**SharePoint Online Continuous Deployment Framework (SPO-CDF)**

#### Author: Paul Borisov

#### 

#### October 12, 2016

Table of Contents

[Used terminology 3](#_Toc464071845)

[Introduction 3](#_Toc464071846)

[Content of this document 3](#_Toc464071847)

[The legacy provisioning solution 3](#_Toc464071848)

[The modern provisioning technique 3](#_Toc464071849)

[Visible drawbacks of the Add-ins (Apps) framework 4](#_Toc464071850)

[Requirements for an alternative simple provisioning solution 4](#_Toc464071851)

[The simple provisioning solution SPO-CDF 6](#_Toc464071852)

[Deployment hub 6](#_Toc464071853)

[Known issue 6](#_Toc464071854)

[Required permissions 6](#_Toc464071855)

[General structure of SPO-CDF 6](#_Toc464071856)

[User interface to request creation of site collections 24](#_Toc464071857)

[How to use SPO-CDF 27](#_Toc464071858)

# Used terminology

Branding: the process of applying a set of common corporate styles and UI-changes, also known as “look-and-feel”, that usually includes specific company’s logo, colour schema etc.

WSP-solution: a standard deployment package used to deploy artifacts in previous versions of SharePoint that used Feature Model Framework currently deprecated and substituted with more modern techniques based on the Remote Provisioning.

Sandbox WSP-solution: a special type of WSP-solution initially targeted for deployment into site collections of the restricted SharePoint environments like SharePoint Online. These solutions can use only a limited set of the SharePoint APIs permitted in so-called Sandbox Execution Model.

Legacy sandbox solutions, sandbox WSP-solutions: old versions of WSP-solutions possibly deployed to and activated in the environment.

CBSS, Code Based Sandbox Solutions: a general term, which describes sandbox solutions that include a compiled DLL. This compiled DLL may include the legacy server-side logic or be empty; in any case activation of any CBSS in the SharePoint Online was recently restricted by Microsoft (compare with NCSS below).

NCSS, No Code Sandbox Solutions: a general term, which describes sandbox solutions that include only the declarative artifacts (XML) and do not contain a compiled DLL. NCSS can be still activated in SharePoint Online without restrictions (compare with CBSS above).

# Introduction

## Content of this document

This document describes the automated deployment solution for SharePoint Online called “SharePoint Online Continuous Deployment Framework”; shortly SPO-CDF.

SPO-CDF allows automating the detection of just created OOB site collections and optional creation of newly requested ones followed by applying common branding and customizations to them. The default package of SPO-CDF includes a number of the fully working samples of similar customizations that can be used as a ready demo in evaluations and as a starting point to make your own changes.

SPO-CDF supports automated creation of modern site collections based on standard SharePoint templates available OOB as well as the legacy styled site collections based on custom templates automatically deployable with sandbox solutions. SPO-CDF recognizes a type of selected template (OOB vs. custom) and performs required adjustments and deployments automatically.

The source code of the solution can be found in the repository <https://github.com/PaulBorisov/SharePoint-Online-Continuous-Deployment-Framework>.

## The legacy provisioning solution

The classic customizations of SharePoint Online applications have been gradually developed and redesigned starting from 2010. At that time the key technology to create and deploy all necessary customizations including the company’s branding was based on so-called Sandbox Solutions usually represented by one or several specially packed WSP-files deployable into a special place of every newly created site collection.

* This document does not overview earlier years and Farm Solutions since they are not applicable to SharePoint Online. The sandbox solutions have been first introduced in SharePoint 2010.

No doubts, the technology of Sandbox Solutions was quite novice and officially promoted by Microsoft in 2009 – 2013, however, nowadays its time has gone. Microsoft has officially declared restrictions of “Code Based Sandbox Solutions” ([CBSS](#CBSS)) in the SharePoint Online since 29 July, 2016. This restriction means the server side code of legacy CBS Solutions that have been previously deployed and already activated in the existing site collections can continue running and working, however, new activations of CBSS are not permitted in any site collection.

Although Microsoft has no foreseeable plans to restrict also so-called “No Code Sandbox Solutions” ([NCSS](#NCSS)), the company does not recommend using any WSP-based customizations in the modern development techniques.

## The modern provisioning technique

What is the substitution for the legacy technique based on sandbox solutions? The modern technique called “the remote provisioning”. In practice, there is not any official solidified framework for the remote provisioning yet available at the moment of writing this document (October 12, 2016). There is a number of open source components and other initiatives, for example, “[OfficeDev/PnP: Office365 Developer Patterns and Practices](https://github.com/OfficeDev/PnP)”, “[Introducing the PnP Provisioning Engine](https://github.com/OfficeDev/PnP-Guidance/blob/551b9f6a66cf94058ba5497e310d519647afb20c/articles/Introducing-the-PnP-Provisioning-Engine.md)”, “[PnP Samples](https://github.com/OfficeDev/PnP/tree/master/Samples)” etc. No doubts, all these initiatives look great and impressive, however, none of them represents a simple “ready-made” solution that could automate creation and branding of site collections, literally, in no time.

This is also reasonable to mention that most of the proposed solutions involve usage of the modern techniques that rely on so-called “[SharePoint-hosted Add-ins](https://msdn.microsoft.com/en-us/library/office/fp179930.aspx)” and “[Provider-hosted Add-ins](https://msdn.microsoft.com/en-us/library/office/fp179930.aspx)”.

* By the definition, “SharePoint-hosted Add-in” deployed to the SharePoint Online cannot perform global operations like the creation of a site collection. Its usual purpose to provide UI-based customizations in a particular sub-site of a site collection.
* In the opposite side, “Provider-hosted Add-in” runs outside of SharePoint Online and can perform global operations like the creation of a site collection; however, it must have full permissions on an Office 365 Tenant to conduct them.
* In order to avoid confusion, Microsoft has replaced the older term “App” with “Add-in” in the reviewed naming convention changes. For example, “SharePoint-hosted Add-in” and “Provider-hosted Add-in” have been previously known as “SharePoint-hosted App” and “Provider-hosted App” accordingly.

## Visible drawbacks of the Add-ins (Apps) framework

Usually, a “Provider-hosted Add-in” is supposed to run in Microsoft Azure or any similar environment that has the exposure to the Internet. This is not strictly necessarily, however, recommended and actively promoted by Microsoft. Why is the exposure to the Internet needed? The matter is SharePoint Online should be able to redirect to location of a “Provider-hosted Add-in” seamlessly. Certainly, redirection to a local address like http://localhost:3789 is acceptable during development, however, is not applicable in a real Production environment.

Also “Provider-hosted Add-in” must have permission level of Tenant Admin (Global Administrator in the terms of Office365) to perform more or less serious actions like the creation of a site collection in the SharePoint Online. Frankly speaking, this required level of permissions looks a bit excessive and is definitely much higher than the actually sufficient “SharePoint administrator”. Why does the role of “SharePoint administrator” look sufficient? For example, you can easily delegate creation of a site collection in the SharePoint admin center (of SharePoint Online) to a person that just belongs to the role of “SharePoint administrator” only.

* This is confirmed by the fact that access to the object of type “Microsoft.Online.SharePoint.TenantAdministration.Tenant” works for a SharePoint Administrator, and does not require the level of Global Administrator.
* An interesting observation, even though some operations over the object “Microsoft.Online.SharePoint.TenantAdministration.Tenant” may raise the errors like “Current user is not a Tenant Admin” when executed under the limited account of SharePoint administrator; those operations usually succeed.
  + A typical example of similar operation is the adjustment of the site property DenyAddAndCustomizePages on the top root site collection of SharePoint Online. You can find more details in the chapter “Know issue” below.

Unfortunately, the current framework of Add-ins does not support this intermediate level of permissions.

* The [supported levels](https://technet.microsoft.com/en-us/library/jj219576(office.15).aspx) include “list”, “web”, “site collection”, and the whole tenant.
* This lack of intermediate level permissions is explainable and seems to come from the unification of the framework architecture for the SharePoint “on-premises” and Online as a part of “Office 365” where the term “Tenant” has slightly different meaning.
* In simple words, the “Tenant” usually corresponds to a single “Web-application” in the SharePoint “on-premises” vs. a private sub-domain of multiple applications like SharePoint, Exchange, Skype for Business etc., in the “Office 365”. Thus SharePoint Online is not a “Tenant”, but rather an application of a tenant in “Office 365”.

And one more disadvantage, developers should usually create custom user interfaces in the “Provider-hosted Add-in”. This is not possible to utilize parts of the existing screens in SharePoint Online for similar purposes because “Provider-hosted Add-in” is running in a separate remote environment.

# Requirements for an alternative simple provisioning solution

Let’s formalize the requirements for a solution.

First of all, the solution does not have to cover every imaginable point of a provisioning pipeline, however, it should provide at least the automated detection and optional creation of new site collections (as the most time consuming and potentially error prone operation) followed by their almost instant branding according to the company’s standards.

Any specialized customizations and other actions can be applied separately and are not discussed in this document.

