Patch.Deploy

Method	Deploy
Module	Patch
Library	Core
Action	Deploy (i.e. download and by default install) either all available patches or specific patches to this device.
Parameters	`Source` (string): The source of patch meta-data, typically the installation mechanism used to handle patch(es). One of: • `CAB`: A file with a `.cab` extension and appropriate binary structure containing update metadata, located somewhere in the file system available to the local agent. The file is used directly by the local Windows Update agent, without SCCM or WSUS being involved. • `SCCM`: System Center Configuration Manager. • `WSUSL`: A local (to the enterprise) corporate Windows Server Update Service server. • `WSUSR`: The remote Windows Server Update Service feed over the internet at microsoft.com . ⚠ Case is not significant. `PatchSpec` (string; optional, defaults to an empty string implying **all** available patches will be deployed): The patches to be actioned. ⚠ For Windows, this is a comma-separated list of Knowledge Base article numbers (*without* the "KB" prefix) or the GUIDs of specific patches, or even a mixture. (See example below.) Case is not significant for the hex digits in GUIDs. `DownloadOnly` (boolean; default `false`): Whether the patch should be downloaded but not be installed if it is not already installed. ⚠ DownloadOnly does not apply if the source is `SCCM`. The SCCM API does not support downloading only, whereas WSUS does. `CabFilePath` (string; default empty): The full path of the location of the `.cab` file if Source is 'CAB'. ⚠ This *must* be specified if `Source` is 'CAB' and should not be specified for any other `Source` value. If the path is specified it must be to a local CAB-file. Shared, i.e. remote, CAB-files are not supported by Windows Update Agent. `CheckOnline` (bool; default `false`): Grant permission (or not) as to whether to go over the network or remain purely local to the host. If this is `false`, patches can only be installed if they have already been downloaded to the local cache. With SCCM this may trigger installs that have already been identified and downloaded but not yet installed (ie they are ready and waiting for the deadline). A value of `true` is incompatible with `CabFilePath`. ⚠ Setting this to `true` may cause a lot of network traffic if many devices run this method simultaneously. For a `Source` of `WSUSR` the "network" in question is the internet, and for `WSUSL` and `SCCM` it is the LAN.

	`Asynchronous` (bool: default `false`): If `false`, then the patch will happen inline with the instruction execution, meaning the instruction could take many minutes to complete and the Agent will be unable to respond to other patch instructions effectively. If `true` then no result set is returned and a follow up call some time later to List would be needed to determine whether the patch is now installed. ⚠ Synchronous (`Asynchronous` = `false`) patching with source SCCM is achieved by eventing and polling WMI classes as the CM client is totally independent of Tachyon and makes its own decisions. Tachyon requests the SCCM client to do something and monitors the progress to see whether the request has started and finished. There is a good chance that the installation will complete within the ten minutes allowed for 'synchronous' completion of this method, however Tachyon may decide that CM Client has taken too long and will report back that the timeout has occurred. The CM Client driven installation will however continue in the background though and the final status can be determined via `Patch.List` using the appropriate PatchSpec.
	`TimeoutSecs` (int; optional, default `1800`): If the Source does not complete within the specified number of seconds, return anyway. A value of 0 means an eternal timeout, i.e. none. A negative value is illegal. Available from v4.0. ⚠ This parameter is incompatible with `Asynchronous` = `true`, which will cause the instruction to return success before the operation has even started, let alone completed. If the timeout occurs the install will continue in the background as though nothing happened but `Success` is returned from the method and for any rows in the result set the `InstallError` and `InstallResult` columns will indicate that the timeout occurred. It will be necessary to call List to determine the new state of the device.
Return values	For each of the supplied patches or if not supplied for all available patches the following is returned: ⚠ If no update information is available then the method will return success with no rows.
	`PatchSpec` (string): The patch identifier. For Windows, a Knowledge Base article number.
	`DownloadOnly` (boolean): Whether the patch was to be just staged (`false`) or also installed (`true`), i.e. the supplied or implicit `DownloadOnly` parameter.
	`NeededDownload` (boolean): Whether the patch actually needed to be downloaded.
	`DownloadResult` (integer): The COM success (0) or error (not 0) codes related to downloading, otherwise empty if no download is attempted.
	`DownloadError` (string): The human readable form of the `DownloadedResult` column if a download error occurred.
	`NeededInstallation` (boolean): Whether the patch actually needed to be installed (`true`) or it was already installed (`false`).
	`InstallResult` (integer): The COM success (0) or error (not 0) codes related to installation, otherwise empty (`null`) if installation is not attempted.
	`InstallError` (string): The human readable form of `InstallResult` if an installation error occurred.
	`RebootRequired` (bool): If `true` then a reboot is required after the patch was installed, if `false` then a reboot is not required. If `DownloadOnly` was requested, indicates whether a reboot will (or may) be required following installation.
Example	`Patch.Deploy(Source:"SCCM", PatchSpec:"2267602,3182545", DownloadOnly:true);`
Platforms	• Windows
Notes	Updates will be installed only if they do not require user input. However, any license agreements are accepted automatically. Windows may refuse to allow installations if a reboot is pending. If no update information is available then the method will return success with no rows. The method does not fail with an error if the `PatchSpec` indicates a patch (or patches) that is not available, whether the source does not have it or the patch simply does not exist. In this case a sparse response record will be returned with empty `DownloadResult` and `InstallResult` values, and the Agent log will contain a "`No updates to install`" warning.