

Sitecore Installation Framework Configuration Guide

Sitecore XP 9.0

Step by step guide to the Sitecore 9.0 configuration process with the Sitecore Install Framework



sitecore[®]
Own the experience[™]

Table of Contents

Chapter 1	Introduction	3
1.1	Getting Started	4
1.1.1	How to Use This Guide	4
Chapter 2	Install the Module	5
2.1	Install the Sitecore Installation Framework Module	6
2.1.1	Installation with Microsoft PowerShell®	6
2.1.2	Manual Installation.....	6
2.1.3	Validate the Installation	7
2.1.4	Import the Sitecore Installation Framework into a Powershell session	8
Chapter 3	Customize the Sitecore Installation Framework	9
3.1	Create and Customize Configurations	10
3.1.1	Tasks.....	10
3.1.2	Parameters	10
3.1.3	Config Functions	11
3.1.4	Variables	11
3.1.5	Modules	11
3.1.6	Settings	12
3.2	Create Tasks.....	13
3.2.1	The CmdletBinding Attribute	13
3.2.2	Task Parameters	13
3.2.3	Return Values from Tasks.....	14
3.2.4	Write to the Logs	14
3.2.5	Include Tasks in a Configuration.....	14
3.3	Create Config Functions	15
3.3.1	Config Function Parameters.....	15
3.3.2	Include Config Functions in a Configuration	15
Chapter 4	Further Guidance & Troubleshooting	16
4.1	Further Usage and Help.....	17
4.1.1	Run Tasks and Config Functions Directly	17
4.1.2	Execution Policies	17
4.1.3	Get Help on the Installation Framework	18
4.2	Troubleshooting.....	22

Sitecore® is a registered trademark. All other brand and product names are the property of their respective holders. The contents of this document are the property of Sitecore. Copyright © 2001-2017 Sitecore. All rights reserved.

Chapter 1

Introduction

This document describes the Sitecore Installation Framework for Sitecore 9.0 rev. 171002 (Initial Release).

This document contains the following chapters:

- **Chapter 1 – Introduction**
An introduction to the Sitecore Installation Framework.
- **Chapter 2 – Install the Module**
Contains information about the steps necessary to install the Sitecore Installation Framework module.
- **Chapter 3 – Customize the Sitecore Installation Framework**
Contains information about extension points that allow you to customize the Framework for your specific installation needs.
- **Chapter 4 – Further Guidance**
Contains information about the built-in help system from Microsoft.

1.1 Getting Started

The Sitecore Installation Framework is a Microsoft® Powershell module that supports local and remote installations of Sitecore, and it is fully extensible. You can install the entire Sitecore solution (XP), or the CMS-only mode (XM) solution.

This Sitecore Installation Framework Configuration Guide describes how to configure and customize the installation process of the Sitecore Experience Platform (XP) to suit your own needs.

1.1.1 How to Use This Guide

This guide describes how to configure your Sitecore installation with the Sitecore Installation Framework, as well as cmdlets and extensibility points. There is also a chapter on further help and guidance.

This guide is intended to be a supplement to the Sitecore 9.0 Installation Guide, to help you customize your installation. Refer to the Installation Guide for information about system requirements and other prerequisites, the general installation process, and the additional configuration tasks that are necessary after installing Sitecore 9.0.

You can download the Sitecore Installation Guide from the Sitecore Downloads page – <https://dev.sitecore.net>.

Chapter 2 Install the Module

This chapter contains information about installing, updating, validating, and importing the Sitecore Installation Framework module.

This chapter contains the following section:

- Install the Sitecore Installation Framework Module

2.1 Install the Sitecore Installation Framework Module

You can install the Sitecore Installation Framework module directly through Microsoft PowerShell®, or you can perform a manual installation by first downloading the Sitecore Fundamentals module and then the Sitecore Installation Framework module as .ZIP packages.

Note

Microsoft PowerShell automation of Public Key Infrastructure (PKI) configuration functions on IIS 8.5 are not yet supported by the Sitecore Installation Framework.

2.1.1 Installation with Microsoft PowerShell®

The Sitecore Installation Framework is made available through the [Sitecore Gallery](#). The Sitecore Gallery is a public MyGet feed that can be used to download and install PowerShell modules created by Sitecore.

To install Sitecore 9.0 with PowerShell:

1. To add the repository, in Windows, launch Microsoft PowerShell® as an administrator and run the following cmdlet:

```
Register-PSRepository -Name SitecoreGallery -SourceLocation  
https://sitecore.myget.org/F/sc-powershell/api/v2
```

2. Install the PowerShell module by running the following cmdlet:

```
Install-Module SitecoreInstallFramework
```

3. When prompted to install the module, press **Y**, and then **Enter**.

```
C:\> Install-Module SitecoreInstallFramework  
  
Untrusted repository  
You are installing the modules from an untrusted repository. If you trust this repository, change its  
InstallationPolicy value by running the Set-PSRepository cmdlet. Are you sure you want to install the modules from  
'SitecoreGallery'?  
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is "N"):
```

Update the Sitecore Installation Framework Module

As new features and bug fixes are periodically released, it is recommended that you update the Sitecore Installation Framework.

Note

This procedure is optional.

To update the Sitecore Installation Framework module:

- In a PowerShell command line, run the following cmdlet:

```
Update-Module SitecoreInstallFramework
```

2.1.2 Manual Installation

The Sitecore Installation Framework is also provided as a .ZIP package. When you manually install the Sitecore Installation Framework, you must first download and install the Sitecore Fundamentals package.

