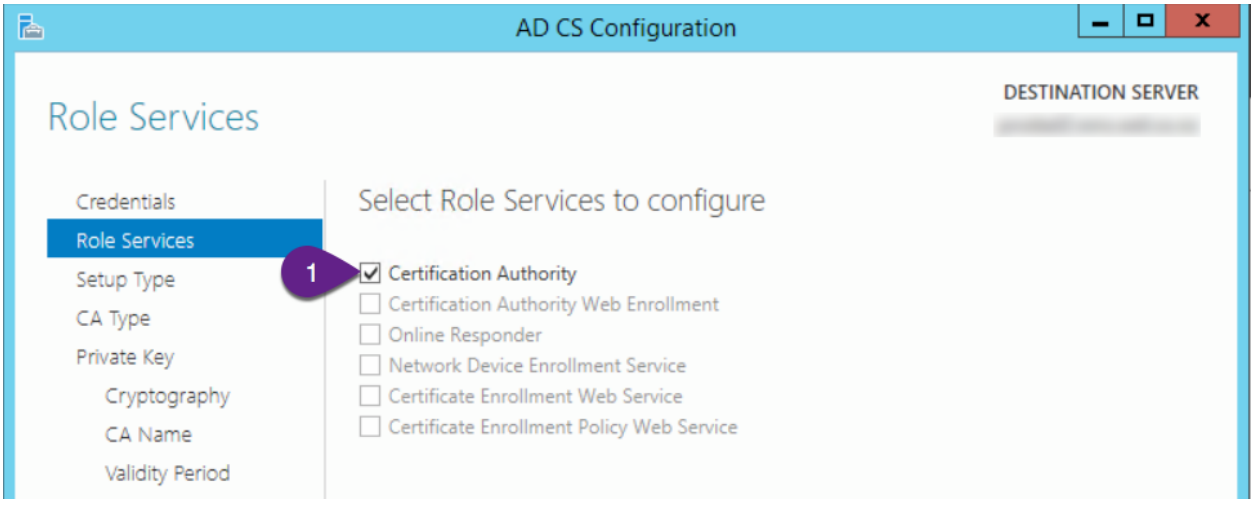
On the “Credentials” screen:

1. Ensure you have entered a valid domain credential
2. Click “Next”



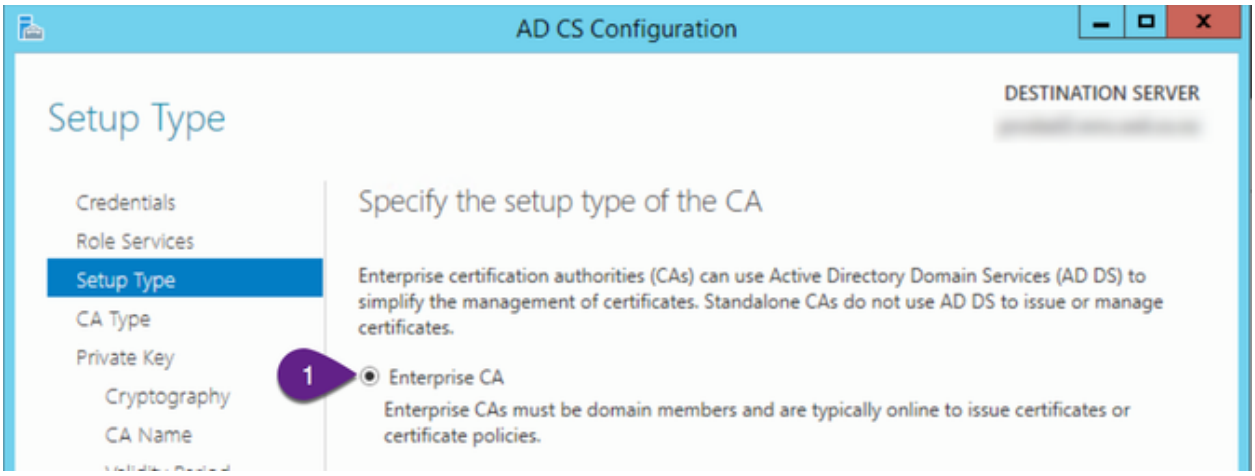
On the “Role Services” screen:

1. Check “Certificate Authority”
2. Click “Next”



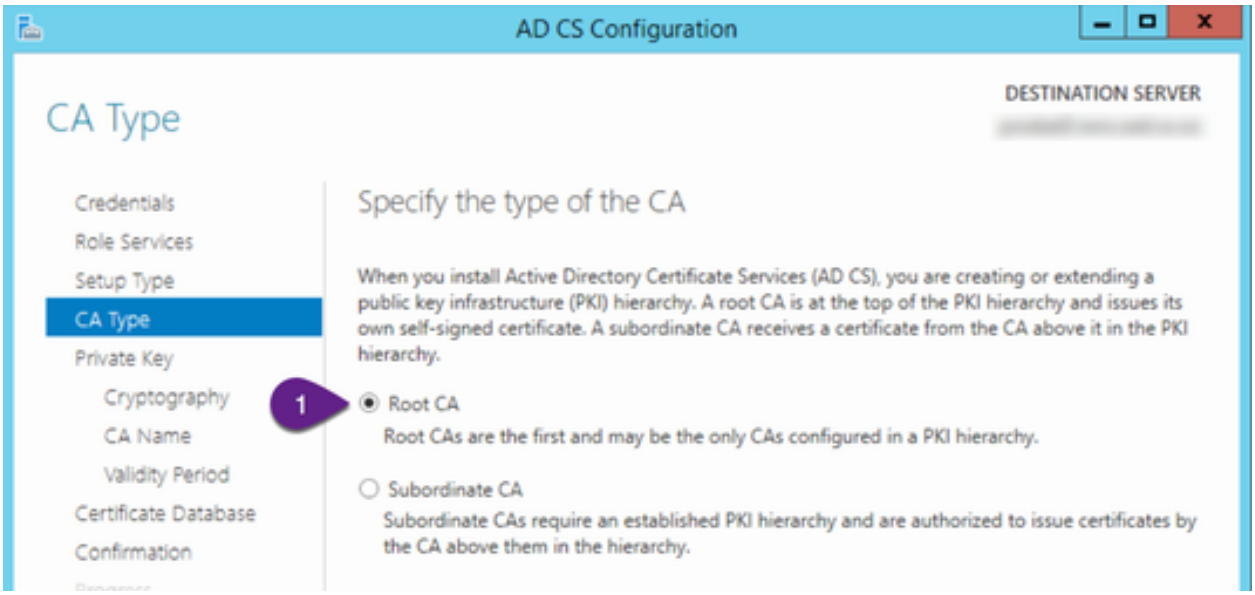
On the “Setup Type” screen:

1. Ensure “Enterprise CA” is selected
2. Click “Next”



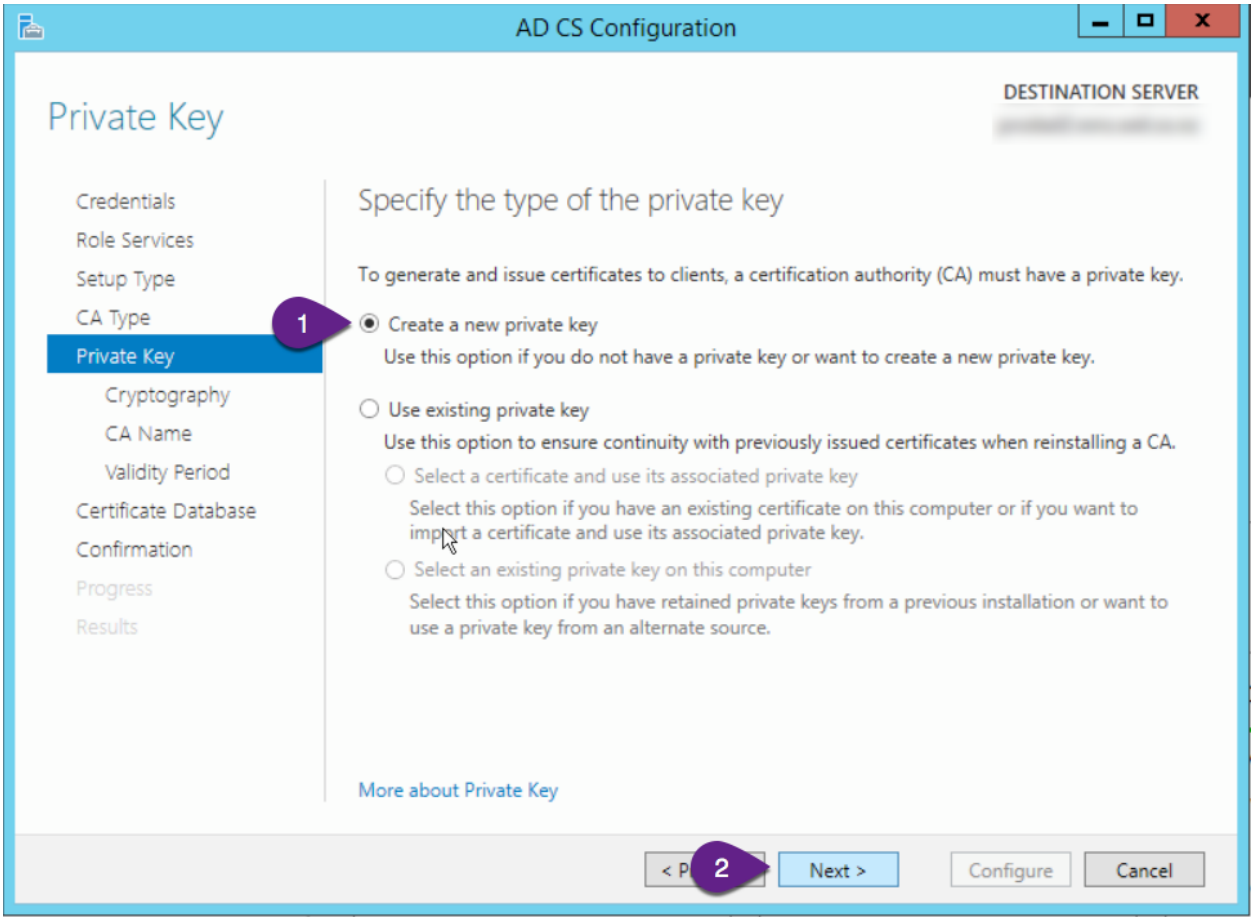
On the “CA Type” screen:

1. Ensure “Root CA” is selected, or “Subordinate CA” if this is the second server your setting up.
2. Click “Next”



On the “Private Key” screen:

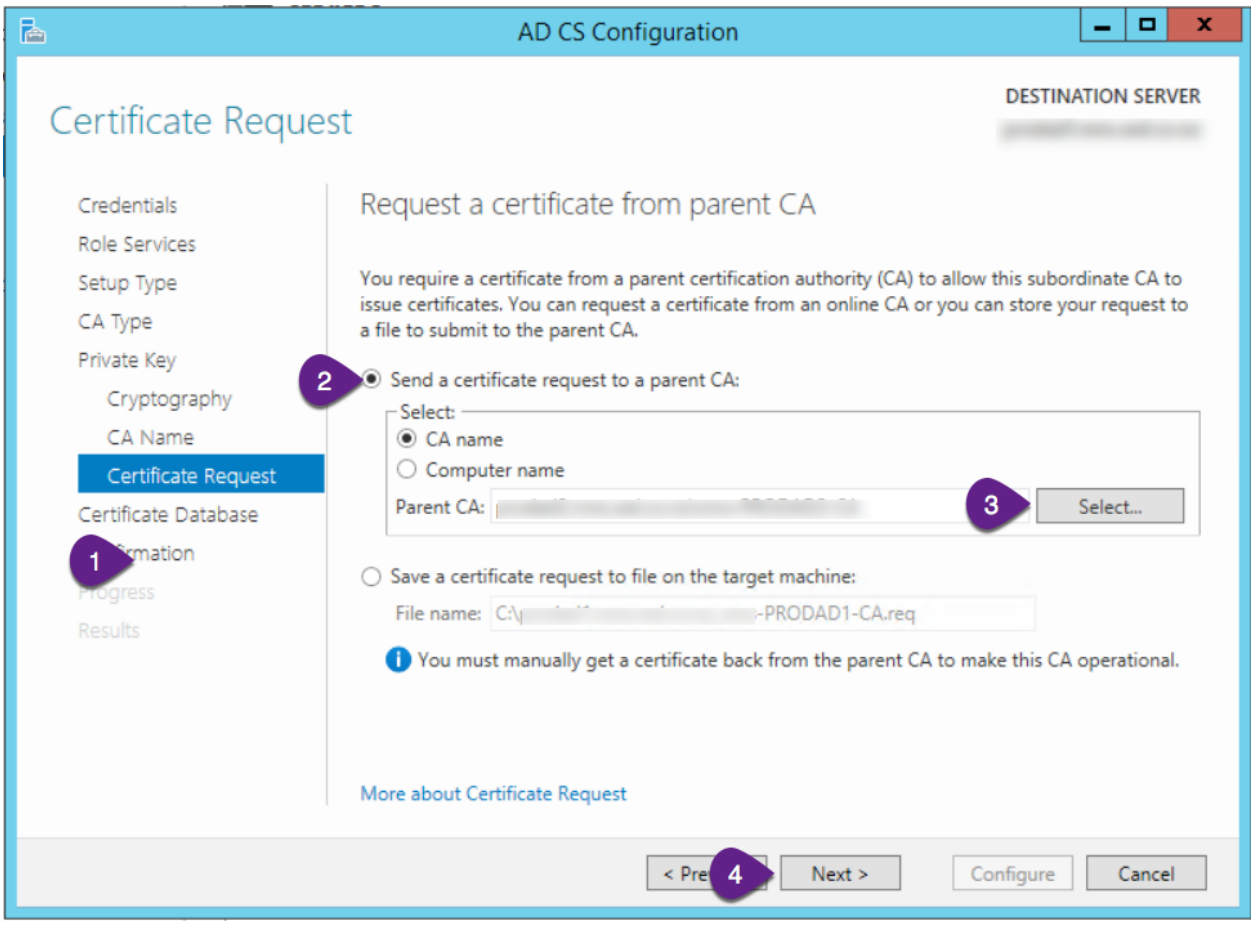
1. Ensure “Create a new private key” is selected
2. Click “Next”



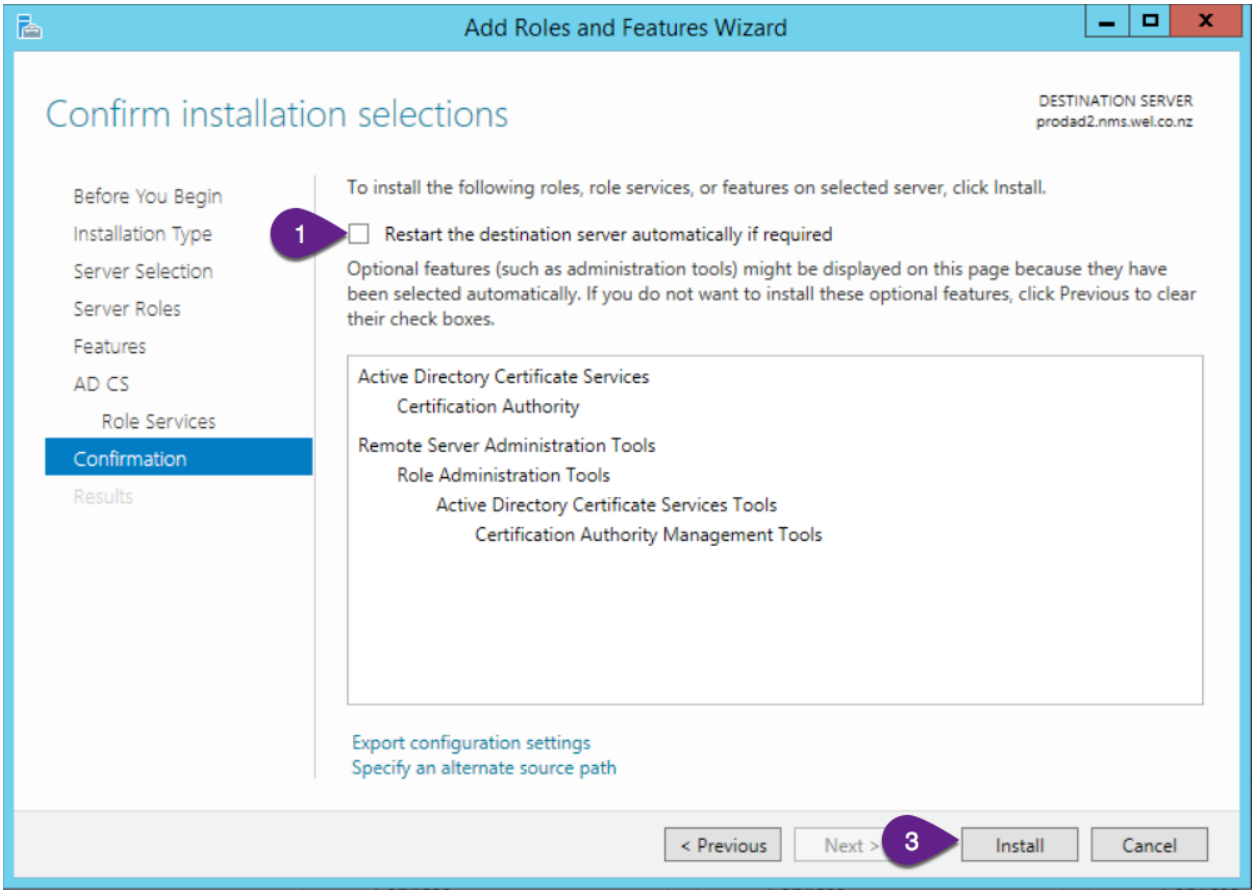
If this IS the first server your setting up and the Root CA, skip this step.

If this is the second server you're setting up, it will be a Subordinate CA and need to request signing from the root CA.

1. Select the "Certificate Request" on the screen list on the side bar
2. Click "Send a certificate request to a parent CA"
3. Click "Select"
4. Click "Next"



1. Click “Confirmation” on the screen list on the side bar
2. Then click “Configure



Repeat the procedure for the other domain controllers in the domain.

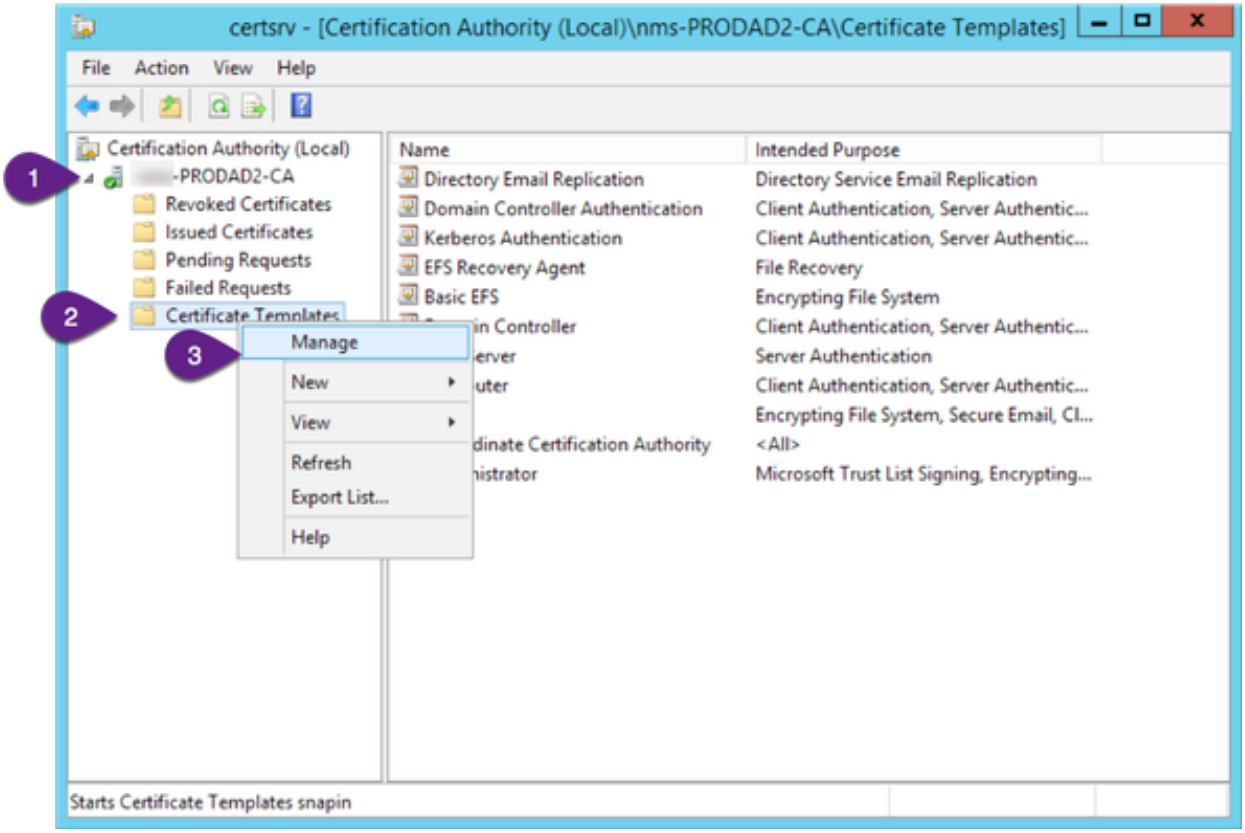
When prompted at the “CA Type”, you will need to select “Subordinate CA” on the subsequent servers.

Configure WinRM Certificate Template

Note: Be sure to check the Certificate Services setting updates in the following section on all domain controllers.

