# Data One 4 PowerShell 2013

## Installation Guide / Manual

**Data One**

Saarbrücken, 2018-02-12

### Description

This document provides instructions on how to install “Data One 4 PowerShell 2013” (Abrrev.: DO4PS) and its features. Apart from the manual itself, a list of frequently asked questions (F.A.Q.) is included. The last section of this guide will then mention limitations that have been observed in the past.

**© Data One GmbH**
www.dataone.de

## Versions

| Version | Date | Author | Changes |
|---|---|---|---|
| 1.0 | 2014-01-03 | Data One | Initial Version |
| 1.1 | 2014-01-30 | Data One | F.A.Q. Updates |
| 1.2 | 2014-10-02 | Data One | F.A.Q Updates |
| 1.3 | 2014-10-07 | Data One | F.A.Q Updates |
| 1.4 | 2014-11-06 | Data One | F.A.Q Updates |
| 1.5 | 2015-01-29 | Data One | Updates UAC, F.A.Q. Updates |
| 1.6 | 2015-02-24 | Data One | Updates PowerModule |
| 1.7 | 2015-04-04 | Data One | Updates Registry access not allowed |
| 1.8 | 2015-04-24 | Data One | F.A.Q Updates / Account |
| 1.9 | 2015-05-12 | Data One | Predefined variables |
| 1.10 | 2015-05-27 | Data One | Translation |
| 1.11 | 2015-06-01 | Data One | Message – Cannot invoke this function |
| 1.12 | 2015-09-08 | Data One | Collections |
| 1.13 | 2015-11-11 | Data One | Rebranded product to “Data One 4 PowerShell” |
| 1.14 | 2015-12-18 | Data One | Update DO4PS / PowerModules documentation / Signing key |
| 1.15 | 2016-04-19 | Data One | Update F.A.Q => Namespace not found |
| 1.16 | 2016-07-12 | Data One | Enterprise Mode |
| 1.17 | 2017-10-17 | Data One | Web.config |
| 1.18 | 2017-10-24 | Data One | Summary: Additional setup steps |
| 1.19 | 2018-02-12 | Data One | Installation |

## Table of Contents

# 1. Installation and Configuration

In 2015 this component was renamed from “PowerActivity” to “Data One 4 PowerShell”. In this documentation we also use the abbreviation “DO4PS”.

## 1.1. Requirements

- **.NET 4.X**
- **Microsoft SharePoint 2013 Server**
- **Nintex Workflow 2013**
- Microsoft PowerShell 3.0 (Included with SharePoint 2013 Server)
- Data One Licensing Feature 2013
- A valid trial, development or production license file
  (Refer to http://www.dataone.de/do4powershellnintexworkflow to get a license file)
- Do not activate Enterprise Mode (Compatibility Mode) for DO4PS design pages.

## 1.2. Installing Data One Licensing

**This step is only required if Data One Licensing has not been installed yet.**

- Copy the "DataOne.SharePoint.Licensing.wsp" to a SharePoint server belonging to your farm e.g. "C:\SPSolutions\".
- Start the SharePoint Management Shell as administrator
- Execute the following PowerShell commands. Note that there will be a restart of application pools which will cause a downtime during restart.

```
Add-SPSolution "C:\SPSolutions\DataOne.SharePoint.Licensing.wsp"

Install-SPSolution -Identity "DataOne.SharePoint.Licensing.wsp" -GACDeployment -Force
```

## 1.3. Installing “Data One 4 PowerShell 2013” for Nintex Workflow 2013

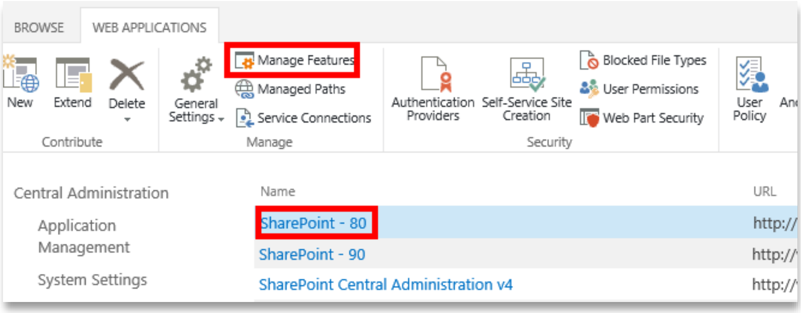
- Copy "DataOne.Nintex.PowerShell.wsp" to a SharePoint server belonging to your farm e.g. "C:\SPSolutions\".
- Start the SharePoint Management Shell as administrator.
- Execute the following PowerShell commands. Note that there will be a restart of application pools which will cause a downtime during restart.

```
Add-SPSolution "C:\SPSolutions\DataOne.Nintex.PowerActivity.wsp"

Install-SPSolution -Identity "DataOne.Nintex.PowerActivity.wsp" –GACDeployment -Force
```

### 1.3.1. Activating the Web Application Feature

Before you can use DO4PS on a SharePoint site, you first have to activate a web application feature called “PowerActivity for Nintex Workflow 2013.

- This needs to be done once for *each web application that you want to use with DO4PS*.
- It is also very important to choose the *SharePoint Central Administration* as the structural level on which to activate this feature. Otherwise you will not able to configure PA2013.



| Name | URL |
|---|---|
| SharePoint - 80 | http:// |
| SharePoint - 90 | http:// |
| SharePoint Central Administration v4 | http:// |

Figure 5: Data One 4 PowerShell – Activating the Web App feature

### 1.3.2. Additional setup steps

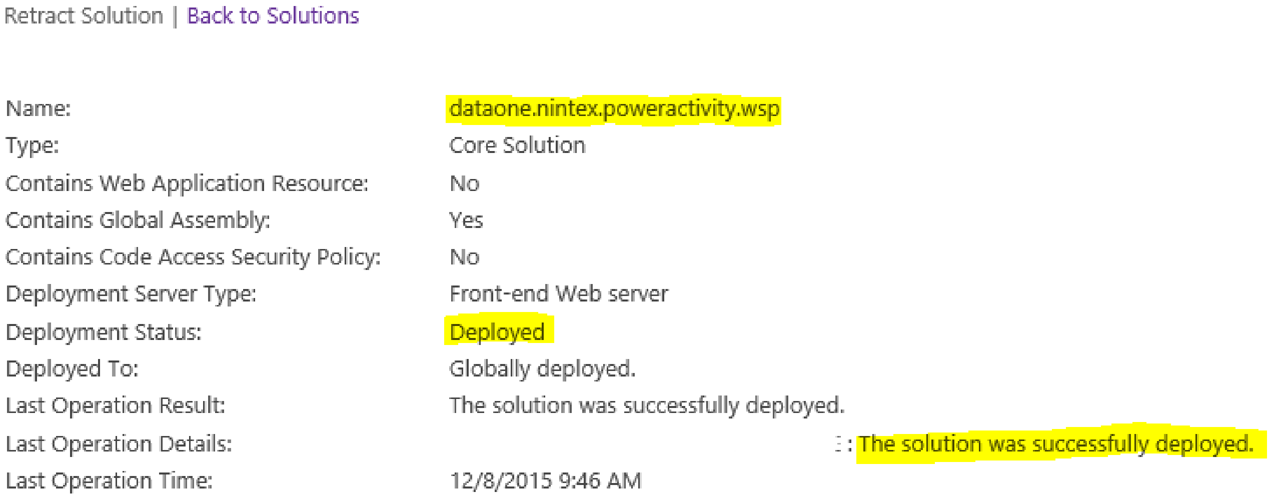
- Legacy CAS: Please refer to chapter 4.9
- DO4PSAccount: If you don't use the default account to execute powershell within DO4PS, you need to grant Shell-Privileges to the account that executes the DO4PS powershell code. Please refer to chapter 1.6.2 / 4.2 (Reason 3)
- UAC: Some CMDLets do not work with UAC. If you receive the error message „Registry access not allowed please refer to chapter 4.2 (Reason 1 / 2)
- Signing Key: This activity uses a signing key to encrypt data. Please refer to chapter 4.1 if you migrate to another environment.

## 1.4. Updating "Data One 4 PowerShell 2013" for Nintex Workflow 2013

- Copy "DataOne.Nintex.PowerShell.wsp" to a SharePoint server belonging to your farm e.g. "C:\SPSolutions\".
- Start the SharePoint Management Shell as administrator.
- Execute the following PowerShell commands. Note that there will be a restart of application pools which will cause a downtime during restart.

```
Update-SPSolution -Identity "DataOne.Nintex.PowerActivity.wsp" -LiteralPath
"C:\SPSolutions\DataOne.Nintex.PowerActivity.wsp" -GACDeployment -Force
```

- Now there will be an IISRESET. Please wait until the component is updated. You can take a look in CentralAdministration -> System Settings -> Manage Farm Solutions. The status of "dataone.nintex.poweractivity.wsp" should be "Deployed"



Name: dataone.nintex.poweractivity.wsp
Type: Core Solution
Contains Web Application Resource: No
Contains Global Assembly: Yes
Contains Code Access Security Policy: No
Deployment Server Type: Front-end Web server
Deployment Status: Deployed
Deployed To: Globally deployed.
Last Operation Result: The solution was successfully deployed.
Last Operation Details: : The solution was successfully deployed.
Last Operation Time: 12/8/2015 9:46 AM

- Now the solution is deployed and we need a feature update. Please replace the URL with your WebApplicationURL. The new version should be 2.1.0.0

```
$webAppUrl = "http://your_web_application_url"
$featureId = "f0439caa-7e60-4581-8fd1-f29199dcd987"

Write-Host "upgrade feature..."
$webApp  = Get-SPWebApplication $webAppUrl
$feature = $webApp.Features|Where {$_.DefinitionId.ToString() -eq $featureId}
Write-Host "updated feature from version " $feature.Version
$feature.Upgrade($false)
Write-Host "updated feature to version " $feature.Version
```

## 1.5. Uninstalling “Data One 4 PowerShell” for Nintex Workflow 2013

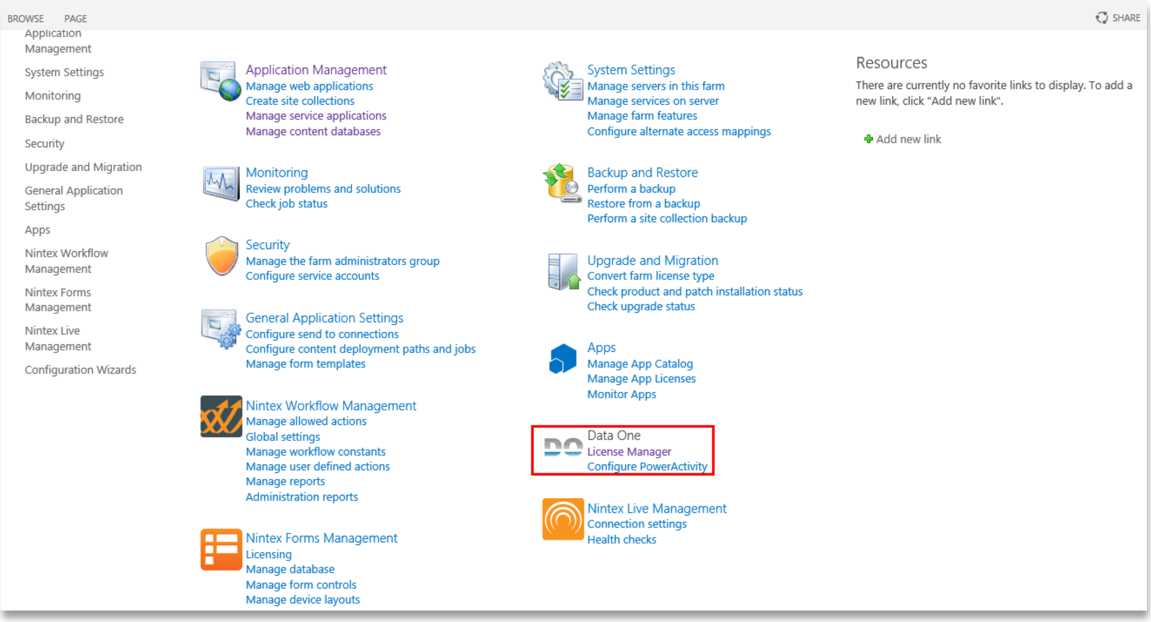
- Start the SharePoint Management Shell.
- Execute the following PowerShell commands. Note that there will be a restart of application pools which will cause a downtime during restart.

```
Uninstall-SPSolution -Identity "DataOne.Nintex.PowerActivity.wsp"

Remove-SPSolution -Identity "DataOne.Nintex.PowerActivity.wsp"
```

## 1.6. Configuring Data One 4 PowerShell for Nintex Workflow 2013

### 1.6.1. License

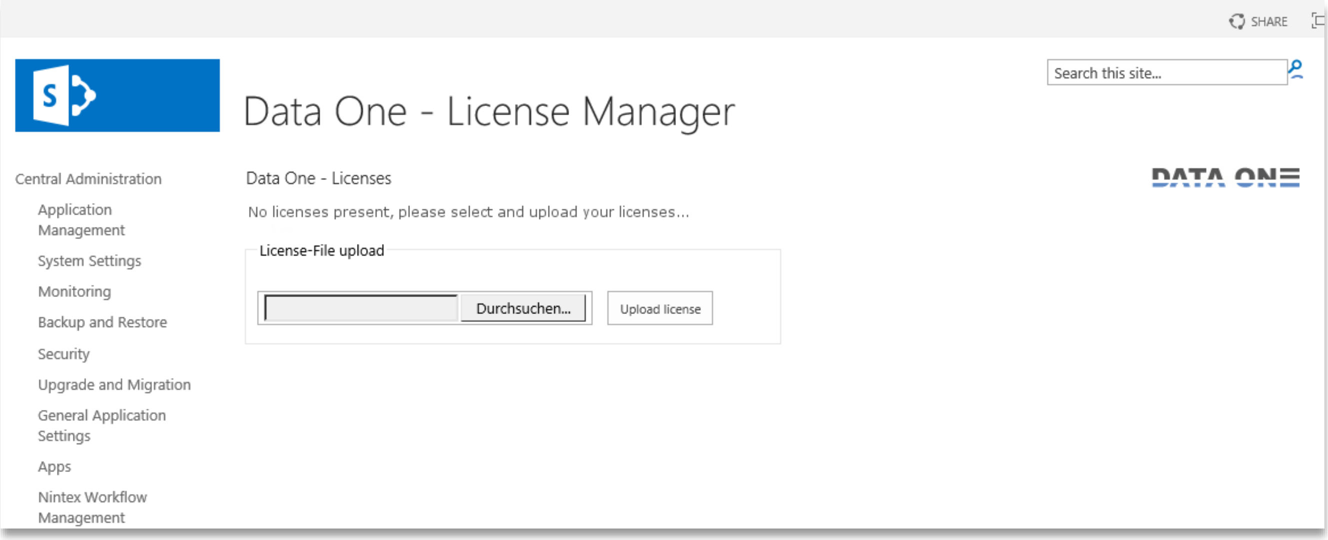
- Open the Central Administration of SharePoint 2013.