So the required simple provisioning solution for the SharePoint Online:

* Should be able to run on any environment that has only the outbound Internet connection. Inbound connections that require dedicated DNS names or IP addresses to exist on your internal environment are highly not desired.
* Should be able to use the permissions available for the role “SharePoint administrator” as the maximum level. Why? Sometimes this looks challenging to convince a customer to grant the top level permissions like Tenant Admin (Global Administrator) to your application.
* Should be able to utilize existing UI parts and screens of SharePoint Online without a necessity to develop your own customized UI from the scratch. For example, this looks reasonable to reuse the modal dialog of SharePoint, and UI of the page “Create site”, which would require just several little tweaks to transform it into “Create site collection”.
* Should provide the customer’s representative a simplified and easily customizable interface to request creation of a new site collection. For example, the OOB dialog “New site collection” available in the SharePoint admin center contains several not absolutely necessary UI-elements like “Time Zone”, “Administrator”, and “Server Resource Quota” that usually confuse non-technical people.
* The customer’s representative should be able to request creation of a new site collection without a necessity to get access to the SharePoint admin center. Ideally, this person should access a special site collection to generate these requests.
* Site collection creation requests must be processed automatically and tolerate expected regular failures of SharePoint Online like timeout, temporary loss of connection, temporarily unavailability of Tenant etc.
* The site collection creation’s logic should not require establishing any special custom infrastructure like Windows Timer Job, Azure WebApp, etc. Ideally, it should be able to run continuously on any available machine that has limited resources and outbound Internet connection.
* Should be able to apply consistent branding and possible common customizations to all newly created site collections, literally, in a couple of minutes. Typical branding requirements usually include company’s logo and colour schema (CSS), common header and footer, common items to the top navigation, responsive UI.
* Should have possibility to store all common branding elements in a single place (“Deployment hub”) without a necessity to deploy similar customizations to every newly created site collection.
* Should support creation of site collections also in the old style from legacy custom web-templates. This requires intermediate automatic deployment of legacy “no code” sandbox solutions to a newly created site collection followed by applying a custom template to the root site of this site collection.
* Should have an option to handle absence of access to particular site collections for the executing account in an already established environment delicately and seamlessly. As mentioned above, the executing account should be able to use the permissions available for the role “SharePoint administrator”. These permissions look sufficient to resolve a possible issue with absence of access to particular site collections.

# The simple provisioning solution SPO-CDF

I have implemented the simple and extendable provisioning solution that covers most of the requirements mentioned above. It is named “SharePoint Online Continuous Deployment Framework”, or shortly, SPO-CDF.

The chosen operational environment is PowerShell 3.0 and a set of five SharePoint CSOM DLLs. This combination provides easy portability of the solution between different machines; it does not raise any special installation and registration requirements (except for copying files), simplifies maintenance and further gradual extensions.

## Deployment hub

Deployment hub is a dedicated site collection that acts as the centralized internal physical storage of customizations (CDN-node) for other site collections that just refer to the necessary files in the Deployment hub without a need to deploy them locally into every site collection

* Obviously, the Deployment hub can also use the customizations stored on it for its own purposes like branding.
* SPO-CDF has an option to support deployment of customizations into every site collection instead of using a centralized deployment hub. This slightly improves overall reliability; however, it makes further maintenance and upgrade of customizations more error prone and time consuming.

I recommend using the top root site collection as a Deployment hub on any new environment of SharePoint Online. If this is not desired, you can choose any other private site collection for the role of the Deployment hub

* For example, you can create /teams/spo-cdf as a standard Team site and use it as a Deployment hub.
* By default all deployed customizations are stored in the folder /\_catalogs/masterpage/customizations of the Deployment hub and its sub-folders.

The requirements for the Deployment hub are simple. The folder /\_catalogs/masterpage/customizations must have at least read permissions for all users of the SharePoint Online. The person who needs to be able to create site collections in the SharePoint Online must be added to the Site collection administrators of the Deployment hub.

## Known issue

Since the beginning of October, 2016 Microsoft periodically disables the option to apply customizations in the top root site of SharePoint Online service of a Tenant including the already existing deployments.

* This is done by turning the value of the setting DenyAddAndCustomizePages to “Enabled” occasionally.
* If need to re-enable the ability to apply customizations to the top root site, you can run the initialization script [\_\_EnvironmentSetup.ps1](#EnvironmentSetup) once again. Refer to the description of this script for details.
* The real necessity to change the content deployed once into the [Deployment hub](#BrandingHub) may appear relatively rarely. However, if this behaviour with “disabled customizations” is undesired just create the Deployment hub in some other location (not in the top root site).

## Required permissions

Executing account must be added to the role “SharePoint administrator” of a Tenant in order to perform all required provisioning actions of SPO-CDF seamlessly without errors of access.

* Note permissions of the “Global Administrator” of a Tenant are not required for the correct work of SPO-CDF at least at the moment of writing this document (October 12, 2016).

## General structure of SPO-CDF

The main folder of the solution contains 6 files of PS-scripts with extension .ps1. The scripts have names that represent the logical order of execution. Each script supports running as a part of the group and separately (standalone).

* \_\_LoadContext.ps1
* 1\_ContinuousDeployment.ps1
* 2\_ProcessAllSiteCollections.ps1
* 3\_UpdateSiteCollection.ps1
* 4\_DeployLegacySolutions.ps1
* 5\_CreateRequestedSites.ps1

There is also a sub-folder “utils” that contains 2 files of PS-scripts.

* \_\_EnvironmentSetup.ps1
* \_\_adjust-internal-wa-to-tenant-admin.ps1; not is use, reserved for the future extensions.

Detailed descriptions of the scripts are given below.

**Script name**

\_\_LoadContext.ps1

**Purpose**

This script loads and optionally initiates the object of the client context connected to a particular initial site collection of the SharePoint Online. It also encapsulates common utility functions used in other scripts.

* Typically, however, not necessarily, it loads the top root site collection https://<subdomain>.sharepoint.com.

The script is usually loaded first by other scripts and stays in memory for the group execution so the first executing script that loads the context leaves it in memory and next ones just check for existence and reuse the loaded context unless their logic needs to connect to a different site collection.

**Parameters defined in the header**

$siteCollection

* [string]
* Absolute URL of either SharePoint Online or "on-premises" site collection.
* "on-premises" is reserved for possible future extension of SPO-CDF.

$username

* [string]
* Login name of either SharePoint Online account or "on-premises" Windows user in format DOMAIN\account
* "on-premises" is reserved for possible future extension of SPO-CDF.

$password

* [string]
* Password of either SharePoint Online account or "on-premises" Windows user
* "on-premises" is reserved for possible future extension of SPO-CDF.

$useLocalEnvironment

* [bool], must always be $false for SharePoint Online
* Set to $true if you intend to use local SharePoint environment. This can be useful in case of "on-premises" site collections (not in the SharePoint Online).
* "on-premises" is reserved for possible future extension of SPO-CDF.

$useDefaultCredentials

* [bool], must always be $false for SharePoint Online
* Set to $true if you intend to use default network credentials instead of username and password.

$pathsToCsomDlls

* [string[]], the array of strings
* Points to location(s) that contain(s) CSOM DLLs. By default all needed SharePoint DLLs are loaded from the sub-folder “csom-dlls” situated below the main folder of SPO-CDF.
* Wildcards are supported, for example:
  + $PSScriptRoot\Microsoft.SharePoint.Client\*.dll, loads all client DLLs
  + $PSScriptRoot\Microsoft.Online.SharePoint.Client\*.dll
* It supports the load from alternative locations, for example:
  + C:\Program Files\Common Files\Microsoft Shared\Web Server Extensions\15\ISAPI\Microsoft.Online.SharePoint.Client\*.dll
  + C:\Program Files\SharePoint Online Management Shell\Microsoft.Online.SharePoint.Client\*.dll

$initContextOnLoad

* [bool], the default value must always be $true
* The value $false is used by an internal logic of some other scripts

**Script name and location**

\_\_EnvironmentSetup.ps1 situated in the subfolder “utils”.

**Purpose**

This script is used to initialize the environment of SharePoint Online and deploy a common set of necessary branding customizations into a dedicated site collection known as a [Deployment hub](#BrandingHub).

The script should be executed standalone as the first one on a new environment. This script loads [\_\_LoadContext.ps1](#LoadContext) and uses its settings to connect to the target environment of SharePoint Online.

* You need to review and adjust the parameters of [\_\_LoadContext.ps1](#LoadContext) before the first execution of the script \_\_EnvironmentSetup.ps1.

**Parameters defined in the header**

$staticUrlWithCustomizations

* [string], the default value is “/” (corresponds to the top root site collection)
* The value contains a server relative URL of a [Deployment hub](#BrandingHub) and can be optionally changed to point to another site collection, for example, to “/sites/spo-cdf”.

$disableCustomizations

* [bool], the default value is $false
* Set to $true if you plan to remove customizations from a [Deployment hub](#BrandingHub)
* In order to redeploy customizations to or upgrade a [Deployment hub](#BrandingHub), first set to $true and execute the script to remove customizations safely and next set to $false and execute the script to add new customizations.
* Removal of customizations just disables them on a [Deployment hub](#BrandingHub); however it does not remove the physical files deployed on it.
* Deployment of new customizations always overwrites previous versions of customization files with new ones.