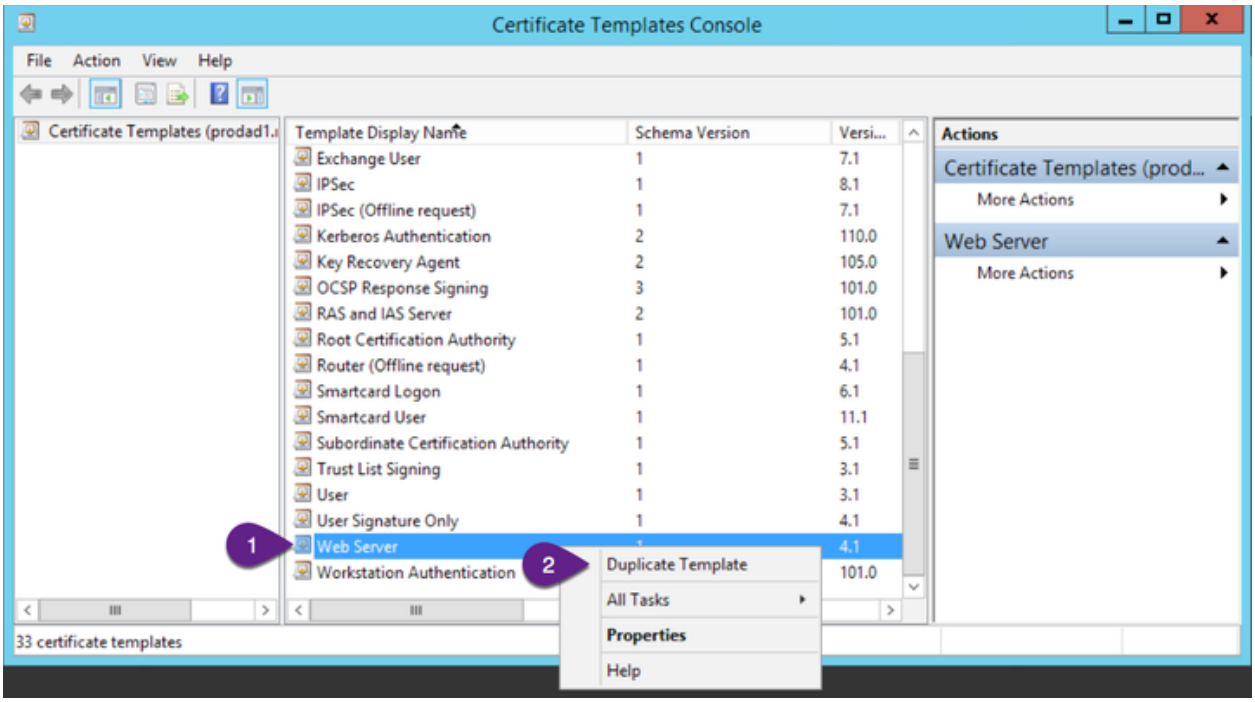
Open “Certificate Authority” (Use the start menu search)

1. Expand the root server
2. Select “Certificate Templates”
3. Right click and select “Manage”



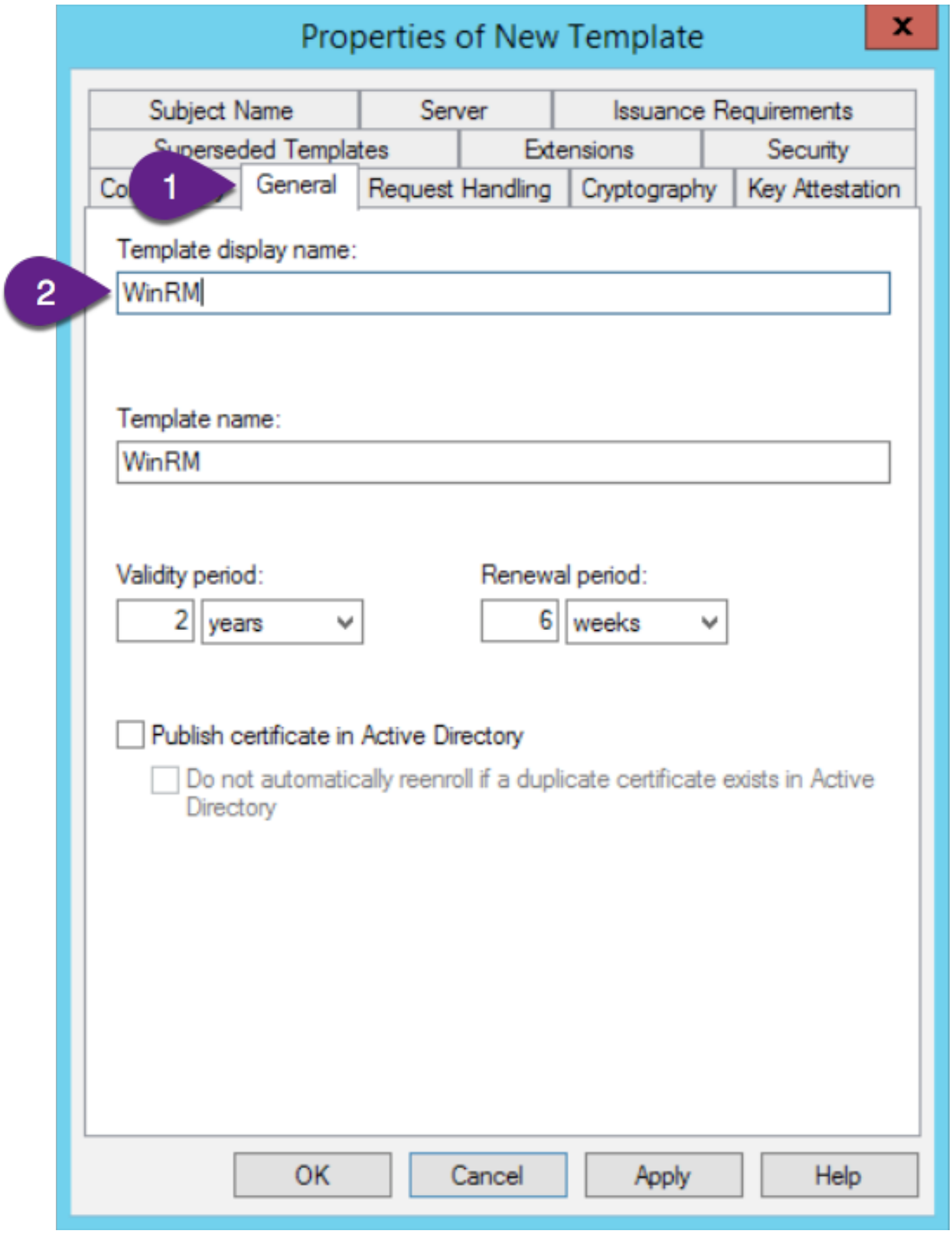
In the “Certificate Template Console” app

1. Find the “Web Server” template in the list
2. Right click and select “Duplicate Template”



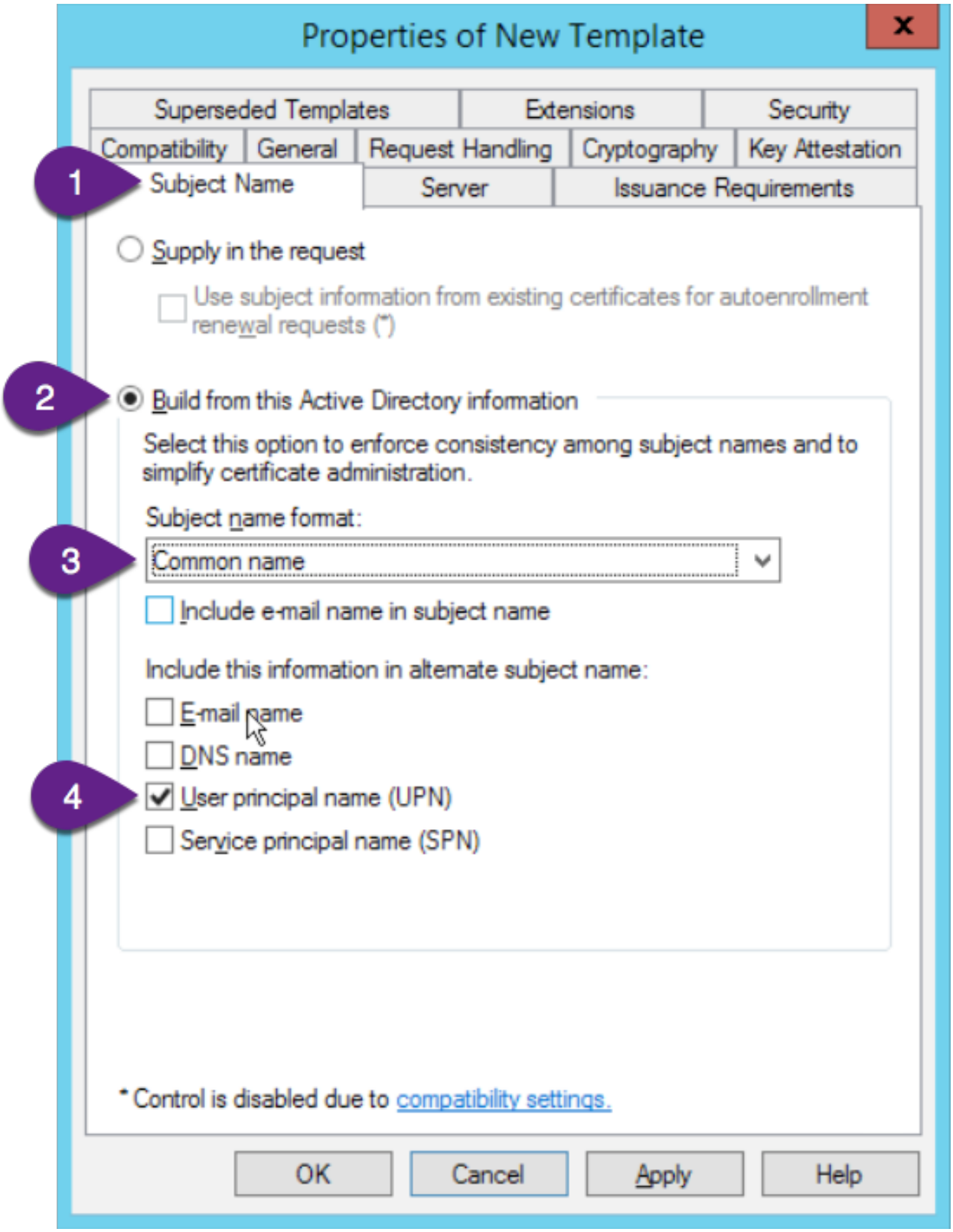
In the “Properties of New Template” app

1. Select the “General” tab
2. Enter “WinRM in the “Template display name”

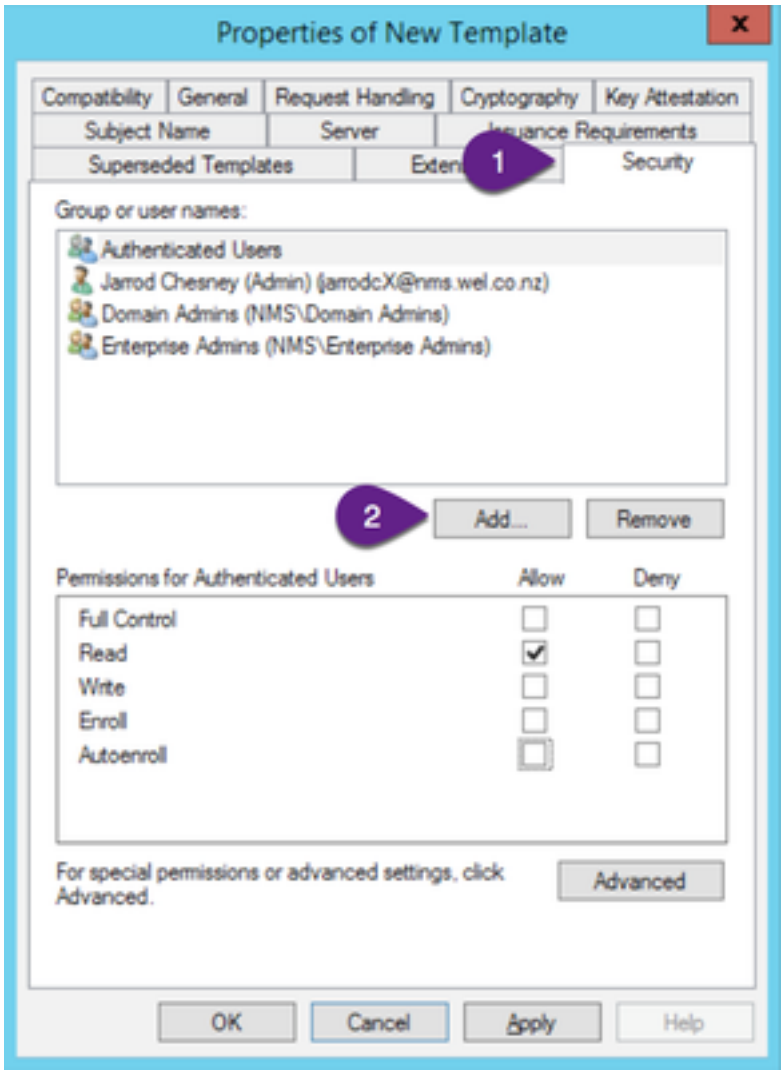


1. Select the “Subject Name” tab
2. Select “Build from this Active Directory information”

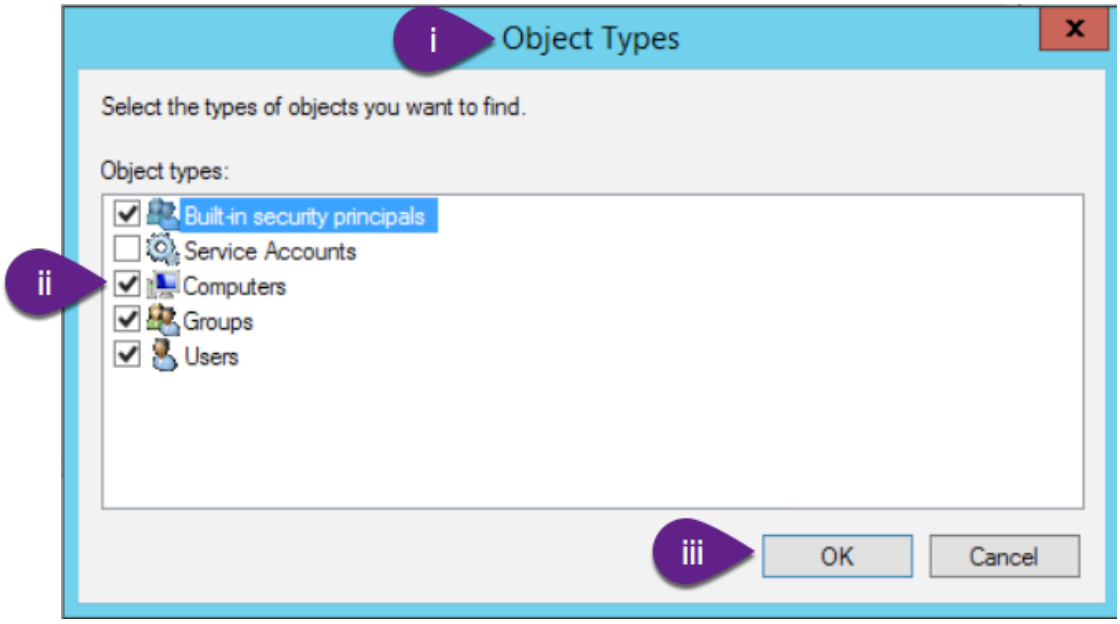
3. Select “Common name” for the “Subject name format”
4. Check “User principle name (UPN)”



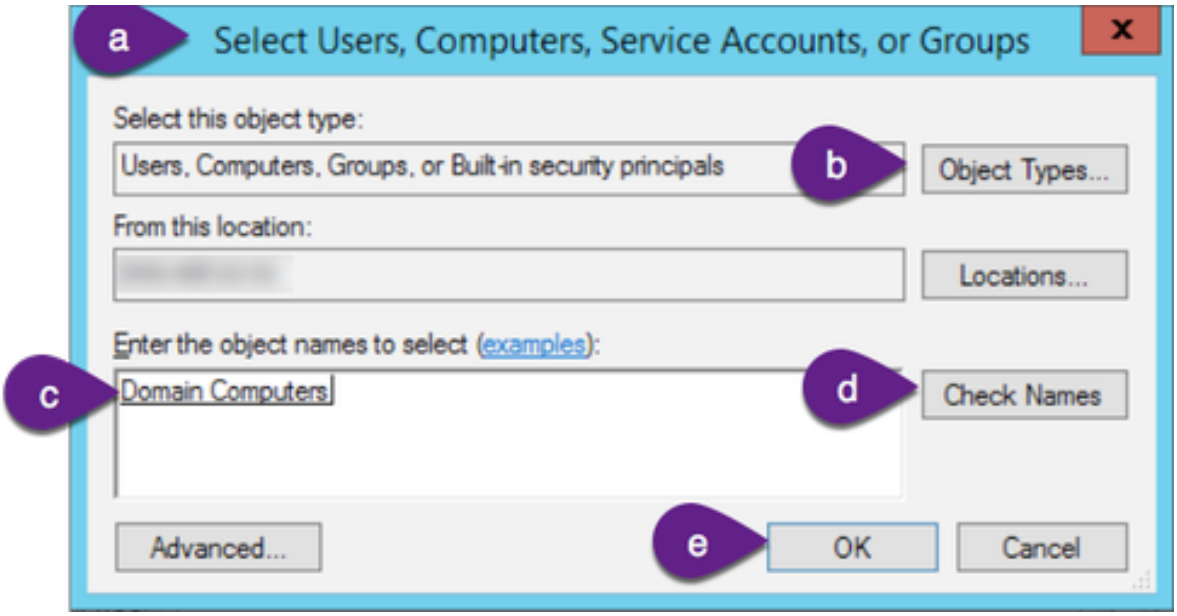
1. Select the “Security” tab
2. Select “Add”



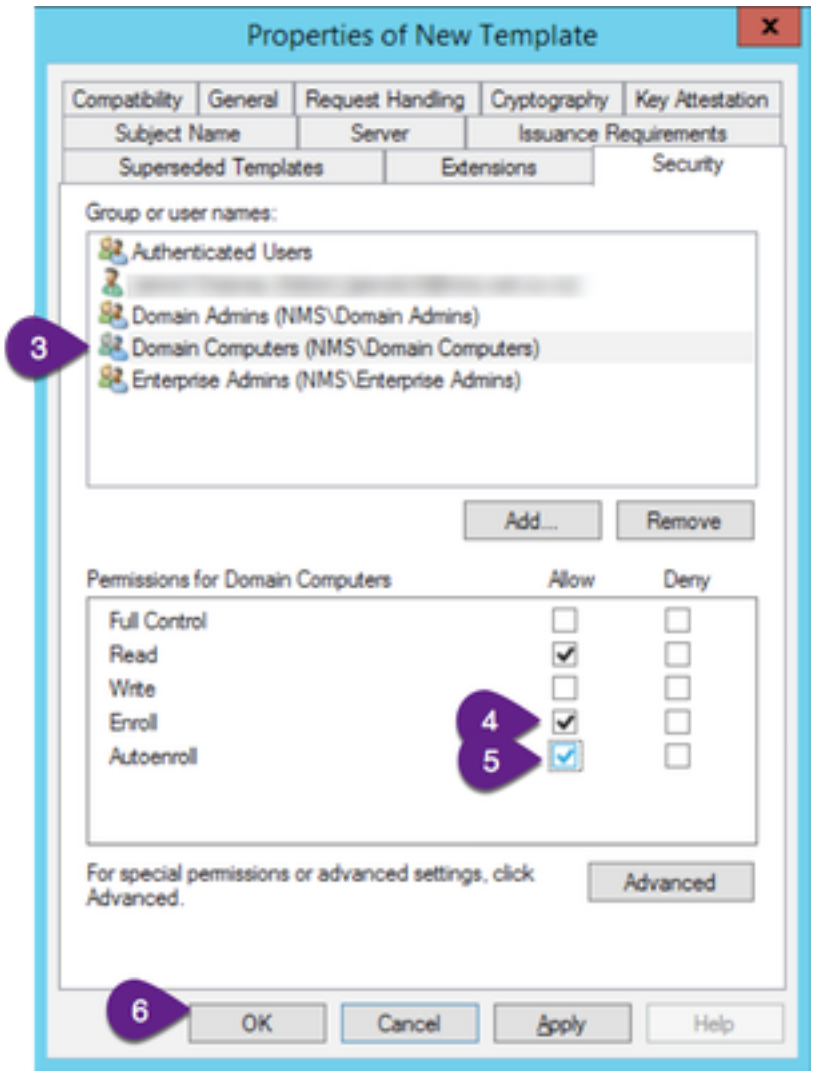
1. On the “Select Users, Computers. . . ” screen:
2. Select “Object Types
    1. On the “Object Types” screen
    2. select “Computers”
    3. click “Ok”



3. Back on the "Select Users, . . . " screen, Enter "Domain Computers" in the "Enter the object names to select" box
4. Click "Check Names"
5. Click "Ok"