There are currently no favorite links to display. To add a new link, click "Add new link".

Figure 1: Central Administration with Data One entries

- In the section Data One, click on License Manager.



Data One - License Manager

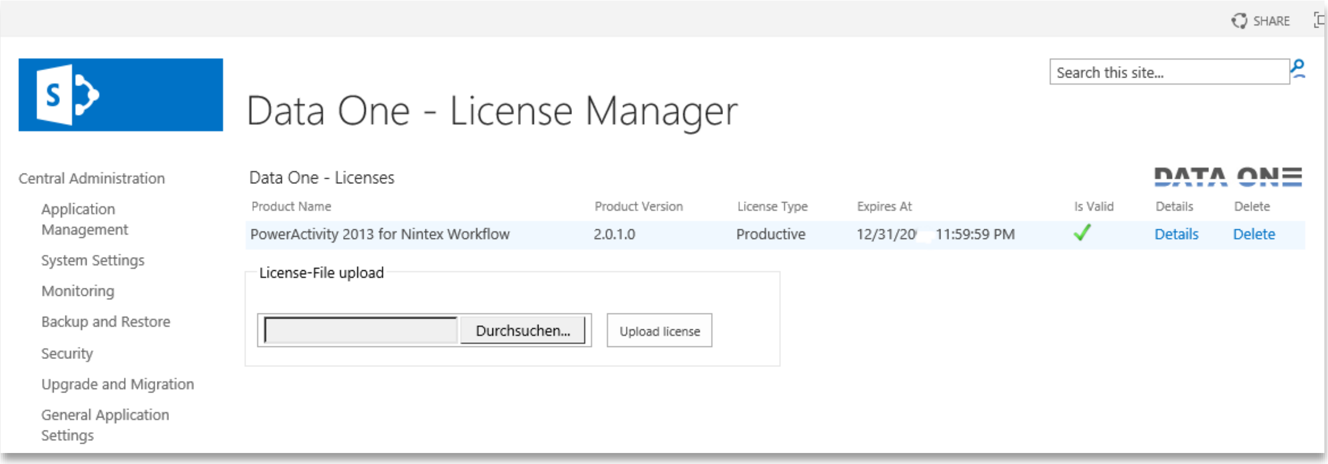
Data One - Licenses

No licenses present, please select and upload your licenses...

License-File upload

Figure 2: License – Upload dialog

- Click the Browse button and choose your license file.
  - Click the Upload button.
  - Once the license has been successfully imported, a page is displayed that shows the relevant product details.



Data One - License Manager

Data One - Licenses

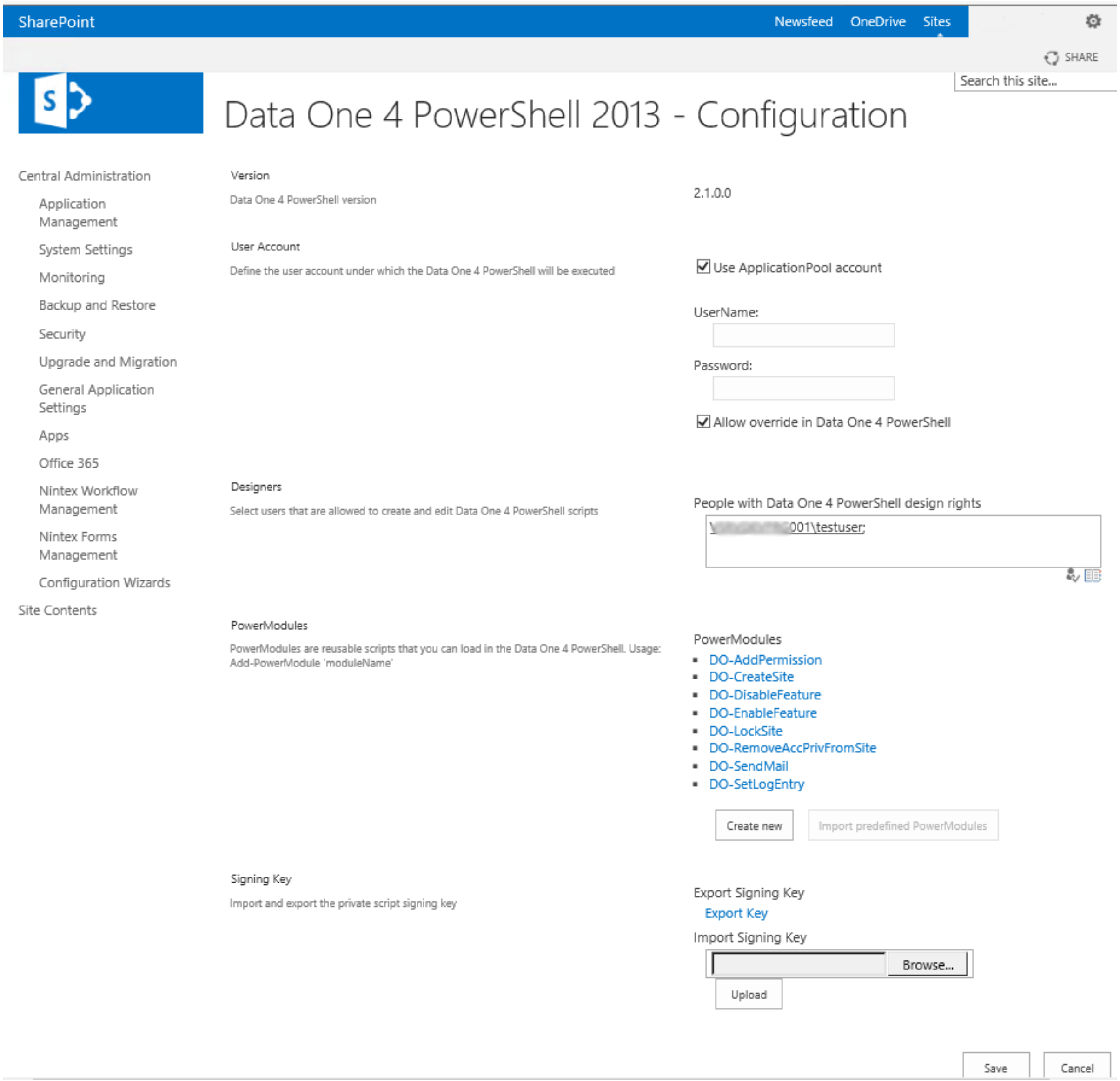
| Product Name | Product Version | License Type | Expires At | Is Valid | Details | Delete |
|---|---|---|---|---|---|---|
| PowerActivity 2013 for Nintex Workflow | 2.0.1.0 | Productive | 12/31/20 11:59:59 PM | ✓ | Details | Delete |

License-File upload

Figure 3: License Dialog with a valid key file

### 1.6.2. Settings

- In order to configure your “Data One 4 PowerShell” action for Nintex Workflow 2013, go to *Central Administration* -> *Data One* -> *Configure Data One 4 PowerShell*.



Data One 4 PowerShell 2013 - Configuration

Version
Data One 4 PowerShell version
2.1.0.0

User Account
Define the user account under which the Data One 4 PowerShell will be executed
☑ Use ApplicationPool account
UserName:
Password:
☑ Allow override in Data One 4 PowerShell

Designers
Select users that are allowed to create and edit Data One 4 PowerShell scripts
People with Data One 4 PowerShell design rights
[redacted]001\testuser;

PowerModules
PowerModules are reusable scripts that you can load in the Data One 4 PowerShell. Usage: Add-PowerModule 'moduleName'
PowerModules
- DO-AddPermission
- DO-CreateSite
- DO-DisableFeature
- DO-EnableFeature
- DO-LockSite
- DO-RemoveAccPrivFromSite
- DO-SendMail
- DO-SetLogEntry

Signing Key
Import and export the private script signing key
Export Signing Key
Export Key
Import Signing Key

Figure 4: Data One 4 PowerShell settings

- The section “Version” contains the DO4PowerShell version number.

- In the section User Account, you can define the windows credentials (“Domain\user”) under which the Data One 4 PowerShell will be executed.

> **By default, Data One 4 PowerShell will run as ApplicationPoolAccount. Consequently, it is necessary that the user specified in the configuration screen above has privileges to execute SharePoint CMDLETS. Otherwise error messages are likely to occur – such as “Registry access not allowed”; “Permission denied”; “The local farm is not accessible” to name but a few.**
>
> **Besides, it is worth mentioning that, by default, if there is a pause in your workflow, the workflow will continue with the user that runs the SharePoint Timer Job.**

To execute SharePoint CMDLETS, the user must have all the privileges that are relevant to the SharePoint object model and to the SharePoint databases (content/admin/config). Use Add-SPShellAdmin to change user privileges if you decide to choose a different account. Of course, you have to repeat these steps for every single SharePoint content/admin/config database the user needs access to.

*Add-SPShellAdmin -UserName ### -database ####*

There is also a Get-SPShellAdmin command.

- In the section “Designers”**,** the users or groups are defined who are allowed to design and publish workflows that contain “Data One 4 PowerShell” actions.

- In the next section, called PowerModules, some additional modules can be selected to be used within “Data One 4 PowerShell”. For example, you may want to choose a module that sets variables and values that are frequently used. If so, just enter *Add-PowerModule '#MODULENAME#'* into the PowerShell Script Mask. Please replace #MODULENAME# and insert the name of the module in question.

  There are some predefined PowerModules included within DO4PowerShell. You can import them with the button “Import predefined PowerModules”. Our predefined PowerModules starts with “DO-“
  They may also receive updates in future versions. If you alter a predefined PowerModule and there is an update you will get a message if you want to overwrite this module or not.

  Ref.: Data One PowerModule – Documentation Chapter 3

- The Signing Key functions as the signature of the workflows on the SharePoint server. This ensures that the imported workflows and custom actions run on the correct machine. You can import and export keys from or to another system to copy workflows or custom actions to a different target system such as from a test system to a productive system after the completion of a test period.

  Impersonation and $credentials are secured with this signing key. If you change your signing key, you have to reenter $credentials and Impersonation and redeploy your workflow.

## 1.7. Checking the Installation and Configuration while Nintex Workflow Designer is Open

Create a new Nintex Workflow and take a look at the workflow action category Data One. Ideally, there should be an action called “Data One 4 PowerShell”.