You can download the Sitecore Fundamentals and the Sitecore Installation Framework packages from the Sitecore Downloads page – <https://dev.sitecore.net>.

Note

When you download the packages, it is possible that the .ZIP packages are marked as *blocked* by Microsoft Windows. To continue the installation of the Sitecore Installation Framework, you must first unblock the .ZIP packages.

Sitecore Installation Framework Configuration Guide

Unblock a .ZIP package

To unblock a .ZIP package:

1. In Windows Explorer, navigate to the folder where you downloaded the .ZIP packages, and right-click the relevant .ZIP file.
2. Click **Properties**.
3. In the **Properties** dialog box, on the **General** tab, click **Unblock**.
4. Click **OK**.

Extract the Sitecore Installation Framework

The installation path that you must use depends on where you want to install the Sitecore Installation Framework. You can install it for all users (global path), for a specific user, or to a custom location.

Usage	Path
All users	C:\Program Files\WindowsPowerShell\Modules
Specific user	C:\Users\ <user>\Documents\WindowsPowerShell\Modules</user>
Custom location	Any path

For example, if you want to make the Sitecore Installation Framework available to all users:

1. Extract the Sitecore Fundamentals .ZIP package to the following path:
C:\Program Files\WindowsPowerShell\Modules\SitecoreFundamentals
2. Extract the Sitecore Install Framework .ZIP package to the following path:
C:\Program Files\WindowsPowerShell\Modules\SitecoreInstallFramework

2.1.3 Validate the Installation

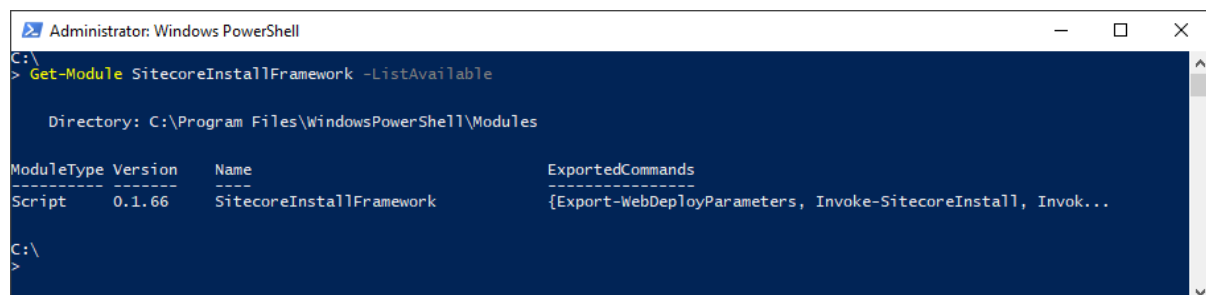
After you install the Sitecore Installation Framework, you can validate its installation to confirm that it is available for use. This procedure is optional.

Note

This validation only works if you have installed the Sitecore Installation Framework to the *All users* (global) path.

To validate the installation:

- In a PowerShell command prompt, run the following cmdlet:
Get-Module SitecoreInstallFramework -ListAvailable



```
Administrator: Windows PowerShell
C:\> Get-Module SitecoreInstallFramework -ListAvailable

Directory: C:\Program Files\WindowsPowerShell\Modules

ModuleType Version Name ExportedCommands
-----
Script 0.1.66 SitecoreInstallFramework {Export-WebDeployParameters, Invoke-SitecoreInstall, Invok...
```

2.1.4 Import the Sitecore Installation Framework into a Powershell session

If you added the Sitecore Installation Framework to either an *All users* or *Specific user* path, you do not have to import it, as this is done automatically, and you can immediately use it in a session by running any cmdlet available in the PowerShell module.

However, if you have installed the Sitecore Installation Framework to a custom location, you must first import the Sitecore Fundamentals package and then import the Sitecore Installation Framework using its full path.

- To import Sitecore Fundamentals, in a PowerShell command line, run the following cmdlet:

```
Import-Module C:\<custompath>\SitecoreFundamentals
```

- To import the Sitecore Installation Framework, in a PowerShell command line, run the following cmdlet:

```
Import-Module C:\<custompath>\SitecoreInstallFramework
```


Chapter 3

Customize the Sitecore Installation Framework

The Sitecore Installation Framework enables you to customize the installation process by using a standard configuration design that you can extend by using custom PowerShell functions. The framework defines a configuration format that supports tasks, parameters, config functions, and variables. For example, you can configure a computer with one or more Sitecore instance, add services, or add custom applications.

All Sitecore Installation Framework configurations are written as .JSON files.

This chapter contains the following sections:

- Create and Customize Configurations
- Create Tasks
- Create Config Functions

3.1 Create and Customize Configurations

You can use tasks, parameters, config functions, and variables to customize your installation process. Custom tasks and config functions can also be packaged as a module, and be included in configurations.

You can base your customization on one of the configurations provided by Sitecore, or you can create your own configuration. This section describes the different components that you can use in a Sitecore Installation Framework configuration.

3.1.1 Tasks

Tasks are actions that are conducted in sequence when the `Install-SitecoreConfiguration` cmdlet is called. A task is implemented as a PowerShell cmdlet.

Each task is identified by a unique name and must contain a `Type` property. A task can have parameters or a collection of parameters passed to it. Tasks map directly to PowerShell functions that are registered with `Register-SitecoreInstallExtension -Type Task`.

