**eBook Gratuit**

# APPRENEZ nsis

eBook gratuit non affilié créé à partir des **contributeurs de Stack Overflow.**

#nsis

# Table des matières

# À propos

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: nsis

It is an unofficial and free nsis ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official nsis.



Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

# Chapitre 1: Démarrer avec nsis

## Remarques

NSIS (Nullsoft Scriptable Install System) est un système open source professionnel permettant de créer des installateurs Windows. Il est conçu pour être aussi petit et flexible que possible et convient donc parfaitement à la distribution sur Internet.

En tant que première expérience d'un utilisateur avec votre produit, un installateur stable et fiable est un composant important d'un logiciel performant. Avec NSIS, vous pouvez créer de tels installateurs capables de faire tout ce qui est nécessaire pour configurer votre logiciel.

NSIS est basé sur des scripts et vous permet de créer la logique nécessaire pour gérer les tâches d'installation les plus complexes. De nombreux plug-ins et scripts sont déjà disponibles: vous pouvez créer des programmes d'installation Web, communiquer avec Windows et d'autres composants logiciels, installer ou mettre à jour des composants partagés, etc.

-> Extrait de: https://sourceforge.net/projects/nsis/

## Versions

| Version | Remarques | Date de sortie |
|---|---|---|
| 2.00.0 | | 2004-02-07 |
| 2,46,0 | Probablement la version la plus répandue dans la nature. | 2009-12-06 |
| 2.51.0 | La version finale 2.x contient des correctifs de sécurité importants. | 2016-04-01 |
| 3.00.0 | Première version avec support officiel Unicode. | 2016-07-24 |

## Examples

### Installation ou configuration

Le programme d'installation de NSIS peut être téléchargé à l' adresse http://nsis.sourceforge.net/Download . Un exe de 1,6 Mo sera téléchargé. Vous pouvez l'installer en utilisant l'assistant. Lors de l'installation, il y a des options à installer

```
1. Full: Installs all the components
2. Lite: Basic and only essential components from the UI
3. Minimal: The NSIS core files only.
4. Custom: It's up to us to install whichever components that we need.
```

## Bonjour le monde!

Code à enregistrer dans „helloworld.nsi“:

```
Name "Hello World"
OutFile "helloworld.exe"
Section "Hello World"
MessageBox MB_OK "Hello World!"
SectionEnd
```

Compilez-le avec:

```
<Path to NSIS>\makensis.exe <Path to script>\helloworld.nsi
```

Lire Démarrer avec nsis en ligne: https://riptutorial.com/fr/nsis/topic/5491/demarrer-avec-nsis

# Chapitre 2: Enregistrement de bibliothèques (RegDLL)

## Introduction

En bref, une DLL est une collection de petit code exécutable, qui peut être appelée au besoin par un programme en cours d'exécution. La DLL permet à l'exécutable de communiquer avec un périphérique spécifique tel qu'une imprimante ou peut contenir du code pour effectuer un nombre quelconque de fonctions particulières. Comme il existe plusieurs méthodes d'implémentation pour ce faire, dans cette rubrique, je vous montrerai comment enregistrer et désenregistrer les DLL que votre application appelle; et nous allons le faire en utilisant la ligne de commande `RegSrv32.exe` .

## Syntaxe

- `${Register::DLL} "codec.ocx" "" $1 $0`
- `${UnRegister::DLL} "volumeCtrl.cpl" /DISABLEFSR $1 $0`
- `${RegisterDLL} "print.drv"`

## Paramètres

| Commutateur | Discription |
|---|---|
| `/u` | Annule le serveur. |
| `/s` | Indique que `regsvr32` doit s'exécuter en mode silencieux et ne pas afficher de message. |
| `/n` | Spécifie de ne pas appeler *DllRegisterServer* . Vous devez utiliser cette option avec `/i` . |
| `/i` **: cmdline** | Appelle *DllInstall en lui* passant une option **[cmdline]** . Lorsqu'il est utilisé avec `/u` , il appelle désinstaller dll. |
| **dllname** | Spécifie le nom du fichier dll à enregistrer. |
| `/?` | Affiche l'aide à l'invite de commande. |

## Remarques

# RegSrv32.exe

L'utilisation de la ligne de commande `RegSvr32.exe` est la méthode que je préfère pour enregistrer

vos bibliothèques. Commençons ici d'abord. Les PC Windows couplés à Internet Explorer 3.0 ou version ultérieure sont disponibles avec `RegSvr32.exe` . Il y a donc de bonnes chances que votre PC soit livré en standard avec cet utilitaire. Maintenant, si vous utilisez une machine 64 bits, vous pouvez envisager deux variantes. Ils peuvent être trouvés dans `$WINDIR\system32` ou `$WINDIR\SysWow32` .