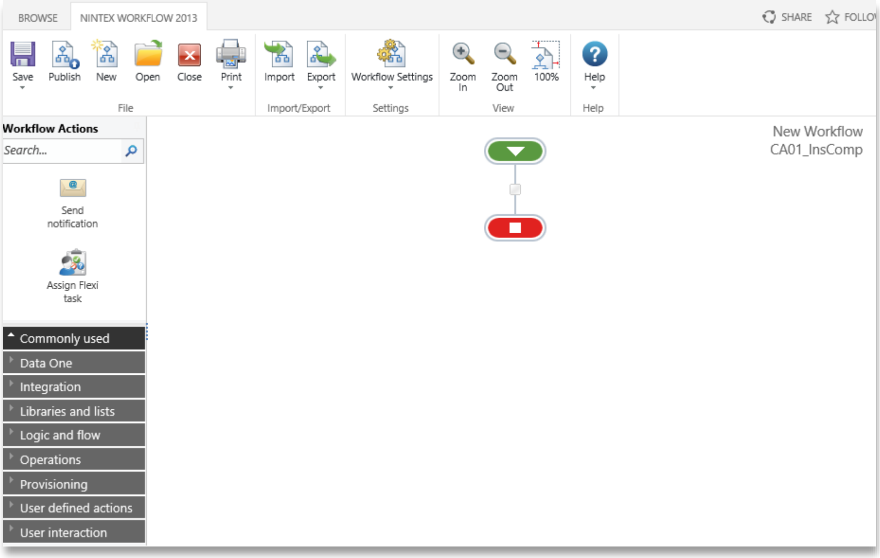


Figure 6: Nintex Workflow Designer

## 1.8. Inserting the Data One 4 PowerShell Action

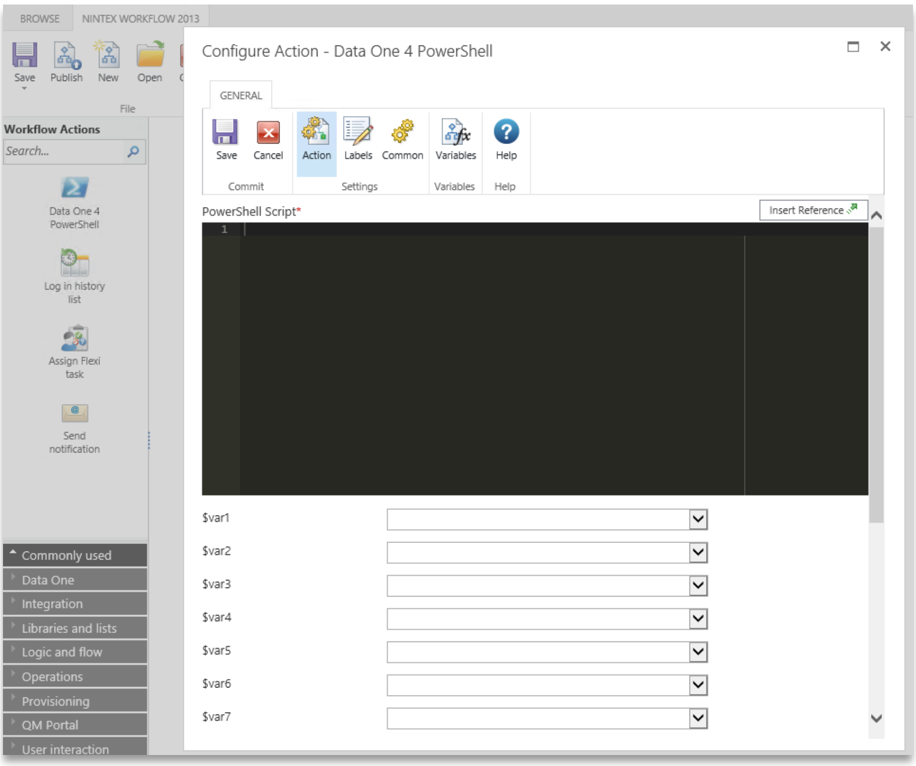


Figure 7: Data One 4 PowerShell – Workflow action configuration

If the “Data One 4 PowerShell” configuration dialog above appears, the installation of DO4PS has been successful.

## 2. Manual: Data One 4 PowerShell for Nintex Workflow 2013

“Data One 4 PowerShell” has been developed to enable users to easily exploit the opportunities inherent to PowerShell within Nintex Workflow. Once DO4PS has been successfully installed, it can be used like any other Nintex Workflow action. To do so, just drag the DO4PS action to your workflow and configure it. The following manual will cover key aspects of using the activity and its features.

### 2.1. How to Use Data One 4 PowerShell

Data One 4 PowerShell action is quick and easy to apply: Open the Nintex Workflow Designer and open or, if needed, create a workflow of your choice. Choose the category “Data One” and drag and drop the DO4PS action to the workflow.

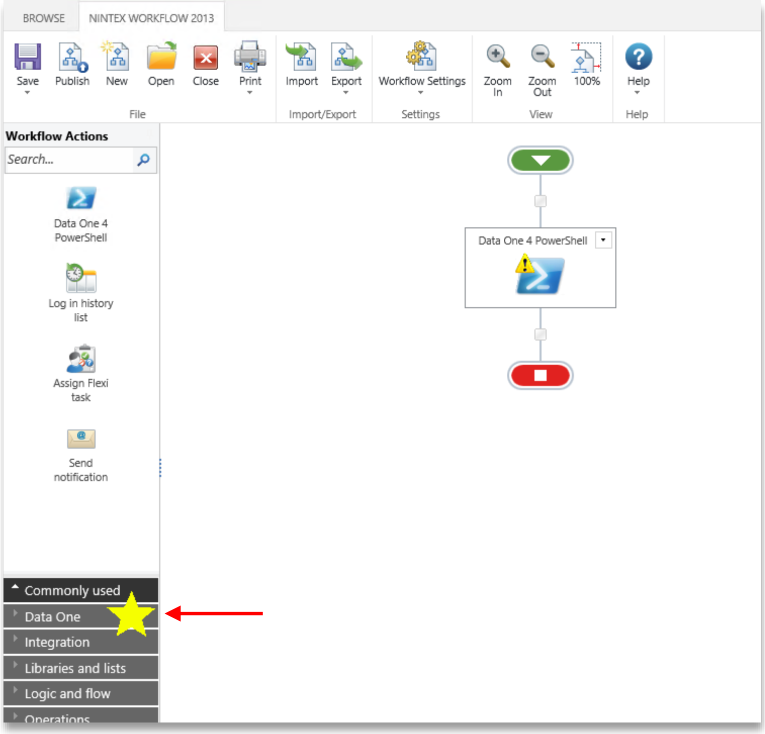


Figure 8: Nintex Workflow Designer with “Data One 4 PowerShell” action

Now, you will have to configure the workflow action. To do so, double click on the action or, alternatively, select “Configure” from the menu in the right upper corner.

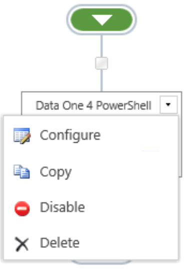
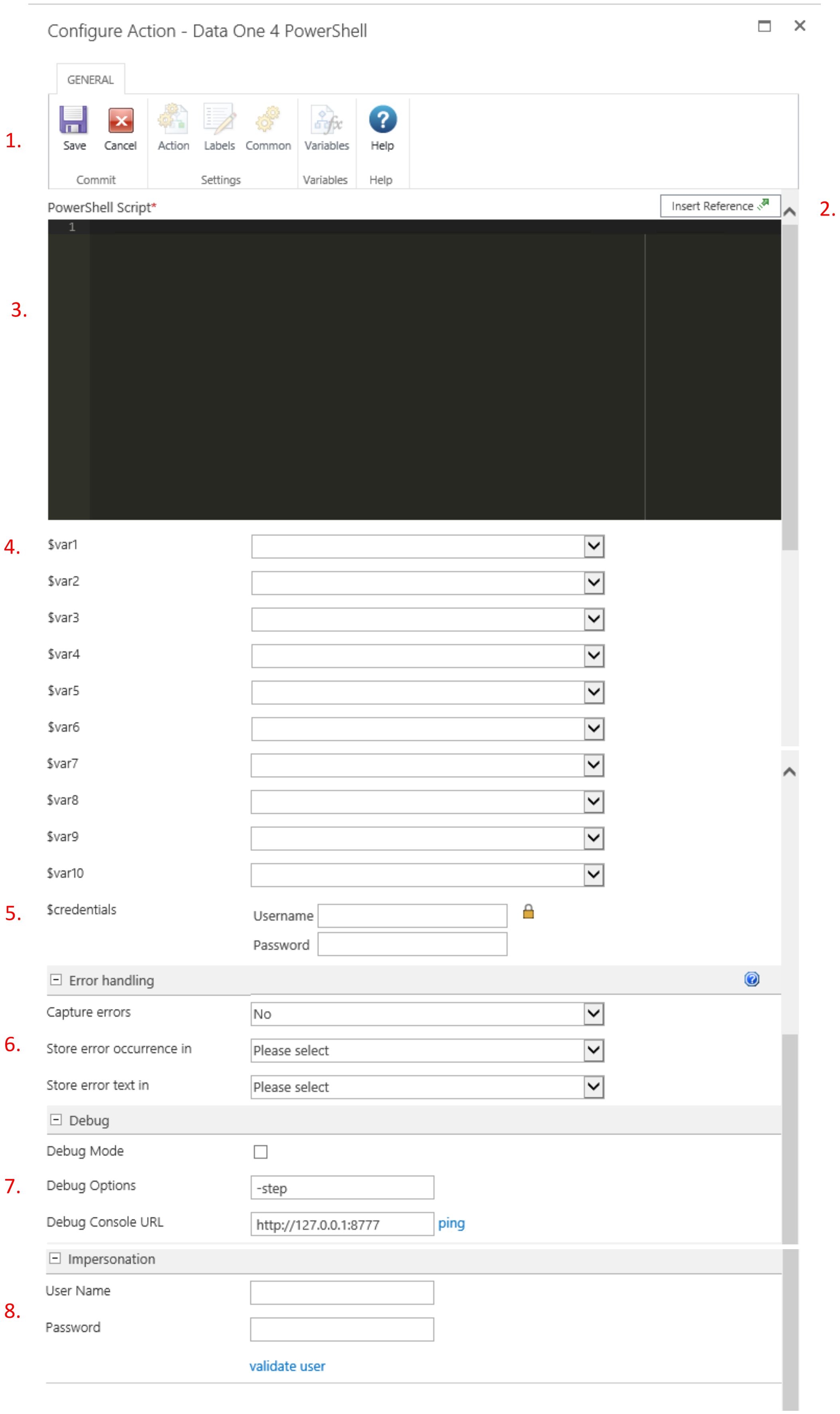


Figure 9: Open configuration

The configuration mask below will open and you can start configuring your DO4PS action.

## 2.2. The Configuration Mask



Capture errors: No

Store error occurrence in: Please select

Store error text in: Please select

Debug Mode ☐

Debug Options: -step

Debug Console URL: http://127.0.0.1:8777

Figure 10: Data One 4 PowerShell – Configuration mask

### 2.2.1. The Ribbon Menu

The Ribbon Menu (1) consists of buttons that correspond to main functions.

| | |
|---|---|
| ***Button Save*** | The Save button as the first button is for saving the current configuration. |
| ***Button Cancel*** | Pressing the next button, the Cancel button, the configuration mask is closed without saving the changes that have been made so far. |
| ***Button Action*** | The Action button contains the main configuration of the activity. Chapter 2.3 to 2.3.9 will cover various aspects in more detail. |
| ***Button Labels*** | Fourth, Labels allows users to define labels which will be displayed in specified places in the action. These labels are visible in the workflow itself and facilitate communication as things are more transparent. |
| ***Button Common*** | Under Common, a log message can be left. It will be listed in the workflow history if the workflow is over. In the section Expected duration, you can enter the time you expect this action to take until completion. This information is used for statistics and reports. |
| ***Button Variables*** | The button Variables enables users to define workflow variables. |
| ***Button Help*** | Clicking on the Help button, general information regarding SharePoint 2013 can be found. |

### 2.2.2. PowerShell Script

In the script dialog, you can define the PowerShell script you want to run at this point of your workflow.

This editor supports:

- PowerShell syntax highlighting
- Code-Completition (CTRL+Space)
- Line numbers
- Bracket matching
- Undo (CTRL+Z) / redo (CTRL+Y)

If you frequently use distinct scripts, you have the possibility to choose predefined modules. You can use these PowerModules by typing *Add-PowerModule '#ModuleName#'*. #ModuleName# is a place holder for the name you specified in the configuration of the DO4PS (Location: *Central Administration* -> *Data One* -> *Configure Data One 4 PowerShell*).

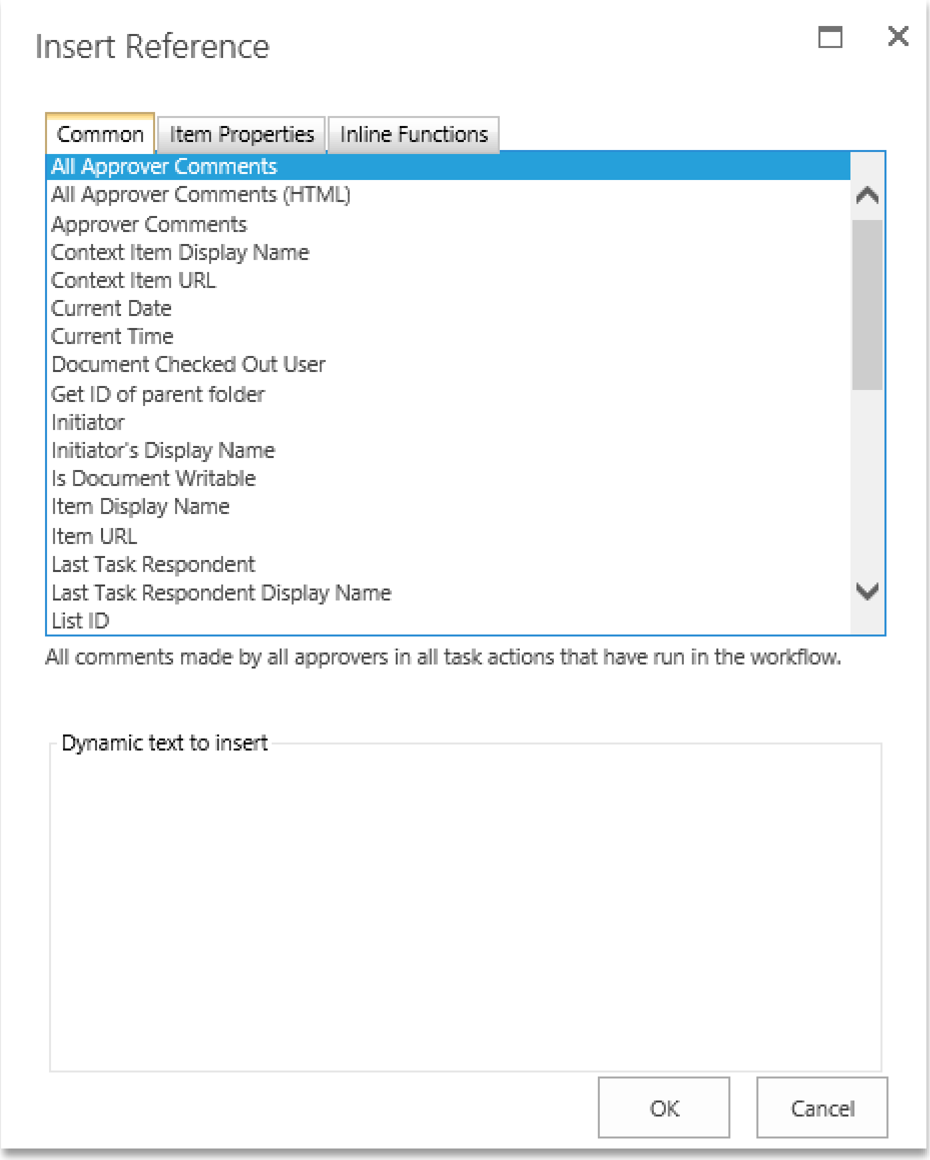
For more information regarding PowerModules, check the example in Chapter 2.3.4.