The following example shows a task of type `Copy` that you can use in a configuration to copy files from one location to another:

```
{
  // Tasks lists all the tasks to be executed
  "Tasks": {
    // The following task has a name of 'CopyFiles'
    "CopyFiles" : {
      // The task invoked will be the 'Copy' task (Invoke-CopyTask)
      "Type": "Copy",
      "Params": {
        // The task receives the 'Source' and 'Destination' parameters
        "Source": "c:\files",
        "Destination": "c:\newfiles"
      }
    }
  }
}
```

Note

To make sure that your configuration is valid, you must ensure that each task has a unique name. This means that the task can be directly executed, and the task can be identified in a log. It also enables the same type of task to be used multiple times in a configuration.

3.1.2 Parameters

Parameters let users change values inside a configuration at runtime. Parameters must declare a type. They can also declare a default value and a description.

When you add a parameter to a configuration, it can be passed into a task using the `parameter` config function. The task then points to the value provided in the configuration or the value passed when `Install-SitecoreConfiguration` is called. For example, to pass the `Source` and `Destination` parameters to the `CopyFiles` task:

```
{
  "Parameters": {
    "Source": { "Type": "string", "Description": "The source of files" },
    "Destination": { "Type": "string", "DefaultValue": "c:\newfiles" }
  },
  "Tasks": {
    "CopyFiles" : {
      "Type": "Copy",
      "Params": {
        "Source": "[parameter('Source')]",
        "Destination": "[parameter('Destination')]"
      }
    }
  }
}
```

```
}
```

On the one hand, the `Source` parameter does not contain a `DefaultValue` property, so it is required when `Install-SitecoreConfiguration` is called.

However, on the other hand, the `Destination` parameter does have a default value. If the user does not provide a value, the `DefaultValue` from the configuration is used.

The values at runtime are then passed to the `CopyFiles` task using the `parameter` config function.

To pass the values at the command line:

- You must use the name of the parameter with standard PowerShell parameter syntax. For example:
`Install-SitecoreConfiguration -Path .\configuration.json -Source c:\sourcefiles`

3.1.3 Config Functions

Config functions allow elements of the configuration to be dynamic, letting you calculate values, invoke functions, and pass these values to tasks so that a configuration can be flexible.

A config function is written as a string enclosed in square brackets `[]`. It is identified by a name and can receive function parameters. For example: `[functionName (param1, param2, param3)]`

Each function maps directly to a PowerShell function that is registered with the following cmdlet:

```
Register-SitecoreInstallExtension -Type ConfigFunction
```

3.1.4 Variables

Variables are values that are calculated within the configuration itself. Unlike parameters, variables can use config functions to determine their value.

Variables cannot be overridden at runtime; however, when they are being calculated, they can use the values from parameters and other variables. In the following example, the `Destination` variable determines the destination path using the `environment` and `concat` config functions:

```
{
  "Parameters": {
    "Source": { "Type": "string", "Description": "The source of files" }
  },
  "Variables": {
    "Destination": "[concat(environment('SystemDrive'), '\\newfiles')]"
  },
  "Tasks": {
    "CopyFiles": {
      "Type": "Copy",
      "Params": {
        "Source": "[parameter('Source')]",
        "Destination": "[variable('Destination')]"
      }
    }
  }
}
```

3.1.5 Modules

The Sitecore Installation Framework provides many tasks and config functions. You can load additional tasks and config functions by including them in a configuration.

If the module is available on `PSModulePath`, you can load modules by name. Alternatively, you must load modules using an explicit path. Modules are loaded at the beginning of an installation.

If the additional features are packaged in a module on your computer, you can add them in the `Modules` section of a configuration to import them into the install session. For example:

- To load a module by name –`"MyCustomModule"`.

- To load a module by directory – "C:\\modules\\custom".
- To load a module by explicit path – "C:\\extensions\\extensions.psml".

```
{
  "Modules": [
    "MyCustomModule",
    "C:\\modules\\custom",
    "C:\\extensions\\extensions.psml"
  ]
}
```

3.1.6 Settings

Settings let you configure the default requirements of the installation process. Some settings can also be overridden at runtime for the user by passing them as parameters to the `Install-SitecoreConfiguration` cmdlet.

For example: `Install-SitecoreConfiguration -WarningAction Stop -InformationAction SilentlyContinue`.

The values in the settings are applied from lowest to highest in the following order:

- Default value – Contained in code.
- Configuration – Set in the configuration file.
- Command line – Passed at the command line.

Name	Default Value	Allowed Values	Command line?	Description
WarningAction	Continue	Continue Ignore Inquire SilentlyContinue Stop Suspend	Yes	The action to take when a warning occurs.
ErrorAction	Stop	Continue Ignore Inquire SilentlyContinue Stop Suspend	Yes	The action to take when an error occurs.
InformationAction	Continue	Continue Ignore Inquire SilentlyContinue Stop Suspend	Yes	The action to take when information is logged.

3.2 Create Tasks

Tasks are PowerShell functions that you can invoke from within a Sitecore Installation Framework configuration. When you invoke a configuration, each task performs an action. Because a task is implemented as a PowerShell function, it benefits from all the features that PowerShell offers.

3.2.1 The CmdletBinding Attribute

When you create a task, you must use the `CmdletBinding` attribute. This provides support for common PowerShell parameters, for example, those that control error handling.

It is best practice to use the `SupportsShouldProcess` parameter so that users can test the actions that the task takes, without applying them. The following example shows the use of the `CmdletBinding` attribute in the `Invoke-UnpackTask` cmdlet:

```
Function Invoke-UnpackTask {
    [CmdletBinding(SupportsShouldProcess=$true)]
    param(
        # Parameters
    )
    # function code
}
```

For more information about the `CmdletBinding` attribute, see the [About Functions CmdletBindingAttribute](#) article on the MSDN website.