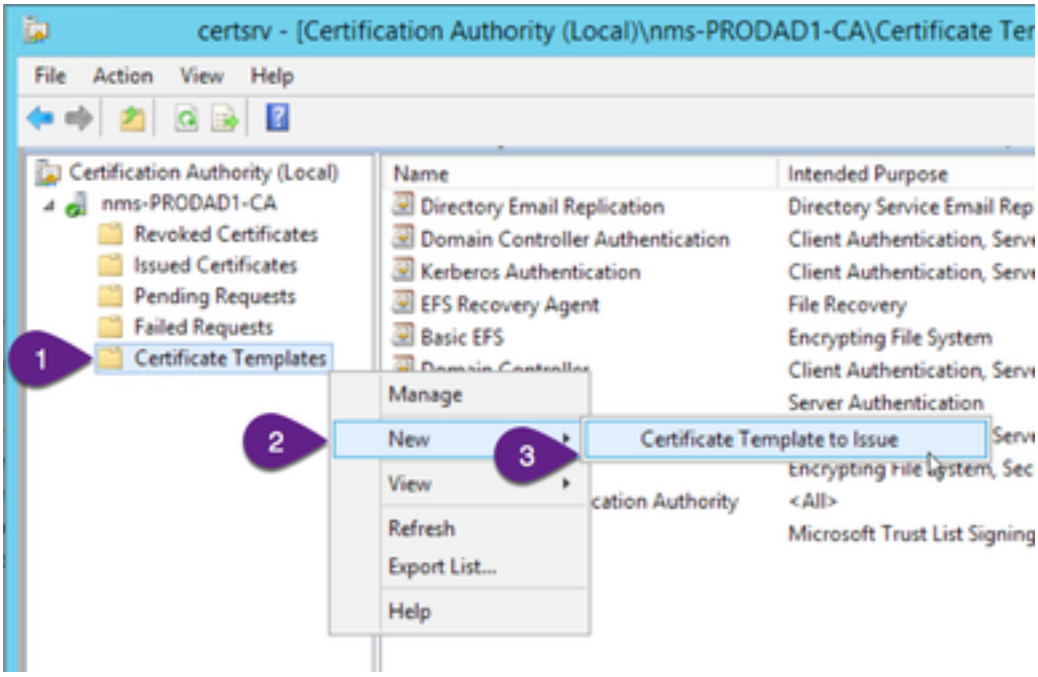


3. Back on the "Properties" screen, Select "Domain Computers"
4. Select "Enrol"
5. Select "Autoenroll"
6. Click "Ok"



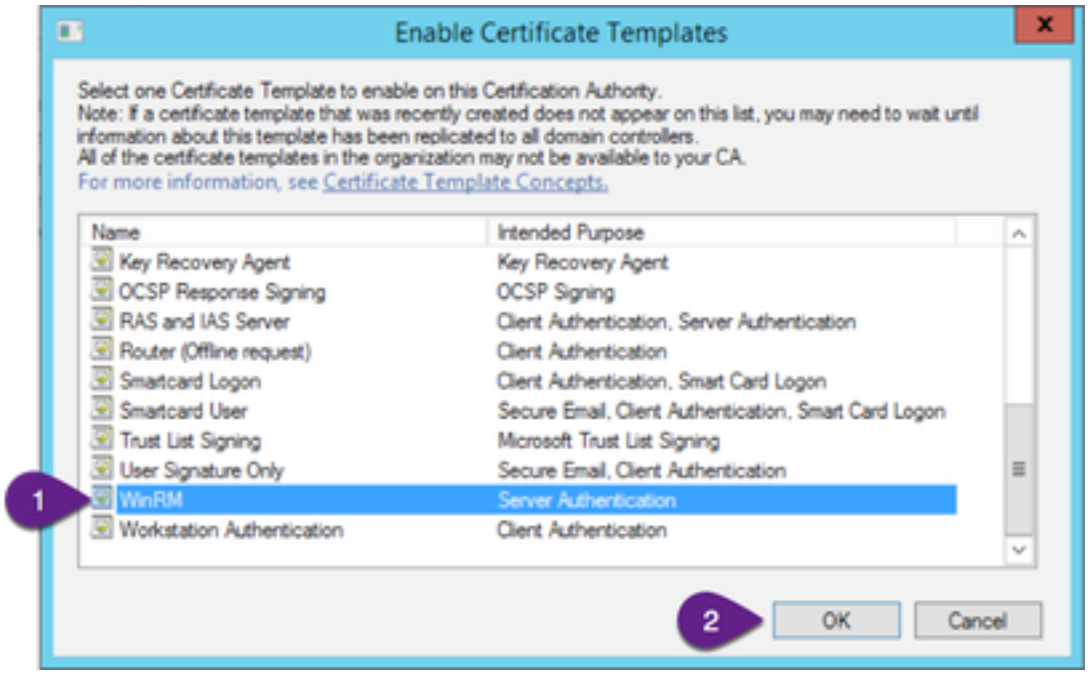
Back in the “Certificate Authority” app

1. Right click on “Certificate Templates”
2. Select “New”
3. Select “Certificate Template to Issue”



On the "Enable Certificate Templates" popup:

1. Find and select the created "WinRM" certificate template.
2. Click "Ok"

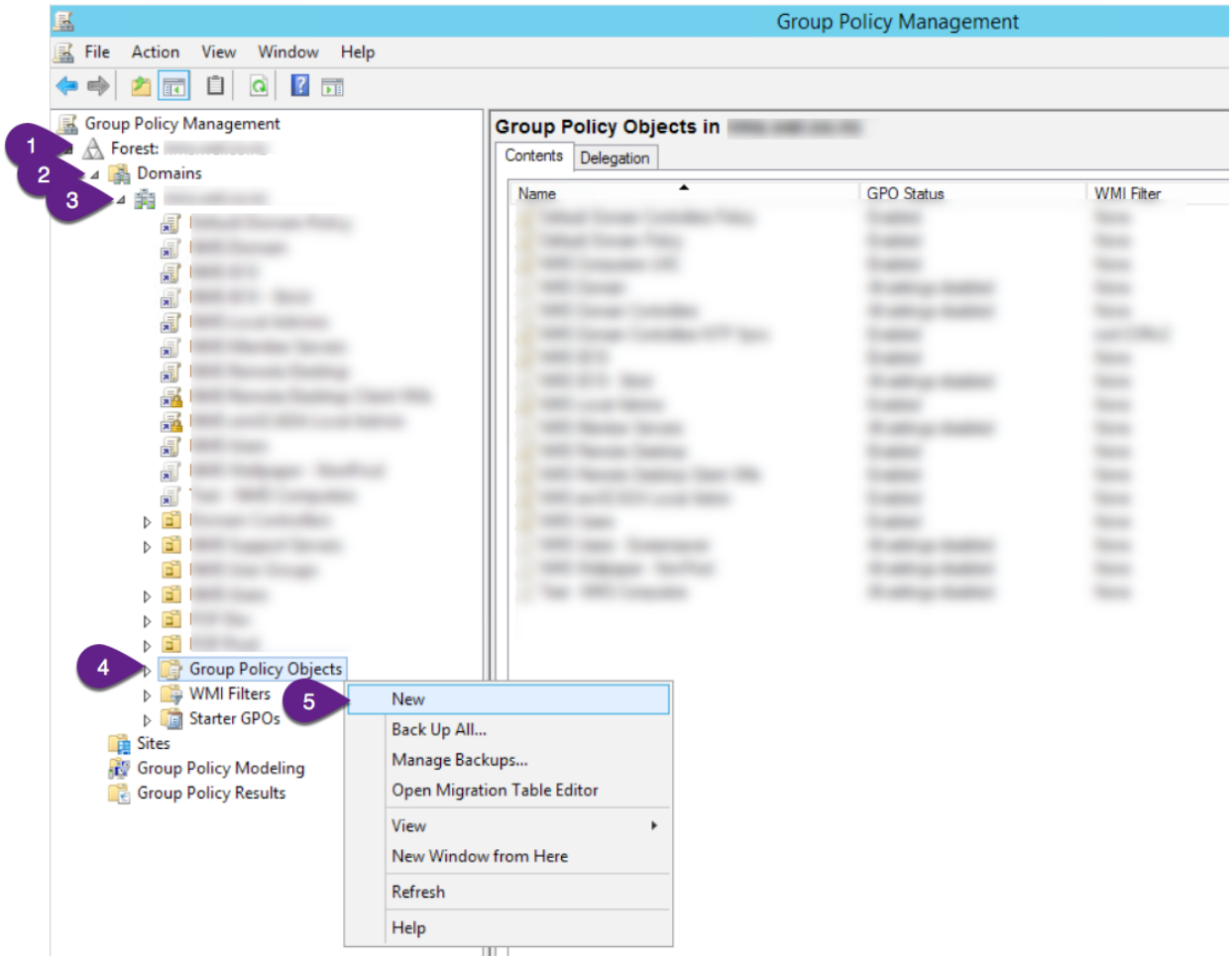


Create the Group Policy Object

The group policy object will automatically enable WinRM on Windows operating systems joined to the domain.

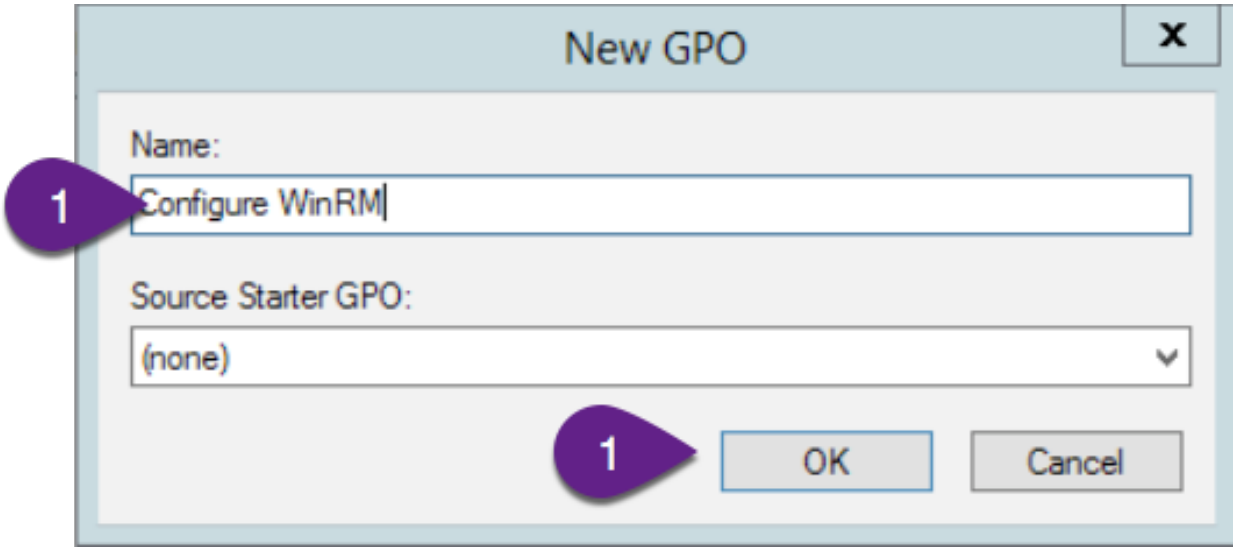
Open the “Group Policy Management” app

1. Expand the Forest
2. Expand the Domains
3. Expand the Domain
4. Select the “Group Policy Objects”
5. Right click and select “New”



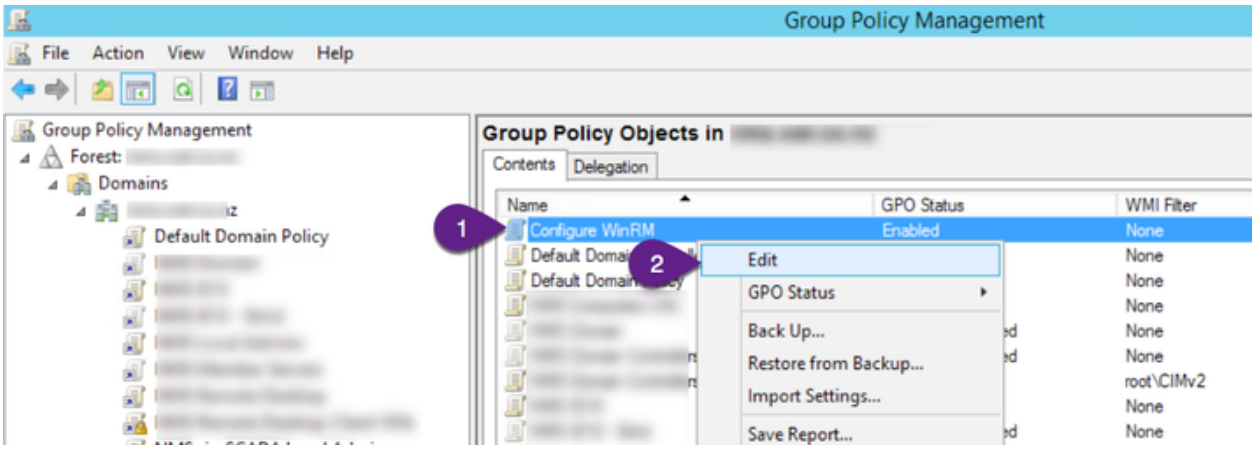
On the “New GPO” dialog

1. Enter “Configure WinRM” in the “Name” field
2. Click “OK”



In the "Group Policy Objects" list:

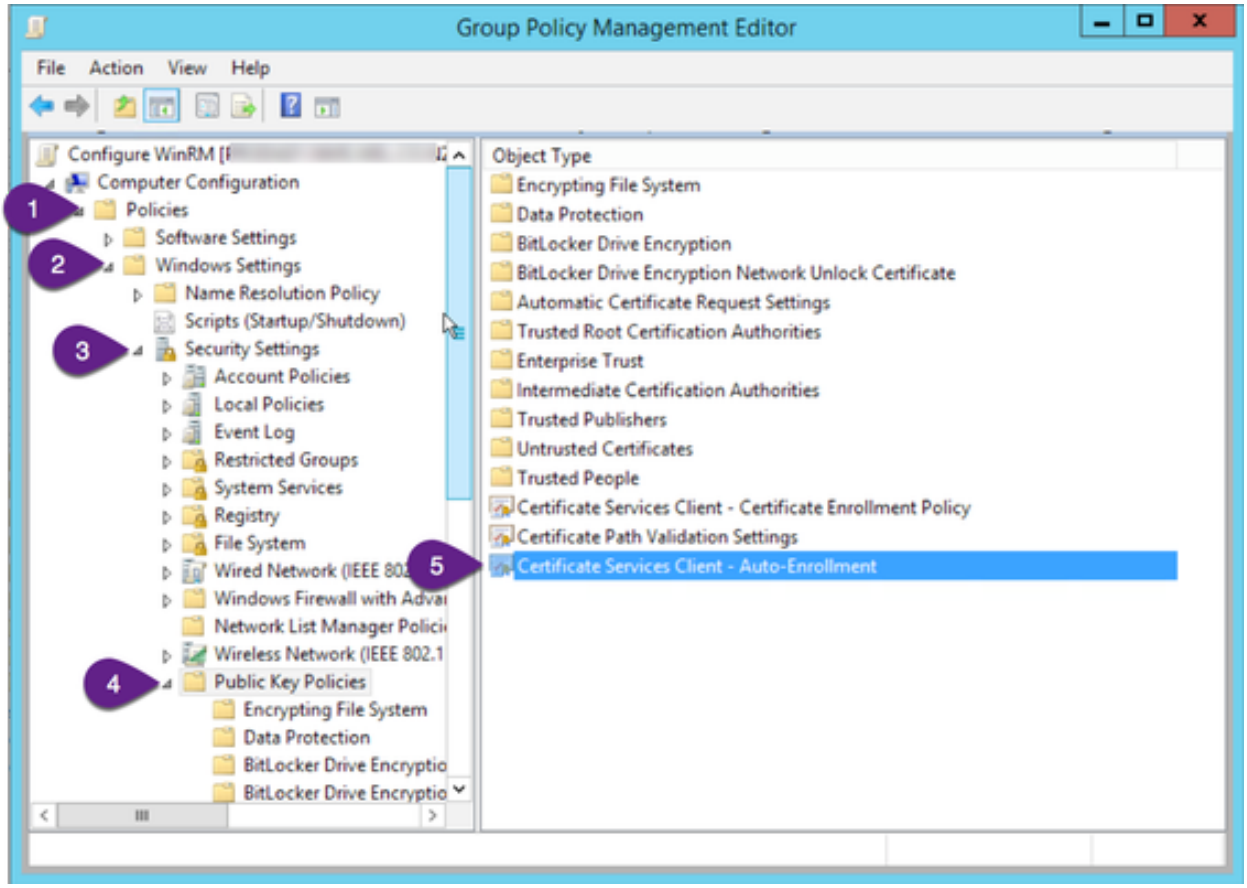
1. Right click on "Configure WinRM"
2. Select "Edit"



Enabling Autoenroll of Certificate Services

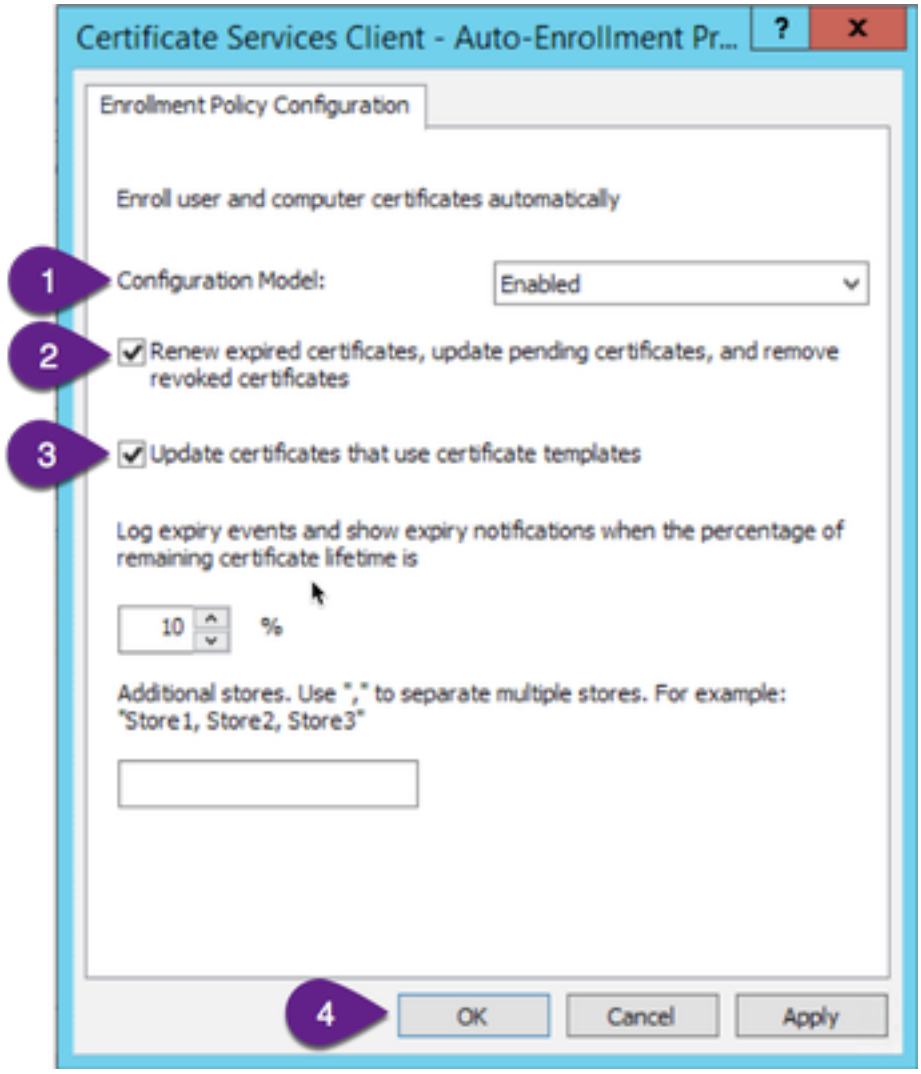
In the "Configure WinRM" Group Policy:

1. Expand "Computer Configuration" → "Policies"
2. Expand "Windows Settings"
3. Expand "Security Settings"
4. Select "Public Key Policies"
5. On the right hand pane, double click "Certificate Services Client – Auto-Enrollement"



In the “Certificate Services Client – Auto-Enroll. . . ” properties:

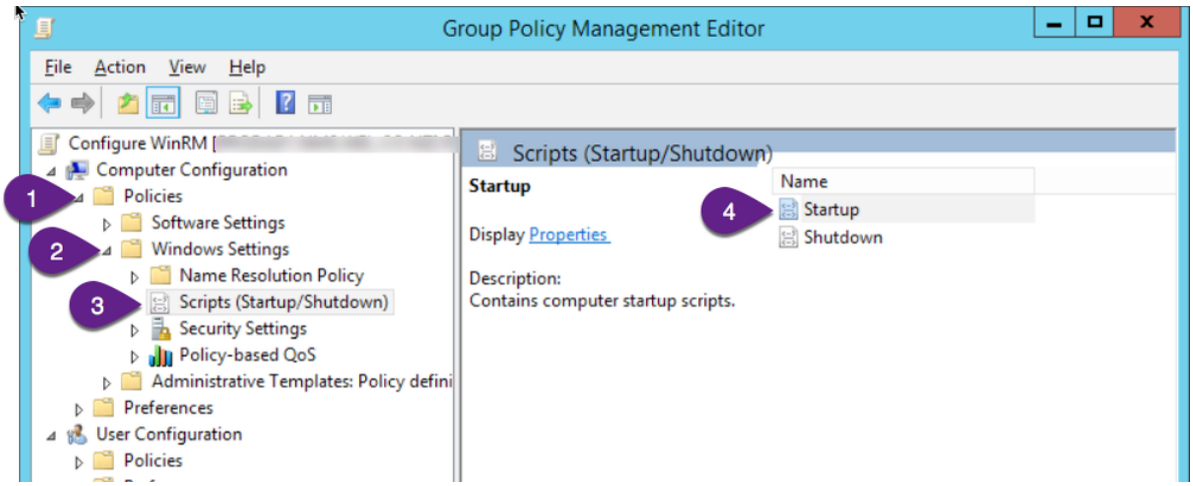
1. Set “Configuration Model” to “Enabled”
2. Check “Renew expired certificates. . . ”
3. Check “Update certificates that user certificate templates”
4. Click “OK”