$webRelativeUrlTargetFolder

* [string], the default value is “\_catalogs/masterpage/customizations/scripts"
* Do not change the default value

$filesForStandardWebTemplates

* [string], the default value is "customizations\scripts\wt-standard.js"
* Do not change the default value
* The physical file customizations\scripts\wt-standard.js contains an auto-generated JavaScript with a set of standard OOB web-templates currently available in a Tenant for creation of site collections.
* This file is automatically re-generated on every execution of the script \_\_EnvironmentSetup.ps1. This allows refreshing the list of available web-templates with possible new ones that may appear in the future.
* You have two options to deploy the newly generated web-templates to a [Deployment hub](#BrandingHub):
  + Execute the script \_\_EnvironmentSetup.ps1 -disableCustomizations $true followed by \_\_EnvironmentSetup.ps1 -disableCustomizations $false. These actions will remove and reinstall all customizations to a [Deployment hub](#BrandingHub). As mentioned above removal of customizations just disables them on a Deployment hub; however it does not remove the physical files deployed on it (thus allowing the connected clients to use them without interruption).
  + Execute the script 3\_UpdateSiteCollection.ps1 -siteCollectionUrl <absolute-url-of-a-branding-hub> -force $true. This should overwrite the existing customizations of a [Deployment hub](#BrandingHub) with new versions.

$defaultLocale

* [int], the default value is 1033 (English).
* This parameter identifies, which language should be selected as the default one in the updated file "customizations\scripts\wt-standard.js". This language will be preselected in the dialog that allows creating site collection requests.
* Do not change the default value if you are uncertain.

$supportedLocales

* [int[]], the default value contains 50 predefined locales to process
* This value is dynamically overwritten by the logic in the case if the next parameter allowOverwritingSupportedLocales is set to $true (by default).
* Do not change the default value

$allowOverwritingSupportedLocales

* [bool], the default value is $true
* If the value is set to $true the value of the parameter supportedLocales mentioned above is dynamically overwritten with available locales retrieved from a sub-site of a [Deployment hub](#BrandingHub).
* Do not change the default value

$compatibilityLevel = 15

* [int], the default value is 15, which corresponds to SharePoint 2016 and 2013 including Online.
* This value configures a list of retrievable web-templates for the file "customizations\scripts\wt-standard.js"
* Setting the value to 14 allows generating the list of available web-templates that correspond to SharePoint 2010 (including obsolete Meeting Workspace etc.).
* Note Microsoft does not guarantee supporting all of the web-templates that correspond to SharePoint 2010; however, you can still use them on your own risk.
* Do not change the default value if you are uncertain.

**Script name and location**

\_\_adjust-internal-wa-to-tenant-admin.ps1 situated in the subfolder “utils”.

**Purpose**

This script is not in use and just reserved for possible future extensions of the solution to SharePoint “on-premises”.

* Technically, nothing prevents adjusting and running SPO-CDF on the “on-premises” Farms of SharePoint 2013 and 2016.
* SharePoint “on-premises” requires enabling a Tenant on the level of each web-application to behave similarly to the SharePoint Online.

**Script name**

1\_ContinuousDeployment.ps1

**Purpose**

This script emulates the work of Windows Scheduler and starts the loop, which executes the script 2\_ProcessAllSiteCollections.ps1 followed by execution of the script 5\_CreateRequestedSites.ps1. The first script starts and executes synchronously and the second one asynchronously. After the script 1\_ContinuousDeployment.ps1 starts the execution of 5\_CreateRequestedSites.ps1 it makes a pause of 30 seconds (“sleeps” to wait for release of resources).

Asynchronous execution of the script 5\_CreateRequestedSites.ps1 can run for significant time because the process of creating site collections in the SharePoint Online is usually continuous. While the script 5\_CreateRequestedSites.ps1 is being executed its next instance does not start so only one session that creates site collection is active at the same time. Obviously, the script 2\_ProcessAllSiteCollections.ps1, which runs synchronously inside 1\_ContinuousDeployment.ps1 also has only one active instance at the same time.

The script 1\_ContinuousDeployment.ps1 keeps a track of processed site collections in the cache file \_\_site-states.csv situated in the subfolder “logs” of the main folder of SPO-CDF.

* \_\_site-states.csv is a tab separated file that can be seamlessly open for review in the Excel. Content of this file is re-saved by the script 1\_ContinuousDeployment.ps1 after each execution of the script 2\_ProcessAllSiteCollections.ps1. The file stores the following information about each processed site collection:
  + Url: absolute URL of each processed site collection; in lower case.
  + LastProcessed: the most recent date and time when a site collection was successfully or unsuccessfully processed by the script 2\_ProcessAllSiteCollections.ps1.
  + Succeeded: true is the processing of a particular site collection has completed without errors and false in the opposite case.
  + Customized: true if the branding and customizations of a particular site collection have been successfully applied to the processed site collection and false in the opposite case (if branding and customizations have been successfully disabled).
  + FailedAttempts: amount of attempts used to apply branding and customizations to a particular site collection. By default, the script 2\_ProcessAllSiteCollections.ps1 performs five retry attempts. If all five attempts have failed the script does not retry applying or removing customizations anymore. Note the value is not reset after each successful attempt, however, you can reset it manually if needed.
* Content of the file \_\_site-states.csv can be easily deleted of modified to change the parameters of processing a particular site collection. In case of accidental mistakes the lines of this file that fail to load are just ignored.

**Parameters defined in the header**

$secondsToRepeat

* [int], the default value is 30
* It specifies the Interval between iterations, in seconds (time to “sleep” between executions of 2\_ProcessAllSiteCollections.ps1)

$maxIterarions

* [int], the default value is 0
* This parameter allows limiting max amount of iterations. 0 or negative value means infinite execution.

$maxFailedAttempts

* [int], the default value is 5
* This parameter allows limiting max amount of failed attempts to apply or disable customizations on a particular site collection. 0 or negative value means infinite amount of retries.

$filePathSiteStates

* [string], the default value is "$PSScriptRoot\logs\\_\_site-states.csv"
* This parameter specifies the path to the file that contains cached statuses of processing particular site collections. After the site collection has been processed and the branding and customizations have been applied to it the trace record is added to the cache file. This record allows to avoid processing of the same site collection in the next iterations (unless value of the parameter $disableCustomization is changed to the opposite in the script 2\_ProcessAllSiteCollections.ps1; in this case processing repeats and an updated status is stored into the cache file \\_\_site-states.csv).

$readCacheFileOnEveryIteration

* [bool], the default value is $true
* If value of this parameter is set to $true, it forces to re-read status information from the cached file after every iteration. This is useful if you need to change processing behaviour for a particular site collection dynamically (to be applied in the next iteration).
* If the value is $false the file is just stored in memory after re-saving, and any manual changes are ignored.

**Script name**

2\_ProcessAllSiteCollections.ps1

**Group and standalone execution**

This script supports a group execution inside [1\_ContinuousDeployment.ps1](#ContinuousDeployment) and a standalone execution on its own. In the first case the script receives, checks and optionally updates the [cache](#CacheFile) data in memory supplied by [1\_ContinuousDeployment.ps1](#ContinuousDeployment). In the case of a standalone execution check of the cache data is voided (dummy empty cache variable is used).

**Purpose**

This script connects to a Tenant of SharePoint Online, finds all SharePoint site collections present in it and iterates through them. If the script identifies a particular site collection has not been processed earlier (checks the [cache](#CacheFile) data), it tries to process it and execute the scripts [3\_UpdateSiteCollection.ps1](#UpdateSiteCollection) or [4\_DeployLegacySolutions.ps1](#DeployLegacySolutions) depending on the required customization scenario (processing a standard site collection vs. processing a legacy site collection).

* The script identifies legacy site collections that use WSP-based customization model by comparing their URL with the value defined by the parameter $legacyUrls; this value contains a regular expression for a comparison (see [below](#legacyUrls)).

After a site collection has been processed the script updates the cache information in memory received from the parent script [1\_ContinuousDeployment.ps1](#ContinuousDeployment) unless it is a standalone execution.

* Note a particular site collection could be processed successfully of with errors. This state is reflected in the cache data for possible future retry attempts. You can see more detailed information about the structure of cache data and the retry attempts in the description of the script [1\_ContinuousDeployment.ps1](#ContinuousDeployment).

**Suppression of restricted permissions**

Particular site collections can have different security settings. Sometimes these settings may prevent the script 2\_ProcessAllSiteCollections.ps1 from accessing a site collection. However since the script is executed under the account that belongs to the role of “SharePoint administrator” it has the ability to suppress the security settings of a site collection temporarily or constantly. Default parameter values of the script permit adjusting the security settings of site collections temporarily, and only if it is really necessary (i.e. no access detected).

* Technically, in this case the script explicitly adds the executing account as an additional site collection administrator using a Tenant object model and keeps this information until the end of processing this site collection. At the end of processing the script removes the executing account from site collection administrators (since it was explicitly added). In the case the site collection administrator has been added but its removal failed the script reports the accident to the file \_\_admins-pending.csv situated in the subfolder “logs” of the main folder of SPO-CDF.
* The behaviour described above is controlled by the parameter $addSiteAdminWhileProcessing, which has the default value $true. If this is set to $false the optional addition of the executing account into the site collection administrators is not performed. Obviously, this may lead to errors of type “Access denied” on attempts to process the restricted site collection. The setting $false is not recommended, however, is useful in the case if the customer has very strict security rules on granting any extra access.
* There is another parameter $keepSiteAdminAfterProcessing with default value $false. If this is set to $true removal of the optionally added executing account from site collection administrators is not performed and the accident is not reported to the file \_\_admins-pending.csv.