### 2.2.3. Insert Reference

Clicking on the Insert Reference button (2), you can choose some common variables, inline functions and workflow specific variables. To add one of these to the DO4PS, just double-click on it and press OK.

Insert Reference is not supported in User Defined Actions (UDAs).

**Security note: The usage of Insert Reference may potentially lead to an increased vulnerability towards script injection attacks. However, the usage of variable binding is secure.**



All comments made by all approvers in all task actions that have run in the workflow.

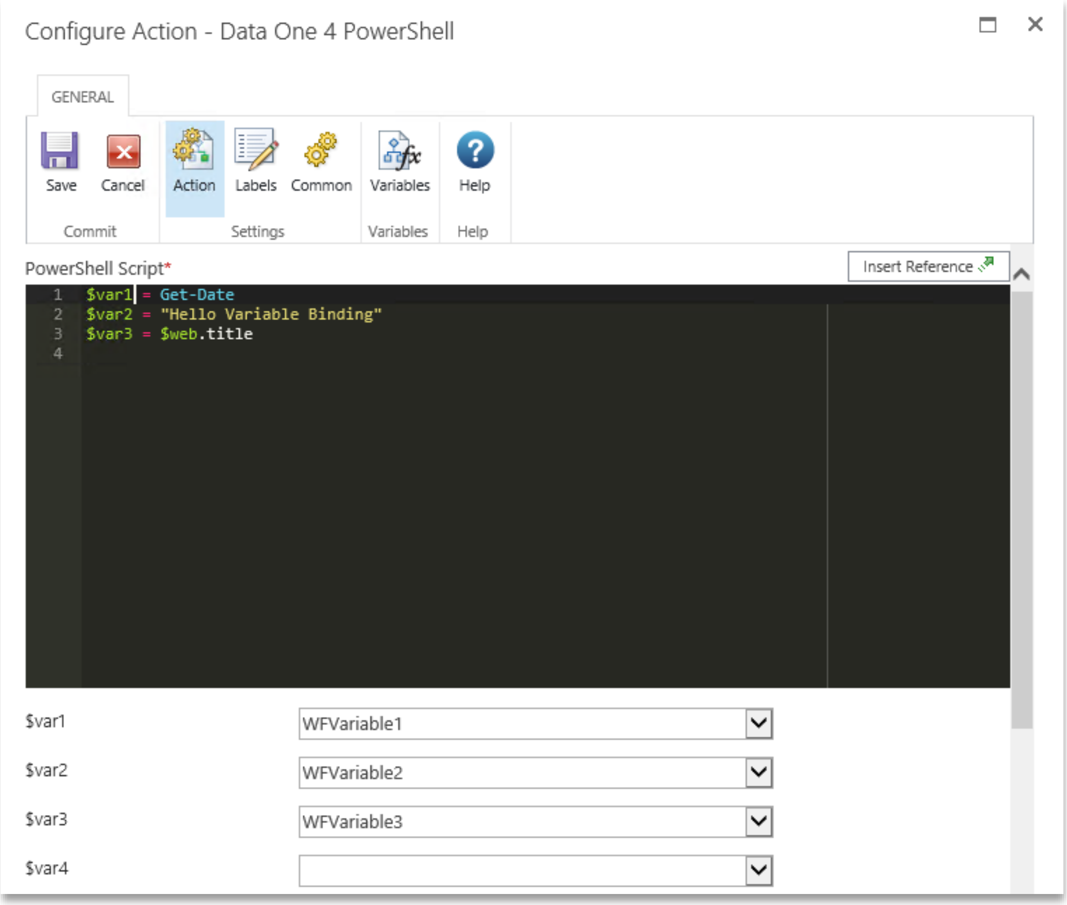
Figure 11: Insert Reference dialogue

### 2.2.4. Variables

Inside of DO4PS configuration, you can use a set of ten variables (4) to link your DO4PS variables to the global workflow variables. The values you assign must be serializable. If they are not, the workflow will fail. The variable binding is bidirectional, which means you can read and modify the values of the variable. In every dropdown field, you can choose the variable you specified before as a workflow variable.

**Example 1: Setting Workflow Variables within "Data One 4 PowerShell" action**

- $var1 = Get-Date
- $var2 = "Hello Variable Binding"
- $var3 = $web.Title



PowerShell Script*

```
$var1 = Get-Date
$var2 = "Hello Variable Binding"
$var3 = $web.title
```

$var1 WFVariable1

$var2 WFVariable2

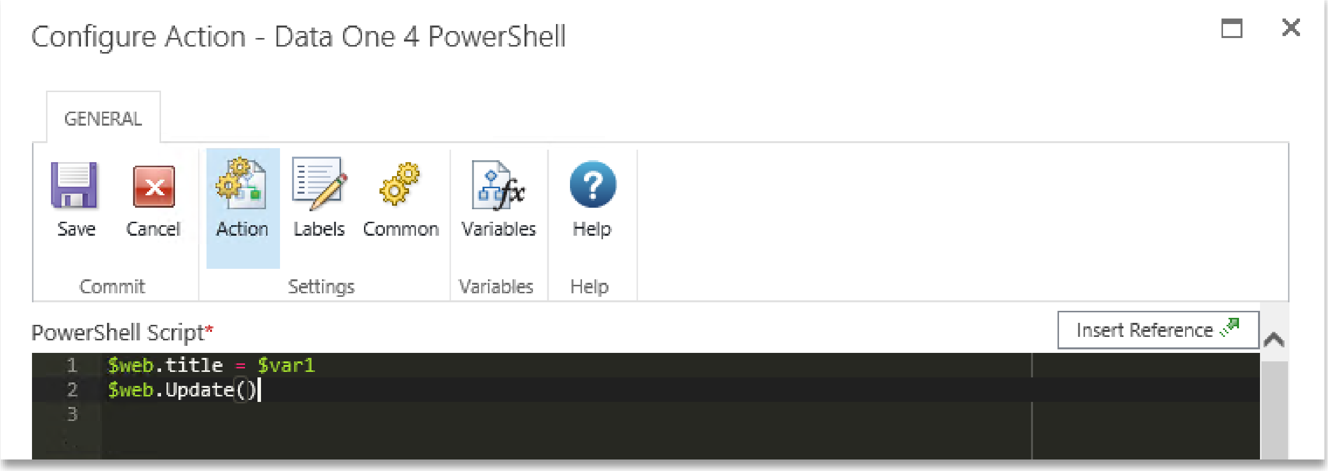
$var3 WFVariable3

$var4

Figure 12: Workflow example 1

**Example 2: Reading Workflow Variables**

- $web.Title = $var1
- $web.Update()



PowerShell Script*

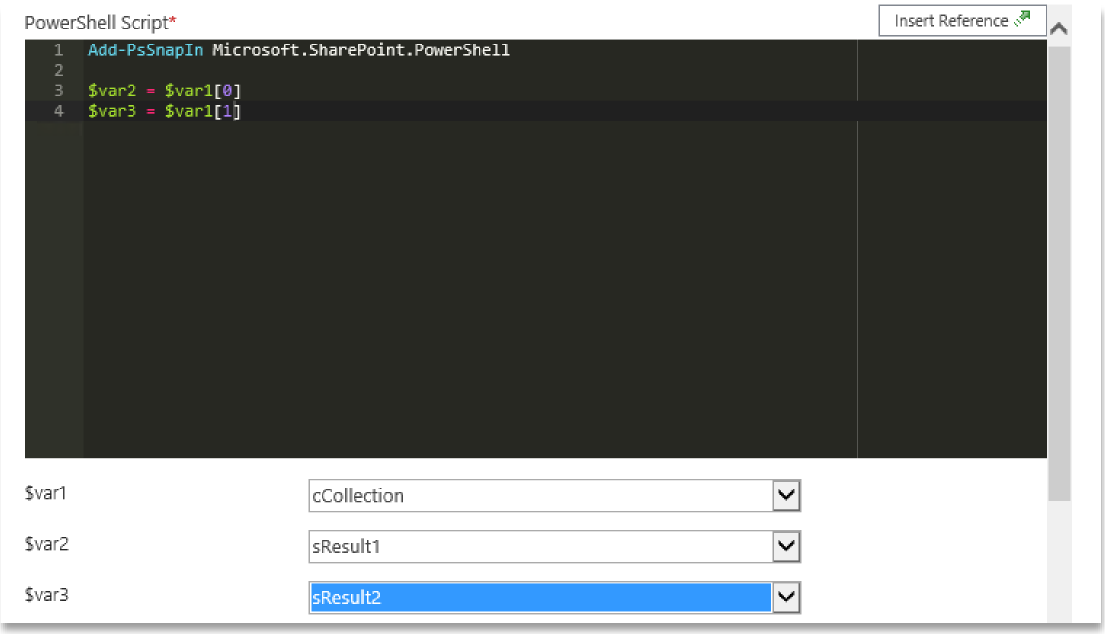
```
$web.title = $var1
$web.Update()
```

Figure 13: Workflow example 2

**Example 3: Collection Variables**

It is also possible to pass NintexWorkflow collection variables to a PowerShell Script. In this case you can assign a collection variable to $var1 to $var10. Within a PowerShell Script you can read those collection values.

For example if there are 2 values you can access them with $var1[0] and $var1[1]



PowerShell Script*

```
Add-PsSnapIn Microsoft.SharePoint.PowerShell

$var2 = $var1[0]
$var3 = $var1[1]
```

$var1 cCollection

$var2 sResult1

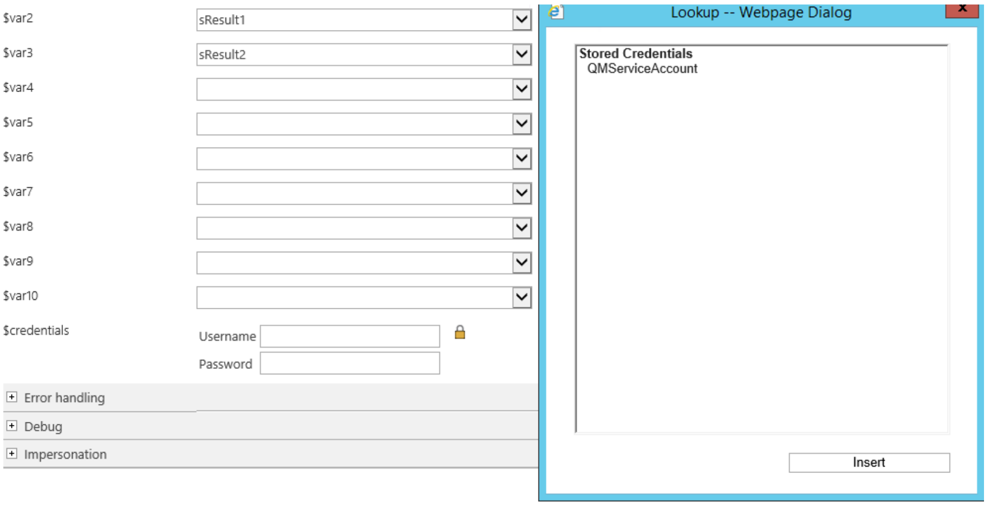
$var3 sResult2

## 2.2.5. Credentials

The credential variable (5) contains a mask for user credential input. If you wish to run a PowerShell script with special credentials, you can specify these there and use them inside of your script with the $credentials variable.

The password will not be shown as plain text.

By choosing the lock symbol, you can choose your stored credentials. Stored credentials are a Nintex feature. You can define them under *Central Administration* → *Nintex Workflow Management* → *Manage Workflow Constants*.



$var2 sResult1

$var3 sResult2

Stored Credentials
QMServiceAccount

Figure 14: Stored Credentials

### 2.2.6. Error Handling

The DO4PS error handling (6) is based on the Nintex error handling. It enables you to catch exceptions.

To use error handling correctly, define two workflow variables – one Boolean, i.e. *error bool,* and one string variable for the error message, i.e. *error_message*. Now, activate error handling by setting *Capture errors?* to Yes. For *Store error occurrence in*, choose a Boolean variable. For the error message, choose a string variable.

Now you can react to every error for example by using 'set a condition' action. You can check the Boolean value and determine consecutive steps from within your workflow.

### 2.2.7. Debugging

With the DO4PS Debug Mode (7), you are able to debug your DO4PS action as well as included scripts.