Configure Enrolment Script

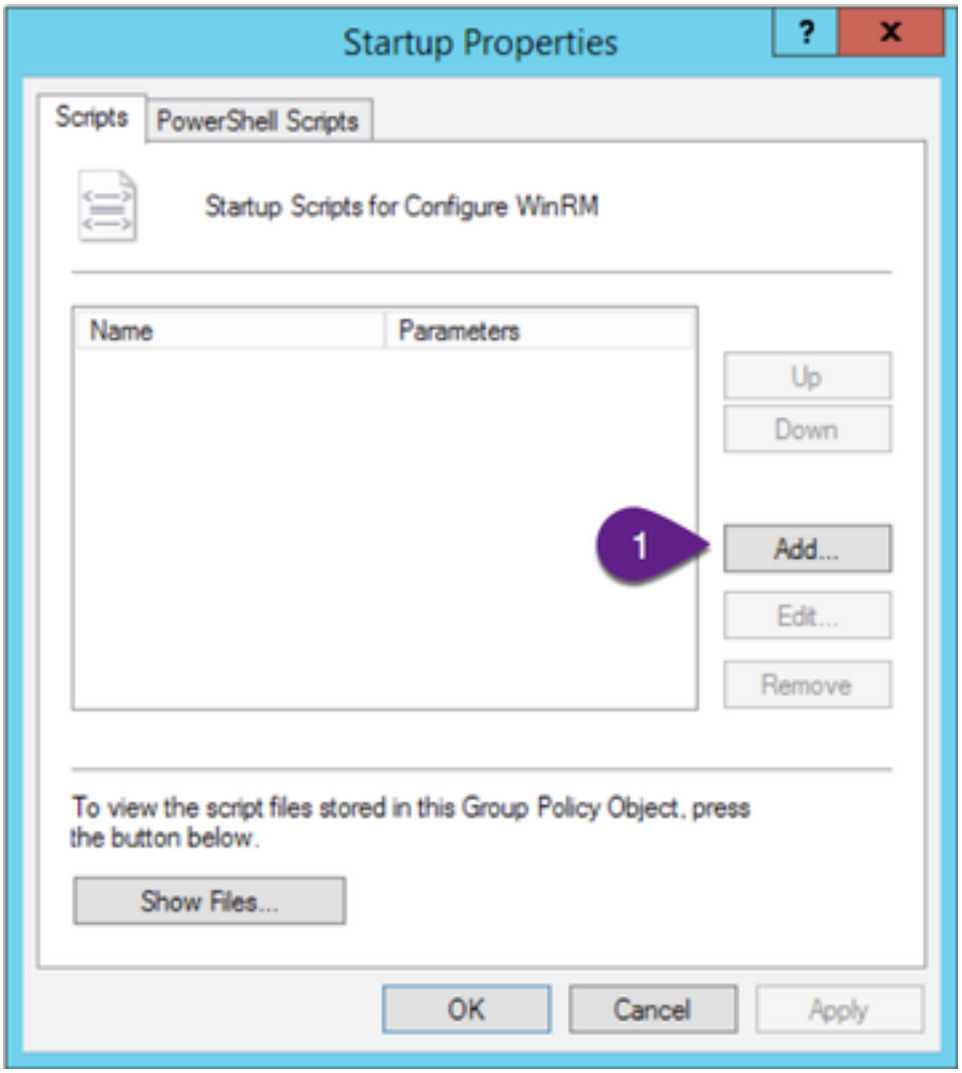
Expand the following

1. Expand “Computer Configuration” → “Policies”
2. Expand “Windows Settings”
3. Expand “Scripts”
4. Double click on “Startup”



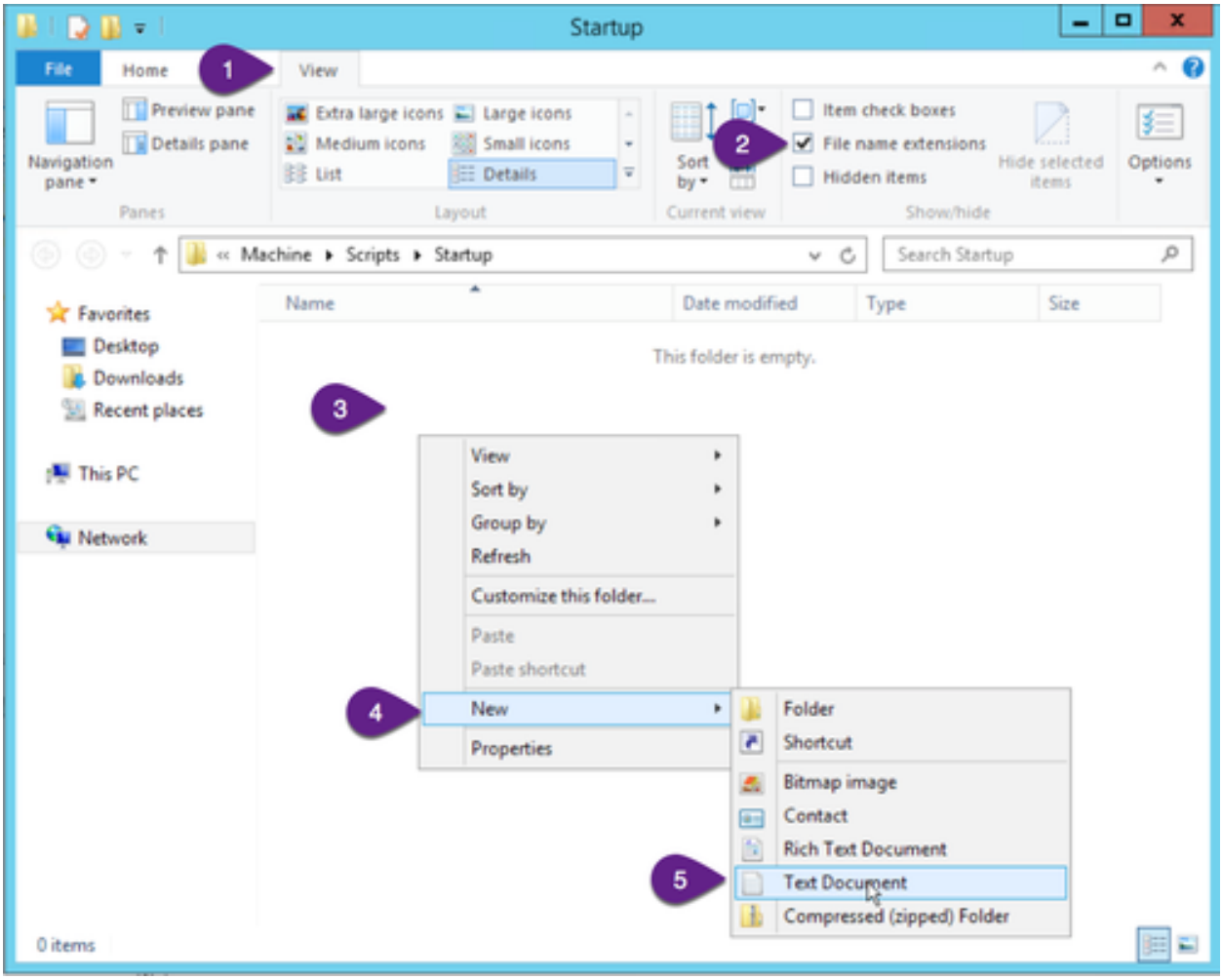
On the "Startup Properties"

1. Click "Show Files"



In windows explorer

1. Click “View”
2. Ensure “File name extensions” is checked
3. Right click on a blank space in the window
4. Select “New”
5. Select “Text Document”

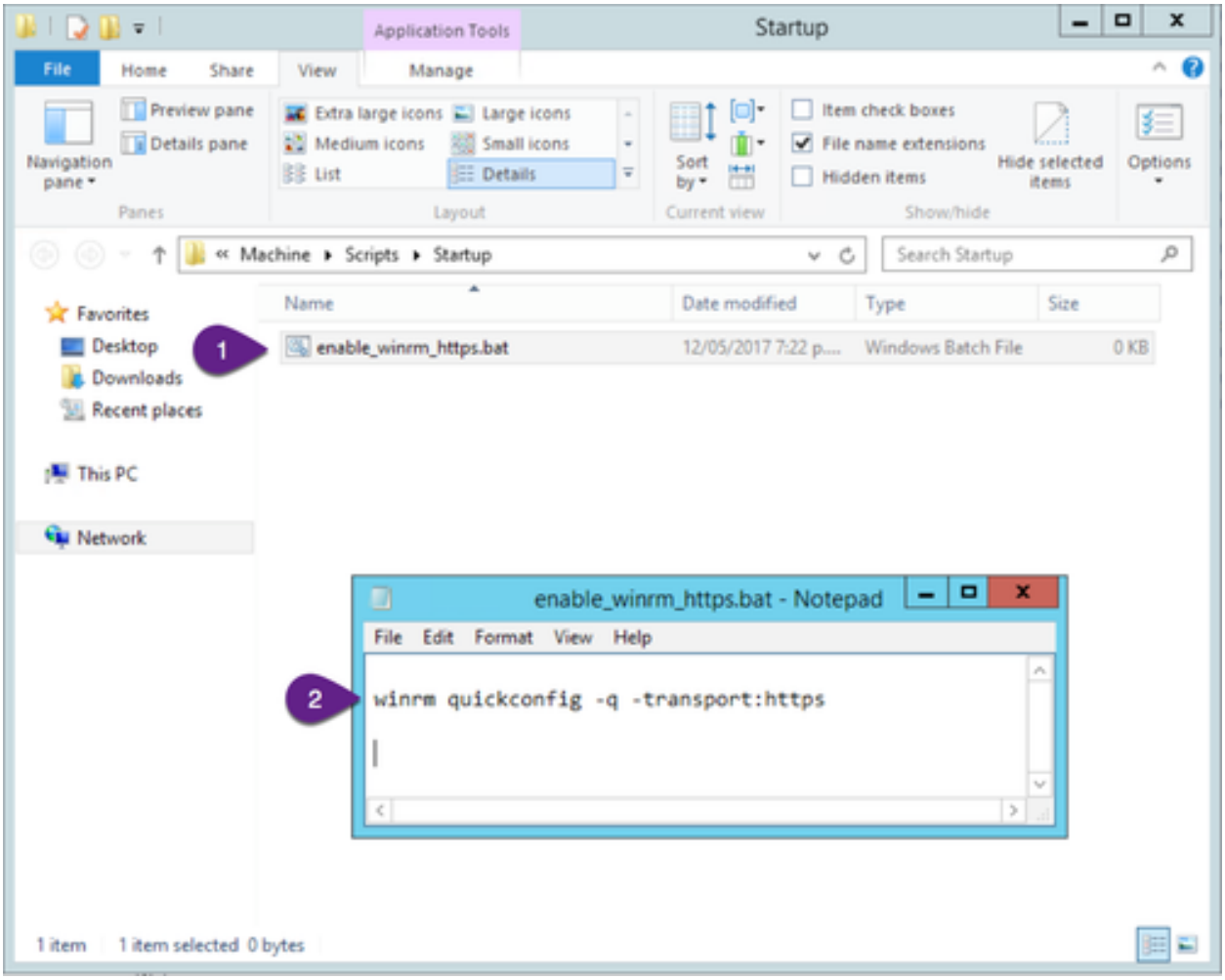


In windows explorer

1. Rename the file to “enable_winrm_https.bat”
2. Enter the following as the file contents

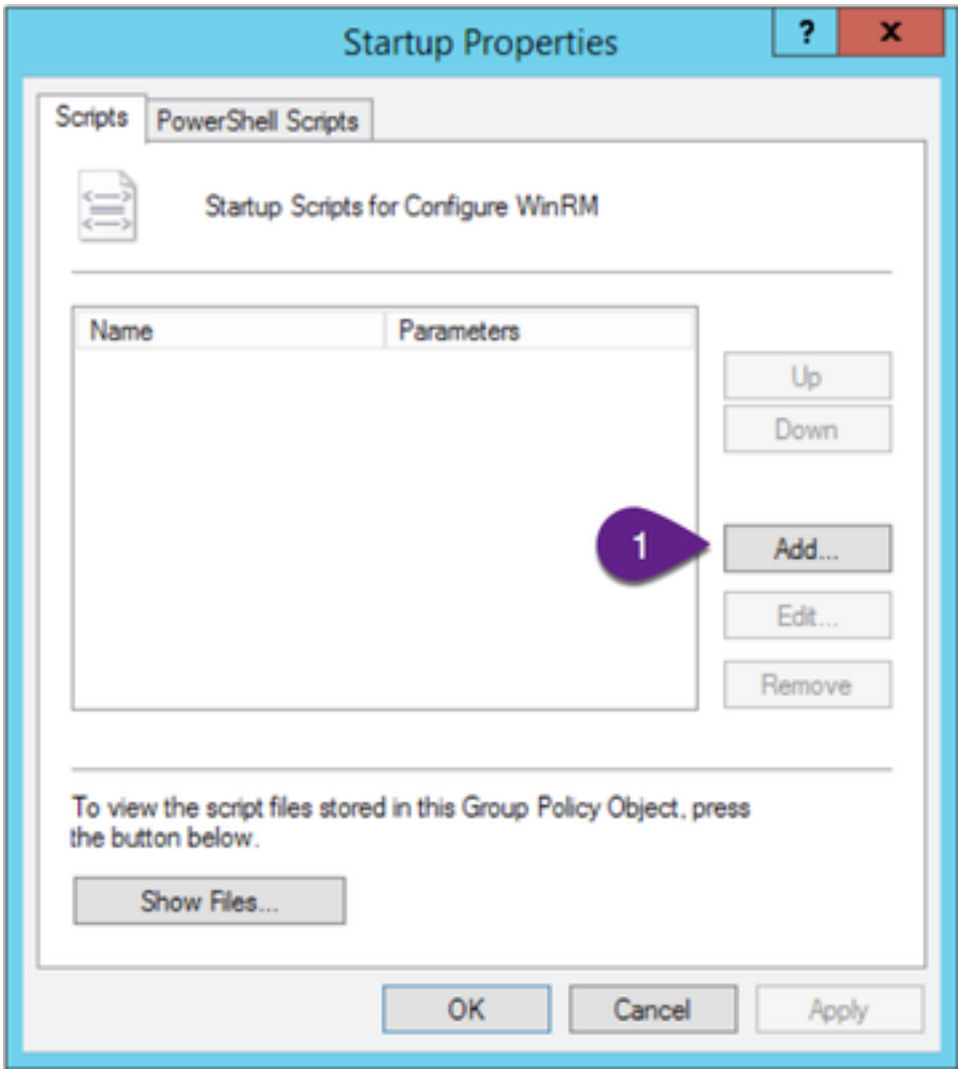
```
winrm quickconfig -q -transport:https
```

3. Save the file and close notepad
4. Close windows explorer

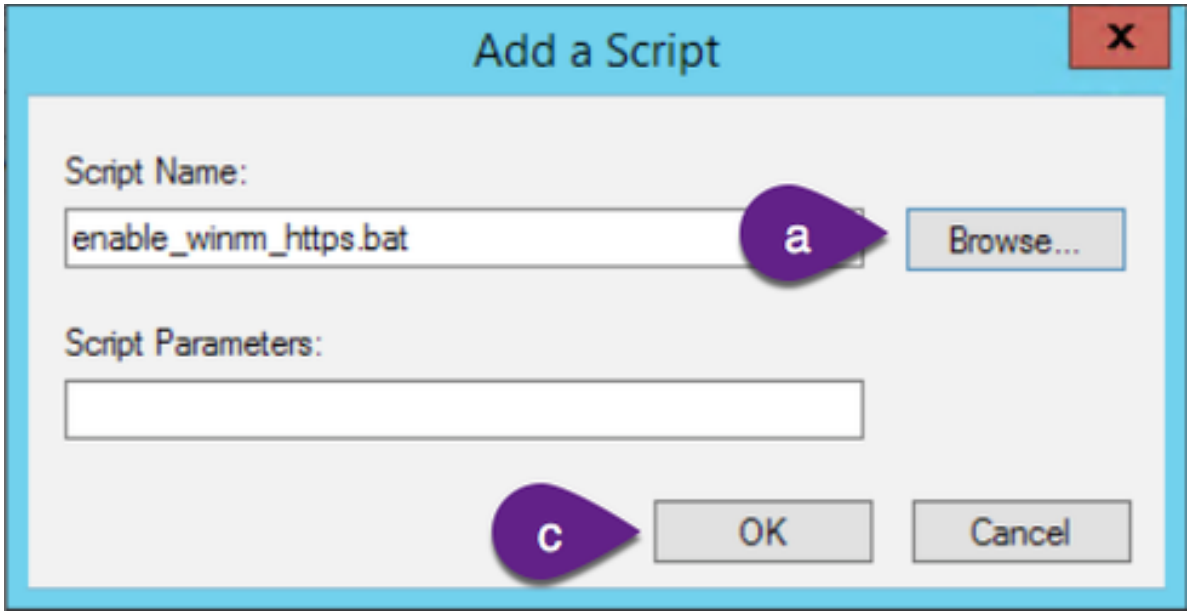


Back at the “Startup Properties” screen

1. Click “Add”



1. On the Add Script Diaglog
    1. Click “Browse”
    2. Select the “enable_winrm_https.bat”
    3. Click Ok

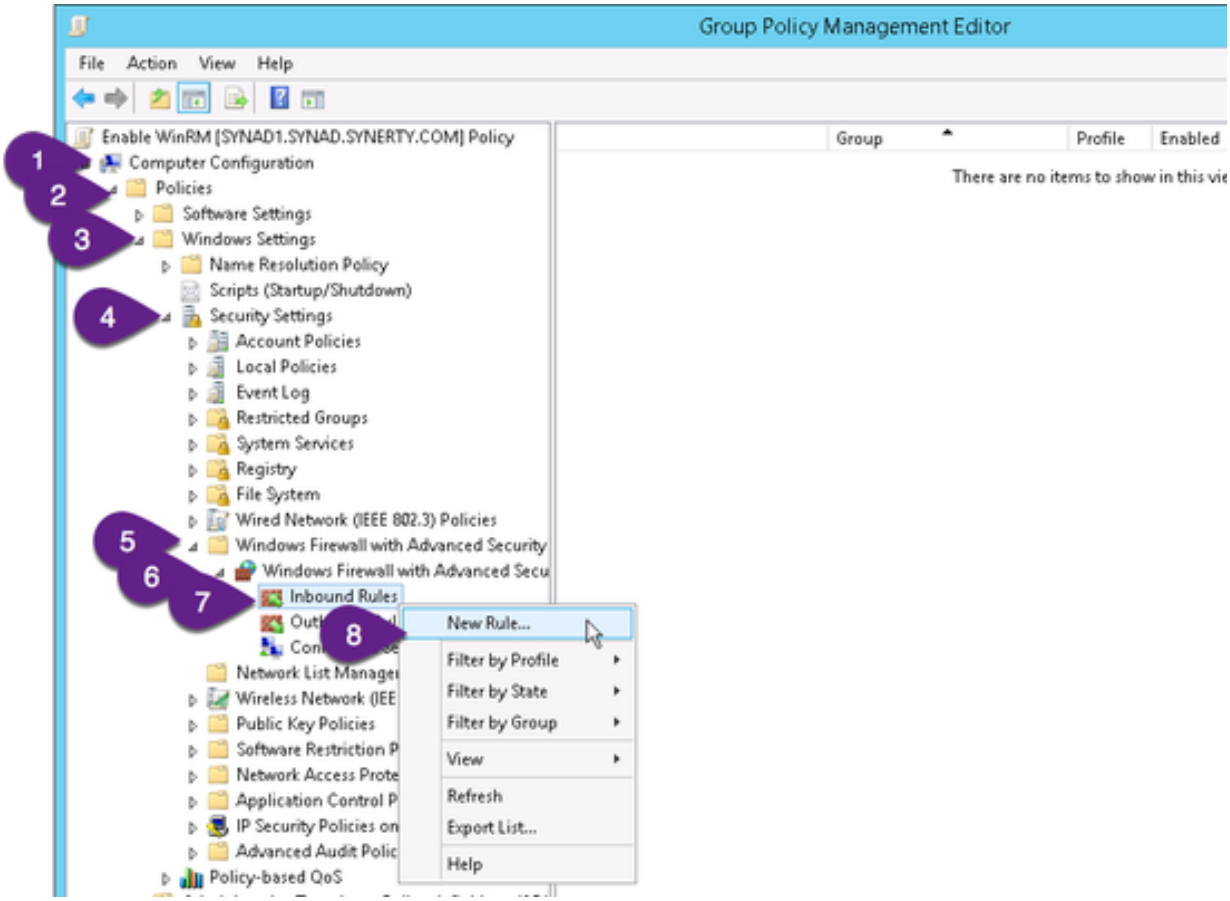


2. Click “Ok” on the Startup Properties dialog.

Configure Firewall for WinRM

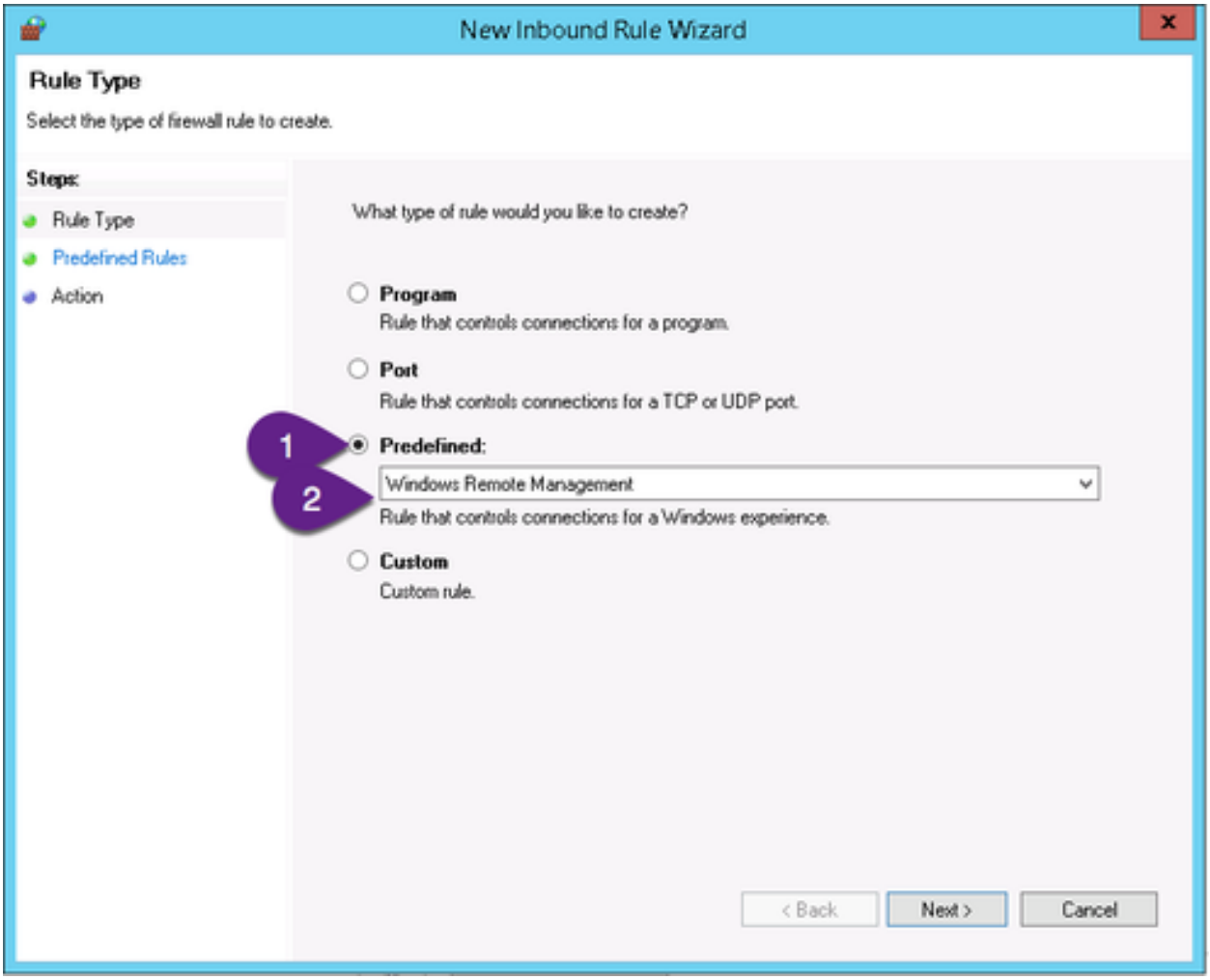
Expand the following

1. Expand “Computer Configuration”
2. Expand “Policies”
3. Expand “Windows Settings”
4. Expand “Security Settings”
5. Expand “Windows Firewall with Advanced Security”
6. Expand “Windows Firewall with Advanced Security – . . . .”
7. Right click on “Inbound Rules”
8. Click “New Rule”