Les paramètres que vous pouvez utiliser avec RegSrv32 sont `/u /s /i /n` . Le commutateur de commande `/u` désenregistrer le fichier. Le commutateur `/i` peut être utilisé avec `/u` pour appeler à la désinstallation de DLL. Le paramètre `/n` n'appellera pas *DllRegisterServer* ; il est utilisé avec `/i` qui est le commutateur d'installation. Si vous utilisez `/s` , qui signifie silencieux, aucune boîte de message ne sera affichée sur Windows XP ou version ultérieure.

Lorsque vous utilisez `RegSvr32.exe` partir de la ligne de commande, vous obtenez des boîtes de message après l'avoir appelé. La fonction *DLLSelfRegister* sera invoquée, sauf si vous utilisez le commutateur susmentionné bien sûr; Si elle réussit, une boîte d'alerte indiquant son succès s'affiche, identique à celle qui génère un message d'erreur.

> D'après mon expérience, `RegSvr32.exe` x64 enregistre correctement les DLL x86 sur Windows Vista et versions ultérieures (exclut XP; consultez cet article pour plus d'informations). En outre, Windows XP est un art en voie de disparition; bénis son coeur. =)

# Examples

## RegSrv32.exe Utilisation simple

```
#= Regardless of architecture we're using just the following

!define REGSVR `$SYSDIR\regsvr32.exe` #= define where RegSrv32 is
!define DLL `$AppDirectory\App\MyLegalProgram\myLegit.dll` #= define the file to register

##=
#= Command line usage is the same for both variants of RegSrv32 as follows
#= regsvr32 [/u] [/s] [/n] [/i[:cmdline]] DLL
#=
##= So in our .nsi file it would be similar to the following:

Exec `"${REGSVR}" /s "${DLL}"`

#= Moreover, you may also use the following

ExecWait `"${REGSVR}" /s "${DLL}"` $0 #= The $0 will contain the error code if any

#= The above will wait for exe to quit it's process before continuing
```

## Macro: Register :: DLL

```
##=
# The variable $Bit will hold either 64 or 32 depending on system architecture
# This is used with ${Register::DLL} and ${UnRegister::DLL}
# Use this in the beginning of your code.
```

```
# This snippet should only be used once.
#
Var Bit
System::Call "kernel32::GetCurrentProcess()i.s"
System::Call "kernel32::IsWow64Process(is,*i.r0)"
StrCmpS $0 0 +3
StrCpy $Bit 64
Goto +2
StrCpy $Bit 32

##=
#= Register::DLL
#
# USAGE:
# ${Register::DLL} "DLL Filename" /DISABLEFSR $0 $1
#
#        ::DLL       = Registers a DLL file.
#        /DISABLEFSR = Disables redirection if x64. Use "" to skip.
#        $0          = Return after call
#        $1          =  ''      ''    ''
#
!define Register::DLL `!insertmacro _Register::DLL`
!macro _Register::DLL _DLL _FSR _ERR1 _ERR2
    StrCmpS $Bit 64 0 +4
    StrCmp ${_FSR} /DISABLEFSR 0 +3
    ExecDos::Exec /TOSTACK /DISABLEFSR `"${REGSVR}" /s "${_DLL}"`
    Goto +2
    ExecDos::Exec /TOSTACK `"${REGSVR}" /s "${_DLL}"`
    Pop ${_ERR1}
    Pop ${_ERR2}
!macroend

##=
# Alternatively you can use this macro found in my travels
# but you should include x64.nsh as this macro makes use
# of it's function.
#
#= USAGE:
# ${RegisterDLL} "SomeLibrary.dll"
#
!define RegisterDLL "!insertmacro _RegisterDLL"
!macro _RegisterDLL _DLL
    ${If} ${RunningX64}
        ${DisableX64FSRedirection}
        ExecWait '"$SYSDIR\regsvr32.exe" /s "${_DLL}"'
        ${EnableX64FSRedirection}
    ${Else}
        RegDLL "${DLL}"
    ${EndIf}
!macroend
```

Le paramètre / DISABLEFSR ne doit être utilisé que sur les machines x64. Cependant, si vous vous trompez, il y a une sécurité intégrée dans les macros qui éviteront cette balle pour vous.