3.2.2 Task Parameters

Task parameters are declared as normal PowerShell parameters. You can use validation and types to restrict the values that can be passed to the cmdlet. This includes marking parameters as mandatory, and support for multiple parameter sets.

When a task is called from a configuration, the `Params` property is mapped to the parameters declared in the PowerShell function. For example, the `Invoke-CopyTask` cmdlet declares the following parameters:

```
Function Invoke-CopyTask {
    [CmdletBinding(SupportsShouldProcess=$true)]
    param(
        [Parameter(Mandatory=$true)]
        [ValidateScript({ Test-Path $ })]
        [string]$Source,
        [Parameter(Mandatory=$true)]
        [ValidateScript({ Test-Path $ -IsValid })]
        [string]$Destination
    )
    # function code
}
```

- The `Source` parameter is mandatory and checks that the given value is a string that points to a path that exists.
- The `Destination` parameter is also mandatory and checks that the given value is a string that is a valid file path.

The following is an example of how to declare the `Copy` task in a configuration where only the mandatory parameters are used:

```
{
    "Tasks": {
        "CopySomeFiles": {
            "Type": "Copy",
            "Params": {
                "Source": "c:\somefile\example.txt",
                "Destination": "c:\copied\"
            }
        }
    }
}
```

3.2.3 Return Values from Tasks

Tasks do not need return values. When invoked through a configuration, values returned from a task are not captured or processed directly. However, any value that you return from a task is shown in the install logs.

3.2.4 Write to the Logs

You can write log information from a task using the `Write-TaskInfo` function. The function receives `Message` and `Tag` parameters.

The task name is discovered by default but it can also be passed by using the `taskname` function. Log information is displayed in the following format: `[<TaskName>]: [<Tag>] <Message>`

For example, if you use the following command during the execution of the `CustomTask` function:

```
Write-TaskInfo -Message Updated -Tag Info
```

The log creates the following entry: `[CustomTask]: [Info] Updated`

The log files are named after the configuration file that is being used, and a time stamp in the following format: `YYMMDD (Year/Month/Day)`. For example: if the configuration file is `sitecore-xp0.json`, and this file was used on July 4, 2017, the following log file is created: `sitecore-xp0.170704.log`.

3.2.5 Include Tasks in a Configuration

Once a task has been written, it must be registered with the Installation Framework. Tasks can be included in a configuration by packaging them as a PowerShell module and adding them to the `Modules` section of a configuration.

By using the `Register-SitecoreInstallExtension` cmdlet, the task is made available for use in configurations. For example the `Copy-CustomItems` cmdlet can be registered as the `CustomCopy` task using the following cmdlet:

- `Register-SitecoreInstallExtension -Command Copy-CustomItems -As CustomCopy -Type Task`

Note

You can replace an existing registered task by using the `-Force` parameter. The following custom cmdlet replaces the default copy task: `Register-SitecoreInstallExtension -Command Copy-CustomItems -As Copy -Type Task -Force`

3.3 Create Config Functions

Config functions are PowerShell functions that you can invoke from within a Sitecore Installation Framework configuration to access and calculate values that can be passed to a task.

Because a config function is implemented as a PowerShell function, it benefits from all the features that PowerShell offers. This includes parameter validation, strict mode, requires, and others.

Config functions must always return a value, and this value can be used by other config functions or by tasks within a configuration.

3.3.1 Config Function Parameters

Config function parameters are declared as normal PowerShell parameters. You can use validation and types to restrict the values that can be passed. This includes marking parameters as mandatory and support for multiple parameter sets.

When a config function is called from a configuration, the parameters are applied in the order that they are declared. If you want them to be applied in a different order, use the [Position argument](#) to specify this.

The `Invoke-JoinConfigFunction` function declares the following parameters:

```
Function Invoke-JoinConfigFunction {
    param(
        [Parameter(Mandatory=$true)]
        [psobject[]]$Values = @(),
        [Parameter(Mandatory=$false)]
        [string]$Delimiter = ","
    )
    # function code
}
```

In a configuration, this can be used as follows:

```
{
  "Parameters": {
    "Values": { "Type": "string[]", "DefaultValue": [ 1,2,3,4,5 ]
  },
  "Variables": {
    "Joined": "[join(parameter('Values'), '-')]"
  }
}
```

When the `Joined` variable is evaluated, it results in a value of: 1-2-3-4-5.

3.3.2 Include Config Functions in a Configuration

Once a config function has been written, it must be registered with the Installation Framework. Config functions can be included in a configuration by packaging them as a PowerShell module and adding them to the `Modules` section of a configuration.

By using the `Register-SitecoreInstallExtension` cmdlet, the config function is made available for use in configurations. For example the `Get-CustomJoin` cmdlet can be registered as the `CustomJoin` config function using the following cmdlet:

```
Register-SitecoreInstallExtension -Command Get-CustomJoin -As CustomJoin -
Type ConfigFunction
```

Note

You can also replace an existing registered config function by using the `-Force` parameter. The following replaces the default `join` config function with a custom cmdlet:

```
Register-SitecoreInstallExtension -Command Get-CustomJoin -As Join -Type
ConfigFunction -Force
```

Chapter 4

Further Guidance & Troubleshooting

The Sitecore Installation Framework contains embedded documentation about tasks and config functions that you can access directly from the PowerShell command line.