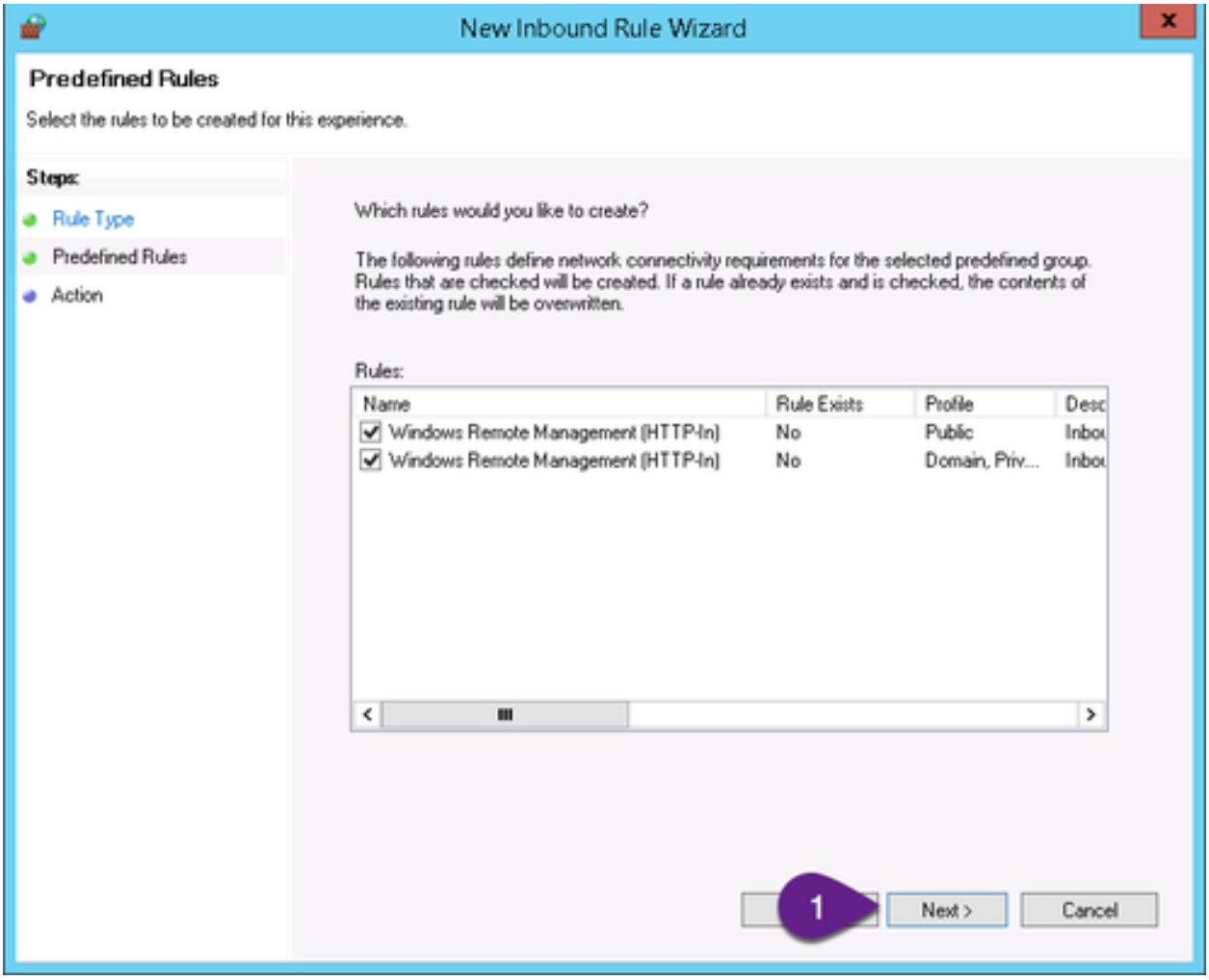
On the “New Inbound rule Wizard”

1. Click “Predefined”
2. Select “Windows Remote Management”
3. Click “Next”



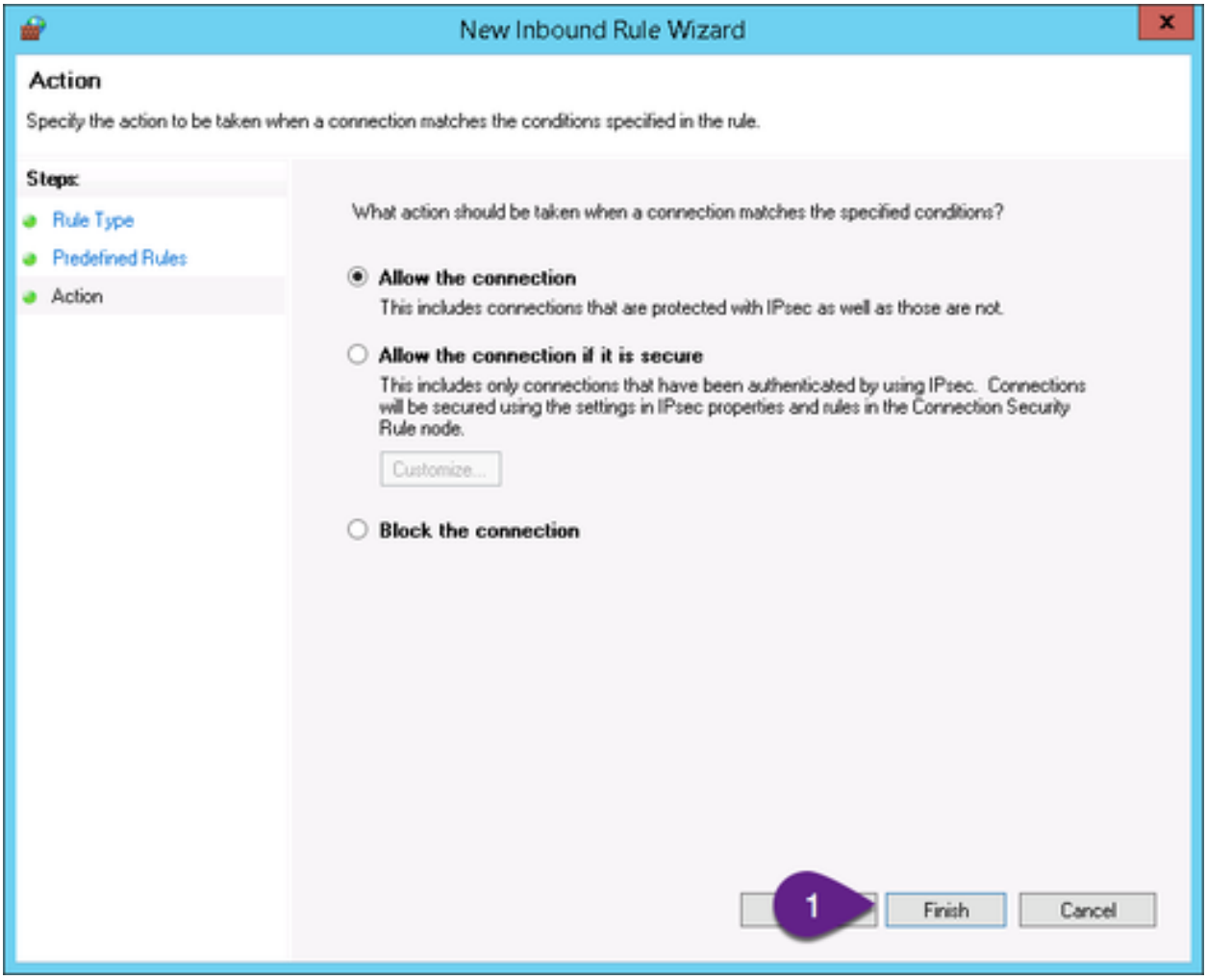
On the “Predefined Rules” screen

1. Click “Next”



On the “Action” screen

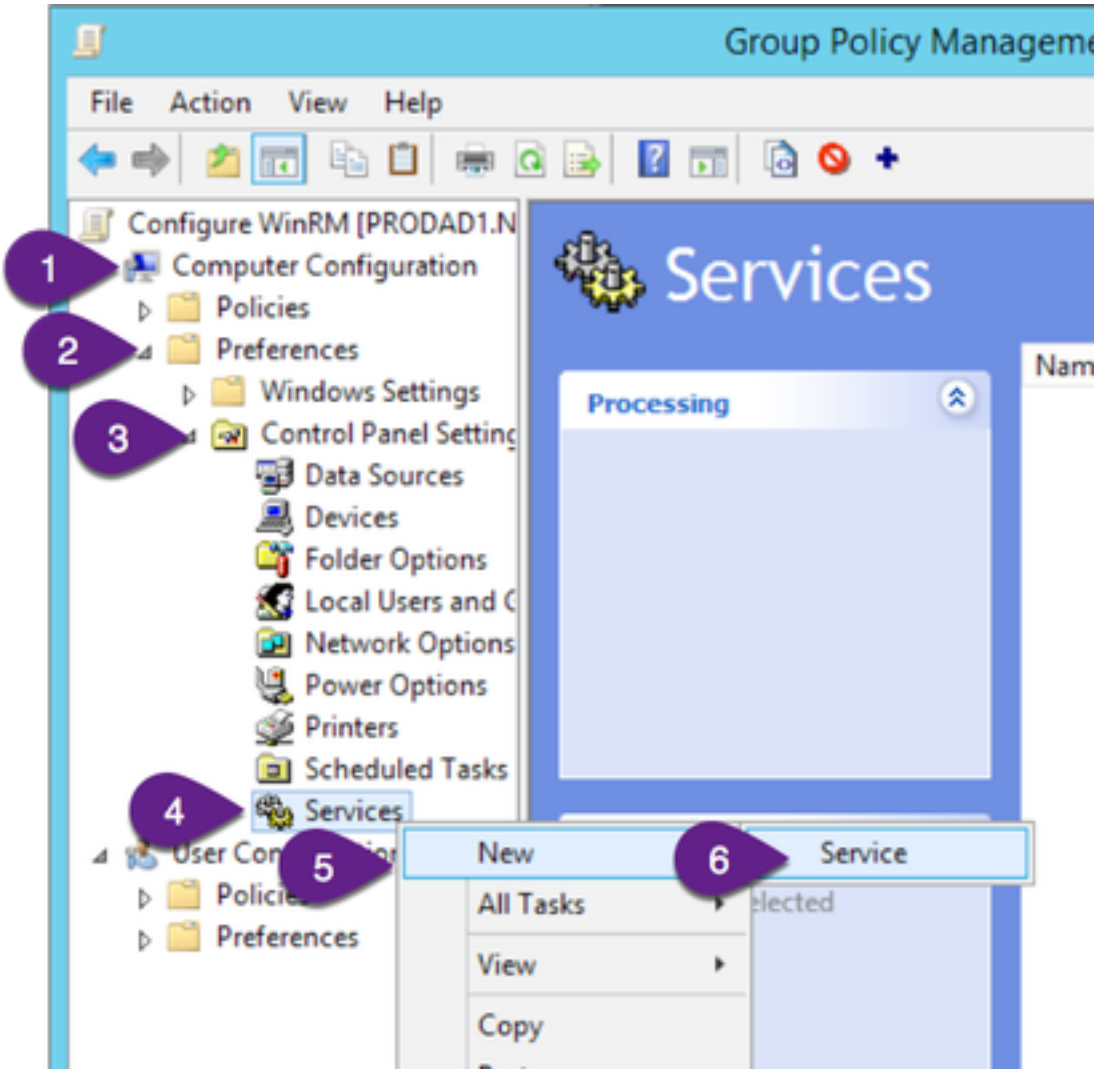
1. Click “Finish”



Enable WinRM

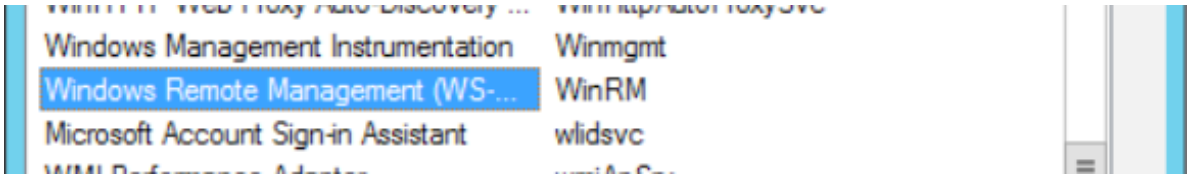
Expand the following

1. Expand “Computer Configuration”
2. Expand “Preferences”
3. Expand “Control Panel Settings”
4. Expand “Services”
5. Right click on “Services”
6. Select “New
7. Select “Service”

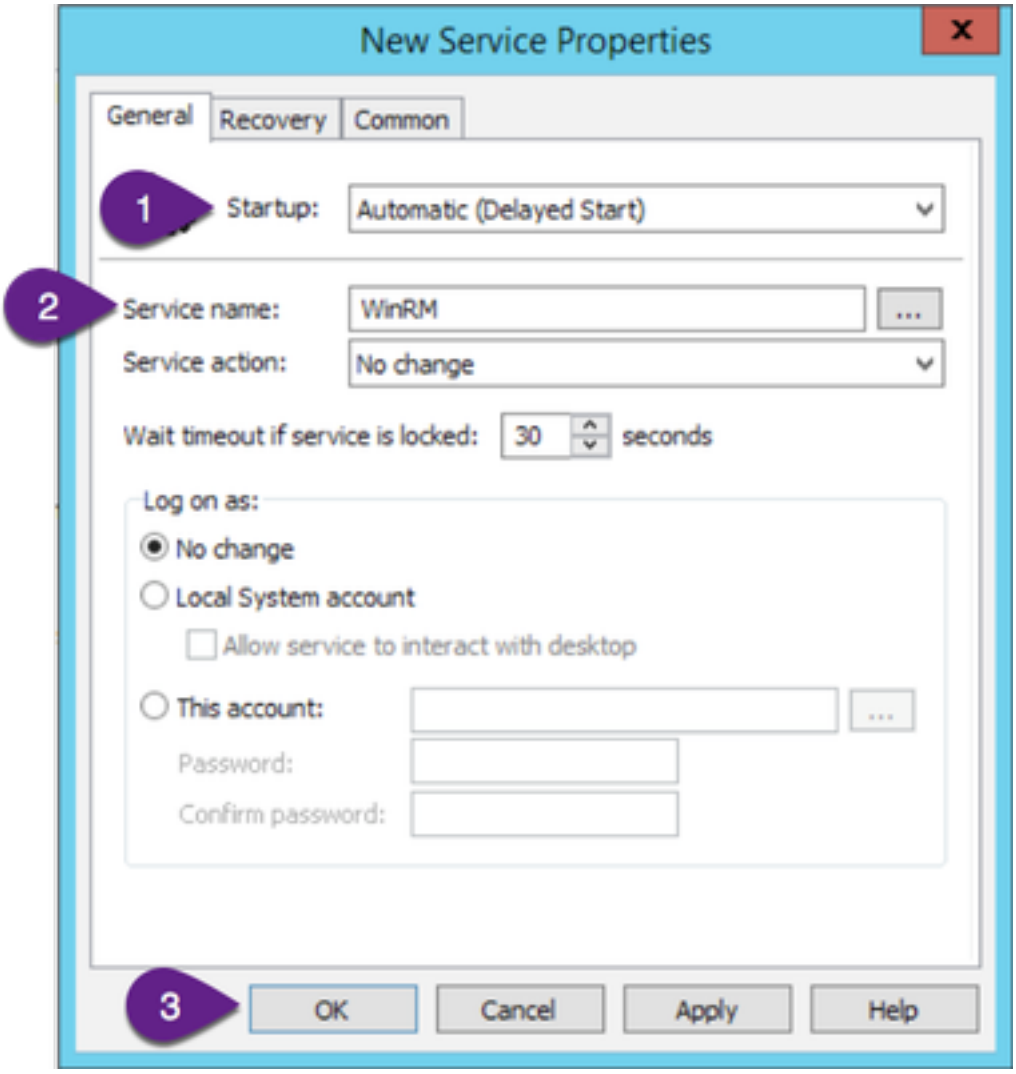


On the "Predefined Rules" screen

1. Change "Startup" to "Automatic (Delayed Start)"



2. Change "Service name:" to "WinRM"

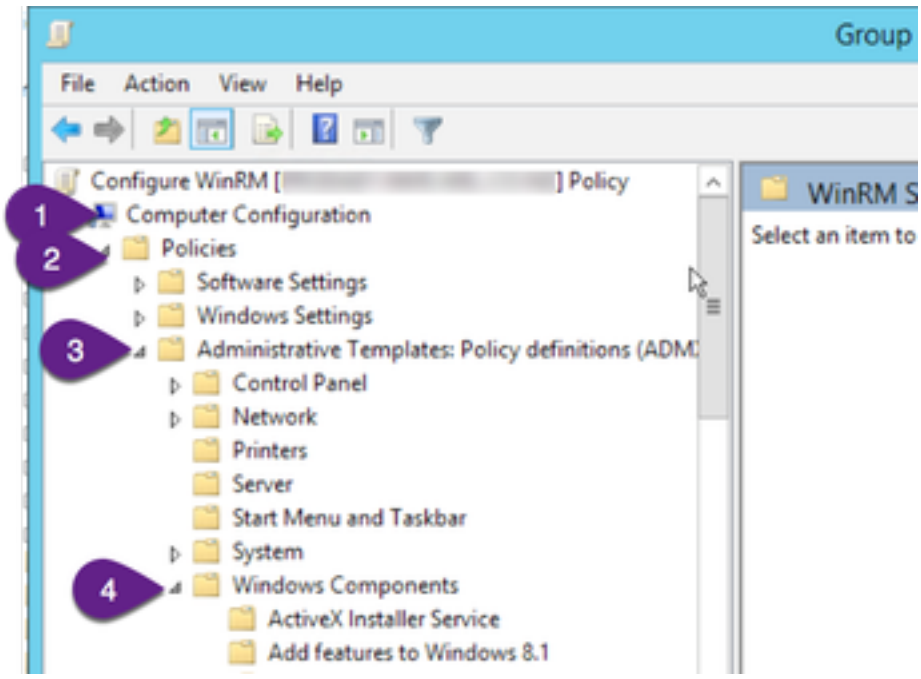


3. Click “OK”

Tweak WinRS

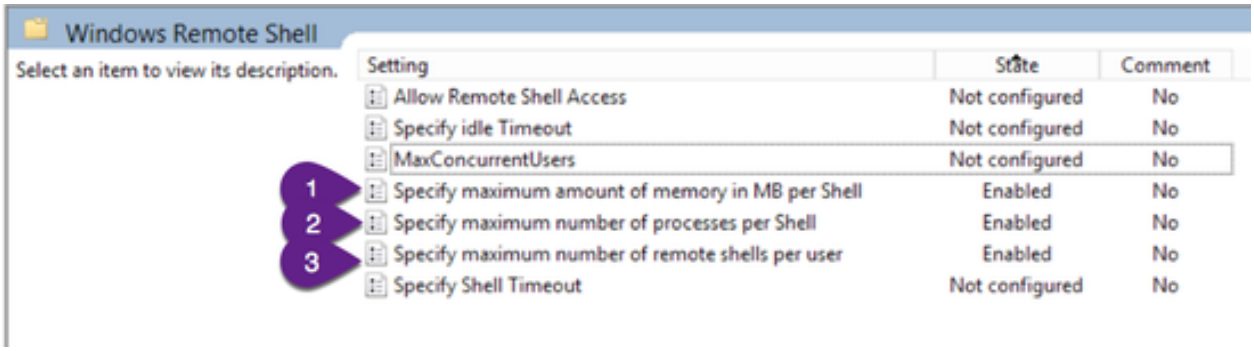
Expand the following

1. Expand “Computer Configuration”
2. Expand “Policies”
3. Expand “Administrative Template Policy”
4. Expand “Windows Components”
5. Expand “Windows Remote Shell



In the Settings pane:

1. Enable and Set **"Specify maximum amount of memory in MB per shell"**, to 1024
2. Enable and Set **"Specify maximum number of processes per shell"**, to 64
3. Enable and Set **"Specify maximum number of remote shells per user"**, to 64



Linking Group Policy

The group policy is now complete. Link the group policy to the desired OUs, and reboot the target servers.