For the integration of the debug functionality, just check the box Debug Mode and enter the IP of the system on which the debug console runs. Start the debug console and push **ping** in order to test whether your debug console is available. If you receive a positive popup message and your debug console shows a message that ping has been received, your debug mode is correctly configured.

To use the debug console, start your workflow and switch to the system on which the debug console runs. For each PowerShell command, an individual prompt will appear. You have five different options:

| | |
|---|---|
| ***[Y] Yes*** | Yes is the default option in cases you press enter without input. The execution of the current command will be accepted |
| ***[A] Yes to all*** | This accepts all commands and no more questions will appear. |
| ***[N] No*** | By choosing the No option, the execution of the present command will stop and the next command starts. |
| ***[L] No to all*** | Similar to Yes to all, this option rejects all upcoming commands. |
| ***[S] Suspend*** | The last option is Suspend. This option enables you to interrupt the workflow. It opens a PowerShell command line prompt within the debug console. You exit the suspend mode through typing exit. In this mode, you have access to all workflow variables and can check their values. You also have access to all PowerShell commands. |

### 2.2.8. Predefined Variables

The following table contains the available variables for the element on which the workflow runs.

Avoid overwriting them as this can give rise to datatype problems.

**Security note: The variables are initialized within the scope of the user who starts the workflow. If you use impersonation, create a new object of the types SPSite, SPWeb etc. to get an additional instance.**

| *Variable* | | *List* | *Site* |
|---|---|---|---|
| *$site* | Microsoft.SharePoint.SPSite | X | X |
| *$web* | Microsoft.SharePoint.SPWeb | X | X |
| *$list* | Microsoft.SharePoint.SPList | X | |
| *$listItemService* | Microsoft.SharePoint.Workflow.IListItemService | X | X |
| *$taskService* | Microsoft.SharePoint.Workflow.ITaskService | X | X |
| *$sharePointService* | Microsoft.SharePoint.Workflow.ISharePointService | X | X |
| *$context* | Microsoft.SharePoint.WorkflowActions.WorkflowContext | X | X |
| *$item* | Microsoft.SharePoint.ListItem | X | |

Table 1: List of predefined variables

### 2.2.9. Impersonation

The last section in the DO4PS configuration dialogue (8) is impersonation. Here, you can overwrite the user credentials which are used to run the PowerShell script on the target system.

Sometimes, it may be necessary to run a script with credentials that are different from the standard ones. In this case, you can use impersonation to integrate these credentials into your workflow.

Note that the username should contain the domain name i.e. DOMAIN\username.

## 2.3. Features

### 2.3.1. Impersonation

After DO4PS is successfully installed, the administrator is able to define a default user which runs the embedded scripts. If the administrator doesn’t define an user, the system account will be used.

Sometimes, it may be necessary to have more or special rights to run a workflow. The impersonation feature allows you to overwrite the standard user of the workflow. In this case, you can define a user inside the DO4PS action configuration. For more information, see chapter 2.2.9.

### 2.3.2. User Defined Actions

You can use DO4PS actions in Nintex User Defined Actions (UDA) like every other Nintex action. To create an UDA that contains a DO4PS action, you have to be a DO4PS Designer. However, for using the UDA later on, you do not need to be a DO4PS designer.

Variable binding is not supported in UDA.

### 2.3.3. Debugging

DO4PS allows you to debug your scripts. You can interrupt your workflow. The debug console is helpful in detecting errors. It is possible to check variables and/or to set up any PowerShell command you want. For more information about the usage, see chapter 2.2.7.

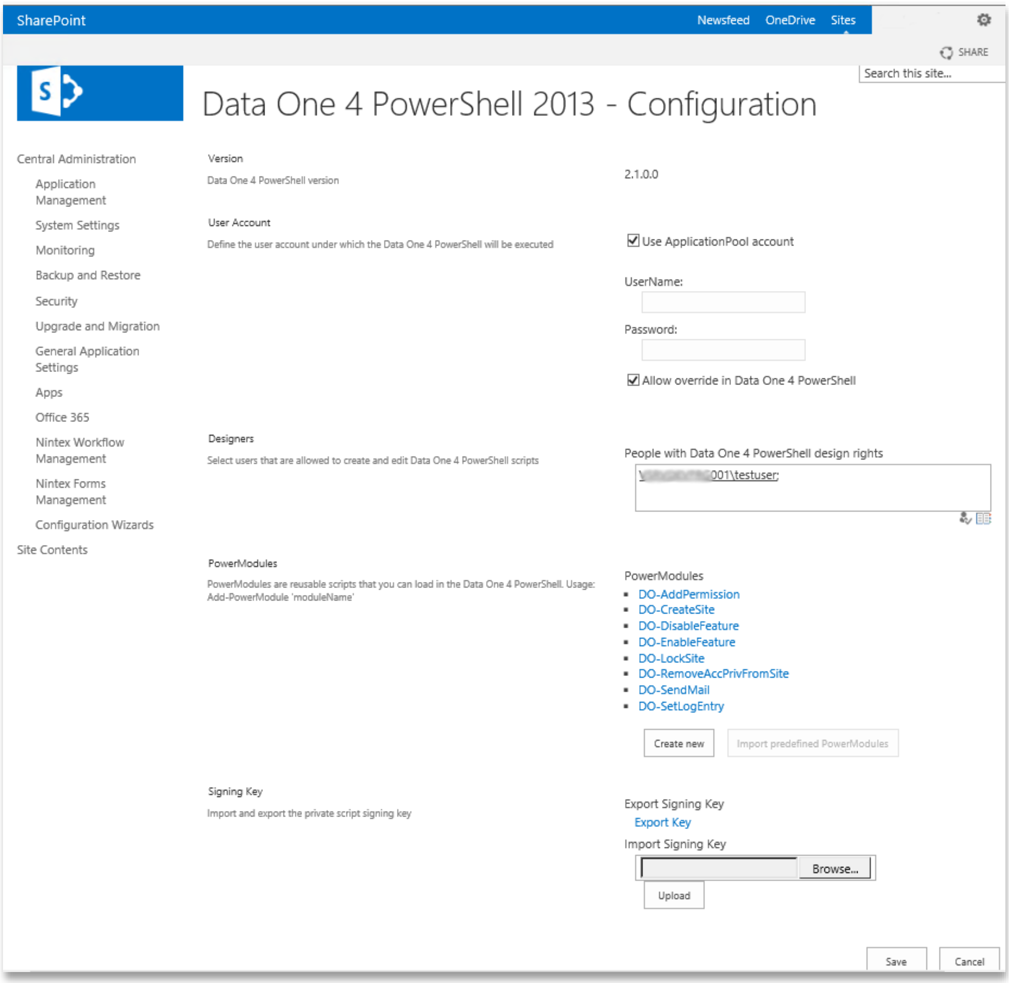
### 2.3.4. PowerModules

Data One 4 PowerShell provides the opportunity to define a set of modules that can be used inside the DO4PS script editor. This saves time and, besides, reduces errors.

The editor of these PowerModules has to define them in the Central Administration. You can find the PowerModules under the DO4PS configuration. As shown in Figure 15: Data One 4 PowerShell – Configuration you can see a list of present PowerModules there. If you want to use a PowerModule, it is necessary to know the name of the respective module.

In the DO4PS script mask, you can use this module by typing **Add-PowerModule 'moduleName'.** Please replace #ModuleName# with the name of the module in question.

It is not supported to use reference variables within PowerModules.



Data One 4 PowerShell 2013 - Configuration

Version
Data One 4 PowerShell version
2.1.0.0

User Account
Define the user account under which the Data One 4 PowerShell will be executed
☑ Use ApplicationPool account
UserName:
Password:
☑ Allow override in Data One 4 PowerShell

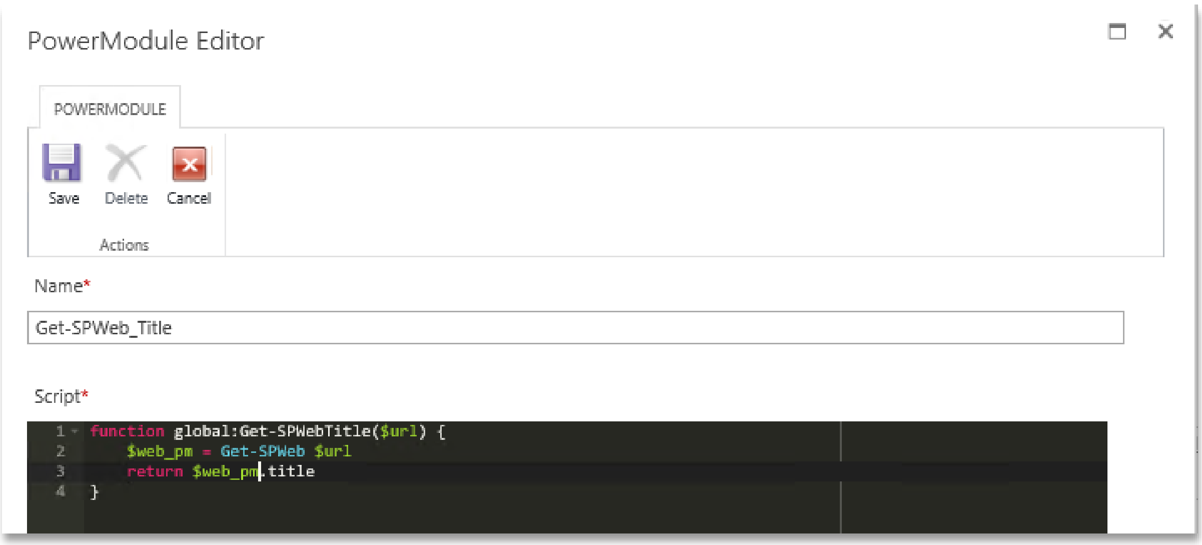
Designers
Select users that are allowed to create and edit Data One 4 PowerShell scripts
People with Data One 4 PowerShell design rights
[redacted]001\testuser;

PowerModules
PowerModules are reusable scripts that you can load in the Data One 4 PowerShell. Usage: Add-PowerModule 'moduleName'

PowerModules
- DO-AddPermission
- DO-CreateSite
- DO-DisableFeature
- DO-EnableFeature
- DO-LockSite
- DO-RemoveAccPrivFromSite
- DO-SendMail
- DO-SetLogEntry

Signing Key
Import and export the private script signing key
Export Signing Key
Export Key
Import Signing Key

Figure 15: Data One 4 PowerShell – Configuration



Name*

Get-SPWeb_Title

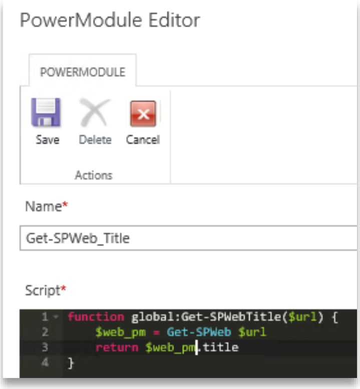
Script*

```
function global:Get-SPWebTitle($url) {
    $web_pm = Get-SPWeb $url
    return $web_pm.title
}
```

Figure 16: PowerModule editor

Example:

➔ Step 1: Adding a PowerModule in Central Administration -> DO4PS -> Configuration -> PowerModules



Name*

Get-SPWeb_Title

Script*

```
function global:Get-SPWebTitle($url) {
    $web_pm = Get-SPWeb $url
    return $web_pm.title
}
```

➔ Step 2: Using a PowerModule within DO4PS action

As soon as you have defined the PowerModule in CentralAdministration, you can call this PowerModule within your DO4PS action:

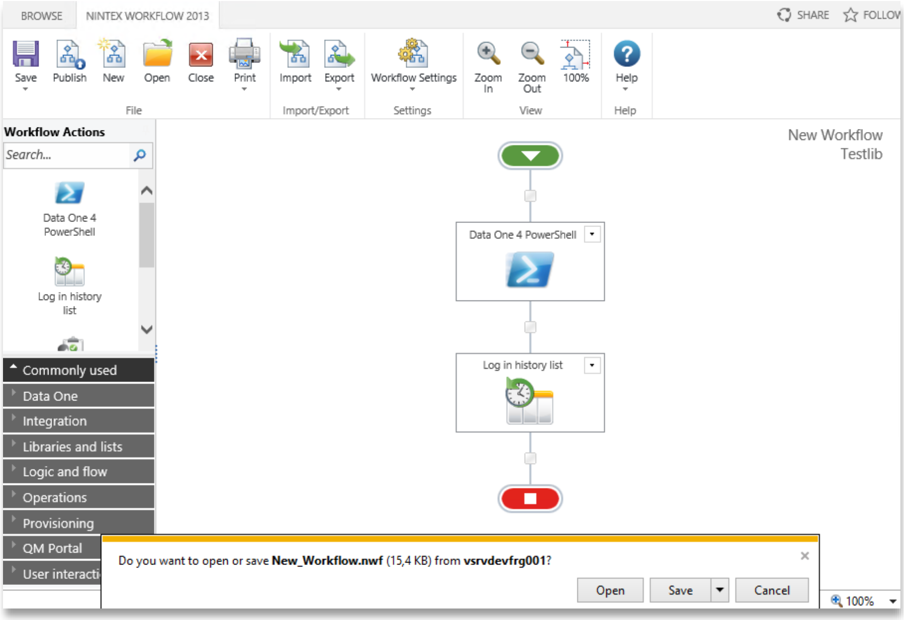
```
Add-PSSnapin Microsoft.Sharepoint.Powershell
Add-PowerModule 'Get-SPWeb_Title'
$var1 = Get-SPWebTitle {ItemProperty:URL}
```

### 2.3.5. Support for Site Workflows

Since version 2010 and up, Nintex Workflow 2013 supports site workflows. With this feature, it is possible to start workflows at site level. The DO4PS action for Nintex Workflow 2013 is also available at site level.