## Macro: UnRegister :: DLL

```
##=
```

```
# The variable $Bit will hold either 64 or 32 depending on system architecture
# This is used with ${Register::DLL} and ${UnRegister::DLL}
# Use this in the beginning of your code.
# This snippet should only be used once.
#
Var Bit
System::Call "kernel32::GetCurrentProcess()i.s"
System::Call "kernel32::IsWow64Process(is,*i.r0)"
StrCmpS $0 0 +3
StrCpy $Bit 64
Goto +2
StrCpy $Bit 32

#=#
#= UnRegister::DLL
#
# USAGE:
# ${UnRegister::DLL} "DLL Filename" /DISABLEFSR $0 $1
#
#        ::DLL       = Unregisters a DLL file.
#        /DISABLEFSR = Disables redirection if x64. Use "" to skip.
#        $0          = Return after call
#        $1          =  ''      ''    ''
#
!define UnRegister::DLL `!insertmacro _UnRegister::DLL`
!macro _UnRegister::DLL _DLL _FSR _ERR1 _ERR2
    StrCmpS $Bit 64 0 +4
    StrCmp ${_FSR} /DISABLEFSR 0 +3
    ExecDos::Exec /TOSTACK /DISABLEFSR `"${REGSVR}" /s /u "${_DLL}"`
    Goto +2
    ExecDos::Exec /TOSTACK `"${REGSVR}" /s /u "${_DLL}"`
    Pop ${_ERR1}
    Pop ${_ERR2}
!macroend

##=
# Alternatively you can use this macro I found in my travels
# but be sure to include the x64 plugin as this macro makes
# use of it's function.
#
#= USAGE:
# ${UnregisterDLL} "SomeLibrary.dll"
#
!define UnregisterDLL "!insertmacro _UnregisterDLL"
!macro _UnregisterDLL _DLL
    ${If} ${RunningX64}
        ${DisableX64FSRedirection}
        ExecWait '"$SYSDIR\regsvr32.exe" /s /u "${_DLL}"'
        ${EnableX64FSRedirection}
    ${Else}
        UnRegDLL "${DLL}"
    ${EndIf}
!macroend
```

Le paramètre / DISABLEFSR ne doit être utilisé que sur les machines x64. Cependant, si vous vous trompez, il y a une sécurité intégrée dans les macros qui éviteront cette balle pour vous.

Lire Enregistrement de bibliothèques (RegDLL) en ligne:

https://riptutorial.com/fr/nsis/topic/10142/enregistrement-de-bibliotheques--regdll-

# Chapitre 3: Point net

## Introduction

Lorsque le .NET Framework est installé sur un ordinateur, le programme d'installation du .NET écrit des clés de registre lorsque l'installation est réussie. Vous pouvez tester si .NET Framework 4.5 ou une version ultérieure est installé en vérifiant la clé de registre `HKLM\SOFTWARE\Microsoft\NET Framework Setup\NDP\v4\Full` pour obtenir une valeur `DWORD` nommée *Release* . L'existence de cette clé indique que .NET Framework 4.5 ou version ultérieure a été installé sur cet ordinateur.

## Remarques

# Valeurs .NET

Vous trouverez ci-dessous un graphique montrant le numéro de version correspondant à la valeur `DWORD` dans la clé *Release* . Voir ci-dessous ce tableau pour un exemple sur la façon de l'utiliser en action. Je ferai de mon mieux pour essayer de garder cette page à jour avec les dernières valeurs, mais vous pouvez visiter cette page pour une liste mise à jour si la version dont vous avez besoin n'est pas listée ici.