Complete

This procedure is now complete, You can Create new Windows Server values in Attune and set the WinRM specification to "WinRM 2.0 HTTPS"

5.3.5 How To Enable Windows Built-in Administrator Account

In PowerShell, you can enable the built-in Administrator account on a local machine by using the net user command or the LocalUser cmdlet available in the *Microsoft.PowerShell.LocalAccounts* module (available in PowerShell 5.1 and later). Here's how you can do it using both methods:

Using 'net user' Command

The net user command is a legacy command available in Command Prompt but can also be executed from PowerShell.

Procedure

Run the following steps in PowerShell with administrative privileges.

Set the desired password, run the following command:

```
$PlainTextPassword = '{password}'  # Replace with the desired password
```

Enable the Administrator account, run the following command:

```
net user Administrator $PlainTextPassword /active:yes
```

This command sets the built-in Administrator account to active.

Using 'LocalUser' Cmdlet

In PowerShell 5.1 and later, you can use the Enable-LocalUser cmdlet from the Microsoft.PowerShell.LocalAccounts module to enable the built-in Administrator account.

Procedure

Run the following steps in PowerShell with administrative privileges.

Set the desired password, run the following command:

```
$PlainTextPassword = '{password}'  # Replace with the desired password
```

Enable the Administrator account, run the following command:

```
# Import the LocalAccounts module
Import-Module Microsoft.PowerShell.LocalAccounts

# Enable the built-in Administrator account
Enable-LocalUser -Name "Administrator"

# Convert the password to a secure string
```

(continues on next page)

(continued from previous page)

```
$SecurePassword = ConvertTo-SecureString -String $PlainTextPassword -AsPlainText -
↪Force

# Set the new password
Set-LocalUser -Name Administrator -Password $SecurePassword
```

This cmdlet enables the built-in Administrator account.

Complete

This procedure is now complete. Your Administrator account is enabled.

Note: The built-in Administrator account is usually disabled by default for security reasons. Enabling it can pose a security risk if not managed properly. Always ensure that it's secured with a strong password and that you understand the implications of enabling it.

5.3.6 How To Add Local Non-Administrator Account to the Remote Management Users Group

To add a local non-administrator account to the "Remote Management Users" group on a Windows machine using PowerShell, you can use the *Add-LocalGroupMember* cmdlet.

Add the account, run the following command:

```
Add-LocalGroupMember -Group "Remote Management Users" -Member "{username}"
```

Verify that the user has been added to the group by using the *Get-LocalGroupMember*.

```
Get-LocalGroupMember -Group "Remote Management Users"
```

Complete

This procedure is now complete.

5.3.7 Script Snippets

Powershell Script Snippets

How To Start Attune From Local User Snippet

```
$action = New-ScheduledTaskAction `
    -WorkingDirectory 'C:\Users\{winUser.user}' `
    -Execute `
    'C:\Users\{winUser.user}\AppData\Local\Programs\attune\resources\app.asar.
↪unpacked\py\attune-server'


$task = New-ScheduledTask -Action $action `
```

(continues on next page)

(continued from previous page)

```

Register-ScheduledTask 'attune-server' -InputObject $task `
   -User "{winUser.user}" `
   -Password "{winUser.password}"

Start-ScheduledTask 'attune-server'

Start-Sleep -Seconds 5

# Check that its running
Get-Process -Name "attune-server"
```

How To Restart a Windows VM Code Snippet

To restart a Windows VM:

```
$WAIT = 10
shutdown /r /t $WAIT /c "Restart from Attune"
Write-Host "Restarting in $WAIT seconds."
```

How To Shutdown Windows Snippet

To shutdown a Windows VM:

```
$WAIT = 10
shutdown /s /t $WAIT /c "Shutdown from Attune"
Write-Host "Shutting down in $WAIT seconds."
```

How To Get Fully Qualified Domain Name Windows Snippet

To get the FQDN for the current host:

```
[System.Net.Dns]::GetHostByName(($env:computerName))
```

How To Windows Commands Exit with the Wrong Code

Windows commands often exit with a 0 code, even if the command fails. Catch errors like this using Select-String; the Powershell equivalent of `grep`.

```
echo "Attempting to sync Group Policy..."
$out = gpupdate
echo $out

$result = Select-String -InputObject $out -Pattern "error"
if ($result.length -ne 0) {
    exit 1
}
```

How To Search for Installed oVirt VIRTIO Drivers

This command will list the oVirt VIRTIO Drivers installed on a machine:

```
PS C:\Users\Administrator> driverquery | Select-String -Pattern "VirtIO"
```

How To Set Timezone Server

```
w32tm /query /status

Write-Host "Setting timezone server as {timezoneServer.fqn}"

w32tm /config /syncfromflags:manual `
    /manualpeerlist:"{timezoneServer.fqn}" `
    /reliable:yes /update
```

If you get the error when running the above command at line 1: `The following error occurred: The service has not been started. (0x80070426)` Run these commands:

```
w32tm /unregister
w32tm /register
net start w32time
```

Bash Script Snippets

How To Iterate Over Files in a Directory

`ls` is not the command that expands globs to filenames, bash does that. In the following snippet, bash expands `/dev/{s,v}*1` and expands it to `/dev/sda1` which is what the for loop is then passed.

```
# This only works if there are no spaces in the file names.
for f in /dev/{s,v}*1
do
    echo "$f"
done
```

How To For Loop Over CSV File Which Contains Spaces

Loops over a comma separated value file that contains spaces. This is perfect for Server Groups in Attune as the server group values are stored as as CSV's.

```
# The following three lines are identical as $IFS=" "
# if $IFS="," then the line with brackets around it would generate an array
# which isn't suitable for an input for this example.
VAR=("Test 1","Test 2","Test 3","Test 4")
VAR="Test 1","Test 2","Test 3","Test 4"
VAR="Test 1,Test 2,Test 3,Test 4"

# Set the field separator to a comma,
# and echo the variables contents into the stdin of read
# tell read to produce a variable of type array
```

(continues on next page)

(continued from previous page)

```
IFS=',' read -ra items <<< $VAR

# Iterate over the the indexes of the array, not the items
# This works with for, because for will split on spaces and
# the indexes don't have spaces in them, eg 1 2 3 4 5
for index in "${!items[@]}"
do
    # Dereference the index of the array,
    # This way, we get our correct value even if it has spaces in it.
    echo "Testing CSV for loop: ${items[index]}"
done
```

Output:

```
Testing CSV for loop: Test 1
Testing CSV for loop: Test 2
Testing CSV for loop: Test 3
Testing CSV for loop: Test 4
```

Examples

The following example will print the name, IP address, and FQDN for each of the servers in the appServers server group.

```
IFS=',' read -ra ip <<< "{appServers.serverIps}"
IFS=',' read -ra fqn <<< "{appServers.serverFqns}"
IFS=',' read -ra name <<< "{appServers.serverNames}"

for i in "${!ip[@]}"
do
    echo ${name[$i]}
    echo ${ip[$i]}
    echo ${fqn[$i]}
done
```

Build a List of Items to Loop Over

Build a list with comments:

```
list=""
list+=" git"            # version control
list+=" vim"            # text editor
list+=" hdparm"         # hard drive tools
list+=" nmap"           # network debugging
list+=" fail2ban"       # brute-force protection

for F in ${list}
do
    echo ${F}
    done
```

Or if you want to keep it simple:

```
list="
    git
    vim
    hdparm
    nmap
    youtube-dl
    fail2ban
"

for F in ${list}
do
    echo ${F}
done
```

How To Iterate Over FileNames With Spaces

This snippet will iterate over file names containing spaces in a directory.

```
echo /my/**/dir | while IFS= read -r file
do
    echo "Doing some work with ${file}"
    ls "${file}"
done
```

This snippet will iterate over files that contain a string found by grep.

```
SCHEMA="newschema"
function listFiles {
    grep -lRi OLDSCHEMA \
       /dir1 \
       /dir2
}

listFiles | while IFS= read -r file
do
    echo "Replacing OLDSCHEMA with ${SCHEMA^^} in ${file}"
    # ^ upercases the first letter, ^^ upper cases them app
    sed-i "s/CORPDBA/${SCHEMA^^}/g" "${file}"
done
```

Conditional Bash Step in Attune

The following code can be used to create a conditional step in attune, that will only execute if an Attune variable is literally `true`.

```
if {attuneCheckVariable}
then
    echo "Check passed, will execute code"
    # code to execute here
else
    echo "Check failed, won't execute code"
fi
```

How To Perform SSL Handshake and Save Certificate to File

The following example will retrieve the SSL certificate from a server and save it to a file named `out.cer`.

```
# Define ip and port

openssl s_client -connect ${ip}:${port} 2>/dev/null </dev/null | openssl x509 -out␣
↪out.cer
```

How To Append Lines To File If They Do Not Exist

```
FILE=/etc/samba/smb.conf

LINES=$(cat <<EOF
[scan]
Comment = Scans
Path = /var/scan
Browseable = yes
Writeable = Yes
only guest = no
guest ok = no
create mask = 0777
directory mask = 0777
Public = no
EOF
)

grep -q "${LINES}" ${FILE} && echo "Nothing to do" || \
    echo "${LINES}" | tee -a ${FILE}
```

How To Grep the Standard Error

The following example:

1. Executes perl Makefile.PL and saves the standard error of this command to a variable.
2. Echoes the standard error so that it appears normal in the Attune output.
3. Searches the standard error for a specific message and fails the step with an exit code of 34 if found.

The standard output is printed as normal.

```
{ stderr="$( { perl Makefile.PL; } 2>&1 1>&3 3>&- )"; } 3>&1
IFS=$'\n' echo "${stderr}" 1>&2
grep -q "Warning: prerequisite" <<< ${stderr} && exit 34
```

Filter and Save Output to File

The following example will write the complete output of a command to a file and print a filtered subset of this output to STDOUT.

```
<command> | tee -a output_file.txt | grep <filter>
```

Exit Codes in Functions

Don't use `exit` in bash functions, they should instead use `return` so that the calling script has the option of what to do on failure.

If `set -o errexit` is set then the script will still exit with failure, but if the calling script doesn't want that to happen then they can add `myFunc || true` to stop it happening.

```
function bad {
  echo "# bye"
  return 1
}
function good {
  echo "# hi"
  return 0
}
```

Output:

```
bad; echo "# $?"
# bye
# 1

good; echo "# $?"
# hi
# 0
```

How To Piping into Functions

Bash can pipe data into functions.

There is one stdin for the whole script, and the function can read the data for this.

```
function pipeMe1 {
    # Just read a line
    read var
    echo "One line read, it's :${var}"

    # More stdin can be left here, `read` more of it or cat the rest.
    # cat may hang if stdin is closed
}

function pipeMe2 {
    # Now read the rest of the stdin
    cat
}

```

(continues on next page)

(continued from previous page)

```
function pipeMe3 {
    # Now read the rest of the stdin
    cat <<EOF
    Here is my heredoc
    $(cat)
    I've wrapped that stdin with more stdin
EOF
}

pipeMe1 <<EOF
line 1
line 2
EOF

pipeMe2 <<EOF
line 1
line 2
EOF

pipeMe3 <<EOF
line 1
line 2
EOF
```

How To Iterate Over RPM Files List and Install RPM if not Already Installed

```
PACKAGE="sqlite-devel"

cd "/tmp/${PACKAGE}"

i=0
# lists the RPMS in the directory
for RPM in *
do
    # remove the extension of the RPM
    RPM_NAME_ONLY=$(echo "${RPM}" |
        sed -e 's/\([^.]*\).*/\1/'
        -e 's/\(.*\)-.*/\1/'
    )

    if rpm -qa | grep "${RPM_NAME_ONLY}";
    then
        echo "${RPM_NAME_ONLY} is already installed."
    else
        echo "Installing ${RPM} offline."

        yum localinstall --assumeyes --cacheonly \
            --disablerepo=* "${RPM}"
    fi

    ((i+=1))
done
```

Explanation of sed REGEX

Reference is at https://www.unix.com/shell-programming-and-scripting/150883-remove-version-numbers-package-lists.html

Using `sqlite-devel-3.7.17-8.el7_7.1.x86_64.rpm` as an example this is what the first stage sed does:

```
echo "sqlite-devel-3.7.17-8.el7_7.1.x86_64.rpm" | sed -e 's/\([^.]*\).*/\1/'
```

Gives this output:

```
sqlite-devel-3
```

In `\([^.]*\).*`:

- The `[^.]*` matches everything that is not a `.` zero or more times.
- The `\([^.]*\)` matches everything that is not a `.` zero or more times and assigns the result to variable 1.
- The `([^.]*\).` matches everything up to but the first `.`.
- The last part `.*` strips away the characters after the first `.`.
- Finally the `\1` substitutes the the match we got in variable 1.

The second stage sed:

```
echo "sqlite-devel-3.7.17-8.el7_7.1.x86_64.rpm" | \
  sed -e 's/\([^.]*\).*/\1/' \
  -e 's/\(.*\)-.*/\1/'
```

Gives this output:

```
sqlite-devel
```

In `(.*\)-.*`:

- `(.*\)-` matches everything before the first `-` and stores the result into variable 1.
- `-.*` strips away everything after the first `-`.
- Finally the `\1` substitutes the the match we got in variable 1.

How To Check Whether SELinux is Enabled

```
if sestatus | grep -i "SELinux status:" | grep -q "enabled"
then
    echo "SElinux is enabled"
else
    echo "SElinux is disabled"
fi
```

How To Download RPM For Offline Use

Here is an example usage of dnf to download cabextract and all it's dependencies.

```
dnf download --resolve --alldeps cabextract
```

SQL (Oracle) Code Snippets

How To Get Readable Results in sqlplus

Use SQL Developer wherever possible. It is much better than sqlplus and has excellent tools for browsing the database and debugging.

If you don't have access to SQL Developer, you can format the sqlplus output as follows.

```
set linesize 400
set pagesize 1000

-- Run a separate command for each column to be displayed, as follows.
-- a20 means the column is 20 characters wide. Use this format for varchar2 fields.
-- For numbers, each digit is represented by a 9

column OBJECT_NAME format a20
column OBJECT_TYPE format a20
column STATUS format a20
column PRICE format 999.99
```

How To Loop over a Node List

Attune node lists (formerly known as server groups) store data in comma-separated values (CSV) format. This example loops over the replicationServers node list and updates the HOST_DETAILS table based on the CSV text values replicationServerHostIds and replicationServerTypeList.

```
DELETE FROM HOST_DETAILS;

set serveroutput on
DECLARE
  hosts VARCHAR2(200) := '{replicationServers.serverHostnames}';
  host_ids VARCHAR2(100) := '{replicationServerHostIds}';
  host_types VARCHAR2(100) := '{replicationServerTypeList}';
BEGIN
  FOR row IN
  (SELECT trim(regexp_substr(hosts, '[^,]+', 1, LEVEL)) host,
          trim(regexp_substr(host_ids, '[^,]+', 1, LEVEL)) host_id,
          trim(regexp_substr(host_types, '[^,]+', 1, LEVEL)) host_type,
          LEVEL priority

   FROM dual
    CONNECT BY LEVEL <= regexp_count(hosts, ',')+1
  )
  LOOP
    dbms_output.put_line('Adding server ' || row.host
                         || ', id=' || row.host_id
                         || ', type=' || row.host_type
```

(continues on next page)

(continued from previous page)

```
                        || ', priority=' || row.priority);
    EXECUTE IMMEDIATE
       'Insert into ENMAC.HOST_DETAILS ('
           || 'HOST_NAME,HOST_PRIORITY,HOST_ID,HOST_LAST_TRANSACTION,HOST_STATUS,'
           || 'HOST_TYPE,HOST_NETWORK,HOST_LAST_ERROR,HOST_GROUP,HOST_SEND_TXNS,HOST_
↪RECV_TXNS,HOST_THIS_SERVER,'
           || 'HOST_LAST_REQUEST_ID,HOST_TRANSACTION_TIME,LAST_SEND_TRANSACTION_
↪SERIAL,HOST_HALT,'
           || 'HOST_ROLL_TRANSACTION_ID'
           || ')'
           || ' VALUES (''' || row.host || ''',' || row.priority || ',''' || row.host_
↪id || ''''
           || ',0,1, '|| row.host_type ||' , ''Default'',0,null,''Y'',''Y'',null,null,
↪null,null,null,null)';
   END LOOP;
 END;
```

How To Check the Grants on a User

Run this query in SQL Developer. You'll be prompted for the user when you run the query. Alternatively run the query in **sqlplus** but substitute a string in place of `:the_user` variable.

```
SELECT GRANTEE, PRIVILEGE
    FROM sys.dba_sys_privs
    WHERE grantee = :the_user
UNION
SELECT GRANTEE, PRIVILEGE
    FROM dba_role_privs rp JOIN role_sys_privs rsp ON (rp.granted_role = rsp.role)
    WHERE rp.grantee = :the_user;
```

How To Create Bash Script to Perform Oracle Commands as Oracle on Target Server

Option 1:

```
F=$(mktemp)
printf "
alter system set COMPATIBLE='19.3.0' scope=spfile;
shutdown immediate;
startup mount;
alter database open;
" > ${F}

{oracleDbHome}/bin/sqlplus / as sysdba @${F}
rm ${F}
```

Option 2:

```
# Ensure ORACLE_HOME is set
# Ensure ORACLE_BIN is set to $ORACLE_HOME/bin
# Ensure ORACLE_BIN is in $PATH

```

(continues on next page)

(continued from previous page)

```
which sqlplus

sqlplus /nolog <<EOF
CONN {oracleUser.user}/{oracleUser.password}@{oracleUser.sid}
SELECT * FROM DUAL
EOF
```

How To Create Dynamic SQL Script from Bash

```
sqlplus splex/$Splex_PW > $Action_File << EOF
set HEADING OFF
set ECHO OFF
set FEEDBACK OFF
set pages 0
set lines 200
select 'alter trigger '||owner||'.'||trigger_name||' $action; from dba_triggers where␣
↪trigger_name like '%SPX%';
select 'alter table '||table_owner||'.'||table_name||' modify t_updated_by_spx date␣
↪default sysdate;' from dba_triggers where trigger_name like '%SPX%' ;
@Action_File
exit
EOF

rm $Action_File
```

How To Use SQL to Create a File that Bash can Query

```
#Query the host_details for the type of host
sqlplus <user>/<userpassword>@<instance> > $tmpFile << EOF
SET ECHO OFF
SET HEADING OFF
SET PAGES 100
SET FEEDBACK OFF
SELECT host_name, host_type
  FROM host_details
  WHERE host_name '$dbname'
  AND host_type IN (1, 4)
EOF

# Get the host type from the file
hostType=$(grep $dbName $tmpFile | awk '{ print $2}' )
```

Mako Template Code Snippets

How To Escape a Mako Variable

To stop Attune from trying to substitute a variable, escape it by enclosing it inside Mako raw text tags. In this example, line 2 will be printed exactly as it is written.

```
<%text>
user=cyrus argv=/usr/lib/cyrus-imapd/deliver -e -r ${sender} -m ${extension} ${user}
</%text>
```

How To Render a Node List value IP addresses

This code snippet renders the IP addresses of nodes in a node list on a single line separated by spaces.

The makoParameter name is appServers.

```
${appServers.serverIps.replace(",", " ")}
```

5.3.8 How to configure GIT remotes with FTPS

This document explains how to configure an Redhat 8 Attune node to use FTP with Explicit for version control. This method is a way of using the FTP protocol but in a secure way. When FTP makes a TCP connection, it is encrypted end to end so the data can not be read in transit.

Install

Install openssl

Check to see if **OpenSSL** is installed on the Attune host:

```
dnf list installed openssl
```

If it is not installed, install it with:

```
dnf install openssl
```

Install GIT

Git only needs to be installed if it is not already, to check if git is installed, run: `dnf list installed git`

To install git, run:

```
dnf install git
```

Install git ftp

GIT_FTP is required for the FTP Transfer. The project can be found here: https://github.com/git-ftp/git-ftp

To install git_ftp run:

```
# as the attune user
curl https://raw.githubusercontent.com/git-ftp/git-ftp/master/git-ftp > $HOME/bin/git-ftp
chmod 755 $HOME/bin/git-ftp
```

This downloads the script and makes it executable in the Attune users binary directory.

Install curl-devel package

The curl-devel package includes the header files and libraries necessary to develop applications that handle HTTP or FTP transactions using the cURL library.

To install the package, run:

```
dnf install -y curl-devel
```

Install openssl-devel package

To install the package, run:

```
#as root
dnf install -y openssl-devel
```

Build curl v8

git_ftp utilised curl for the ftp transfer. curl Version 8 or greater needs to be installed but this is no always available in RHEL8 repositories. It is important that this version of curl is installed in the users `bin` directory so that replacing the system version does not cause anything to break.

Install **Dev Tools**:

```
sudo yum groupinstall -y "development tools"
```

Download **curl version 8**:

```
curl --output bin/curl-8.5.0.tar.gz https://curl.se/download/curl-8.5.0.tar.gz
```

Extract the file:

```
tar -xvf curl-8.5.0.tar.gz
```

Change into the directory:

```
cd curl-8.5.0
```

Configure **curl**

```
./configure --with-openssl --prefix=$HOME
```

Build **curl**

```
make
```

Warning: Ensure that the install path is `/home/attune/bin`

Install **curl**

```
make install
```

Add the Attune users bin directory to the **PATH**.

```
cat <<'EOF' >> ~/.bashrc

# Add Path for GIT FTP
export PATH="$HOME/bin:$PATH"

EOF
```

Source the bashrc file

```
source ~/.bashrc
```

Check that the version is `v8`

```
curl --version
```

Note: If the version is not v8, you need to check that `/home/attune/bin` is in the `$PATH` variable.

Configure the Repository

Configure the git ftp repository settings

In your project repository, run:

```
git config git-ftp.user "<username>"
git config git-ftp.url "<URL>"
git config git-ftp.password "<password>"
```

Warning: It is not secure to store the users password in plain text. git_ftp can use the -P flag to prompt the users password.

Note: The url must specify explicit TLS like `ftpes://<hostname>:<port><path_to_file>.git` The path is the relative path from the FTP session root.

Run GIT INIT

Running git init in an existing repository is safe. It will not overwrite things that are already there. The primary reason for rerunning git init is to pick up newly added template..