**Suppression of the default restriction of changes on the top root site collection**

In some cases, Microsoft restricts the top root site collection of the SharePoint Online from changes of the general structure. This is done via setting the special site collection property DenyAddAndCustomizePages to “Enabled”. The script automatically identifies presence of this setting on the top root site collection of the SharePoint Online and disables it.

* If desired, you can review and change this default behaviour in the method GetSitePropertiesViaTenant.
* If DenyAddAndCustomizePages is set to Enabled in the site collection properties any attempt to process similar site collection ends up with “Access denied” errors. Sometimes this can be very misleading.

**Log of processing actions**

The process of applying branding and customizations to a particular site collection is complex and potentially error prone. This is reasonable to report the progress and the errors of processing each site collection for possible future review of the problems. The script 2\_ProcessAllSiteCollections.ps1 logs all the actions and messages on processing all site collections into the files named “log-<year>-<month>-<day>-<hours>-<minutes>.txt” and situated in the subfolder “logs” of the main folder of SPO-CDF.

* Note each execution of the script 2\_ProcessAllSiteCollections.ps1 always reports actions on all processed site collections into a single log file.
* The script automatically removes old log files if their total amount in the subfolder “logs” exceeds 1440 (i.e. ~24 hours old and earlier in the case of constant execution from the context of [1\_ContinuousDeployment.ps1](#ContinuousDeployment)).

**Parameters defined in the header**

$unattended

* [bool], the default value is $true
* This parameter can be used for debugging purposes. If it is set to $false the script asks to confirm most of processing actions explicitly (y/n).
* Do not change the default value if you are uncertain.

$disableCustomizations

* [bool], the default value is $false
* This parameter allows applying or reverting customizations on all site collections in a single run.
* Do not change the default value if you are uncertain.

$forceOnFailedOnly

* [bool], the default value is $true
* This parameter instructs the script to force applying or reverting customizations on a particular site collection in the case when the last processing has failed and the maximum amount of retry attempts has not been exceeded.
* Default settings defined in the parent script 1\_ContinuousDeployment.ps1 permit up to 5 retry attempts in total. Refer to the description of [1\_ContinuousDeployment.ps1](#ContinuousDeployment) for more details.
* Do not change the default value if you are uncertain.

$maxFailedAttempts

* [int], the default value is 0, which means “infinite retry attempts”
* Value of this parameter is overwritten by the value of similar parameter defined in [1\_ContinuousDeployment.ps1](#ContinuousDeployment) unless the script 2\_ProcessAllSiteCollections.ps1 is executed standalone
* This parameter duplicates a similar parameter 1\_ContinuousDeployment.ps1 defined in the parent script and is used in the case of a standalone execution to support the logic’s flow.
* Do not change the default value if you are uncertain.

$forceOnSucceededOnly

* [bool], the default value is $false
* This parameter instructs the script to force applying or reverting customizations on a particular site collection in the case when the last processing has succeeded.
* Maximum number of retry attempts has no influence on this parameter in compare with its sibling parameter $forceOnFailedOnly.
* Do not change the default value if you are uncertain.

$preferSearchQuery

* [bool], the default value is $false
* This parameter instructs the script’s logic to prefer search query to get list of sites in case when the Tenant temporarily malfunctions.
* Set value of this parameter to $true only in the really critical cases when the Tenant is not available or malfunctions (unfortunately, this occasionally happens in the SharePoint Online)
* Remember that a newly created site collection is not instantly available in search results due to a crawl processing delays.
* Security trimming is always applied to search results thus some content can be unavailable in compare with standard access via Tenant. That’s why the default logic of the script uses retrieval of site collection properties using a Tenant object model. Using a Tenant is more reliable in compare to search queries; however, Tenant’s services can be randomly unavailable from time to time.
  + If Tenant is unavailable (i.e. the site https://<tenant>-admin.sharepoint.com is down), temporarily change $preferSearchQuery to $true
  + If $preferSearchQuery = $true the filters defined in the parameter $excludeBySearchProperties are used instead of $excludeBySiteProperties. You can find more details below,
* Do not change the default value without a necessity.

$excludeBySiteProperties

* [Hashtable], the default value is

$excludeBySiteProperties = @{

# Explicitly exclude site collections restricted

DenyAddAndCustomizePages = @{Value = "Enabled"; Match = $true};

# Include only unlocked site collections and exclude locked ones

LockState = @{Value = "Unlock"; Match = $false};

# Include only active site collections and exclude inactive ones

Status = @{Value = "Active"; Match = $false};

# Explicitly exclude site collections having these URLs

Url = @{Value = "(?i)((/portals/hub)|(/portals/community)|(-public\.))"; Match = $true};

# Include site collections with this URL pattern only

#Url2 = @{Value = "(?i)((/teams/ucd)|(/sites/ts)|(/sites/pp)|(/sites/ncss))"; Match = $false};

#Url2 = @{Value = "(?i)/sites/t"; Match = $false};

# Include site collections of these templates and exclude others

#Template = @{Value = "(?i)((STS)|(CMSPUBLISHING))"; Match = $false};

# Explicitly exclude site collections of these templates

Template2 = @{Value = "(?i)((SPSMSITEHOST)|(SPSPERS))"; Match = $true};

}

* This parameter allows managing filters on processing certain site collections on a variety of its properties. The logic uses the algorithm “exclude if the site collection property matches the pattern”.
  + Example 1, DenyAddAndCustomizePages = @{Value = "Enabled"; Match = $true}. This condition means if a site collection has the property DenyAddAndCustomizePages value of which matches with Enabled this site collection should be omitted from further processing.
  + Example 2, Status = @{Value = "Active"; Match = $false}. This condition means if a site collection has the property Status value of which does NOT match with Active this site collection should be omitted from further processing.
  + Example 3, Url2 = @{Value = "(?i)/sites/t"; Match = $false}. Trailing numeric value of the property is ignored and only the property name itself is used; in this case Url2 🡪Url. This allows using plural “exclude” and “include” statements for the same site collection property processed one after another (Url, Url2, Url3 etc.). The condition of the Example 3 shown above means if a site collection has the property Url (trailing 2 is ignored) value of which does not match with "(?i)/sites/t" this site collection should be omitted from further processing.
* Do not change the default value if you are uncertain.

$excludeBySearchProperties

* [Hashtable], the default value is

$excludeBySearchProperties = @{

Path = $excludeBySiteProperties.Url;

#Path2 = $excludeBySiteProperties.Url2;

#WebTemplate = $excludeBySiteProperties.Template1;

WebTemplate2 = $excludeBySiteProperties.Template2

}

* This parameter duplicates the work of $excludeBySiteProperties for the case of using a search service instead of a Tenant object model. Properties of a site collection returned in the search results have different names in compare with names of Tenant’s SiteProperties used in the parameter $excludeBySiteProperties
* The most important properties for the search results are mapped to the correspondent ones specified in the parameter $excludeBySiteProperties.
* Do not change the default value if you are uncertain.

$legacyUrls

* [string], regular expression, the default value is $null
* This parameter identifies legacy sites by their URLs.
* The script identifies legacy site collections that use WSP-based customization model by comparing their URL with the value defined by this.
* The default value $null means there are no legacy SharePoint site collections in your Tenant.
* If you change the value to “.” all site collections of your Tenant will be considered as legacy ones that require processing via execution of [4\_DeployLegacySolutions.ps1](#DeployLegacySolutions) instead of [3\_UpdateSiteCollection.ps1](#UpdateSiteCollection) in the case of standard site collections (4\_DeployLegacySolutions.ps1 performs deployment of sandbox solutions into the legacy site collections).
* Do not change the default value if you are uncertain.

$logFile

* [string], the default value is "$PSScriptRoot\logs\log" + (get-date).ToString("-yyyy-MM-dd-HH-mm") + ".txt"
* This parameter specifies the name and location of the log file produced on running each session of this script. The script uses PowerShell-transcript engine to output the execution information into the log file.
* A new log file is generated in each single execution of this script unless the previous execution has been completed in the current minute (in this case the data of a new execution can be appended to the previous log file).
* Do not change the default value without strong necessity.

$logFileAdminsPending

* [string], the default value is "$PSScriptRoot\logs\\_\_admins-pending.csv"
* This parameter contains path to the incident reporting file in the case when the script has successfully added the executing account to site collection administrators to suppress restricted permissions on a particular site collection, however, has failed to remove it later.
* Refer to the chapter [Suppression of restricted permissions](#SuppressRestrictedPermissions)above in this document for more details.
* Do not change the default value without strong necessity.

$maxLogFiles

* [int], the default value is 1440
* This parameter defines maximum amount of log files generated on each execution of this script to keep in the subfolder “logs” of the main folder of SPO-CDF.
* Refer to the chapter [Log of processing actions](#LogProcessingActions)above in this document for more details.
* Do not change the default value without strong necessity.

$addSiteAdminWhileProcessing

* [bool], the default value is $true
* This parameter allows suppressing error messages on possible lack of permissions in site collections for the executing account. It instructs to add the executing account temporarily to site collection administrators in case of no access while processing a particular site collection and remove it after the processing is done.
* If the executing account has been added to site collection administrators, however, could not be removed after the processing has been finished the incident is reported to the file defined by the parameter $logFileAdminsPending and may require manual removal.
* Refer to the chapter [Suppression of restricted permissions](#SuppressRestrictedPermissions)above in this document for more details.
* Do not change the default value without strong necessity.