> Notez particulièrement comment il existe deux lignes pour `4.7` , `4.6.2` , `4.6.1` et `4.6` . Ces versions exigent que vous vérifiiez les deux valeurs lorsque les systèmes d'exploitation varient.

| Version | Système opérateur | Valeur de libération |
|---|---|---|
| 4.7 | Mise à jour du créateur Windows 10 | 460798 |
| 4.7 | Tous sauf la mise à jour du créateur de Windows 10 | 460805 |
| 4.6.2 | Mise à jour de l'anniversaire de Windows 10 | 394802 |
| 4.6.2 | Tous sauf mise à jour d'anniversaire de Windows 10 | 394806 |
| 4.6.1 | Mise à jour de Windows 10 novembre | 394254 |
| 4.6.1 | Tous sauf Windows 10 novembre Update | 394271 |
| 4.6 | Windows 10 | 39 3295 |
| 4.6 | Tous sauf Windows 10 | 393297 |
| 4.5.2 | Tout | 379893 |
| 4.5.1 | Windows 8.1 ou Windows Server 2012 R2 | 378675 |

| Version | Système opérateur | Valeur de libération |
|---|---|---|
| 4.5.1 | Windows 8 et Windows 7 | 378758 |
| 4.5 | Tout | 378389 |

# Conclusion

> Je n'ai pas complètement testé ces fonctions, donc si vous rencontrez des problèmes avec l'un d'entre eux, faites-le moi savoir et je ferai de mon mieux pour vous aider. Vous pouvez voir cette fonction fonctionner en action avec SharpDevelop Portable .

Comme je l'ai dit ci-dessus, je ferai de mon mieux pour essayer de garder ces mises à jour, mais si vous avez besoin d'une version que ces fonctions ne vérifient pas, faites-le moi savoir pour pouvoir les réviser et mettre à jour cette page.

## Examples

### Exemple d'échantillon

Voici un exemple d'utilisation des valeurs du registre pour vérifier la présence de dotNET 4.5 ou supérieur. Je recommande de mettre cet extrait de code quelque part comme `.OnInit` car cela s'exécutera avant que quelque chose se passe; De cette façon, il vérifie la présence de .NET avant que les fichiers ne soient copiés ou que les modifications du registre ne soient effectuées.

```
;=#
;= Define the registry key we're looking for.
!define DOTNET  `SOFTWARE\Microsoft\NET Framework Setup\NDP\v4\Full`

Section "Main"
    ClearErrors ;= Clear any errors we may have encountered before hand.
    ReadRegDWORD $0 HKLM `${DOTNET}` `Release` ;= Read the value of Release.
    IfErrors +3 ;= If there is an error than there is no .NET installed so we jump 3 lines
down which lands on MessageBox.
    IntCmp $0 394806 +5 0 +5    ;= Compares the value for v4.6.2 if it matches then we jump 5
lines and avoids the MessageBox
    IntCmp $0 394802 +4 0 +4    ;= Remember to check for Windows 10's value aswell as the
above line won't.
    MessageBox MB_ICONSTOP|MB_TOPMOST `You must have v4.6.2 or greater of the .NET Framework
installed. Launcher aborting!` ;= If the check failed then we alert the user the required
version wasn't found.
    Call Unload ;= We call the Unload function here because we failed the .NET check.
    Quit ;= Closes the Launcher
SectionEnd
```

### Fonction avec macro

Fonction dotNETCheck

Sinon, vous pouvez utiliser la fonction que j'ai écrite ci-dessous. Celui-ci utilise *LogicLib.nsh* . Il

devrait fonctionner sans avoir à connaître la valeur des versions .NET à partir de la clé de *version* dans le registre. Comme il est écrit maintenant, il ne vérifie que les versions entre `4.5` et `4.7` .