This chapter contains the following section:

- Further Usage and Help
- Troubleshooting

4.1 Further Usage and Help

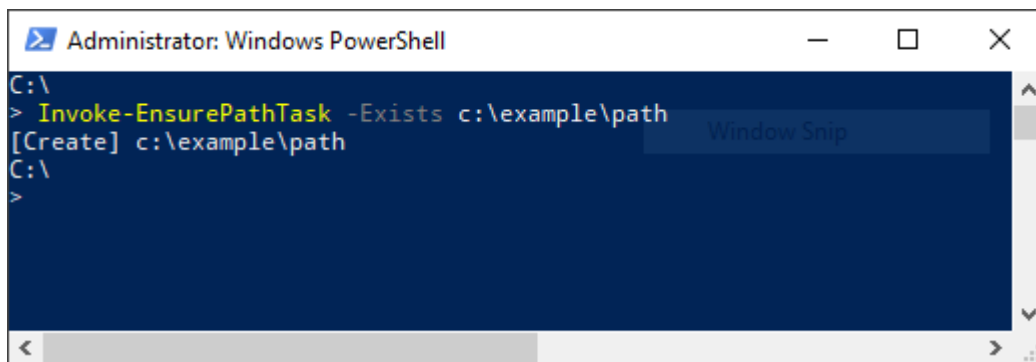
This section contains the following additional information that might be useful when you are using the Sitecore Installation Framework:

- Run Tasks and Config Functions Directly
- Execution Policies
- Get Help on the Installation Framework

4.1.1 Run Tasks and Config Functions Directly

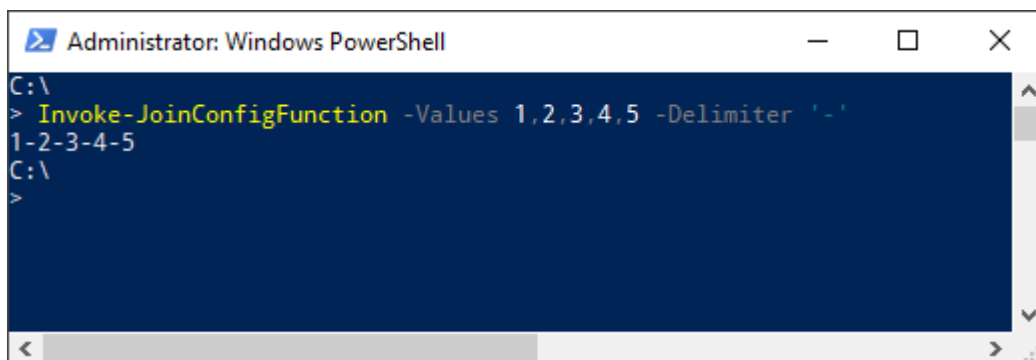
Both tasks and config functions are implemented as PowerShell cmdlets. When the Sitecore Installation Framework has been installed, you can directly invoke the cmdlets using standard PowerShell syntax.

Running tasks or config functions directly allows you to test the results at the command line. You can also integrate the commands into your own PowerShell scripts. For example, you can directly invoke the `EnsurePath` task by using its full PowerShell syntax:



```
Administrator: Windows PowerShell
C:\
> Invoke-EnsurePathTask -Exists c:\example\path
[Create] c:\example\path
C:\
>
```

Similarly, you can directly invoke the `Join` config function by using its full PowerShell syntax:



```
Administrator: Windows PowerShell
C:\
> Invoke-JoinConfigFunction -Values 1,2,3,4,5 -Delimiter '-'
1-2-3-4-5
C:\
>
```

You can see the next chapter in this guide to get more [help and guidance](#) on using tasks and config functions directly in the PowerShell module.

4.1.2 Execution Policies

The execution policies in PowerShell let you restrict the conditions in which scripts and modules are loaded.

You can set policies for the computer, the user, or for the current session. You can also apply them by using a Group Policy. The Sitecore Installation Framework is digitally signed. This means that the module can be imported and executed in a PowerShell session running under any execution policy, except **Restricted**.

For more information about PowerShell Execution Policies, see the following [link](#).

4.1.3 Get Help on the Installation Framework

The Sitecore Installation Framework contains information about each task and config function, and there is also documentation about the general use of the framework:

- `about_SitecoreInstallFramework`: contains general information about the framework.
- `about_SitecoreInstallFramework_Extending`: contains information about extension points that let you customize the framework for your specific installation needs.
- `about_SitecoreInstallFramework_Configurations`: contains examples of how to use tasks, scripts, and modules if you extend the framework.

There are three ways to view the help documentation:

- In the PowerShell window.
- With a markdown reader.
- As HTML pages.