$keepSiteAdminAfterProcessing

* [bool], the default value is $false
* This parameter allows keeping the executing account temporarily added to site collection administrators in the case of no access to a particular site collection. If value of this parameter is set to $true removal of the optionally added executing account from site collection administrators is not performed and the accident is not reported to the file \_\_admins-pending.csv.
* Refer to the chapter [Suppression of restricted permissions](#SuppressRestrictedPermissions)above in this document for more details.
* Do not change the default value without strong necessity.

$suppressTranscript

* [bool], the default value is $false
* This is an internal service flag used during initialization of the environment.
* Do not change the default value of this parameter.

$suppressSiteCollectionUpdate

* [bool], the default value is $false
* This is an internal service flag used during initialization of the environment.
* Do not change the default value of this parameter.

$processedSites

* [Hashtable], the default value is @{}
* This parameter represents a [cache](#CacheFile) data supplied to this script by the parent script [1\_ContinuousDeployment.ps1](#ContinuousDeployment) in the case of automated execution (a looped group execution).
* The default value is used to maintain consistency of the script’s logic in the case of a standalone execution.
* Do not change the default value of this parameter.

**Useful property filters (regex)**

You can easily limit the scope of site collections that should be processed to a single only if you uncomment the setting Url2 and specify its value as shown below:

* $excludeBySiteProperties = @(

…

Url2 = @{Value = "(?i)<absolute-or-server-relative-url-of-your-site-collection> "; Match = $false};

…

)

* This is also recommended to uncomment and modify the parameter $excludeBySearchProperties as

$excludeBySearchProperties = @{

…

# The next parameter comes into use instead of $excludeBySiteProperties.Urls2

# if the Tenant is temporarily unavailable and the processing logic falls back to using the search engine

Path2 = $excludeBySiteProperties.Url2;

…

}

You can also easily limit the scope of site collections that should be processed to the ones created from a specific standard web-template only if you uncomment the setting Template and specify its value as shown below:

* $excludeBySiteProperties = @(

…

Template = @{Value = "(?i)^STS$"; Match = $false};

…

}

* This is also recommended to uncomment and modify correspondent settings of the parameter $excludeBySearchProperties as

$excludeBySearchProperties = @{

…

# The next parameter comes into use instead of $excludeBySiteProperties.Template

# if the Tenant is temporarily unavailable and the processing logic falls back to using the search engine

WebTemplate = $excludeBySiteProperties.Template

…

}

* Note names of custom web templates cannot be specified above because they are eventually turned into the standard web templates in the CSOM property Site.RootWeb.WebTemplate.

You can set a value of the parameter $legacyUrls to the URL of specific site collection to deploy legacy sandbox solutions from the subfolder “legacy\sandbox-wsps” into those site collections while the script is processing them.

* For example, $legacyUrls = "(?i)((/sites/legacy-site-collection-4)|( (/sites/legacy-site-collection-5)))"

**Script name**

3\_UpdateSiteCollection.ps1

**Purpose**

This script supports a group execution from the context of [1\_ContinuousDeployment.ps1](#ContinuousDeployment) and a standalone execution on its own. In the first case the script receives some of the execution parameters from the parent script [2\_ProcessAllSiteCollections.ps1](#ProcessAllSiteCollections). In the case of a standalone execution the script uses the default values of own parameters defined in its header.

This script performs branding and customization of a particular single site collection. By default, it supports reusing physical files stored in the [Deployment hub](#BrandingHub) via setting correspondent references to them. Optionally, the script also supports deploying physical files into a particular site collection.

* Refer to the chapter [How to disable the Deployment hub](#DisableDeploymentHub) for more details.

**Parameters defined in the header**

$siteCollectionUrl

* [string], the default value is $null. In this case the site collection specified in \_\_LoadContext.ps1 will be processed.
* In the case of a group execution from the context of [1\_ContinuousDeployment.ps1](#ContinuousDeployment) the value of this parameter contains an absolute URL of a particular site collection supplied by the parent script [2\_ProcessAllSiteCollections.ps1](#ProcessAllSiteCollections) during iteration through site collections.
* Do not change the default value of this parameter.

$disableCustomizations

* [bool], the default value is $false
* This parameter allows applying or reverting customizations on a site collection.
* In the case of a group execution from the context of [1\_ContinuousDeployment.ps1](#ContinuousDeployment) a value of this parameter is always overwritten by the parent script 2\_ProcessAllSiteCollections.ps1, which also has its own parameter with the same name, value of which affects processing of plural site collections.
* Do not change the default value if you are uncertain.

$force

* [bool], the default value is $false
* In the case of a group execution from the context of [1\_ContinuousDeployment.ps1](#ContinuousDeployment) value of this parameter may vary dynamically depending on conditions and the settings of parameters $forceOnFailedOnly, $maxFailedAttempts and forceOnSucceededOnly defined in the parent scripts 2\_ProcessAllSiteCollections.ps1 and 1\_ContinuousDeployment.ps1.
* In the case of a standalone execution of this script, the parameter can be used to re-deploy customizations forcibly to a site collection that already contains them. Set the value to $true and make sure $disableCustomizations = $false.
* In the case of a standalone execution of this script, the parameter can be also used to guarantee complete removal of customizations forcibly from a site collection that does not seem to contain them. Set the value to $true and make sure $disableCustomizations = $true.
* Do not change the default value if you are uncertain.

$evaluateOnly

* [bool], the default value is $false
* The parameter exists only in this script and is never overwritten from the parent ones in the case of a group execution.
* If you change its value to $true, it allows emulating the complete processing without making the actual changes to the processed site collection. This can be useful for debugging and preparation purposes when you want to save time that would be spend on the actual changes,
* Do not change the default value if you are uncertain.

$staticUrlWithCustomizations

* [string], the default value is "/", which identifies the top root site collection as the [Deployment hub](#BrandingHub).
* The value of this parameter is never overwritten from the parent ones in the case of a group execution.
* If the value is set to the empty string (“”) this forces deployment of physical files with customization into a processed site collection instead of using the customizations stored in the [Deployment hub](#BrandingHub).
* You should never set the value of this parameter to $null to avoid errors.
* Do not change the default value if you are uncertain.

$webRelativeUrlTargetFolder

* [string], the default value is "\_catalogs/masterpage/customizations"
* This parameter defines the target folder of a site collection where the physical files of customizations should be stored. In case of using a [Deployment hub](#BrandingHub) value of this parameter is used to refer to a correspondent folder of the hub.
* Do not change the default value of this parameter without a strong necessity.

$rootFolderWithCustomizations

* [string], the default value is “$PSScriptRoot\customizations"
* This parameter identifies physical location of customizations in the file system. By default, customizations are stored in the subfolder “customizations” of the main folder of SPO-CDF.
* Do not change the default value of this parameter without a strong necessity.

$customActionFiles

* [string[]], an array of strings, the default value is shown below.

$customActionFiles = @(

"scripts/jquery-3.1.0.min.js", "scripts/custom-ui.js", "css/custom-ui.css",

"scripts/SPO-Responsive.js", "css/SPO-Responsive.css"

)

* This parameter contains short references to custom JavaScripts and CSS-files that should be automatically loaded on every page below each particular site collection including its subsites.
  + Technically, a dedicated User Action (also known as Custom Action) is created for every script or CSS-file mentioned in the value of this parameter.
  + This is not a mistake in the description, a CSS-file can be also loaded by a User Action via a relatively simple dynamic logic that can be observed in the function AddSiteScriptCustomAction of the current PS-script.
* Reference to the copyright: the script SPO-Responsive.js and its supplemental CSS-file SPO-Responsive.css are taken from the open source project “[SharePoint 2013/2016/Online Responsive UI](https://github.com/OfficeDev/PnP-Tools/tree/master/Solutions/SharePoint.UI.Responsive)”. As found in tests this combination works pretty well for most of the site types; big thanks for the excellent job to the author Paolo Pialorsi.

$customActionLegacyFiles

* [string[]], an array of strings, the default value is @("scripts/custombranding.js")
* The value of this parameter is only used to identify if a site has particular legacy customizations, for example, some JavaScript deployed as a custom action. You can observe a sample of verification logic in the function HasLegacyCustomizations of this PS-script. This is just an example, which can be easily adjusted to your specific needs.

$customActionMenuItems

* [Hashtable], the default value is shown below.

$customActionMenuItems = @{

CreateSite = @{

Url = "scripts/custom-sa-create-site.js";

LocalizedTexts = @{

Default = 1033;

1033 = @{Title = "Add new site collection"; Description = ""};

1035 = @{Title = "Lisää uusi sivustokokoelma"; Description = ""}

};

Order = 1;

Rights = "FullMask";

SiteAdminsOnly = $true;

PermittedOnUrls = @($staticUrlWithCustomizations)

}

}

* This parameter allows defining custom menu items that can be added to the standard menu of Site Actions. The default value is self-explaining.
  + The functionality of the added custom menu item can require specific rights to be visible (OOB protection) or can be limited to site collection administrators (an additional logical protection, not tight).
  + Custom menu items can also be limited to specific URLs of site collections managed via PermittedOnUrls.