### 2.3.6. Import and Export

With “Data One 4 PowerShell 2013”, you can import and export workflows that contain a DO4PS action. In order to export a Nintex workflow that contains a DO4PS action, open the Nintex Workflow Designer. Open or create the workflow you want to export. In the ribbon, press the Export button.



Do you want to open or save **New_Workflow.nwf** (15,4 KB) from **vsrvdevfrg001**?

Figure 17: Exporting a Nintex Workflow which contains a Data One 4 Powershell action

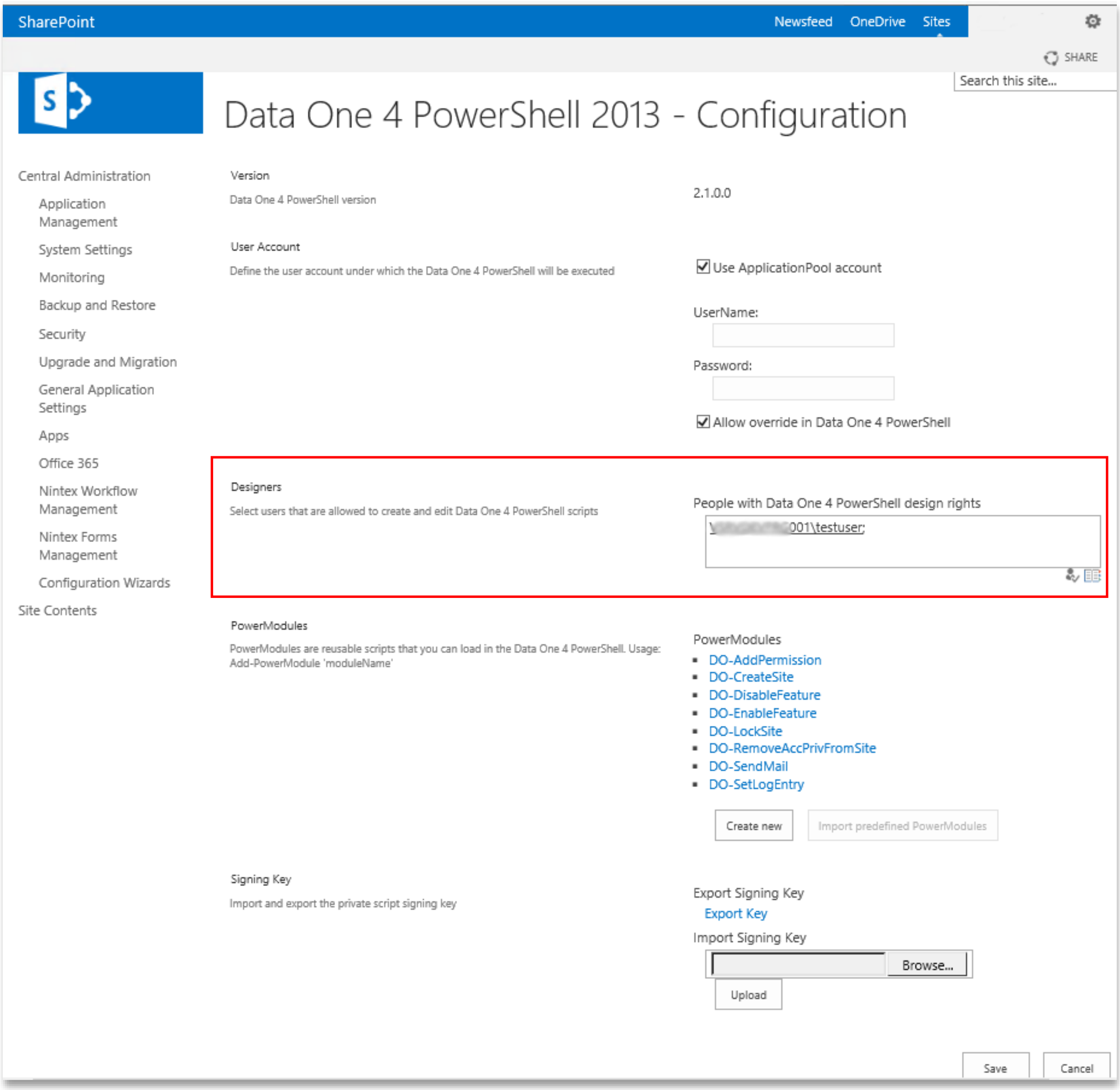
If you want to import a workflow, just press the import button and choose the *.nwf file (export file of a Nintex Workflow).

### 2.3.7. Defining “Data One 4 PowerShell” Users

Not every Nintex workflow designer has the necessary rights to use the DO4PS action correctly. Non-trusted users can drag the action on the workflow. Yet they are not able to make any configuration.

If you want to assign permissions to a user that enable this user to configure DO4PS action, you have to add this user to the Designers group. To do so, switch to *Central Administration* -> *Data One* -> *Data One 4 PowerShell 2013* -> *Configuration*.

In the section Designer, add the respective user to the list of designers.



SharePoint Newsfeed OneDrive Sites

SHARE

Search this site...

Data One 4 PowerShell 2013 - Configuration

Central Administration
- Application Management
- System Settings
- Monitoring
- Backup and Restore
- Security
- Upgrade and Migration
- General Application Settings
- Apps
- Office 365
- Nintex Workflow Management
- Nintex Forms Management
- Configuration Wizards

Site Contents

**Version**
Data One 4 PowerShell version

2.1.0.0

**User Account**
Define the user account under which the Data One 4 PowerShell will be executed

☑ Use ApplicationPool account

UserName:

Password:

☑ Allow override in Data One 4 PowerShell

**Designers**
Select users that are allowed to create and edit Data One 4 PowerShell scripts

People with Data One 4 PowerShell design rights
[redacted]001\testuser;

**PowerModules**
PowerModules are reusable scripts that you can load in the Data One 4 PowerShell. Usage: Add-PowerModule 'moduleName'

PowerModules
- DO-AddPermission
- DO-CreateSite
- DO-DisableFeature
- DO-EnableFeature
- DO-LockSite
- DO-RemoveAccPrivFromSite
- DO-SendMail
- DO-SetLogEntry

Create new | Import predefined PowerModules

**Signing Key**
Import and export the private script signing key

Export Signing Key
Export Key
Import Signing Key
Browse...
Upload

Save | Cancel

Figure 18: Granting “Data One 4 PowerShell” permissions

### 2.3.8. Using Nintex Error Handling

Within DO4PS action for Nintex Workflow 2013, you can use Nintex Workflow error handling. To achieve this, you have to make the relevant configurations inside of the DO4PS action. For more details, see Chapter 2.2.6.

### 2.3.9. Predefined Variables

The DO4PS action for Nintex Workflow 2013 was delivered with some predefined variables. Please check chapter 2.2.8 for a list of available variables. Always keep in mind that there are differences between list and site workflows due to the different structural levels and scopes.

# 3. PowerModules Documentation

There are some PowerModules integrated with Data One 4 PowerShell. Please refer to chapter 2.3.4 for a detailed description on how to use those PowerModules. The following chapter describes the integrated PowerModules.

You install them by clicking on “Import predefined PowerModules” on the configuration page in Central Administration.

## 3.1. PowerModule DO-AddPermissionClaims

*Description*

This function adds a permission level to a given user or user group in a SharePoint web. You can grant permissions to either a single user or group at one time. It is recommended that the webapplication uses claims.

*Parameter*

| PARAMETER | MANDATORY | DESCRIPTION |
|---|---|---|
| **WEBURL** | YES<br>[string] | This parameter defines the address of the SharePoint Web in which you want to add permissions. |
| **PERMLEVEL** | YES<br>[string] | Defines the permission level which you want to grant the user or user group. You must provide the display name of the wanted permission level.<br>By default SharePoint delivers the following permission levels:<br>View Only, Limited Access, Read, Contribute, Edit, Design, Full Control<br>Please take care of localized SharePoint descriptions.<br>For further information please see:<br>https://technet.microsoft.com/en-us/library/cc721640.aspx |
| **GROUPNAME** | NO<br>[string] | This string must contain the name of the SharePoint group you want to authorize. |
| **USERNAME** | NO<br>[string] | Name of the user to add (example domain\user). |

*Example*

Grant Full Control permissions to a user and a SharePoint group:

```
Add-PSSnapin Microsoft.Sharepoint.Powershell

Add-PowerModule 'DO-AddPermission'

DO-AddPermission -webUrl "http://sharepointUrl/" -permLevel "Full Control" -groupName "TestGroup"

DO-AddPermission -webUrl "http://sharepointUrl/" -permLevel "Full Control" -userName "domain\usr"
```

## 3.2. PowerModule DO-CreateSite

*Description*

This function creates a new SharePoint Site Collection by given parameters.

### *Parameter*

| PARAMETER | MANDATORY | DESCRIPTION |
|---|---|---|
| **URL** | YES<br>[string] | This parameter defines the wanted address of the new SharePoint Site Collection. |
| **OWNER** | YES<br>[string] | Defines the login name of the user which should be the owner of the newly created Site Collection. |
| **NAME** | NO<br>[string] | This is the name of the newly created Site Collection. |
| **TEMPLATENAME** | NO<br>[int] | The identifier of a defined SharePoint Site Template.<br>You find a list of Site Templates here:<br>http://www.funwithsharepoint.com/sharepoint-2013-site-templates-codes-for-powershell/<br>If there is no template defined, SharePoint suggests you a list of templates when opening the new site for the first time. |

### *Example*

Create a new Team Site with title “Created by Data One 4 PowerShell”:

```
Add-PSSnapin Microsoft.Sharepoint.Powershell

Add-PowerModule 'DO-CreateSite'

DO-CreateSite -URL "http://sharepointUrl/sites/newSite"-owner "domain\user_name"
-templateName "STS#0" -Name "Created by Data One 4 PowerShell"
```

## 3.3. PowerModule DO-DisableFeature

### *Description*

This function allows you to disable a SharePoint feature. Features can be of scope “Web” or “Site”.

It is useful because disabling features is by default not possible in a Data One 4 PowerShell script.

### *Parameter*

| PARAMETER | MANDATORY | DESCRIPTION |
|---|---|---|
| **SCOPE** | YES<br>[string] | This specifies the scope of the feature you want to disable.<br>Allowed values are “Web” and “Site”. |
| **URL** | YES<br>[string] | Defines the address of the SharePoint Web or Site in which you want to disable the specified feature. |
| **FEATUREID** | YES<br>[string] | This is the global identifier of the specified feature which was converted to text. |

### *Example*

Disable a feature in a given SharePoint Web:

```
Add-PSSnapin Microsoft.Sharepoint.Powershell

Add-PowerModule 'DO-DisableFeature'

DO-DisableFeature -scope "Web" -URL "http://sharepointUrl/subWeb/"
-featureId "12345678-e545-4d32-897a-bab6f5846e18"
```

### 3.4. PowerModule DO-EnableFeature

*Description*

This function allows you to enable a SharePoint feature. Features can be of scope “Web” or “Site”.

It is useful because enabling features is by default not possible in a Data One 4 PowerShell script.

*Parameter*

| PARAMETER | MANDATORY | DESCRIPTION |
|---|---|---|
| **SCOPE** | YES<br>[string] | This specifies the scope of the feature you want to enable.<br>Allowed values are “Web” and “Site”. |
| **URL** | YES<br>[string] | Defines the address of the SharePoint Web or Site in which you want to enable the specified feature. |
| **FEATUREID** | YES<br>[string] | This is the global identifier of the specified feature which was converted to text. |

*Example*

Enable a feature in a given SharePoint Site:

```
Add-PSSnapin Microsoft.Sharepoint.Powershell

Add-PowerModule 'DO-EnableFeature'

DO-EnableFeature -scope "Site" -URL "http://sharepointUrl/"
-featureId "12345678-e545-4d32-897a-bab6f5846e18"
```

### 3.5. PowerModule DO-LockSite

*Description*

This function gives you the possibility of easily locking a SharePoint Site, or to unlock it again.

*Parameter*

| PARAMETER | MANDATORY | DESCRIPTION |
|---|---|---|
| **SITEURL** | YES<br>[string] | This parameter defines the address of the SharePoint Site which you want to edit. |
| **UNLOCK** | NO<br>[bool] | If you omit this parameter the specified Site will be locked.<br>Deliver ‘$true’ for this parameter to unlock the Site again. |

*Example*

Lock and unlock a SharePoint Site:

```
Add-PSSnapin Microsoft.Sharepoint.Powershell

Add-PowerModule 'DO-LockSite'

DO-LockSite -siteUrl "http://sharepointUrl/"
DO-LockSite -siteUrl "http://sharepointUrl/" -unlock $true
```

## 3.6. PowerModule DO-RemoveAccPrivFromSite

*Description*

This function removes userprivileges for a given site. It is possible to skip the root web. Privileges will only be removed from all webs included in this site.

*Parameter*

| PARAMETER | MANDATORY | DESCRIPTION |
|---|---|---|
| **SITEURL** | YES<br>[string] | This parameter should contain the url of the sitecollection<br>For example: http://server/site/sitenamex |
| **ACCOUNTNAME** | YES<br>[string] | This parameter should contain the username to delete the privileges |
| **SKIPROOTSITE** | NO<br>[boolean] | If you skrip the root site, the privileges are deleted on all subwebs only<br>Default is $false |

*Example*

Remove privileges for “domain\account” on http://server/sites/site

```
Add-PSSnapin Microsoft.Sharepoint.Powershell

Add-PowerModule 'DO-RemoveAccPrivFromSite'

DO-SetLogEntry -siteURL "http://server/sites/site" -accountName "domain\account" -skipRootSite
$false
```

## 3.7. PowerModule DO-SendMail

*Description*

This function allows you to send emails during the runtime of your Data One 4 PowerShell script. The PowerModule uses SPUtilitySendMail.