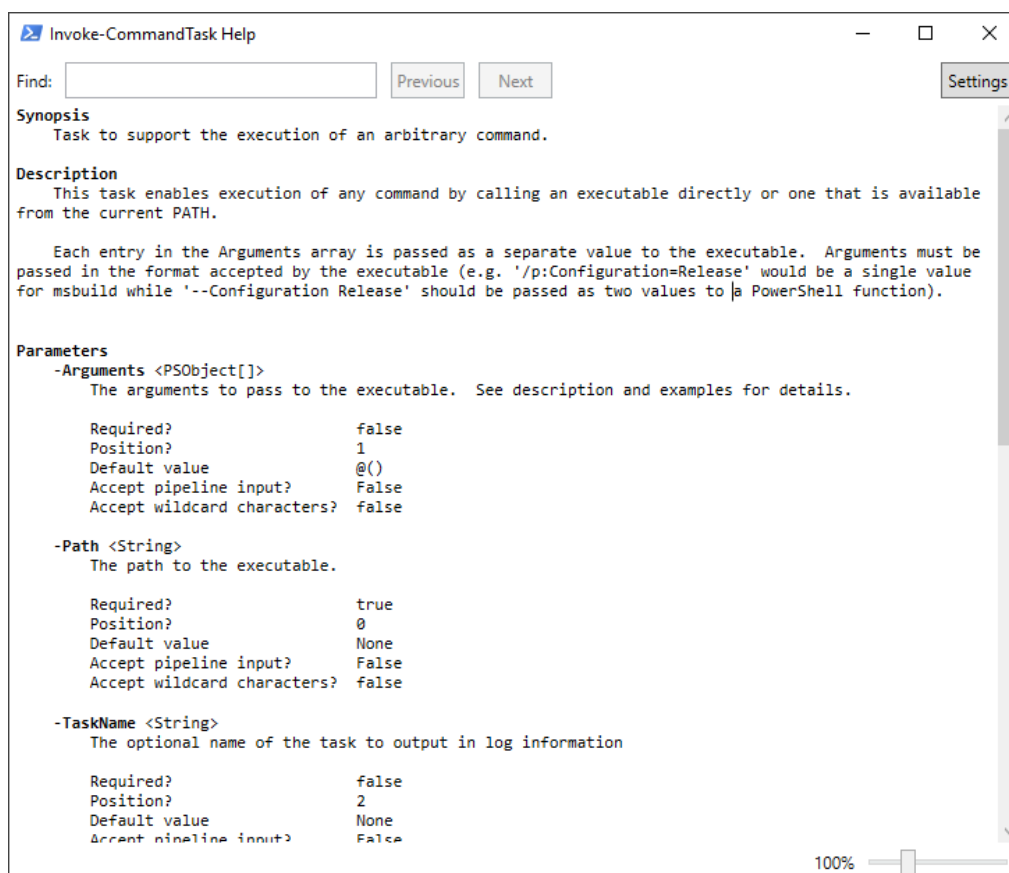
In addition to the help for individual tasks and functions, it is also possible to get a list of all available tasks or config functions using the `Get-Command` cmdlet.

View Help in the PowerShell Window

The PowerShell module has embedded help documentation that you can read.

To access the help for a specific PowerShell command or function:

- In PowerShell, enter the `Get-Help` cmdlet. When you do this, the help is displayed in the command line. If you want to open the help in a separate window, add the `ShowWindow` parameter to the `Get-Help` command. For example: `Get-Help Invoke-CommandTask -ShowWindow`



Sitecore Installation Framework Configuration Guide

View Help with a Markdown Reader

When you unpack the Sitecore Installation Framework, it contains a folder of markdown documentation that you can read with any text editor or markdown reader.

To read the documentation:

- In Windows, navigate to the `SitecoreInstallFramework\docs` folder and using a text editor or markdown reader open the relevant topic.

The following screenshot shows a subset of the topics that are available:

about_SitecoreInstallFramework.md	20/06/2017 13:33	Markdown Source...
about_SitecoreInstallFramework_Configu...	20/06/2017 13:33	Markdown Source...
about_SitecoreInstallFramework_Extendi...	20/06/2017 13:33	Markdown Source...
Export-WebDeployParameters.md	20/06/2017 13:33	Markdown Source...
Invoke-AddXmlTask.md	20/06/2017 13:33	Markdown Source...
Invoke-AppPoolTask.md	20/06/2017 13:33	Markdown Source...
Invoke-CommandTask.md	20/06/2017 13:33	Markdown Source...
Invoke-ConcatConfigFunction.md	20/06/2017 13:33	Markdown Source...
Invoke-CopyTask.md	20/06/2017 13:33	Markdown Source...
Invoke-CreateServiceTask.md	20/06/2017 13:33	Markdown Source...
Invoke-DownloadFileTask.md	20/06/2017 13:33	Markdown Source...
Invoke-EnsurePathTask.md	20/06/2017 13:33	Markdown Source...
Invoke-EnvironmentConfigFunction.md	20/06/2017 13:33	Markdown Source...
Invoke-FilePermissionsTask.md	20/06/2017 13:33	Markdown Source...

View Help as HTML Pages

The Sitecore Installation Framework contains a folder of HTML pages that you can read with any browser.

To read the documentation:

1. In Windows Explorer, navigate to the `SitecoreInstallFramework\docs\html` folder.
2. To open it in a browser window, double-click the relevant topic.

Get a List of the Available Tasks and Config Functions

To see the tasks and config functions available:

- In a PowerShell command line, run the `Get-SitecoreInstallExtension` cmdlet.

```
Administrator: Windows PowerShell
C:\>
> Get-SitecoreInstallExtension

Name                Type                Command                Module
----                -
AddXml              Task                Invoke-AddXmlTask      SitecoreInstallFramework
AppPool             Task                Invoke-AppPoolTask     SitecoreInstallFramework
Command            Task                Invoke-CommandTask     SitecoreInstallFramework
Copy               Task                Invoke-CopyTask        SitecoreInstallFramework
CreateService       Task                Invoke-CreateServiceTask SitecoreInstallFramework
DownloadFile        Task                Invoke-DownloadFileTask SitecoreInstallFramework
EnsurePath          Task                Invoke-EnsurePathTask  SitecoreInstallFramework
FilePermissions     Task                Invoke-FilePermissionsTask SitecoreInstallFramework
HostHeader          Task                Invoke-HostHeaderTask  SitecoreInstallFramework
HttpRequest         Task                Invoke-HttpRequestTask SitecoreInstallFramework
ManageAppPool       Task                Invoke-ManageAppPoolTask SitecoreInstallFramework
ManageService       Task                Invoke-ManageServiceTask SitecoreInstallFramework
ManageSolrConfig    Task                Invoke-ManageSolrConfigTask SitecoreInstallFramework
ManageSolrCore      Task                Invoke-ManageSolrCoreTask SitecoreInstallFramework
ManageSolrSchema    Task                Invoke-ManageSolrSchemaTask SitecoreInstallFramework
ManageWebsite       Task                Invoke-ManageWebsiteTask SitecoreInstallFramework
RemoveService       Task                Invoke-RemoveServiceTask SitecoreInstallFramework
SetXml              Task                Invoke-SetXmlTask      SitecoreInstallFramework
Unpack              Task                Invoke-UnpackTask      SitecoreInstallFramework
WebBinding          Task                Invoke-WebBindingTask  SitecoreInstallFramework
WebDeploy           Task                Invoke-WebDeployTask   SitecoreInstallFramework
Website             Task                Invoke-WebsiteTask     SitecoreInstallFramework
WebsiteClientCert   Task                Invoke-WebsiteClientCertTask SitecoreInstallFramework
And                 ConfigFunction      Invoke-AndConfigFunction SitecoreInstallFramework
Concat              ConfigFunction      Invoke-ConcatConfigFunction SitecoreInstallFramework
Environment          ConfigFunction      Invoke-EnvironmentConfigFunction SitecoreInstallFramework
Equal               ConfigFunction      Invoke-EqualConfigFunction SitecoreInstallFramework
If                  ConfigFunction      Invoke-IfConfigFunction SitecoreInstallFramework
Join                ConfigFunction      Invoke-JoinConfigFunction SitecoreInstallFramework
JoinPath            ConfigFunction      Invoke-JoinPathConfigFunction SitecoreInstallFramework
Not                 ConfigFunction      Invoke-NotConfigFunction SitecoreInstallFramework
Or                  ConfigFunction      Invoke-OrConfigFunction SitecoreInstallFramework
ReadJson            ConfigFunction      Invoke-ReadJsonConfigFunction SitecoreInstallFramework
ResolveCertificatePath ConfigFunction      Invoke-ResolveCertificatePathConfigFunction SitecoreInstallFramework
ResolvePath         ConfigFunction      Invoke-ResolvePathConfigFunction SitecoreInstallFramework
SqlConnectionString ConfigFunction      Invoke-SqlConnectionStringConfigFunction SitecoreInstallFramework

C:\>
>
```

Note

You can filter the list by passing a value to the `Type` parameters. The following command only returns the tasks: `Get-SitecoreInstallExtension -Type Task`.

To return the tasks and config functions available when a particular configuration is run, you must pass the path to the configuration file to the `Path` parameter.

For example, `Get-SitecoreInstallExtension -Path c:\configuration.json` displays the default tasks and config functions, as well as the extra registrations brought in by the configuration.

Sitecore Installation Framework Configuration Guide

```
Administrator: Windows PowerShell
C:\> Get-SitecoreInstallExtension -Path C:\configuration.json
Importing Module => c:\extensions.psm1

Name                Type                Command              Module
----                -
AddXml              Task                Invoke-AddXmlTask    SitecoreInstallFramework
AppPool             Task                Invoke-AppPoolTask   SitecoreInstallFramework
Command            Task                Invoke-CommandTask   SitecoreInstallFramework
Copy               Task                Invoke-CopyTask      SitecoreInstallFramework
CreateService       Task                Invoke-CreateServiceTask SitecoreInstallFramework
DownloadFile        Task                Invoke-DownloadFileTask SitecoreInstallFramework
EnsurePath          Task                Invoke-EnsurePathTask SitecoreInstallFramework
FilePermissions     Task                Invoke-FilePermissionsTask SitecoreInstallFramework
HostHeader          Task                Invoke-HostHeaderTask SitecoreInstallFramework
HttpRequest         Task                Invoke-HttpRequestTask SitecoreInstallFramework
ManageAppPool       Task                Invoke-ManageAppPoolTask SitecoreInstallFramework
ManageService       Task                Invoke-ManageServiceTask SitecoreInstallFramework
ManageSolrConfig    Task                Invoke-ManageSolrConfigTask SitecoreInstallFramework
ManageSolrCore      Task                Invoke-ManageSolrCoreTask SitecoreInstallFramework
ManageSolrSchema    Task                Invoke-ManageSolrSchemaTask SitecoreInstallFramework
ManageWebsite       Task                Invoke-ManageWebsiteTask SitecoreInstallFramework
RemoveService       Task                Invoke-RemoveServiceTask SitecoreInstallFramework
SetXml              Task                Invoke-SetXmlTask    SitecoreInstallFramework
Unpack             Task                Invoke-UnpackTask    SitecoreInstallFramework
WebBinding          Task                Invoke-WebBindingTask SitecoreInstallFramework
WebDeploy           Task                Invoke-WebDeployTask SitecoreInstallFramework
WebSite            Task                Invoke-WebSiteTask   SitecoreInstallFramework
WebsiteClientCert   Task                Invoke-WebsiteClientCertTask SitecoreInstallFramework
And                 ConfigFunction      Invoke-AndConfigFunction SitecoreInstallFramework
Concat              ConfigFunction      Invoke-ConcatConfigFunction SitecoreInstallFramework
Environment          ConfigFunction      Invoke-EnvironmentConfigFunction SitecoreInstallFramework
Equal               ConfigFunction      Invoke-EqualConfigFunction SitecoreInstallFramework
If                  ConfigFunction      Invoke-IfConfigFunction SitecoreInstallFramework
Join                ConfigFunction      Invoke-JoinConfigFunction SitecoreInstallFramework
JoinPath            ConfigFunction      Invoke-JoinPathConfigFunction SitecoreInstallFramework
Not                 ConfigFunction      Invoke-NotConfigFunction SitecoreInstallFramework
Or                  ConfigFunction      Invoke-OrConfigFunction SitecoreInstallFramework
ReadJson            ConfigFunction      Invoke-ReadJsonConfigFunction SitecoreInstallFramework
ResolveCertificatePath ConfigFunction      Invoke-ResolveCertificatePathConfigFunction SitecoreInstallFramework
ResolvePath         ConfigFunction      Invoke-ResolvePathConfigFunction SitecoreInstallFramework
SqlConnectionString ConfigFunction      Invoke-SqlConnectionStringConfigFunction SitecoreInstallFramework
Write               ConfigFunction      Write-Host            Microsoft.PowerShell.Utility

C:\>
```