For example:

```
git ftp init -P --disable-epsv -v --insecure "ftpes://10.211.55.2:21/trace-
↪configuration-script.git"
```

- -P asks the user to manually enter the password
- –disable-epsv Disables Extended Passive Mode
- – insecure Does not check the certificate

How to use Git FTP

To push a file

In your repository, run:

```
git ftp push -P --disable-epsv -v --insecure
```

Git FTP Reference

For a list of git-ftp reference, see: *git_ftp reference guide*

Troubleshooting

For troubleshooting instructions, see *How to Troubleshoot Git FTP*

Appendix

Prerequisites

The Prerequisites for FTPES are:

- GIT installed.
- GIT_FTP installed.
- CURL version 8
- OpenSSL is installed
- Curl Devel

5.4 Reference guides

The following reference guides are the technical descriptions of how to operate Attune.

5.4.1 Architecture and Deployment

This page provides a high level overview of Attune's software architecture.

Attune is available as a desktop application or deployed as a Server for the web application.

Attune Desktop Application

The Attune App that can be downloaded, installed and run in minutes. The Attune App only runs jobs while the app is open. The Attune application is supported on the follow operating systems:

- MacOS
- Windows

App Resource Requirements:

Minimum: 1 CPU, 512MB RAM, 500MB

Recommended: 2 CPUs, 2GB RAM, 1GB

Download Attune App

Attune Server and Web Application

The Attune Server is a headless, standalone server running in a cloud or server environment, using a Web App as the admin interface. The following web browsers are supported:

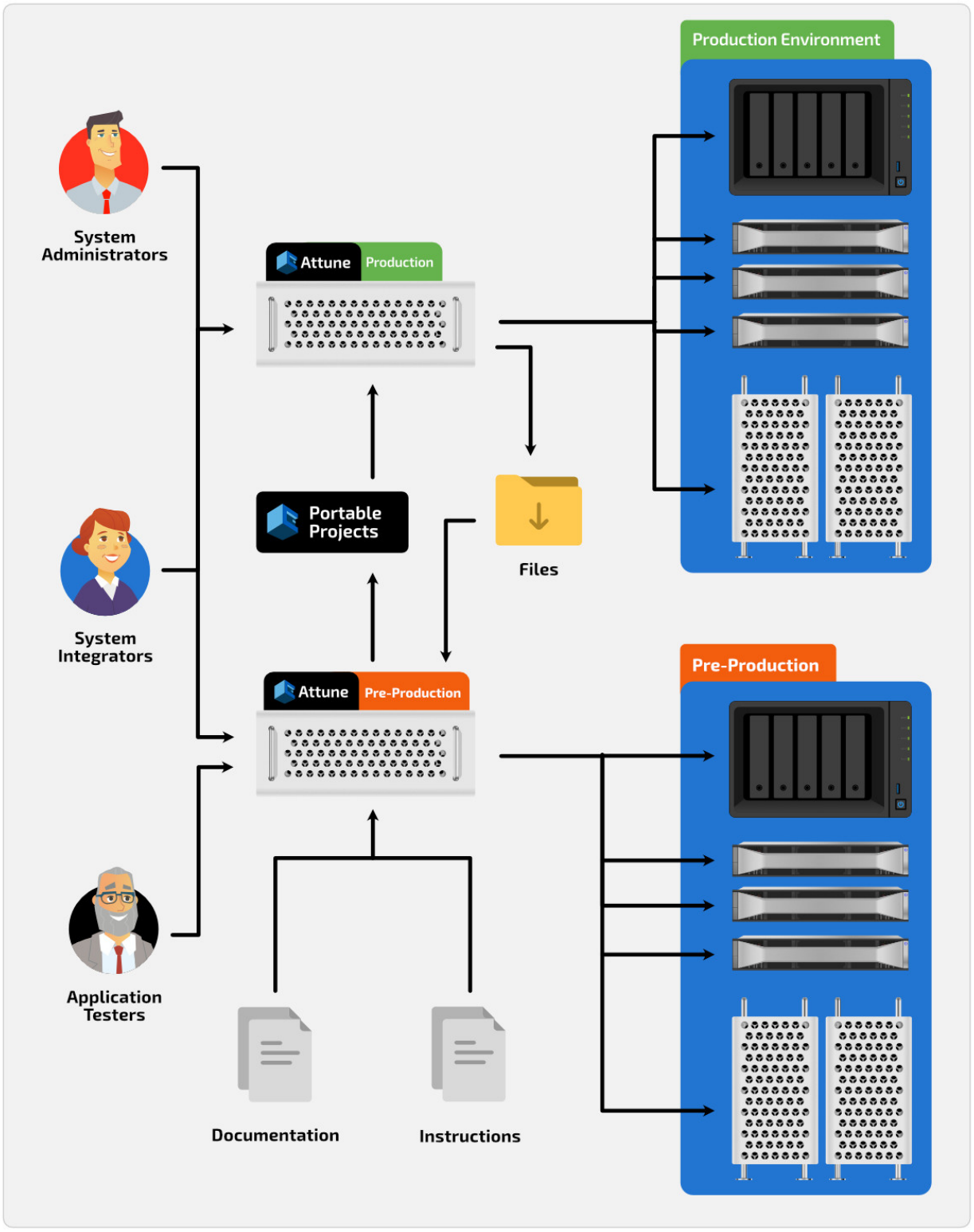
- Brave
- Chrome
- Mozilla Firefox
- Microsoft Edge

Attune Server is deployed as a VM Template and is supported on the following operating systems:

- RHEL 7

- RHEL 8
- CentOS 7
- CentOS 8
- Oracle Linux 7
- Oracle Linux 8

The following diagram shows two instances of Attune server deployed in a Pre-Production and Production environments.



App Resource Requirements:

Minimum: 1 CPU, 512MB RAM, 6GB

Recommended: 8 CPUs, 8GB RAM, 500GB

Learn more, book an online demonstration!

5.4.2 Design Workspace

Your Automated and Orchestrated IT processes are developed and modified in individual Projects. The Design Workspace contains your Projects.

Projects

Projects are self contained and sharable. Your Projects contain your:

- Parameters,
- Files,
- Steps &
- Blueprints.

You can have many Projects and Projects can contain many Parameters, Files, Steps and Blueprints.

Projects are version controlled with GIT integration. You can clone, push and pull Projects with repositories such as GitHub

Parameters

Parameters are used in Blueprints and substituted with Values when a job is run. Values are mapped against Parameters in the Plan. This provides maximum Blueprint re-usabiliity.

All Parameters have the same basic attributes regardless of their type.

A project as a minimum requires a Node type and Credential type Parameters to create a Step in a Blueprint. Parameters can be created in the Step.

To learn how to reference Parameters in a Script read: Referencing Parameters in Scripts

Files

Your Files can be stored centrally in Attune. Files are divided into two categories:

- Version Controlled,
- Large Files.

Version Controlled Files

Version Controlled Files are maintained in the GIT repository. These are typically small text files.

Dynamically create your **Configuration Files** when your Job is run. Attune has a template engine that allows you to deploy files which contain markup using Mako, an embedded Python language. The template engine processes the files in a configuration archive and renders the final outputs.

More information about Mako templates, and the syntax used for the configuration archives files can be found at http://www.makotemplates.org

Large Files

Large Files are not stored in the GIT repository. These can be large zipped directories, installation file, ISO's, etc.

Steps & Blueprints

Blueprints orchestrate the sequence and paralleling of Steps. Steps automate individual scripted actions. Blueprints from other Projects can be linked into a Blueprint.

5.4.3 Plan Workspace

Your Jobs are planned to run Blueprints with substituted Values controlled via the Self-Service Portal, Jobs Workspace or Scheduler. The Jobs are created in the Plan Workspace.

The **Plan Workspace** contains your:

- Values,
- Plans &
- Schedules.

Values

Values are substituted into Parameters when a Jobs is run. Values have different attributes based on their type.

To learn the Values type attributes read: *Referencing Parameters in Scripts*

Plans

Plans allow you to assign Values to the Parameters that appear in a Blueprint. You can create many Plans for a single Blueprints with different Values assigned.

Schedules

Schedules can be configured to run many Plans either sequentially or in parallel. The Attune service must be running for the Scheduler to run.

5.4.4 Run Workspace

Control your Jobs, debug your Steps and review your Historical Logs in the Run Workspace.

The **Run Workspace** contains your:

- Jobs &
- Historical Logs.

Jobs

Jobs allow you to start, pause or abort individual Jobs. Individual Steps can be inspected and modified. Jobs can be run from any Step in the procedure allowing you to skip sections or repeat a Step.

If a Step is modified it will be updated in the Blueprint. If the Blueprint structure of Steps is modified the Job will need to be recreated to populate the new, deleted or reorganised Steps.

Recreating the Job will archive the current log for each Step.

Historical Logs

Historical Logs are captured after each Scheduled Job, Self-Server Portal run job and recreated Job.

The Historical Logs can be viewed in Attune or exported in either PDF or HTML formats.

5.4.5 Referencing Parameters in Scripts

Blueprints are composed of **Steps**, and each **Step** has a unique name and a number of other attributes, including a **Script**, where the actual commands for executing the **Step** are written. In these commands, curly braces { } are used to delimit **Values** which are referenced by **Parameters**.

In order to be used in a **Script** the **Parameters** names are stripped of any separating spaces between words and converted to camelcase - with only the first letter of the second, and any subsequent words, capitalised.

So if a **Parameter** is named - *This is my Name*, it would be converted to: `thisIsMyName`.

Values are referenced in scripts as: `{placeHolderName.placeValueAttribute}`.

So for a **Parameter** with the name *This is my Name* of type **Server**, we can refer to the IP address in a script as: `{thisIsMyName.ip}`.

The expressions contained in the curly braces are evaluated prior to **Job** being **Run**, and the curly brace expressions are replaced by actual **Values** . In the example above `{thisIsMyName.ip}` would be substituted with the IP attribute of the **Value** that was assigned to the **Parameter** - *THIS IS my name* in the Plan.

This substitution of **Values** is performed by Attune, so using the curly Brace {} notation to reference attribute **Values** works with all **Script** regardless of the type of **Script**.

Value Attributes

This section details the **Value** attributes for each **Parameter** type.

OS Credential

Attributes of *OS Credential* **Parameter** type can be referenced as follows:

Value Attribute	Attribute Name	Script Reference
Name	name	`{myPlaceHolder.name}`
User	user	`{myPlaceHolder.user}`
Password	password	`{myPlaceHolder.password}`
Comment	comment	`{myPlaceHolder.comment}`

SQL Credential

Attributes of *SQL credential* **Parameter** type can be referenced as follows:

Value Attribute	Attribute Name	Script Reference
Name	name	`{myPlaceHolder.name}`
User	user	`{myPlaceHolder.user}`
SID	sid	`{myPlaceHolder.sid}`
Password	password	`{myPlaceHolder.password}`
As Sysdba	asSysDba	`{myPlaceHolder.asSysDba}`
Comment	comment	`{myPlaceHolder.comment}`

Node

Attributes of *Node* **Parameter** type are referenced as follows:

Value Attribute	Attribute Name	Script Reference
Name	name	`{myPlaceHolder.name}`
IP Address	ip	`{myPlaceHolder.ip}`
Hostname	hostname	`{myPlaceHolder.hostname}`
Domain Name	domain	`{myPlaceHolder.domain}`
Comment	comment	`{myPlaceHolder.comment}`

Node List

Node List is a comma separated list of node **Parameters** (CSV).

Attributes of *Node List* **Parameter** type are referenced as follows:

Value Attribute	Attribute Name	Script Reference
Name	name	`{myPlaceHolder.name}`
Server	serverNames	`{myPlaceHolder.serverNames}` as CSV
Server	serverIps	`{myPlaceHolder.serverIps}` as CSV
Server	serverHostnames	`{myPlaceHolder.serverHostnames}` as CSV
Server	serverTypes	`{myPlaceHolder.serverTypes}` as CSV **The types are:** • 1 = Linux • 2 = Windows
Server	serverDomains	`{myPlaceHolder.serverDomains}` as CSV
Server	serverFqns	`{myPlaceHolder.serverFqns}` as CSV
Comment	comment	`{myPlaceHolder.comment}`

IPv4 Subnet

Attributes of *IPv4 Subnet* **Parameter** type are referenced as follows:

Value Attribute	Attribute Name	Script Reference
Name	name	`{myPlaceHolder.name}`
Subnet	subnet	`{myPlaceHolder.subnet}`
Netmask	netmask	`{myPlaceHolder.netmask}`
Gateway IP	gateway	`{myPlaceHolder.gateway}`
DNS IP 1	dns1	`{myPlaceHolder.dns1}`
DNS IP 2	dns2	`{myPlaceHolder.dns2}`
Comment	comment	`{myPlaceHolder.comment}`

Text

Attributes of *Text* **Parameter** type are referenced as follows:

Value Attribute	Attribute Name	Script Reference
Name	name	`{myPlaceHolder.name}`
Value	value	`{myPlaceHolder}` Please note the text value is referenced without the attribute.
Comment	comment	`{myPlaceHolder.comment}`

5.4.6 git_ftp reference guide

Git-ftp is an FTP client using Git to determine which local files to upload or which files to delete on the remote host. For more information, see the git_ftp manual: https://git-ftp.github.io/

Syntax

```
git ftp <action>[<options>][<url>]
```

Actions

Some of the common options are listed here:

- `init` : Uploads all git-tracked non-ignored files to the remote server and creates the .git-ftp.log file containing the SHA1 of the latest commit.
- `push` : Uploads files that have changed and deletes files that have been deleted since the last upload.
- `log` : Downloads last uploaded SHA1 from log and hooks *git log*.
- `help` : Shows the help screen.

Options

- `-u [username]`, `--user [username]` : FTP login name. If no argument is given, local user will be taken.
- `-p [password]`, `--passwd [password]` : FTP password. See -P for interactive password prompt. (note)
- `-P`, `--ask-passwd` : Ask for FTP password interactively.
- `-a`, `--all` : Uploads all files of current Git checkout.
- `-c`, `--commit` : Sets SHA1 hash of last deployed commit by option.
- `-A`, `--active` : Uses FTP active mode. This works only if you have either no firewall and a direct connection to the server or an FTP aware firewall. If you don't know what it means, you probably won't need it.
- `-b [branch]`, `--branch [branch]` : Push a specific branch
- `-h`, `--help` : Prints some usage information.
- `-v`, `--verbose` : Be verbose.
- `-vv` : Be as verbose as possible. Useful for debug information.
- `--insecure` : Don't verify server's certificate.
- `--cacert <file>` : Use as CA certificate store. Useful when a server has a self-signed certificate.
- `--disable-epsv` : Tell curl to disable the use of the EPSV command when doing passive FTP transfers. Curl will normally always first attempt to use EPSV before PASV, but with this option, it will not try using EPSV.
- `--auto-init` : Automatically run init action when running push action

URL

```
protocol://host.domain.tld:port/path
```

For example

```
ftp://host.example.com:2121/mypath
```

Supported Protocols are:

- `ftp://...` : FTP (default if no protocol is set)
- `sftp://...` : SFTP
- `ftps://...` : FTPS
- `ftpes://...` : FTP over explicit SSL (FTPES) protocol

5.5 FAQ

Here you'll find answer to many common questions.

5.5.1 Attune Frequently Asked Questions

Why can't I import or export large blueprints?

Several reasons may cause large blueprint files to fail to import or export from the Attune Web App.

1. Check that the `/tmp` and `/home` directories have enough storage with `df -h`
2. Check that the browser is not enforcing a download size limit by trying another browser.
3. Ensure that it is not a firewall preventing the download by connecting to the Attune server from the same subnet.

5.5.2 Troubleshooting

This page contains advice about errors and problems commonly encountered.

Attune Server and Web Application

Logs

Inspect the Attune log:

```
less /home/attune/attune.home/attune.log`
```

Inspect the system logs:

```
sudo journalctl -f
```

System Disk Space Usage

List filesystems

```
df -h
```

List large files in the */home/* directory:

```
find /home/ -type f -size +100M -exec du -h {} +
```

Restart the Attune Service

Restart the attune service and tail the log.

```
sudo systemctl restart attune
tail -F ~/attune/attune.home/attune.log
```

Monitor the log.

Attune Application on Windows

Logs

Inspect the Attune log:

```
C:\Users\{account_name}\AppData\Local\com.servertribe.attune\attune.log
```

Attune Application on macOS

Logs

Inspect the Attune log:

```
/Users/{account_name}/Library/Application Support/com.servertribe.attune/attune.log
```

Step Errors

HTTPSConnectionPool

The following error may occur if the network latency is high:

```
HTTPSConnectionPool(host='{targetNodeIp}', port=5986): Max retries exceeded with url:␣
↪/wsman (Caused by ConnectTimeoutError(<urllib3.connection.HTTPSConnection object at␣
↪0x1162b8e20>, 'Connection to {targetNodeIp} timed out. (connect timeout=5)'))
```

The error message you're seeing is related to a network connection issue. Attune is attempting to establish an HTTPS connection using a connection pool. Connection pools are used to manage multiple network connections efficiently.

Try re-running the step after 1 minute.

5.5.3 How to Troubleshoot Git FTP

Verbose Flags

When troubleshooting the git-ftp command, the first step is to either add the `-v` or `-vv` flag to the command. This will cause the commands output to be more verbose.

- `-v` Increase the output to be more verbose.
- `-vv` Is more verbose that the `-v` flag.

Firewall Rules

Git FTP will attempt to connect the port you specify, (FTP is port 21 by default) to the FTP server.

- **Active FTP**: The client initiates a command connection from a random unprivileged port (greater than 1023) to the FTP server's command port (21). Then, the client starts listening on its original port + 1 and sends this port number to the FTP server, which initiates a data connection from its port 20 to this port. So, the client's firewall needs to allow outbound traffic on port 21 and inbound traffic on random unprivileged ports for the data connection.
- **Passive FTP**: The client initiates a command connection from a random unprivileged port to the FTP server's command port (21). When the client needs data, it sends a PASV command over the command channel. The server replies with its chosen unprivileged port (greater than 1023), and the client initiates a data connection to this port. So, the client's firewall needs to allow outbound traffic on port 21 and also on random unprivileged ports for the data connection.

TLS Handshake

A successful TLS handshake between the client and server is essential for the FTPES protocol. During the handshake, the client and server acknowlege each other, exchange certificates and keys, and agree on a cipher.

If there is a reason that the handshake is not successful, the connection will close. The most common problem is when the client can not verify the certificate. When the `--insecure` flag is set for the git_ftp command, it will not attempt to verify the certificate.

Below is an example of a successful TLS handshake.

```
* TLSv1.3 (IN), TLS handshake, Server hello (2):
{ [88 bytes data]
* TLSv1.3 (OUT), TLS change cipher, Change cipher spec (1):
} [1 bytes data]
* TLSv1.3 (OUT), TLS handshake, Client hello (1):
} [588 bytes data]
* TLSv1.3 (IN), TLS handshake, Server hello (2):
{ [161 bytes data]
* TLSv1.3 (IN), TLS handshake, Encrypted Extensions (8):
{ [6 bytes data]
* TLSv1.3 (IN), TLS handshake, Finished (20):
{ [52 bytes data]
* TLSv1.3 (OUT), TLS handshake, Finished (20):
} [52 bytes data]
* SSL connection using TLSv1.3 / TLS_AES_256_GCM_SHA384 / [blank] / UNDEF
* Server certificate:
*  subject: C=NZ; ST=CAN; L=CHC; O=Synerty; OU=DevOps; CN=ftp.local
*  start date: Dec 14 04:35:29 2023 GMT
*  expire date: Dec  9 04:35:29 2043 GMT
*  issuer: C=NZ; ST=CAN; L=CHC; O=Synerty; OU=DevOps; CN=ftp.local
*  SSL certificate verify result: self signed certificate (18), continuing anyway.
*   Certificate level 0: Public key type ? (2048/112 Bits/secBits), signed using␣
↪sha256WithRSAEncryption
{ [5 bytes data]
* TLSv1.3 (IN), TLS handshake, Newsession Ticket (4):
```

To check to see if the servers certificate can be verified, use:

```
openssl s_client -connect hostname:port
```

Client vs Server Certificates

In FTP active mode, the server will try and establish a TCP/TLS connection to the client for the data connection, in this case, the server will expect a certificate from the client.

In FTP passive mode, the client is responsible for making the command and data connections and the server will not request a clients certificate.

This behaviour can be seen when running the git_ftp command in very verbose mode.

5.6 Glossary

Blueprints A Blueprint is the orchestration of Steps.

Jobs Jobs are created in the Plan Portal, pairing Values with the Parameters that are configured in the Blueprint. Running a job will perform the tasks configured in the Blueprint using the Values configured in the Parameters.

Parameters Steps contains Parameters for credentials, node details, text, and more.

Steps A Step contains the connection details and action(s) to be performed.

Values Values are substituted into a Job Parameters when a Job is run.