This method supports sending email messages only to valid SharePoint user email addresses.

Please refer to:

https://msdn.microsoft.com/EN-US/library/office/microsoft.sharepoint.utilities.sputility.sendemail.aspx

*Parameter*

| PARAMETER | MANDATORY | DESCRIPTION |
|---|---|---|
| **WEBURL** | YES<br>[string] | An Web Object that represents the site |
| **TO** | YES<br>[string] | This parameter specifies the email address of the person to which the mail should be sent. SendMail supports sending email messages to valid SharePoint user email addresses. |
| **BODY** | YES<br>[string] | This is the text message of the email. You should enter a plain text. |
| **SUBJECT** | YES<br>[string] | This is the subject of the new email. |
| **APPENDHTMLTAG** | NO<br>[boolean] | TRUE to append an HTML tag to the message; otherwise false |
| **HTMLENCODE** | NO<br>[boolean] | TRUE to encode the message and replace characters in HTML Tags with entities; otherwise false |

This function returns $true, if SendMail was able to send the mail. Otherwise it returns $false

*Example*

Send an email:

```
Add-PSSnapin Microsoft.Sharepoint.Powershell

Add-PowerModule 'DO-SendMail'

$result_mail = DO-SendMail -webUrl "http://weburl.com/sites/xyz" -to "mailadress@xyz1123.com" -
body "Test" -subject "Test"
```

## 3.8. PowerModule DO-SetLogEntry

*Description*

This function creates or updates a logfile. The developer can define the destination file, LogLevel, DateTime and a text to log.

To use this function you have to add the SharePoint-Snapin to your script first.

*Parameter*

| PARAMETER | MANDATORY | DESCRIPTION |
|---|---|---|
| **LOGFILE** | YES<br>[string] | This parameter defines where to save the logfile. The user executing the PowerShell Script must have the privileges to write this file. If this file does not exists, it will be created.<br>For example D:\Logs\logfile.txt |
| **LOGFORMAT** | NO<br>[string] | Defines the Log-Date and Log-Time format contained in your logfile. If you want to use a different format please refer to:<br>https://technet.microsoft.com/en-us/library/ee692801.aspx |

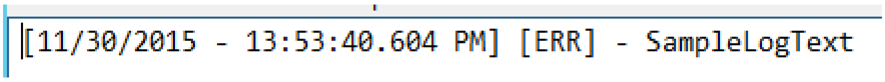
| | | |
|---|---|---|
| | | Default value is: MM/dd/yyyy - HH:mm:ss.fff tt |
| **LOGLEVEL** | YES<br>[string] | This string must contain a loglevel of your choice like ERR, WRN, INF. |
| **LOGSTRING** | YES<br>[string] | This string contains the information or errormessage itself and will be added to the logfile. |

### *Example*

Write “SampleLogText” to PowerMod1.log with loglevel “ERR”:

```
Add-PSSnapin Microsoft.Sharepoint.Powershell

Add-PowerModule 'DO-SetLogEntry'

DO-SetLogEntry -LogFile "D:\log\PowerMod1.log" -LogLevel "ERR" -LogString "SampleLogText"
```

This is a script logentry example:



```
[11/30/2015 - 13:53:40.604 PM] [ERR] - SampleLogText
```

# 4. Frequently Asked Questions

## 4.1. When Starting the Workflow, I Get an Error Message that Reads: Invalid Script Signature.

You get this error message if you migrate and start a Nintex Workflow that contains a DO4PS component from one system to another such as for example from development to production or from 2010 to 2013. In so doing, you have probably forgotten to transfer the signing key from source to target system. For more information, see chapter 1.6.2.

Depending on your requirements, there are different ways to approach this problem:

*If there is currently NO OTHER workflow deployed on target system that contains DO4PS*:

> If your target system already contains workflows with DO4PS actions and these workflows have already been published on the target system at an earlier point in time, you have to republish them after importing the new key. In this scenario, the number of signing keys, is limited to a single key.
>
> Go to source system *Central Administration* -> *Data One* -> *Configure Data One 4 PowerShell*-> *Signing Key* and click Export Key to save the signing key.
>
> Go to target system *Central Administration* -> *Data One* -> *Configure Data One 4 PowerShell*-> *Signing Key* and click Upload. Choose the source key and upload it.

*If you want to use different signing keys and want to republish your workflow on target system:*

> If you want to use different signing keys for source and target system, you have to republish the workflow on the target system. This needs to be done once per import of this workflow from source to target system.

If you migrate from SharePoint 2010 to 2013 it is strongly recommended to first export the signature key from the source system (2010) and import it into the target system (2013) – after configuring DO4PS and before importing or creating any workflow that contains DO4PS actions on the target system.

## 4.2. Windows User Access Control: Requested Registry Access is not allowed

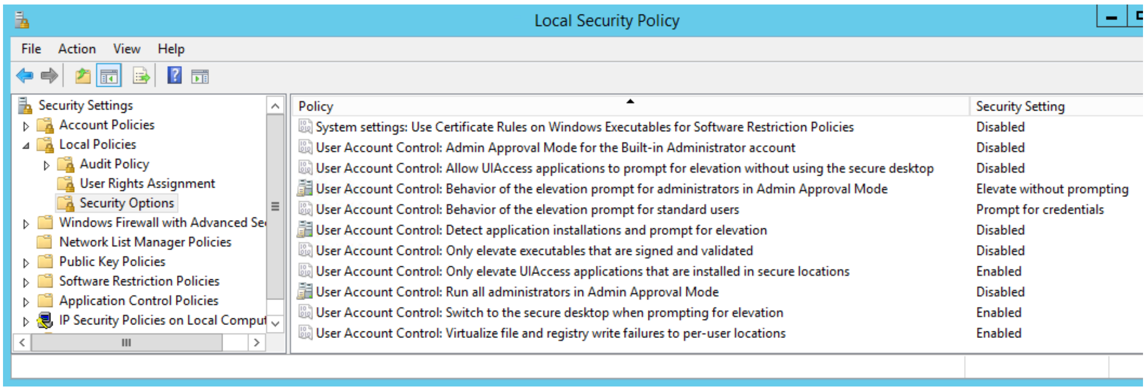
**Reason 1 – Windows UAC**

Depending on the CMDLETS you use, this error occurs. It is for example likely to do so for New-SPSite.

This message indicates that Windows UAC blocks the execution of the command.

If you want to get a distinct statements working without deactivating UAC, use the security policies:

Open the Local Security Policy Editor (secpol.msc) and set the UserAccountControl as displayed below:


Local Security Policy
File
Action
View
Help
Security Settings
Account Policies
Local Policies
Audit Policy
User Rights Assignment
Security Options
Windows Firewall with Advanced Se
Network List Manager Policies
Public Key Policies
Software Restriction Policies
Application Control Policies
IP Security Policies on Local Comput

| Policy | Security Setting |
|---|---|
| System settings: Use Certificate Rules on Windows Executables for Software Restriction Policies | Disabled |
| User Account Control: Admin Approval Mode for the Built-in Administrator account | Disabled |
| User Account Control: Allow UIAccess applications to prompt for elevation without using the secure desktop | Disabled |
| User Account Control: Behavior of the elevation prompt for administrators in Admin Approval Mode | Elevate without prompting |
| User Account Control: Behavior of the elevation prompt for standard users | Prompt for credentials |
| User Account Control: Detect application installations and prompt for elevation | Disabled |
| User Account Control: Only elevate executables that are signed and validated | Disabled |
| User Account Control: Only elevate UIAccess applications that are installed in secure locations | Enabled |
| User Account Control: Run all administrators in Admin Approval Mode | Disabled |
| User Account Control: Switch to the secure desktop when prompting for elevation | Enabled |
| User Account Control: Virtualize file and registry write failures to per-user locations | Enabled |

Of course, it is also possible to deactivate UAC to get rid of this error. To deactivate the Windows UAC, set the registry key to 0 following the instruction below:

- Navigate to the following registry subkey:
- HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System
- In the details pane (Right Pane), locate the EnableLUA key (REG_DWORD type). Double-click on Modify.
- In the data box Value, type 0 (zero). Click OK.
- Exit the registry editor.
- Restart the computer.

Should the problem still exist: Grant the WSS_Admin_WPG Group read access with regards to the following registry key:

HKEY_Users\<userid>\Environment

<userid> is the ID of the concerned application pool account.

Depending on the configuration of the farm, it can be necessary to assign read permissions to different users and/or for further keys as well. In this respect, the decisive factor the user context in which the workflow runs.

**Reason 2 – ApplicatonPool Account deviates from FarmAdmin Account**

The error message “Registry access is not allowed” can also happen if you use an ApplicationPoolAccount that is different from your farm admin. Generally, it is recommended that the two accounts are different from one another. As mentioned in 1.6.2, by default, DO4PS actions are executed as ApplicationPoolAccount. Consequently, your ApplicationPoolAccount is not apt to execute CMDLETS. Make sure your ApplicationPool Account has enough privileges. Your scripts should be run with this user account if you do not use another account.

1) Add the SPShell Admin for content DB to your AppPoolAccount:

   Example: `Add-SPShellAdmin -UserName -database #GUID_OF_YOUR_CONTENT_DBs#`

2) Add the SPShell Admin to SP-Config DB to your AppPoolAccount:

   ```
   Get-SPDatabase | Where-Object {$_.WebApplication -like "SPAdministrationWebApplication"} | Add-SPShellAdmin DOMAIN\Username
   ```

   In the example above, a new user named User1 is added to the SharePoint_Shell_Access role in both – the specified Central Administration content database and the configuration database.

3) Your ApplicationPool Account should be local admin on SharePoint-Farm servers.

4) To make these changes pervasive, open shell and execute an iisreset.

## 4.3. Using PowerShell with a Remote Access

- Configure Windows Remote Management:

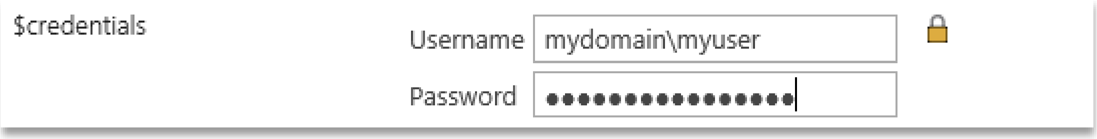
http://msdn.microsoft.com/en-us/library/aa384372%28v=VS.85%29.aspx.

- Ensure that the remoting works as expected with the PowerShell console.
- DO4PS action has to run as the System Account (Central Administration -> Data One -> Configure Data One 4 PowerShell -> User system account (checked). Impersonation does not work. Besides, you have to use explicit credentials to open a remote session. The following example opens a remote session to myserver, invokes a commando and closes the session.

```
# generate a secure password from the credentias variable
$password = convertto-securestring -String $credentials.Password -AsPlainText -force


# create the PowerShell credentials that you need for New-PSSession
$remoteCred  =  new-object  System.Management.Automation.PSCredential($credentials.UserName,
$password)


# open a session to myserver using the credentials from above
$session = New-PSSession -ComputerName myserver -Credential $remoteCred


# invoke a command on myserver and pass in the website title as argument
$result = Invoke-Command -Session $session -ArgumentList $web.Title -ScriptBlock {
    param($param1)

   # simply returns hello website title
    return "Hello " + $param1

}

# log the result to the workflow history log

$result | out-host


# close the session
Remove-PSSession $session
```

The variables that you pass to and return from the remote commands have to be serializable.

The credentials are defined here:



$credentials

Username mydomain\myuser

Password ●●●●●●●●●●●●●●●●

Figure 19: $credentials

## 4.4. Best Practice: How to Update List Items

Avoid updating list items in the following way:

```
$item["Title"] = "Test"

$item.SystemUpdate()
```

This script could cause an infinite loop because updating (even SystemUpdate()) will trigger the event receivers. Using the script from above in a workflow that automatically starts as soon as an item is changed is likely to give rise to an endless loop. To safely update a list item, use the following script instead:

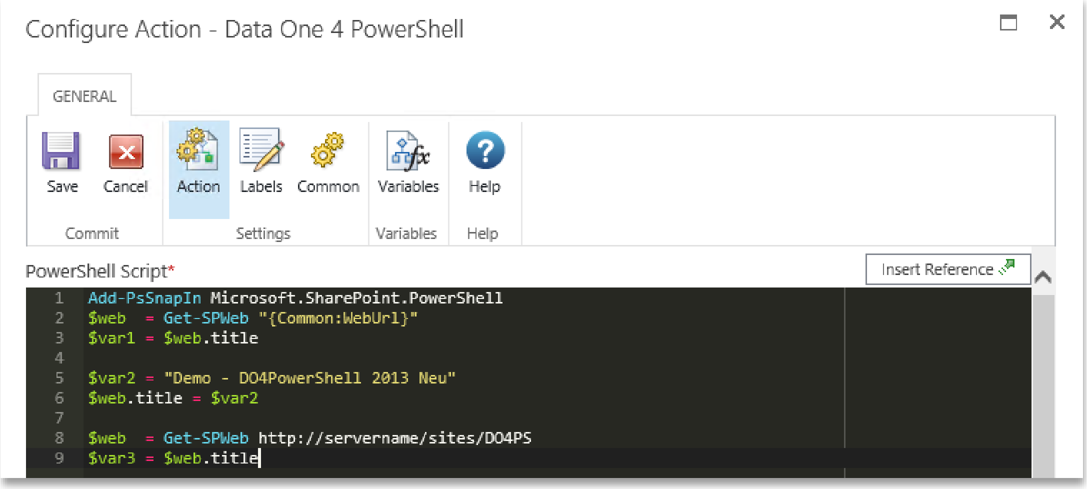
$this.UpdateListItem($item, @{"Title" = "Test"; "SecondFieldName"="Second Value"})

As far as the field name is concerned, you are free to use either the title or the internal name of the field.