$disableFeatures

* [string[]], an array of strings, the default value is @([guid]("87294c72-f260-42f3-a41b-981a2ffce37a")); it disables the feature “Minimal Download Strategy”, which optimizes the load of Team Sites, however, makes the loaded URL of a welcome page ugly.
* This parameter allows specifying a set of undesired features of SharePoint that should be disabled on a processed site collection. The underlying script’s logic first processes the Site Collection features and next the Site Features. If it finds feature with given GUIDs it disables them.

$enableFeatures

* [string[]], an array of strings, the default value is an empty array.
* This parameter allows specifying a set of features of SharePoint that should be enabled on a processed site collection. The underlying script’s logic first processes the Site Collection features and next the Site Features and tries enabling features with given GUIDs.

$navigationNodes

* [Hashtable], the default value is shown below.

$navigationNodes = @{

Default = 1033;

1033 = @{

"Guidelines" = @{Url = "/Pages/guidelines.aspx"; IsExternal = $true; Order = 1};

"A Service Information" = @{Url = "/Pages/service-information.aspx"; IsExternal = $true; Order = 2}

};

1035 = @{

"Suuntaviivat" = @{Url = "/Sivut/suuntaviivat.aspx"; IsExternal = $true; Order = 1};

"Tietoa palvelusta" = @{Url = "/Sivut/tietoa-palvelusta.aspx"; IsExternal = $true; Order = 2}

}

}

* This parameter allows defining custom navigation items that can be automatically added to the standard top navigation menu bar of the root site of a site collection. The default value is self-explaining.
  + The parameter “Default” defines the default locale to be used in the case when the root site has a locale (.Web.Language) that does not contain explicit navigation items specified in $navigationNodes. For example, if the root site has the value .Web.Language = 1053 (Swedish), the navigation nodes of the default English locale (1033) will be added to that site.
  + This is recommended to keep the value of the parameters IsExternal = $true. In the opposite case the logic will always evaluate the existence of actual URLs.
  + The parameter Order reflects order of navigation items being added (a Hashtable does not guarantee the correct order).

$webTemplatesWithNoWelcomePage

* [string], the default value is "(?i)((POLICYCTR)|(OFFILE)|(BDR))"
* This is a service parameter used by the internal logic to identify sites of special types that do not have explicitly defined welcome page.
* Do not change the default value of this parameter without necessity.

$webTemplatesWithNoTopNavigation

* [string], the default value is "(?i)POLICYCTR"
* This is a service parameter used by the internal logic to identify sites of special types that do not have explicitly defined top navigation bar. The value of the parameter is used in conjunction with $navigationNodes
* Do not change the default value of this parameter without necessity.

**How to disable the Deployment hub**

There is an option to disable the Deployment hub and to force installing customizations into every site collection. In order to do this, it is enough to set the value of the parameter $staticUrlWithCustomizations to the empty string (i.e. $staticUrlWithCustomizations = “”) in the header of the script 3\_UpdateSiteCollection.ps1.

**Script name**

4\_DeployLegacySolutions.ps1

**Purpose**

This script supports optional deployment of the “old fashioned” legacy customizations that could have been developed earlier using the classic SharePoint Feature Framework and packed into a number of sandbox solutions.

The script provides automated (re)deployment and (re)activation of all sandbox WSP-solutions from a folder defined in a script’s parameter to a particular site collection.

**Parameters defined in the header**

$siteCollectionUrl

* [string], the default value is $null. In this case the site collection specified in \_\_LoadContext.ps1 will be processed.
* In the case of a group execution from the context of [1\_ContinuousDeployment.ps1](#ContinuousDeployment) the value of this parameter contains an absolute URL of a particular site collection supplied by the parent script [2\_ProcessAllSiteCollections.ps1](#ProcessAllSiteCollections) during iteration through site collections.
* Do not change the default value of this parameter.

$pathToFolderWIthWspSolution

* [string], the default value is "$PSScriptRoot\legacy\sandbox-wsps"
* This parameter specifies full path to the folder that stores WSP-packages of deployable sandbox solutions.
* Note if you need to activate solutions in a specific order use correspondent prefix in file names, for example, 1\_firstSolution.wsp, 2\_anotherSolution.wsp, 3\_extraSolution.wsp

$regexDeploymentUrls

* [string], regular expression, the default value is $null
* This parameter provides additional filters to limit the deployment of sandbox solutions only to those site collections that have URLs matching the specified regular expression.
* In the case of a group execution from the context of [1\_ContinuousDeployment.ps1](#ContinuousDeployment) the value of this parameter is always replaced with the value of $legacyUrls supplied by the parent script [2\_ProcessAllSiteCollections.ps1](#ProcessAllSiteCollections) during iteration through site collections.
* In the case of a standalone execution the default value $null permits deployment of sandbox solutions to any site collection.
* The custom logic that permits or disallows deployment of sandbox solutions to particular site collections can be found and easily adjusted in the function AllowDeployment of this script.
* Do not change the default value of this parameter without necessity.

$disableCustomizations

* [bool], the default value is $false
* This parameter controls activation of sandbox solutions on a site collection. If its value is $false the logic of the script tries to deactivate sandbox solutions in the case they already exist, (re)deploy and (re)activate them. If its value is $true the logic of the script just tries to deactivate possibly existing sandbox solutions.
* In the case of a group execution from the context of [1\_ContinuousDeployment.ps1](#ContinuousDeployment) a value of this parameter is always overwritten by the parent script 2\_ProcessAllSiteCollections.ps1, which also has its own parameter with the same name, value of which affects processing of plural site collections.
* Do not change the default value if you are uncertain.

$force

* [bool], the default value is $false
* In the case of a group execution from the context of [1\_ContinuousDeployment.ps1](#ContinuousDeployment) value of this parameter may vary dynamically depending on conditions and the settings of parameters $forceOnFailedOnly, $maxFailedAttempts and forceOnSucceededOnly defined in the parent scripts 2\_ProcessAllSiteCollections.ps1 and 1\_ContinuousDeployment.ps1.
* In the case of a standalone execution of this script, the parameter can be used to re-apply sandbox solutions forcibly to a site collection that already contains them. Set the value to $true and make sure $disableCustomizations = $false.
* In the case of a standalone execution of this script, the parameter can be also used to ensure complete deactivation of sandbox solutions forcibly in a site collection. Set the value to $true and make sure $disableCustomizations = $true.
* Do not change the default value if you are uncertain.

$dismissAdditionalSolutions

* [string[]], array of strings, the default value is an empty array, @().
* This parameter allows preliminary deactivation of any sandbox solutions that possibly exist in a site collection before the main logic starts (re)deployment and (re)activation of sandbox solutions from the folder specified in the parameter $pathToFolderWIthWspSolutions into this site collection.
* This parameter is useful when the old and the newly deployed solutions may conflict with each other.
* Do not change the default value if you are uncertain.

**Script name**

5\_CreateRequestedSites.ps1

**Purpose**

This script provides automatic creation of requested site collections by the parameters stored in the custom list “Deployment requests” usually situated by the site relative URL Lists/DeploymentRequests of the [Deployment hub](#BrandingHub).

**Parameters defined in the header**

$staticUrlWithCustomizations

* [string], the default value is “/” (corresponds to the top root site collection)
* The value contains a server relative URL of a [Deployment hub](#BrandingHub) and can be optionally changed to point to another site collection, for example, to “/sites/spo-cdf”.

$recreateSiteIfExists

* [bool], the default value is $false
* This parameter can be used in the case of intensive development. It allows a forcible automated recreation of site collections that may already exist in the system.
* Note changing the default value of this parameter to $true is potentially unsafe in the case of accidental misuse because it instructs to delete existing site collection and remove this deleted instance from the Recycle Bin.
* Do not change the default value if you are uncertain.

$removeSiteOnly

* [bool], the default value is $false
* This parameter can be used in the case of intensive development. It allows forcible automated removal of possibly existing site collections without recreating them.
* If the value is $true, this parameter has a priority over $recreateSiteIfExists; a value of the latter one is ignored in this case.
* Note changing the default value of this parameter to $true is potentially unsafe in the case of accidental misuse because it instructs to delete existing site collection and remove this deleted instance from the Recycle Bin.
* Do not change the default value if you are uncertain.

$listUrlDeploymentRequests

* [string], the default value is "Lists/DeploymentRequests"
* This is an internal service parameter that contains site relative URL of the custom list “Deployment Requests” that stores the necessary data used to create the requested site collections.
* Do not change the default value of this parameter.

$compatibilityLevel

* [int], the default value is 15, which corresponds to SharePoint 2016 and 2013 including Online.
* This is an internal service parameter used by the logic of the script. Value of this parameter is used in the internal verification of a web template and as a part of site creation information to identify the necessary compatibility level of a site collection being created.
* Do not change the default value of this parameter.

$storageMaximumLevel

* [int], the default value is 100
* This is an internal service parameter used by the logic of the script.
* Do not change the default value of this parameter without necessity.

$userCodeMaximumLevel

* [int], the default value is 100
* This is an internal service parameter used by the logic of the script. Value of this parameter corresponds to the setting “Server Resource Quota” in the SharePoint admin center (of SharePoint Online). It allows controlling the amount of resources dedicated to a site collection being created.
* Do not change the default value of this parameter without necessity.