---

```
##
# This one requires the use of LogicLib.nsh
# Copy and paste this code somewhere like .OnInit
Function dotNETCheck
    !define CheckDOTNET "!insertmacro _CheckDOTNET"
    !macro _CheckDOTNET _RESULT _VALUE
        Push `${_VALUE}`
        Call dotNETCheck
        Pop ${_RESULT}
    !macroend
    Exch $1
    Push $0
    Push $1

    ${If} $1 == "4.7"
        StrCpy $R1 460798
    ${ElseIf} $1 == "4.6.2"
        StrCpy $R1 394802
    ${ElseIf} $1 == "4.6.1"
        StrCpy $R1 394254
    ${ElseIf} $1 == "4.6"
        StrCpy $R1 393295
    ${ElseIf} $1 == "4.5.2"
        StrCpy $R1 379893
    ${ElseIf} $1 == "4.5.1"
        StrCpy $R1 378675
    ${ElseIf} $1 == "4.5"
        StrCpy $R1 378389
    ${Else}
        Goto dotNET_FALSE
    ${EndIf}

    ReadRegDWORD $R0 HKLM `SOFTWARE\Microsoft\NET Framework Setup\NDP\v4\Full` `Release`
    IfErrors dotNET_FALSE

    IntCmp $R0 $R1 dotNET_TRUE dotNET_FALSE

    dotNET_TRUE:
    StrCpy $0 true
    Goto dotNET_END

    dotNET_FALSE:
    StrCpy $0 false
    SetErrors

    dotNET_END:
    Pop $1
    Exch $0
FunctionEnd

##
# USAGE
# ${CheckDOTNET} $0 "Version Number"
##
# $0 Will hold the version number of the installed .NET
# If $0 is empty ($0 == "") then the error flag is set.
```

```
${CheckDOTNET} $0 "4.5"
IfErrors 0 +4
MessageBox MB_ICONSTOP|MB_TOPMOST `You must have v4.5 or greater of the .NET Framework
installed. Launcher aborting!`
Call Unload
Quit
StrCmpS $0 true 0 -3
```

Lire Point net en ligne: https://riptutorial.com/fr/nsis/topic/10139/point-net

# Chapitre 4: Prestations de service

## Introduction

Lors de l'installation d'un nouveau programme ou de la mise à jour d'une installation, il est recommandé d'arrêter une application installée et tout élément associé avant de remplacer un de ses fichiers. La même chose vaut pour les services. Nous devons nous assurer que le service exécuté localement (le cas échéant) est arrêté avant de pouvoir installer ou mettre à niveau notre programme. Dans cette rubrique, je partagerai quelques macros allant de la création d'un service à l'interrogation d'un service en passant par la suppression d'un tout.

## Syntaxe

- $ {Service :: Create} "NAME" "PATH" "TYPE" "START" "DEPEND" / DISABLEFSR $ 0 $ 1
- $ {Service :: Remove} "NAME" "" $ 0 $ 1
- $ {Service :: QueryConfig} "NAME" / DISABLEFSR $ 0 $ 1

## Paramètres

| $ {Service :: Create} | La description |
|---|---|
| PRÉNOM | Le nom du service |
| CHEMIN | BinaryPathName au fichier .exe |
| TYPE | posséder, partager, interagir, noyau, filesys, rec |
| DÉBUT | démarrage, système, auto, demande, désactivé, auto retardé |
| DÉPENDRE | Dépendances (séparées par / (barre oblique)) |
| / DISABLEFSR | Désactive la redirection si x64. Utilisez "" pour sauter. |
| 0 $ | Retour après appel |
| 1 $ | Retour après appel |

## Examples

### Service :: Créer

```
##=
# The variable $Bit will hold either 64 or 32 depending on system architecture
# This is used with all the ${Service::} macros
# Use this in the beginning of your code.
```

```
# This snippet should only be used once.
#
Var Bit
System::Call "kernel32::GetCurrentProcess()i.s"
System::Call "kernel32::IsWow64Process(is,*i.r0)"
StrCmpS $0 0 +3
StrCpy $Bit 64
Goto +2
StrCpy $Bit 32

##=
#= Service::Create
#
# USAGE:
# ${Service::Create} "NAME" "PATH" "TYPE" "START" "DEPEND" /DISABLEFSR $0 $1
#
#    ::Create    = Creates a service entry in the registry and Service Database
#    NAME        = The Service name
#    PATH        = BinaryPathName to the .exe file
#    TYPE        = own|share|interact|kernel|filesys|rec
#    START       = boot|system|auto|demand|disabled|delayed-auto
#    DEPEND      = Dependencies(separated by / (forward slash))
#    /DISABLEFSR = Disables redirection if x64. Use "" to skip.
#    $0          = Return after call
#    $1          =   ''    ''    ''
#
!define Service::Create `!insertmacro _Service::Create`
!macro _Service::Create _SVC _PATH _TYPE _START _DEPEND _FSR _ERR1 _ERR2
    StrCmpS $Bit 64 0 +7
    StrCmp "${_FSR}" /DISABLEFSR 0 +6
        StrCmp "${_DEPEND}" "" 0 +3
        ExecDos::Exec /TOSTACK /DISABLEFSR `"${SC}" create "${_SVC}" DisplayName=
"${FULLNAME}" binpath= "${_PATH}" type= "${_TYPE}" start= "${_START}"`
        Goto +7
        ExecDos::Exec /TOSTACK /DISABLEFSR `"${SC}" create "${_SVC}" DisplayName=
"${FULLNAME}" binpath= "${_PATH}" type= "${_TYPE}" start= "${_START}" depend= ""${_DEPEND}""`
        Goto +5
    StrCmp "${_DEPEND}" "" 0 +3
    ExecDos::Exec /TOSTACK `"${SC}" create "${_SVC}" DisplayName= "${FULLNAME}" binpath=
"${_PATH}" type= "${_TYPE}" start= "${_START}"`
    Goto +2
    ExecDos::Exec /TOSTACK `"${SC}" create "${_SVC}" DisplayName= "${FULLNAME}" binpath=
"${_PATH}" type= "${_TYPE}" start= "${_START}" depend= ""${_DEPEND}""`
    Pop ${_ERR1}
    Pop ${_ERR2}
!macroend
```

