Windows Server Update Services

Windows Server Update Services (WSUS) est un service permettant de distribuer les mises à jour pour Windows et d'autres applications Microsoft sur les différents ordinateurs fonctionnant sous Windows au sein d'un parc informatique. WSUS est un rôle pour serveur Windows lui permettant ainsi de devenir un serveur de mises à jour local (ou proxy de mises à jour). Ce serveur télécharge et stocke ponctuellement l'ensemble des mises à jour disponibles auprès des serveurs Windows Update de Microsoft et rend possible le contrôle de la diffusion de celles-ci dans le parc.

Outils autour de WSUS

Wsus Package Publisher permet de publier vos propres mises à jour sous forme de fichiers MSI, MSP ou EXE. Ainsi, vous pouvez déployer des applications telles qu'Adobe Reader, Java, Flash Player ou Symantec Endpoint Protection. Et les mettre à jour.

Vous pourrez importer des mises à jour à partir de catalogues d'éditeur (Adobe, Dell, HP...). Afin de publier des pilotes ou firmware pour votre matériels.

Official git repo : https://github.com/DCourtel/Wsus_Package_Publisher
Tutoriel Vidéo : PlayList de Francis Bonnamour

WAPT

WAPT est un logiciel de déploiement son noyau est sous licence GPLv3.

Site Officiel : https://wapt.fr/
Documentation : https://wapt.fr/en/doc/
Official git repo : https://github.com/tranquilit/WAPT

WSUS Offline

WSUS Offline Update est un logiciel de gestion des mises à jour destiné à certaines applications et systèmes d'exploitation de Microsoft. Contrairement au service WSUS de Microsoft, WSUS Offline Update permet d'enregistrer les mises à jour sur un média pour les distribuer vers différents postes d'un parc informatique hors-ligne (offline) au réseau Internet.
Site Officiel : https://www.wsusoffline.net/

Script

Ici, je vais positionné certains scripts que j'ai trouvé pour gagner de la place.

Function Invoke-WsusDeclineAllSuperSeded

Celui-ci utilise l'api Web de WSUS pour décliner l'ensemble des updates dites “Superseded”, il faudra ensuite lancer un nettoyage dans le serveur.

```
function Invoke-WsusDeclineAllSuperSeded {
    <#
    .SYNOPSIS
        On Wsus, performs a find all updates with IsSuperseded with value
True to Decline this update via web api.
    .EXAMPLE
        PS> Invoke-WsusDeclineAllSuperSeded
        Performs a find all updates with IsSuperseded with value True to
Decline this update via local web api.
    .EXAMPLE
        PS> Invoke-WsusDeclineAllSuperSeded -FQDN wsus.example.org -
portNumber 8531 -useSecureConnection $true
        Performs a find all updates with IsSuperseded with value True to
Decline this update via remove web api, use the port number 8531 and use
secure connection.
    .PARAMETER FQDN
        Set the fully qualified domain name like this : myhost.example.com.
The FQDN uniquely distinguishes the device from any other hosts called
myhost in other domains.
    .PARAMETER portNumber
        Set the communication port, by default this parameter is set on
8530.
    .PARAMETER useSecureConnection
        If your Wsus use a SSL certificate, please set this parameter to
$true, by default this parameter is set on $false.

    #>
    param(
        [Parameter(HelpMessage = 'Set the fully qualified domain name like
this : myhost.example.com. The FQDN uniquely distinguishes the device from
any other hosts called myhost in other domains.')]
        [String]$FQDN,
        [Parameter(HelpMessage = 'If your Wsus use a SSL certificate, please
set this parameter to $true, by default this parameter is set on $false.')]
        [Boolean]$useSecureConnection = $false,
        [Parameter(HelpMessage = 'Set the communication port, by default
this parameter is set on 8530')]
        [Int32]$portNumber = 8530
    )

    if (!$FQDN) {
        [String]$FQDN = $env:COMPUTERNAME + $(if (($null -eq
$env:USERDNSDOMAIN) -eq $false) { '.' + $env:USERDNSDOMAIN })
    }
```

```
    # Load .NET assembly

[void][reflection.assembly]::LoadWithPartialName("Microsoft.UpdateServices.A
dministration")

    [Int32]$count = 0

    # Connect to WSUS Server

    Try {
        $updateServer =
[Microsoft.UpdateServices.Administration.AdminProxy]::getUpdateServer($FQDN,
$useSecureConnection, $portNumber)
        write-output "Connected sucessfully"
    }
    Catch {
        Write-Warning "An error occurred:"
        Write-Error $_
        Break
    }


    $updateServer.GetUpdates($(New-Object
Microsoft.UpdateServices.Administration.UpdateScope)) | ForEach-Object {
        if ($_.IsSuperseded -eq 'True') {
            Write-Output ("Decline Update : $_.Title")
            $_.Decline()
            $count = $count + 1
        }
    }
    Total Declined Updates: $count

    trap {
        Write-Warning -Message 'Error Occurred'
        Write-Warning -Message 'Exception Message: '
        Write-Error $_.Exception.Message
        Write-Error $_.Exception.StackTrace
        exit
    }
}
# Please Set your settings here
Invoke-WsusDeclineAllSuperSeded
# EOF
```