5.7 Release Notes

5.7.1 Release notes - Attune v23.7.x

v23.7.3

Bug

AT-1332 Push Compiled Files no longer fails when Binary Files in the Files Archive

AT-1402 Steps in step screen lists in macOS App display correctly

v23.7.2

Bug

AT-1412 Windows App successfully clones projects

v23.7.1

Bug

AT-1411 Windows App no longer fails to start with NoneType has no attribute externalUuid4

5.7.2 Release notes - Attune v23.0.x

v23.0.4

Bug

AT-1383 Optimised Project Website Page Headings Structure

AT-1386 Push Compiled Files Screen Shows Correct Files in Archive

AT-1390 Made Project Website Structure SEO-friendly and Readable

AT-1394 Windows archive clobber/overwrites existing files

AT-1399 Upgrading to plan tree successful if Attune has no plans

AT-1405 Date times in Attune run screen logs are correct timezone

AT-1406 Renaming step updates step list in "link step" drop down

Task

AT-1398 Added responsive navigation bar and footer

AT-1407 Upgraded to "@synerty/vortexjs@3.3.11" for websocket improvements

v23.0.3

Bug

AT-1386 Push Compiled Files Screen Now Shows Correct Files in Archive

v23.0.2

Bug

AT-1377 Edit Step Name and Step Comment causes Project error

AT-1378 Adding a text parameter that doesn't exist to a text script throws an exception

AT-1379 Changing a parameter dropdown throws exception on run

AT-1380 Committing project fails with No attribute sid

AT-1382 Project Website Setup Meta Tags: Formatting and Arrangements

AT-1384 Remove project from parameter now also removes duplicates

v23.0.1

Improvement

AT-566 Handling Step Failures

Bug

AT-1374 Job Run Steps don't show warning icon in tree

v23.0.0

Bug

AT-1268 LDAP fails to find groups for users with 'and' in their name

AT-1269 Rename parameter does not cascade to deploy compiled file mako step

AT-1270 Copy-Step fails if the step exists in the destination

Improvement

AT-1255 Step-details replaced tabs with scrollable sections

AT-1256 Replaced all Add buttons in filter boxes with Add links at bottom of list

AT-1257 Moved action buttons at top of step tree into tree

AT-1265 Add Create Step Under Menu Option

AT-1266 Added UUID4 field for all storage items

AT-1277 Rearranged Project so git remote is with git commit

AT-1300 Uploaded EL7 releases based on branch name

AT-1310 Removed second level navigation bar

AT-1311 Decreased vertical gap between step details forms

AT-1347 Captured Attune startup exception and log them to attune log

Task

AT-731 Improve user interface to remember selected items when moving between tabs

AT-865 Enable jobs to run with Project Link steps in them

AT-1217 Implemented new plant tree UI for scrolling panels

AT-1242 Created Step De-duplicate Feature

AT-1258 Added support for mapping to any project + parameter key tree node mapping

AT-1291 Implemented Plan and Node Tree Delete

AT-1294 Updated to @synerty/vortexjs@3.3.6 vortexpy==3.4.6

AT-1338 Added logging for __hasErrors for steps

5.7.3 Release notes - Attune v5.1.x

v5.1.7

Bug

AT-1268 LDAP fails to find groups for users with "and" in their name.

AT-1269 Rename parameter does not cascade to deploy compiled file mako step

AT-1270 Copy-Step fails if the step exists in the destination

v5.1.6

Bug

AT-1226 Remove Jobs list horizontal scroll

AT-1227 Create Push Files Step, Archive dropdown too small

AT-1228 Create "Execute Linux Script" step dropdown too small

AT-1232 Schedules fail to generate HTML for emails

AT-1233 Copy Step and Move Step doesn't work properly

AT-1234 Schedules think they are failed after first job runs successfully

v5.1.4

Bug

AT-1141 No project is defaulted to when opening a fresh Attune instance for the first time

AT-1202 Showing archive list fails on larger Attune instances

AT-1204 Attune does not deploy unticked files in Push Compiled Files Step

AT-1205 Attune Parameters are no longer editable if exist in Blueprint but not parameter list

Tasks

AT-1190 Revert Project Export button http call changes

v5.1.3

Bug

AT-1187 Loading Files Dropdown in Steps Component performance issue

v5.1.2

Bug

AT-1184 Committed changes are lost on Attune restart

v5.1.1

Bug

AT-1108 Migration scripts do not set the PostgreSQL sequences correctly

AT-1148 Double click on create and export actions

AT-1149 Required Values are unknown before the job is created

AT-1150 Job created without Blueprint being selected

5.7.4 Release notes - Attune v4.3.x

Attune v4.3.14

Bug

AT-643 Attune fails to connect CIFS when SMB2 signature is required

AT-715 Attune with HTTPS enabled doesn't enable the SSL Websocket, and nothing in the browser works.

AT-823 Replace Twisted sendmail with smtplib

AT-824 Attune scheduler does not detect Schedule failures correctly

AT-838 Logging in using LDAP user produces a list indices must be integers or slices, not str

Attune v4.3.13

Bug

AT-715 Attune with HTTPS enabled doesn't enable the SSL Websocket, and nothing in the browser works.

Attune v4.3.12

Bug

AT-549 Show as Text not working for Blueprints with Windows steps

Attune v4.3.11

Bug

AT-544 TypeError in ScheduleManager

AT-545 Markdown to HTML conversion no displaying tables

Attune v4.3.10

New Feature

AT-473 Implement Blueprint Documentation Export

Attune v4.3.9

New Feature

AT-473 Implement Blueprint Documentation Export

Attune v4.3.8

New Feature

AT-473 Implement Blueprint Documentation Export

Attune v4.3.7

New Feature

AT-473 Implement Blueprint Documentation Export

Attune v4.3.6

New Feature

AT-473 Implement Blueprint Documentation Export

Attune v4.3.5

New Feature

AT-473 Implement Blueprint Documentation Export

Attune v4.3.4

Improvement

AT-484 Create full blueprint to zip file export for DIFFs

New Feature

AT-473 Implement Blueprint Documentation Export

Attune v4.3.3

Bug

AT-474 SUDO steps do not work with Mac SSH server

Attune v4.3.2

Bug

AT-466 Values in plan are not sorted alphabetically

AT-474 SUDO steps do not work with Mac SSH server

Attune v4.3.1

Bug

AT-467 Wrong tooltip displayed when selecting Add in Windows Node Values

AT-471 Schedule jobs do not appear in new jobs UI

AT-472 Creating a new blueprint links it under the currently selected group

Task

AT-469 Pull files step shows "Deploy Path"

Attune v4.3.0

Improvement

AT-456 Add SSH port support to Attune

Bug

AT-455 RootResource.py", line 55 - No module named 'app'

AT-457 New windows servers don't have default win rm type selected RM_V2_HTTPS_NTLM

AT-458 Tooltips pop up and get in the way of buttons

AT-460 StepShellPrompted fails with sudo: no password was provided

AT-461 Users are not prompted to enter the domain field

5.7.5 Release notes - Attune v4.2.x

v4.2.5

Bug

AT-417 v4.2.0 pre-release testing bug fixes

AT-453 Opening files in Files screen sometimes fails

Task

AT-391 Code Sign Windows App

v4.2.4

Bug

AT-417 v4.2.0 pre-release testing bug fixes

AT-446 "npm run electron-prod" builds cause the job result log to not show time, severity or colour

AT-450 Attune procedures fail to export on windows

v4.2.3

Bug

AT-416 Angular JS styling to match new Angular 11 screens

AT-417 v4.2.0 pre-release testing bug fixes

AT-422 Opening a Dynamic File causes a Save Successful message

AT-435 Attune fails to open app window macOS sometimes

AT-440 App, Deploy Dynamic Files, assert exists == os.path.exists

AT-441 Uploading file on windows fails "[Errno 13] Permission denied"

AT-442 Directory object no longer deletes files

AT-443 Win: MakoPaths and Archive Editing fails

Task

AT-439 BUG: Job Details / Mapping screens continually reload data

v4.2.2

Bug

AT-429 v4.2.x Whole page scrolls for all panels, in Jobs, Inputs, etc

AT-430 'PlanUnsetPasswordTuple' object has no attribute 'id'

AT-436 Plan Mapping scrolls the whole page, and not a lot is shown on the page

AT-437 v4.2, Attune App scrolling is broken on inputs and job pages

Task

AT-438 Make config.cfg file redundant for app

v4.2.1

Bug

AT-426 Cannot create new Job instance

v4.2.0

Improvement

AT-407 Rename parameters, values, blueprints and files

AT-414 Upgrade Dashboard to Vortex + Ant.Design

Bug

AT-416 Angular JS styling to match new Angular 11 screens

AT-417 v4.2.0 pre-release testing bug fixes

AT-418 Scheduler UI has no data in v4.{0,1}.x Attune

AT-419 4.x job screen doesn't download job instructions

AT-420 v4 Job edit, showAll is not working

AT-421 Running job from dashboard fails to display job failure alerts

AT-424 Disable Plan export to DocX for App Builds

Task

AT-412 Add serverId to password encryption key

AT-413 Schedules' Plans Deleted or Not Displaying

5.7.6 Release notes - Attune v4.1.x

v4.1.7

Bug

AT-366 Server Group Values aren't displaying correctly

AT-369 Unable to Abort or Add New Job after List Index Out of Range Error

v4.1.6

Improvement

AT-297 Duplicate Step

AT-395 Implement pure python file deployments to Linux

AT-401 Implement Pull Archive support

Bug

AT-365 Steps Page UI Not Working

AT-366 Server Group Values aren't displaying correctly

AT-369 Unable to Abort or Add New Job after List Index Out of Range Error

AT-373 New Jobs UI Doesn't Show Error When Plan has Unmapped Variables

AT-377 Deploy RPM Archive Fails

AT-378 'Steps' page on Synerty's Orion Environment Attune Server is not displaying correctly

AT-396 Fix SQLite errors for procedure imports

AT-397 container-fluid doesn't work on plan and step screens

AT-400 Trying to run a job raises List Index Out of Range after steps have been moved

v4.1.0

Improvement

AT-251 Make LDAP config more flexible by taking the full LDAP stringa

Bug

AT-364 V4 Drag and Drop upgrades do not work

Task

AT-367 WIN: Setup Development Environment

AT-375 WIN: Package app to exe

AT-380 DevOps: Create Windows Gitlab Runner

AT-381 DevOps: Setup packaging to create Attune tags.

AT-382 DevOps: Create windows package pipelines

AT-383 Release: Create attune updates and analytics server

5.7.7 Release notes - Attune v4.0.x

v4.0.0

Improvement

AT-268 Upgrade Python to 3.6.8

AT-317 Move Attune LargeFileRequest to txHttpUtil

AT-319 Adapt Attune to use the latest VortexJS/VortexPY from Peek

AT-324 Implement PLPython3 calls

AT-325 Write all job updates to the DB and remove memory caching (Log, Progress, State)

AT-326 Change all loads of job status to load from the DB with PLPython3

AT-332 Create new Angular10 jobs page

AT-358 Style Login Screen

Bug

AT-253 gitpython 3.x.x doesn't run in Python 2.7

AT-316 Attune 'Deploy Archive' steps use blowfish cipher for large files by default.

AT-363 Unable to export procedures as text or .atp files

Task

AT-351 Reformat with Black, Prettier and fix TSLink

5.7.8 Release notes - Attune v3.1.x

v3.1.1

Improvement

AT-349 Package Attune with pysmb==1.1.27

v3.1.0

Task

AT-345 Update Attune navbar to show ServerTribe, update Fonts

5.7.9 Release notes - Attune v3.0.x

v3.0.29

Bug

AT-342 LDAP doesn't work for simple / non role based use cases

v3.0.28

Bug

AT-337 Job List and Dashboard Lists cause high Attune server load

v3.0.27

Bug

AT-304 WinRM steps fail with "Preparing modules for first use"

v3.0.26

Bug

AT-304 WinRM steps fail with "Preparing modules for first use"

AT-307 Fix kerberos dependency issues when deploying Attune offline

v3.0.25

Bug

AT-283 TCP Ping steps take too long to timeout, and end up not noticing servers being down.

AT-304 WinRM steps fail with "Preparing modules for first use"

v3.0.22

Improvement

AT-306 Add "Download Procedure as Text" support

Bug

AT-307 Fix kerberos dependency issues when deploying Attune offline

v3.0.21

Improvement

AT-306 Add "Download Procedure as Text" support

Bug

AT-259 SECURITY Plan Manual export DOESN'T mask passwords - It needs to do this.

AT-303 Emails from scheduled tasks fail in v3.0.20

AT-304 WinRM steps fail with "Preparing modules for first use"

v3.0.20

Task

AT-299 Add support for Windows Kerberos authentication

AT-300 Highlight changes when importing procedures

v3.0.19

Improvement

AT-290 Archives: Make Table of Contents list update on Lock/Unlock

v3.0.18

Improvement

AT-320 IPv6 Subnet - Add variable for subnet bits, not 255.255.255.192

Bug

AT-275 Creating job creates duplicate steps when steps are present twice in procedure

AT-279 Plans some times show duplicate variables

AT-280 Attune doesn't print the last line of log output

AT-281 Attune doesn't handle private key decryption failures.

Task

AT-288 Change release build from pip download to wheel

AT-291 Add PowerCLI and Boot ISO steps to Attune install instructions

v3.0.17

Improvement

AT-255 Make deploy locked tar files use the OS ssh command instead of piping it through pythons SSH

AT-256 Improve performance of SQL queries to update the UI due to slow HyperV performance

AT-258 Implement support for SSL

New Feature

AT-249 Implement support for parallel steps

Bug

AT-236 Download plan as HTML links don't work

AT-247 "Product of Synerty" should be "Powered By Synerty"

AT-254 Fix compatibility issues for macOS

AT-260 Template config step Generic Server parameters don't show

AT-262 Scheduler no longer stops on a procedure that fails, it carries on

AT-263 Add more ForiegnKey ON_DELETE constraints

AT-264 UI Job undefined execute value bug fix

AT-265 Schedules fail to generate job log

AT-267 Schedule Repeating

AT-272 Deploying windows archives exceeds attunes file-max ulimit

AT-273 WinRM steps fail with error message XML "Preparing modules for first use."

AT-274 Plans should allow windows/linux server/cred values for the "generic" variables

v3.0.14

Bug

AT-243 Server Group value items don't have an order

v3.0.8

Improvement

AT-238 Add support for LDAP Auth against different AD DS folders

Bug

AT-239 Improve error message from database integrity errors

v3.0.2

Bug

AT-232 Error while importing procedure : null value in column “osCredId” violates not-null constraint

AT-234 Download results fail when there is a # in the name

Task

AT-235 Update DnD upgrade to v3.x.x

v3.0.1

Improvement

AT-228 Convert Attune update and packaging to use pip and systemctl for init

AT-229 Add log rotater to logging setup

Bug

AT-231 Values are not imported with the correct inherited class

v3.0.0

Improvement

AT-220 Add support for RHEL7 installs

AT-221 Update licensing metrics to schedules and target servers

AT-222 Add support for AD authentication, via LDAP

AT-223 Add support for multiple uers

AT-224 Move built in product docs to read-the-docs

Story

AT-230 Exporting plan to HTML silently fails when passwords are not set

Task

AT-225 Upgrade UI to angular 6

5.7.10 Release notes - Attune v2.0.x

v2.0.7

Bug

AT-219 Proc Import fails to convert Values from Generic

v2.0.6

Bug

AT-213 Scheduler - can't compare offset-naive and offset-aware datetimes

v2.0.5

Bug

AT-192 Procedure import existing variable matching is case sensitive

AT-196 Password can not be decrypted, it needs to be reset.

AT-197 "successfull" spelt with one l

AT-198 Job screen shows UTC times on job screen

AT-199 SSH step deletes askpass file before sudo reads it

AT-204 failure to import Attune v1.4 procedure into Attune v2+

AT-205 Support expired email is sent every 5 seconds.

Task

AT-59 Logging in as sysdba with no user/pass and with SID fails

v2.0.4

Bug

[AT-191](https://servertribe.atlassian.net/browse/AT-191) Procedure import/export has old placeholder / placevalue names

v2.0.3

Bug

AT-165 Some plans can't be deleted because BuildExecuteResult exist

AT-188 Can not download archive files that have # in the name

v2.0.2

Bug

AT-189 createExecuteBlocking - not persistent within this Session

AT-190 Attune v2.0 doesn't create fresh db correctly

v2.0.1

Bug

AT-175 'NoneType' object has no attribute 'compileTemplate'

AT-185 compileTemplateBlocking() takes exactly 3 arguments (2 given)

AT-186 Deploy templates can't access ServerList.servers, detached attribute

AT-187 Licensing fails with Settings.py unicode problem

v2.0.0

Improvement

AT-166 Rename Placeholders to "Variables"

AT-167 Rename Placevalues to "Values"

AT-168 Rename "Executions" to "Jobs"

AT-170 add "Select all / unselect all" buttons to procedure export

AT-177 Replace the executions and procedure tree

AT-178 Encrypt passwords in database

New Feature

AT-102 Encrypt stored passwords and ssh keys in DB

Bug

AT-153 Scheduler has issues around the month barrier

AT-162 CIFS File deployments fail on windows 10

AT-164 Attune never deletes archived exceptions

AT-169 Add "Mask response text" feature to interactive shell step

AT-179 download large procedures fails

5.7.11 Release notes - Attune v1.4.x

v1.4.3

Bug

AT-156 Deploy archive to windows fails (connection refused)

v1.4.2

Improvement

AT-158 The IPv4 subnet placevalue needs a default gateway

Bug

AT-50 'DeployCifsDirectory' object has no attribute 'kill'

AT-157 Plan HTML/Word exports are missing support for "Tcp Ping"

AT-159 Replace existing deploy template archive step doesn't update linked files

AT-160 Spelling mistake in execute tooltip

v1.4.1

Improvement

AT-149 Doc Update - Add IPv4 placevalue to attributes page

v1.4.0

Improvement

AT-123 Archives screen needs search/filter box at top left similar to placevalues, etc

AT-133 Procedures screen needs search box

AT-143 Show placeholder comments in plan

Bug

AT-122 Db session in deferToThread need to expunge all and close

AT-125 Shell Step, with Perl doesn't load environment

AT-127 Typo, "schell" in "There is a \r in the *schell* script, this will be removed"

AT-134 Deploying tar.bz2 generic archives fail

AT-136 SSH authentication fails on fresh RHEL 6.8 VM

AT-137 Deploying generic tar.gz unlocked archive fails.

AT-138 Archives list doesn't independently scroll

AT-139 Download locked archive fails with ng-route

AT-140 Create IPv4 Network placeholder / placevalue

AT-141 Show text placeholder name in placeholder screen

AT-142 SshCmdResponder is missing cursorForward method

AT-144 Deploying template archive fails when SSH keys are missing

AT-145 Implement HTML Plan Document Download

AT-146 Total progress fails to show when remainder has decimal place

Task

AT-106 Create Attune documentation for v1.4.0 +

5.7.12 Release notes - Attune v1.3.x

v1.3.4

Improvement

AT-119 Add support for referencing Server List from text

Bug

AT-118 Attune can't use place value attributes that are ints

AT-120 sort drop down list for place values in plan screen

v1.3.3

Bug

AT-114 sudo: unable to resolve host causes ServerID gen failure

AT-115 License spelling mistake in license entry box label

AT-116 RHEL5 requires full path to dmidecode for license

AT-117 License fails, "sudo: sorry, you must have a tty to run sudo"

v1.3.2

New Feature

AT-113 ServerID generation repeats last 4 chars twice.

Bug

AT-111 attune ignores sudo dmidecode if it fails

AT-112 Scheduler power button isn't set properly on page load

v1.3.1

New Feature

AT-108 Add licensing

AT-109 Add master scheduler enable/disable switch

AT-110 Add "show password" checkbox to placevalues

Improvement

AT-97 Format the seconds count for progress bars, 3m, 40s instead of 220s

AT-98 PG doesn't create indexes on foreign keys by default. Add these to the ORM model

AT-100 Add "with_ploymorphic = '*' to improve performance of loading joined table inheritance.

Bug

AT-96 Stopping an execution causes it to display green in the dashboard and the execute list

AT-103 Uploading files to Attune archive causes error message

AT-104 Showing Last 20/20 line only ever shows 20/20 on execute screen

Task

AT-99 Move auto-list-panel-height directive to RapUI

v1.3.0

Improvement

AT-23 Resize Plan and Group Step comment fields

AT-71 Convert remainder of Plan and Execute screens to use AngularJS

AT-72 Change Attune Web App from full html page reloads to dynamically loading content as required.

AT-73 Add support for archiving executions to disk

AT-75 On execute screen, highlight failed executions as red

AT-76 On execute screen, add total progress bar

AT-77 Add support for filtering out scheduled executions

AT-78 Add success/failure colours to scheduled task list

AT-79 Display the summation of the time taken by sub steps in the group step on the execute screen

AT-80 Add copyright and separate build number on Attune version

Story

AT-63 Create dashboard for running plans

Bug

AT-22 Old style handlers, such as the edit step popup, send data to all clients

AT-29 Show/Hide button missing from plan popup

AT-31 Executions need to ignore updates if its older than the last update.

AT-62 Setting keep executions to 0 causes Attune to only run the first procedure

AT-70 Some parts of Attune still refer to the software as SynBUILD

AT-74 Deploy archive does not auto bootstrap linux ssh keys

AT-81 StepWindowsCommandRunner throws exception when it fails to connect

AT-82 151204 Alembic won't upgrade due to python import dependencies

AT-83 Rename the system menu option to upgrade

AT-84 Improve performance of execution data base interaction

AT-85 Buffer and cache the result updates to the UI, including log lines

AT-86 Add build number to Attune build file name

5.7.13 Release notes - Attune v1.2.x

v1.2.4

Bug

AT-44 Bootstrap fail causes invalid ssh keys to be used

AT-45 Step shell prompted fails with expected string, got tuple

AT-58 First deploy template config step fails in fresh attune server

AT-65 Prompted response responds with twice to some inputs.

v1.2.3

Improvement

AT-32 Make procedure export contain date and server exported from

AT-42 Create deploy script, run cmd with script

AT-52 Move to use our PyWinRM Fork

AT-53 WinRM, Lack of useful error messages when script is broken

Bug

AT-35 Scheduler continued to next procedure when last procedure failed

AT-37 Running steps in thread causes issue when adding log lines

AT-40 Convert execute progress bar to simple green progress bar

AT-46 Deploy template archive errors on rsyncing to git

AT-49 Procedure import needs facility to upgrade schema on import

AT-51 Infinite loop if the directory can't be created when deploying CIFS archives

AT-54 Running really long powershell scripts exceeds windows command arg length

AT-56 Plan shows both drop down and text place values in chrome

v1.2.2

Improvement

AT-21 Set web page title based on server name in settings

Bug

AT-18 ShellPrompted step doesn't clean \r from script

Task

AT-19 StepGroup and new Sql, Shell fields missing from import "replace existing"

AT-20 Make Attune auto bootstrap SSH access

v1.2.1

Bug fixes and added interpreter option support for shell scripts.

Improvement

AT-14 Add interpreter option support for shell scripts

Bug

AT-11 Oracle step commits at the end, sometimes this is undesirable

AT-12 Importing a procedure with no content archives errors out

AT-13 Successful imports need to refresh the procedure screen

AT-15 Imported steps don't link to new archives

AT-16 Imported new archives were using existing archive files

AT-17 Pending sessions can cause drag and drop upload to lockup