$timeoutToClearHangingRequestsMinutes

* [int], the default value is 20
* This is an internal service parameter used by the logic of the script. Value of this parameter specifies the maximum time of a long lasting operation over the site collection being processed, which may accidentally hang. These operations include the ones that set temporary processing statuses into correspondent rows of the list “Deployment Requests” ending with “ing” suffixes, for example, Deleting, Removing, and Creating.
* Do not change the default value of this parameter without necessity.

$dateTimeStampFormatLong

* [string], the default value is "yyyy-MM-dd HH:mm"
* This is an internal service parameter used by the logic of the script. It defines the long format of date and time outputs to the screen made by the logic while the script is processing.
* Do not change the default value of this parameter without necessity.

$dateTimeStampFormatShort

* [string], the default value is "HH:mm:ss"
* This is an internal service parameter used by the logic of the script. It defines the short format of date and time outputs to the screen made by the logic while the script is processing.
* Do not change the default value of this parameter without necessity.

$deployLegacySolutionsForCustomWebTemplates

* [bool], the default value is $true
* This is an internal service parameter used by the logic of the script. It defines the flag that allows the automated deployment of sandbox solutions by the script [4\_DeployLegacySolutions.ps1](#DeployLegacySolutions) to a newly created site collection that has a custom web template specified in the request details for this site collection (in the correspondent row of the list “Deployment Requests”). Custom web templates are the legacy web templates that are usually deployed by correspondent custom features of sandbox solutions.
* Do not change the default value of this parameter without necessity.

$maxSitesToProcessInSingleRun

* [int], the default value is 10.
* This is an internal service parameter used by the logic of the script. It limits max amount of site collections that can be processed in a single execution of this script (5\_CreateRequestedSites.ps1). If there are more pending requests they should be processed later in the new instances of this script.
* Do not change the default value of this parameter without necessity.

$defaultSiteAdministrators

* [string[]], array of strings, the default value is an empty array, @().
* This parameter allows adding extra accounts of site collection administrators to all newly created site collections by default.
* The parameter can be used in the case of intensive development.

$defaultVisitors

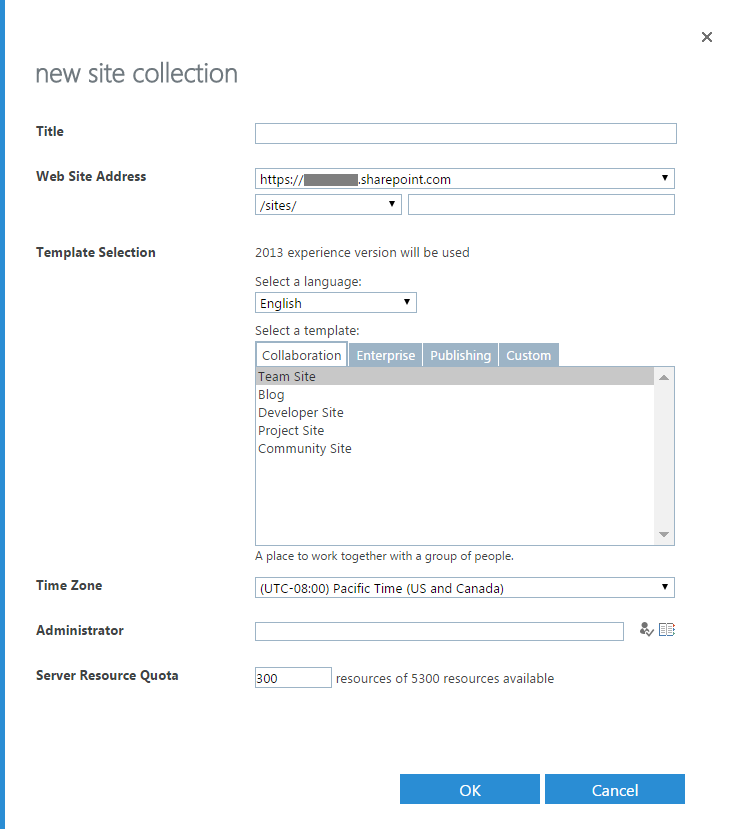
* [string[]], array of strings, the default value is an empty array, @().
* This parameter allows adding extra accounts to the standard group of Visitors of all newly created site collections by default.
* The parameter can be used in the case of intensive development.

The main folder also includes 5 sub-folders

* **csom-dlls** , this folder includes standalone copies of the necessary Microsoft CSOM DLLs.
* Microsoft.Online.SharePoint.Client.Tenant.dll
* Microsoft.SharePoint.Client.dll
* Microsoft.SharePoint.Client.Publishing.dll
* Microsoft.SharePoint.Client.Runtime.dll
* Microsoft.SharePoint.Client.Search.dll
* Microsoft.SharePoint.Client.Taxonomy.dll
* Microsoft.SharePoint.Client.UserProfiles.dll
* **customizations** , this folder stores all deployable customizations; its content can be easily altered.
* **css**
* custom-ui.css
  + - *This CSS-file is loaded by default in every page of a site collection by a specially designed User Action activated when the branding is applied to a site collection.*
* SPO-Responsive.css
  + - *This CSS-file provides a basic* [*Responsive UI*](https://github.com/OfficeDev/PnP-Tools/tree/master/Solutions/SharePoint.UI.Responsive) *for SPO site collections. It is loaded by default in every page of a site collection by a specially designed User Action activated when the branding is applied to a site collection.*
* **Images**
  + logo.png
    - *This is a default file of logo loaded by the CSS-file “custom-ui.css”. You can easily replace this image with your own one and adjust related CSS-styles in “custom-ui.css” accordingly.*
  + title-row-bg.png
    - *This is an optional background image of a title row in the standard master page “seattle.master” of SharePoint Online (“s4-workspaces”). This is loaded by the CSS-file “custom-ui.css”. You can easily replace this image with your own one and adjust related CSS-styles in “custom-ui.css” accordingly.*
* **scripts**
* custom-sa-create-site.js
  + - *This script adds a special menu item to the standard menu “Site Actions” of the* [*Deployment hub*](#BrandingHub) *site collection. It also provides the custom UI and logic that allows making the postponed requests to create site collections by SPO-CDF. You can see more details below in this document. The logic of this script is available for the site collection administrators only (“soft” restriction).*
* custom-ui.js
  + - *This script is loaded by default in the custom action deployed by SPO-CDF when the branding is applied to a site collection.*
* jquery-3.1.0.min.js
  + - *This script is loaded by default in the custom action deployed by SPO-CDF when the branding is applied to a site collection.*
* SPO-Responsive.js
  + - *This script is loaded by default in the custom action deployed by SPO-CDF when the branding is applied to a site collection.* *This is a part of* [*Responsive UI*](https://github.com/OfficeDev/PnP-Tools/tree/master/Solutions/SharePoint.UI.Responsive) *package and works in conjunction with its CSS-file SPO-Responsive.css.*
* wt-custom.js
  + - *This script is loaded and used by the internal logic of “custom-sa-create-site.js”. It contains a self-executing JS-function, which defines the custom web-templates visible in the custom UI generated by the execution of “custom-sa-create-site.js”.*
* wt-standard.js
  + - *This script is loaded and used by the internal logic of “custom-sa-create-site.js”. It contains a self-executing JS-function, which defines the standard web-templates visible in the custom UI generated by the execution of “custom-sa-create-site.js”.*
    - *Content of this file is (re)created by running the environment initiation script* [*\_\_EnvironmentSetup.ps1*](#EnvironmentSetup)*.*
* **legacy**
* **sandbox-wsps** , this folder stores all optionally deployable legacy sandbox solutions.
  + CDF.LegacyBranding.wsp
  + CDF.LegacyWebTemplates.wsp
  + CDF.LegacyListsAndLibraries.wsp
* **logs** , this folder stores various log and cache files produced by SPO-CDF.
* \_\_site-states.csv
* \_\_admins-pending.csv
* log-2016-10-12-12-07.txt
* log-2016-10-12-12-08.txt
* **utils** , this folder stores various initiation scripts for the environment described in the correspondent chapters.
* \_\_adjust-internal-wa-to-tenant-admin.ps1
* \_\_EnvironmentSetup.ps1

## User interface to request creation of site collections

SPO-CDF supports branding and customizations of OOB site collections created using the standard tools of SharePoint Online, for example, the dialog “New site collection” available through the SharePoint admin center.



By obvious security reasons, this dialog is not available outside the context of the SharePoint admin center. These restrictions make it impossible to reuse the URL and the functionality of this dialog in the custom UI.

However, the OOB dialog “New SharePoint Site” that provides creation of a subsite is available in any site collection for authorized users. In general, it contains the UI elements and the functionality, which looks similar to the OOB dialog “New site collection”.