Invoke-DeclineUpdates

Une version alternative que j'ai trouvé ici, il y a de bonne idée mais, j'ai préféré ne pas mettre en place, car il utilise les rsat de WSUS.

```
<#
.Synopsis
   Sample script to decline superseded updates from WSUS, and run WSUS
cleanup if any changes are made

.DESCRIPTION
   Declines updates from WSUS if update meets any of the following:
        - is superseded
        - is expired (as defined by Microsoft)
        - is for x86 or itanium operating systems
        - is for Windows XP
        - is a language pack
        - is for old versions of Internet Explorer (versions 7,8,9)
        - contains some country names for country specific updates not
filtered by WSUS language filters.
        - is a beta update
        - is for an embedded operating system

    If an update is released for multiple operating systems, and one or more
of the above criteria are met, the versions of the update that do not meet
the above will not be declined by this script

.EXAMPLE
   .\Invoke-DeclineUpdates -WSUSServer WSUSServer.Company.com -WSUSPort 8530

# Last updated 13 July 2016

# Author
Nick Eales, Microsoft
#>


Param(
    [Parameter(Mandatory=$false,
    ValueFromPipeline=$true,
    ValueFromPipelineByPropertyName=$true,
    ValueFromRemainingArguments=$false,
    Position=0)]
    [string]$WSUSServer = "Localhost", #default to localhost
    [int]$WSUSPort=8530,
    [switch]$reportonly
    )

Function Invoke-DeclineUpdates{
    Param(
        [string]$WsusServer,
        [int]$WSUSPort,
        [switch]$ReportOnly
    )
    write-host "Connecting to WSUS Server $WSUSServer and getting list of
updates"
```

```
    $Wsus = Get-WSUSserver -Name $WSUSServer -PortNumber $WSUSPort
    if($Null -eq $WSUS){
        write-error "unable to contact WSUSServer $WSUSServer"
    }else{
        $Updates = $wsus.GetUpdates()
        write-host "$(($Updates | Where-Object {$_.IsDeclined -eq $false} |
Measure-Object).Count) Updates before cleanup"
        $updatesToDecline = $updates | Where-Object {$_.IsDeclined -eq
$false -and (
        $_.IsSuperseded -eq $true -or   #remove superseded updates
        $_.PublicationState -eq "Expired" -or #remove updates that have been
pulled by Microsoft
        $_.LegacyName -match "ia64" -or #remove updates for itanium
computers (1/2)
        $_.LegacyName -match "x86" -or  #remove updates for 32-bit computers
        $_.LegacyName -match "XP" -or   #remove Windows XP updates (1/2)
        $_.producttitles -match "XP" -or #remove Windows XP updates (1/2)
        $_.Title -match "Itanium" -or   #remove updates for itanium
computers (2/2)
        $_.Title -match "language\s" -or  #remove langauge packs
        $_.title -match "Internet Explorer 7" -or #remove updates for old
versions of IE
        $_.title -match "Internet Explorer 8" -or
        $_.title -match "Internet Explorer 9" -or
        $_.title -match "Japanese" -or #some non-english updates are not
filtered by WSUS language filtering
        $_.title -match "Korean" -or
        $_.title -match "Taiwan" -or
        $_.Title -match "Beta" -or     #Beta products and beta updates
        $_.title -match "Embedded"     #Embedded version of Windows
        )}
        write-host "$(($updatesToDecline | Measure-Object).Count) Updates to
decline"
        $changemade = $false
        if($reportonly){
            write-host "ReportOnly was set to true, so not making any
changes"
        }else{
            $changemade = $true
            $updatesToDecline | ForEach-Object{$_.Decline()}
        }

        #Decline updates released more then 3 months prior to the release of
an included service pack
        # - service packs updates don't appear to contain the supersedance
information.
        Foreach($SP in $($updates | Where-Object title -match "^Windows
Server \d{4} .* Service Pack \d")){
            if(($SP.ProductTitles |Measure-Object ).count -eq 1){
                $updatesToDecline = $updates | Where-Object {$_.IsDeclined -
eq $false -and $_.ProductTitles -contains $SP.ProductTitles -and
```

```
$_.CreationDate -lt $SP.CreationDate.Addmonths(-3)}
                if($null -ne $updatesToDecline){
                    write-host "$(($updatesToDecline | Measure-
Object).Count) Updates to decline (superseded by $($SP.Title))"
                    if(-not $reportonly){
                        $changemade = $true
                        $updatesToDecline | ForEach-Object{$_.Decline()}
                    }
                }
            }
        }
        #if changes were made, run a WSUS cleanup to recover disk space
        if($changemade -eq $true -and $reportonly -eq $false){
            $Updates = $wsus.GetUpdates()
            write-host "$(($Updates | Where-Object {$_.IsDeclined -eq
$false} | Measure-Object).Count) Updates remaining, running WSUS cleanup"
            Invoke-WsusServerCleanup -updateServer $WSUS -
CleanupObsoleteComputers -CleanupUnneededContentFiles -
CleanupObsoleteUpdates -CompressUpdates -DeclineExpiredUpdates -
DeclineSupersededUpdates
        }

    }
}

Invoke-DeclineUpdates -WSUSServer $WSUSServer -WSUSPort $WSUSPort -
reportonly:$reportonly
```