## 4.5. Using Insert Reference Values

If you use Insert Reference values such as {Common:WebURL}, be aware of the fact that they are replaced by the respective content before a DO4PS activity starts.

Example: *$web=Get-SPWeb "{Common:WebUrl}"* will be replaced and executed as *$web=Get-SPWeb "http://.. "*



```
Add-PsSnapIn Microsoft.SharePoint.PowerShell
$web  = Get-SPWeb "{Common:WebUrl}"
$var1 = $web.title

$var2 = "Demo - DO4PowerShell 2013 Neu"
$web.title = $var2

$web  = Get-SPWeb http://servername/sites/DO4PS
$var3 = $web.title
```

Figure 20: Using insert reference values

## 4.6. Best Practice: How to Access SharePoint Lists

Avoid using *$item["Column"]* to get item values. Because you have to use the display name if you do so.

Instead you should use a CAML-Query and within the CAML-Query use the internal name of your column:

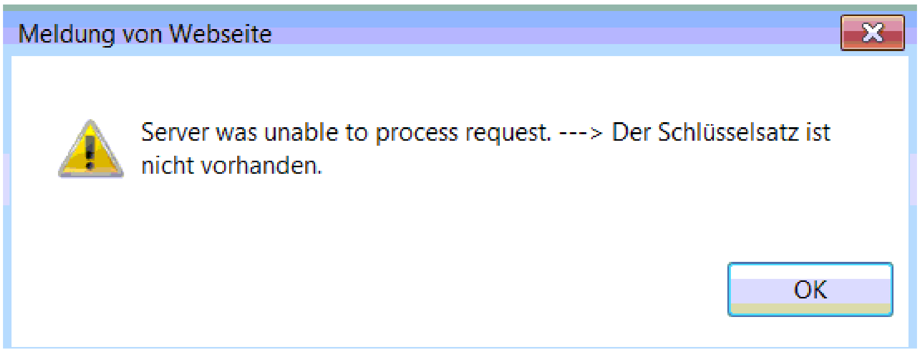
```
$web=Get-SPWeb -Identity

$spWeb = Get-SPWeb -Identity '###YOURWEB###'

$spList = $spWeb.Lists['###YOURLIST###']
```

```
$spQuery = New-Object Microsoft.SharePoint.SPQuery

$spViewFields= '<FieldRef Name="ID" /><FieldRef Name="Keywords" />'

$spQuery.ViewFields=$spViewFields

$camlQuery = '<Where><Eq><FieldRef Name="ID"/><Value Type="Integer">####VALUE#####</Value></Eq></Where>'

$spQuery.Query = $camlQuery

$spListItems = $spList.GetItems($spQuery)
```

## 4.7. Potential Errors when Saving or Publishing a Workflow

### 4.7.1. Key set does not exist / Schlüsselsatz nicht vorhanden

DO4PS uses the keys store of the machine to save the signing key which is necessary to sign the power shell scripts. The error message shown in figure 21 is received, if the permissions for the registry entry are wrong.



Server was unable to process request. ---> Der Schlüsselsatz ist nicht vorhanden.

Figure 21: Key set error

If you get this message, take the following steps:

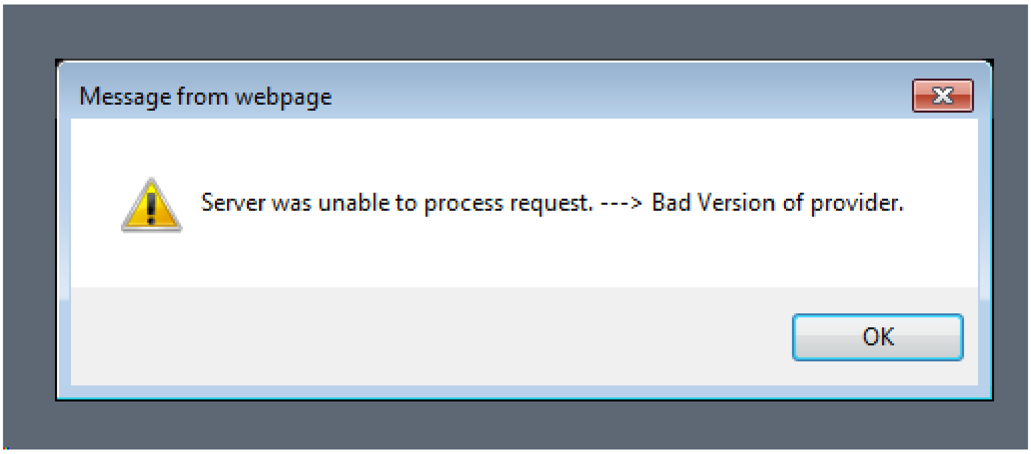
(1) Go to the folder: **c:\Programm Data\Microsoft\Crypto\RSA\MachineKeys**

(2) Assign the permissions **read, write, modify** to the group **Everyone**.

### 4.7.2. Bad Version of Provider

DO4PS uses the keys store of the machine to save the signing key which is necessary to sign the PowerShell scripts.

The error "Bad version of provider" occurs if the signing key does not meet the respective requirements and/or is ill-defined such as chosen too long.



Server was unable to process request. ---> Bad Version of provider.

Figure 22: Bad Version of provider

If you get this message, take the following steps:

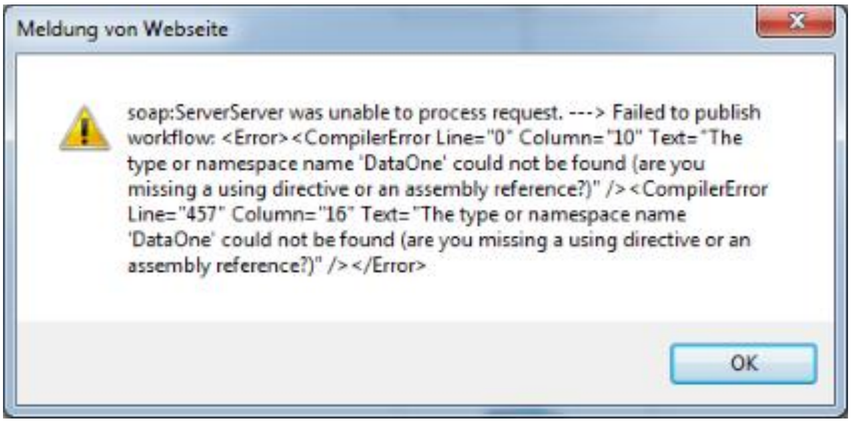
(1) Send a mail to the Data One Customer Care Center. Indicate the exported signing key.

(2) You will get a new signing key. Import this key.

### 4.7.3. Namespace DataOne could not be found

Did you activate the “Data One 4 PowerShell” WebApplication-Feature for each WebApplication you are using the activity? If you don’t activate the feature, you will get this error.

If you still receive the error message below, check the web.config file of your WebApplication where you activated the feature. You have to check this on every WFE-Server in your farm.



soap:ServerServer was unable to process request. ---> Failed to publish workflow: <Error><CompilerError Line="0" Column="10" Text="The type or namespace name 'DataOne' could not be found (are you missing a using directive or an assembly reference?)" /><CompilerError Line="457" Column="16" Text="The type or namespace name 'DataOne' could not be found (are you missing a using directive or an assembly reference?)" /></Error>

Figure 23: DataOne namespace is missing

If the following entry is missing in the section <authorizedTypes>, insert it:

```
<authorizedType Assembly="DataOne.Nintex.PowerActivity, Version=1.0.0.0, Culture=neutral, PublicKeyToken=7d3c850701dacb81" Namespace="DataOne.Nintex" TypeName="PowerActivity" Authorized="True" />
```

If you use more than one WFE server, you might also get the errormessage “Failed on Start” – “Workflow was cancelled by system account”. In this case please also validate the web.config on each WFE Server.

## 4.8. The Local Farm is not accessible

You can receive this error message even if you start DO4PS action with an account that belongs into the group of farm Administrators. This error occurs because you do not have the required rights to execute commands. In the past, you have probably changed the user that executes DO4PS scripts (1.6.2 Settings).

You change these rights by using the *Add-SPShellAdmin* cmdlet.

Syntax: *Add-SPShellAdmin -UserName Domain\User -database DBGUID*

You find additional information at: https://technet.microsoft.com/en-us/library/ff607596.aspx

Again, keep in mind that, by default, DO4PS action are executed as ApplicationPoolAccount and not as FarmAdmin.

## 4.9. System.Management.Automation.SessionStateScope Threw an Exception

This error occurs if CAS support is enabled. The usage of DO4PS presupposes that CAS is disabled.

If you want to use DO4PS and do not use legacy SP2010 mode, please set legacy CAS to false:

- Web.config

  The web.config file of each WebApplication contains a line which reads LegacyCasModel. In this line, replace true by false.

  ```
  <trust legacyCasModel="false"/>
  ```

- OWSTIMER.EXE.CONFIG

  The owstimer.exe.config contains such a line as well. Again, replace true by false. If not, you will get this error when using a pause action.

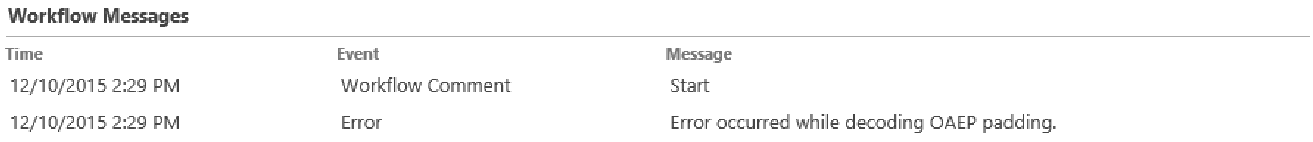
  ```
  <NetFx40_LegacySecurityPolicy enabled="false" />
  ```

  Default folder:

  C:\Program Files\Common Files\microsoft shared\Web Server Extensions\15\BIN\OWSTIMER.EXE.CONFIG

## 4.10. Error occurred while decoding OAEP padding

This error occurs if you use $credentials within your DO4PS component and the workflow was deployed with a different DO4PS signing key. For security reasons the $credentials variable is secured with your signing key.



**Workflow Messages**

| Time | Event | Message |
|---|---|---|
| 12/10/2015 2:29 PM | Workflow Comment | Start |
| 12/10/2015 2:29 PM | Error | Error occurred while decoding OAEP padding. |

Please export your signing key if you migrate to another environment if you don’t want to reenter $credentials and impersonation and if you don’t want to deploy the workflows again.

For migration please refer to 4.1

# 5. Data One 4 PowerShell 2013 – Known Limitations

## 5.1. Get-SPWebApplication or Get-SPFarm

You cannot directly use this CMDLET’s within DO4PS. You will get a message that says that you need FarmAdmin access. Even if you have these access privileges, you will still get this message.

Instead you can use the SharePoint API.

- SharePoint API for Farm object:

```
Add-PSSnapin Microsoft.SharePoint.PowerShell

$farm = [Microsoft.SharePoint.Administration.SPFarm]::Local

$var1 = $farm.BuildVersion.toString()
```

- SharePoint API for WebApplication

```
Add-PSSnapin Microsoft.SharePoint.PowerShell

$webApplication = [Microsoft.SharePoint.Administration.SPWebApplication]::Lookup("http://app")

$var1 = $webApplication.UserDefinedWorkflowMaximumComplexity
```

## 5.2. CMDLET - Enable-SPFeature - How to Activate SharePoint Features

It is impossible use *Enable-SPFeature* in order to activate features. You will receive a message about missing privileges.

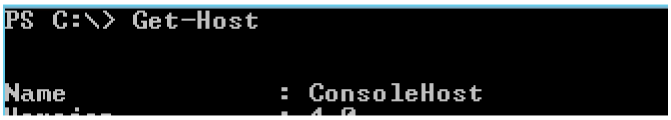
Instead, activate a feature by using this code:

```
## Activate WEB-Feature

$site_api = New-Object Microsoft.SharePoint.SPSite("{Common:WebUrl}")

$web_api   = $site_api.OpenWeb("{WorkflowVariable:WebName}")

$web_api.Features.Add("###YOUR FEATURE ID###")

$web_api.Update()
```

## 5.3. Write-Host / Out-Host / Read-Host

DO4PS isn’t a ConsoleHost instance of PowerShell, so it is not possible to use CMDLETs like Write-Host, Out-Host or Read-Host. Doing so will result in an error: “Cannot invoke this function because the current host does not implement it”

To get current value you can use “Get-Host” CMDLET.



```
PS C:\> Get-Host

Name                 : ConsoleHost
```

# 6. Used Components

## 6.1. Ace

https://ace.c9.io/

Copyright (c) 2010, Ajax.org B.V.

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
* Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
* Neither the name of Ajax.org B.V. nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL AJAX.ORG B.V. BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.