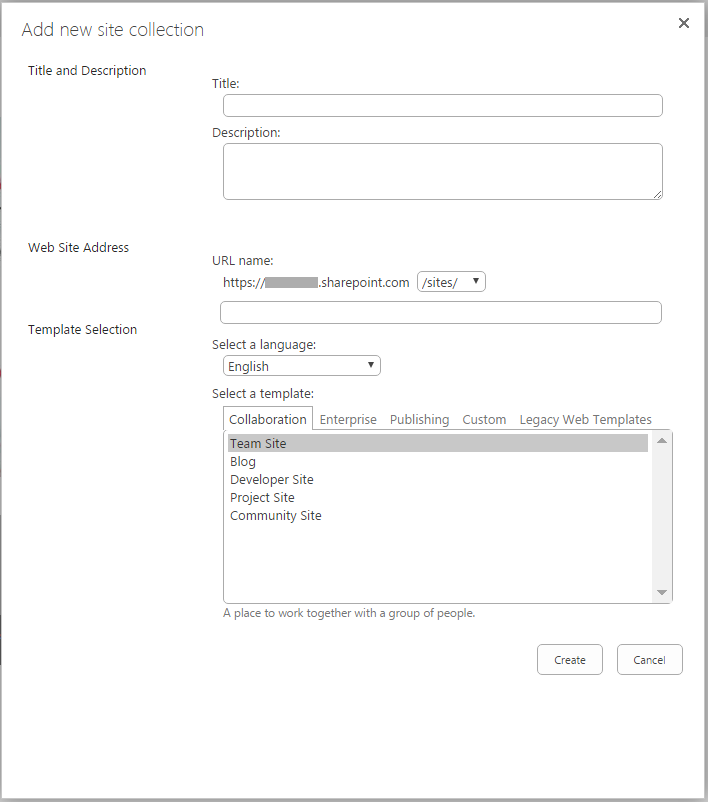
So I decided to reuse the OOB dialog “New SharePoint Site” in SPO-CDF and dynamically adjust its UI to add missing elements required by the logic that creates site collections and hide redundant ones.

* In compare with the OOB dialog “New site collection” the standard dialog “New SharePoint Site” has significantly reduced amount of available web-templates and their descriptions in multiple languages. It also does not have the selection of managed path. Other missing options do not look too important and can be easily replaced with predefined default values (selection of time zone, administrator, and server resource quota).
* Redundant elements of the dialog “New SharePoint Site” mainly include multiple options to adjust permissions; they can be easily made hidden.

As mentioned earlier in this document, there is a file “custom-sa-create-site.js”, which SPO-CDF uploads to the [Deployment hub](#BrandingHub); its default site relative URL corresponds to “\_catalogs/masterpage/customizations/scripts/ custom-sa-create-site.js“.

The script [3\_UpdateSiteCollection.ps1](#UpdateSiteCollection) also adds a custom menu item “Add new site collection” to the “Site Actions” with the functionality available to the site collection administrators of the [Deployment hub](#BrandingHub) (“soft” protection).

When the administrator clicks on this menu item it opens a dialog that has the custom UI made of the adjusted OOB dialog “New SharePoint Site”. This dialog looks like shown on the picture below; you can compare it with the screenshot above.

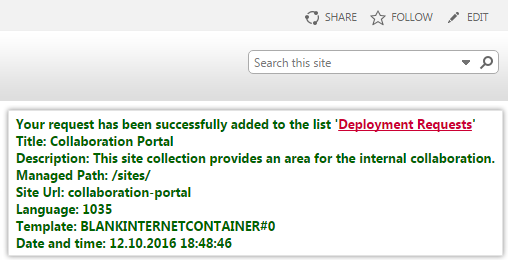


The logic of the adjusted dialog looks familiar to the user and still supports OOB validation of empty fields, formats of entered URLs etc.

* The reuse of the OOB dialog allowed minimizing level of customizations in compare with a fully tailor made UI.
* In general, the standard operations like switching the selected language or navigation through the selected categories work very similarly to the ones found in the OOB dialog ”New site collection”.
* There is a simple protection against undesired reloads of the dialog, which wipes out the applied UI adjustments. If the user tries to reload the content of the adjusted dialog the special verification logic identifies the reloaded state and just closes the dialog (the process of identification and closing takes ~2.5 seconds by default).
* Optionally, the user can request creation of a site collection from a predefined set of Legacy Web Templates usually deployable via sandbox solutions with auto-activated site-scoped features. Obviously, this option is not available in the OOB dialog “New site collection”; however, it could be easily added into the adjusted UI.
  + The configurable set of predefined Legacy Web Templates is (re)deployed via the script customizations\scripts\wt-custom.js described earlier in this document and deployable via the script [\_\_EnvironmentSetup.ps1](#EnvironmentSetup).

**Usage and the logic behind it**

After the user has entered all necessary parameters required to create a new site collection, he or she clicks the button “Create”. The creation process does not start immediately. Instead, the creation request is added to the custom list “Deployment Requests” and the user sees the confirmation box similar to the one shown below (or the error message in case of accidental failure). The list “Deployment Requests” is automatically created on the same site, if not yet exists (usually, on the [Deployment hub](#BrandingHub)).



**Creation process**

Postponed creation of the site collection is performed by the script [5\_CreateRequestedSites.ps1](#CreateRequestedSites) that executes periodically by the script [1\_ContinuousDeployment.ps1](#ContinuousDeployment) as a part of the group execution.

In general, while the script site [5\_CreateRequestedSites.ps1](#CreateRequestedSites) processes a request to create a site collection it changes values of the fields “Status“, “Status message” and optionally “Updated Status Message” in a correspondent list item.

* There are 9 predefined statuses that reflect the current state of a site collection request being processed. Those statuses can be easily monitored for each requested site collection in the list “Deployment Requests”.
  + Requested
  + Creating
  + Created
  + CreatedNeedsCustomTemplate
  + CreatedCustomTemplateFailed
  + CreatedCustomTemplateApplied
  + Deleting
  + Deleted
  + Failed
* If a user has requested creating a site collection with a custom template, the script first creates a site collection with an empty template, uploads sandbox solutions into the created site collection, searches through the available custom templates, and, in the case of success, applies a found template to the root site of a site collection. This option allows supporting the existing legacy solutions.
* If the operation “Creating” or ”Deleting” has failed in the middle the status is automatically restored to “Requested” after the predefined interval. Refer to the description of [5\_CreateRequestedSites.ps1](#CreateRequestedSites) for details.
* If another operation has failed the previous status of not automatically restored.
* The operations to delete the existing site collections are not enabled in the default configuration of SPO-CDF, however, can be easily supported by changing the correspondent parameters. Refer to the description of [5\_CreateRequestedSites.ps1](#CreateRequestedSites) for details.

# How to use SPO-CDF

You can run all the components of SPO-CDF on any Windows machine that has [Powershell 3.0](https://www.microsoft.com/en-us/download/details.aspx?id=34595) installed. Note the default package of SPO-CDF does not require any extra components like SharePoint Online Management Shell, Azure Powershell or SDK, etc.

Try the work of SPO-CDF first in some non-critical environment. I would recommend using a Trial Tenant of Office 365 for the testing purposes.

The usage of SPO-CDF is fairly simple.

* Unpack the archive into any appropriate location (folder in the file system).
  + Do not put it too deep in the file system. Optimally, <dvive>:\spo-cdf
* Open the script file [\_\_LoadContext.ps1](#LoadContext) and adjust the parameters in the header:
  + $siteCollectionUrl, specify here the top root site of your SharePoint Online service
  + $username, use the account that belongs at least to the role of SharePoint Administrator. This is also possible to use the account of Global Admin however this level of privileges looks a bit excessive.
  + $password
* Run the script utils\ \_\_EnvironmentSetup.ps1 to initialize your environment of SharePoint Online and create a [Deployment hub](#BrandingHub).
  + The Deployment hub is created in the top root site collection by default.
  + You can create it in another site collection by adjusting the parameter $staticUrlWithCustomizations of the script utils\ [\_\_EnvironmentSetup.ps1](#EnvironmentSetup). If you do so, you should also adjust the parameter with the same name in the scripts [3\_UpdateSiteCollection.ps1](#UpdateSiteCollection) and [5\_CreateRequestedSites.ps1](#CreateRequestedSites).
* Run the script [1\_ContinuousDeployment.ps1](#ContinuousDeployment). This script starts the infinite loop with the default interval of 30 seconds between sessions. Refer to the description of this script for details.
  + You can adjust the default interval to any other one if you find it inappropriate for your environment.

Obviously, you can change any deployable customizations stored in the subfolders “customizations” and “legacy” below the main folder of SPO-CDF. The descriptions of the content stored in these subfolders have been given above in this document.

## Troubleshooting connection problems

If your connection attempt failed with the error "The remote server returned an error: (403) Forbidden." make sure you are connecting to the correct environment using valid credentials and correct configuration parameters.

* Open the script file [\_\_LoadContext.ps1](#LoadContext).
* Verify that the value of the parameter $siteCollectionUrl corresponds to your target environment.
* Verify the credentials specified in the configuration parameters $username and $password are correct and correspond to the chosen environment.
  + You can also try changing the account to the Global Administrator to see if it has any influence (there should not be any difference with more limited role of “SharePoint Administrator” in regular conditions).
  + Occasionally, the connection to a Tenant may become temporarily unavailable. This condition can be identified by the error messages that state “underlying connection was closed” and “could not connect to Tenant”. In this case it may help to wait for some time before retrying connection attempts.
* When you connect to SharePoint Online both configuration parameters $useLocalEnvironment and $useDefaultCredentials must be set to $false.
  + The value $true is reserved for the possible future extensions to the local SharePoint environment; not in use by SPO-CDF at the moment.