4.2 Troubleshooting

This section describes some of the issues you might encounter when using the Sitecore Installation Framework, and how to resolve them.

Internal Server Error

After a successful installation, the CM instance cannot be started and the following error is displayed:

```
HTTP Error 500.19 - Internal Server Error
```

```
The requested page cannot be accessed because the related configuration data for the page is invalid.
```

```
Error Code 0x8007000d
```

To resolve this error, you must install the URL Rewrite module for IIS. To do this:

- Install URL Rewrite 2.1 using the [Web Platform Installer](#).

Error When Invoking the Web Deploy Task

When you execute the Web Deploy task (`Invoke-WebDeployTask`), you can see the following error:

```
ERROR_SCRIPTDOM_NEEDED_FOR_SQL_PROVIDER
```

This error appears when the web deploy package installs databases using the SQL DACFx framework, and the provider has not been registered.

To resolve this error, ensure that you have the following components installed:

- [SQL Server System CLR Types](#) (2016 version)
- [SQL Server Transact-SQL ScriptDom](#) (2016 version)
- [SQL Server Data-Tier Application](#) (2016 version)

Note

If you are running the Sitecore installation from a 64-bit computer (x64), you must install both the 32-bit (x86) and 64-bit versions of the SQL Server components.

If you still receive the error after installing the SQL Server components, you must directly register the ScriptDom components.

To register the ScriptDom component:

1. Find the path to the ScriptDom library. In Windows Explorer, navigate to the folder `C:\Program Files (x86)\Microsoft SQL Server`.
2. The *Microsoft SQL Server* folder contains one or more subfolders. Click the subfolders (`\90, \100, \110, \120, \130`) and find the `\DAC\bin\Microsoft.SqlServer.TransactSql.ScriptDom.dll` file.
3. Copy or write down the path where the `.dll` file is located.
4. Launch PowerShell and navigate to the *NETFX 4.5.1 Tools* folder. (`C:\Program Files (x86)\Microsoft SDKs\Windows\v8.1A\bin\NETFX 4.5.1 Tools`).
5. Invoke the `gacutil` application and enter the path that you copied in step 3. For example:

```
gacutil.exe /i C:\Program Files (x86)\Microsoft
SDKs\Windows\v8.1A\bin\NETFX 4.5.1
Tools\120\DAC\bin\Microsoft.SqlServer.TransactSql.ScriptDom.dll
```

Sitecore Installation Framework Configuration Guide

Missing Modules

Some features within the Sitecore Installation Framework require that other modules be loaded as well. You might see warnings that certain tasks cannot be loaded when importing the module.

You can continue to use other features in the module, however, the features that displayed warnings cannot be executed. For example, if the `WebAdministration` module is not available, you see the following warnings:

```
> Import-Module .\SitecoreInstallFramework
WARNING: The script 'Invoke-AppPoolTask.ps1' cannot be run because the following modules that are specified by the
"#requires" statements of the script are missing: WebAdministration.
WARNING: The script 'Invoke-WebBindingTask.ps1' cannot be run because the following modules that are specified by the
"#requires" statements of the script are missing: WebAdministration
WARNING: The script 'Invoke-WebsiteTask.ps1' cannot be run because the following modules that are specified by the
"#requires" statements of the script are missing: WebAdministration
```

When this happens, the module is loaded but the following tasks are not available:

- `Invoke-AppPoolTask`
- `Invoke-WebBindingTask`
- `Invoke-WebsiteTask`

Note

To install the `WebAdministration` module, you must first configure IIS on your computer. For more details and steps to configure IIS, see the *Configuring IIS* section in the Sitecore Experience Platform 9.0 rev. 171002 Installation Guide that you can download from the Sitecore Downloads page – <https://dev.sitecore.net>.

Administrator Permissions

To run the Sitecore Installation Framework, you must run PowerShell as an Administrator.

Important

If you try to use the Sitecore Installation Framework in a non-admin window, you can see one of the following error messages and you will not be able to install Sitecore.

```
Import-Module : The required module 'SitecoreFundamentals' is not loaded. Load the module or
remove the module from 'RequiredModules' in the file
```

```
import-module : The script 'SitecoreFundamentals.psml' cannot be run because it contains a
"#requires" statement for running as Administrator. The current Windows PowerShell session is
not running as Administrator. Start Windows PowerShell by using the Run as Administrator
option, and then try running the script again.
```