Le paramètre / DISABLEFSR ne doit être utilisé que sur les machines x64. Cependant, si vous vous trompez, il y a une sécurité intégrée dans les macros qui éviteront cette balle pour vous. Cela s'applique à toutes les macros de service répertoriées ici.

## Service :: QueryConfig

```
##=
#= Service::QueryConfig
#
# USAGE:
# ${Service::QueryConfig} "NAME" /DISABLEFSR $0 $1
```

```
#
#    ::QueryConfig = The service's binary path is returned.
#    NAME          = The Service name
#    /DISABLEFSR   = Disables redirection if x64. Use "" to skip.
#    $0            = Return after call | 1 = success
#    $1            =   ''    ''    ''  | Should be the file path
#
# $1 will now hold the path to it's binary executable or an error
#
!define Service::QueryConfig `!insertmacro _Service::QueryConfig`
!macro _Service::QueryConfig _SVC _FSR _ERR1 _ERR2
    ReadEnvStr $R0 COMSPEC
    StrCmpS $Bit 64 0 +4
    StrCmp "${_FSR}" /DISABLEFSR 0 +3
    ExecDos::Exec /TOSTACK /DISABLEFSR `"$R0" /c "${SC} qc "${_SVC}" | FIND
"BINARY_PATH_NAME""`
    Goto +2
    ExecDos::Exec /TOSTACK `"$R0" /c "${SC} qc "${_SVC}" | FIND "BINARY_PATH_NAME""`
    Pop ${_ERR1}
    Pop ${_ERR2}
!macroend
```

## Service :: Etat

```
##=
#= Service::State
#
# USAGE:
# ${Service::State} "NAME" /DISABLEFSR $0 $1
#
#    ::State     = The service's status is returned.
#    NAME        = The Service name
#    /DISABLEFSR = Disables redirection if x64. Use "" to skip.
#    $0          = Return after call | 1 = success
#    $1          =   ''    ''    ''  | 1 = running
#
# $1 will now hold "1" if running or "0" if not
#
!define Service::State `!insertmacro _Service::State`
!macro _Service::State _SVC _FSR _ERR1 _ERR2
    ReadEnvStr $R0 COMSPEC
    StrCmpS $Bit 64 0 +4
    StrCmp "${_FSR}" /DISABLEFSR 0 +3
    ExecDos::Exec /TOSTACK /DISABLEFSR `"$R0" /c "${SC} query "${_SVC}" | find /C "RUNNING""`
    Goto +2
    ExecDos::Exec /TOSTACK `"$R0" /c "${SC} query "${_SVC}" | find /C "RUNNING""`
    Pop ${_ERR1}
    Pop ${_ERR2}
!macroend
```

## Service :: Start

```
##=
#= Service::Start
#
# USAGE:
# ${Service::Start} "NAME" /DISABLEFSR $0 $1
```

```
#
#    ::Start     = Start a service.
#    NAME        = The Service name
#    /DISABLEFSR = Disables redirection if x64. Use "" to skip.
#    $0          = Return after call
#    $1          =   ''    ''    ''
#
# $1 will now hold "1" if running or "0" if not
#
!define Service::Start `!insertmacro _Service::Start`
!macro _Service::Start _SVC _FSR _ERR1 _ERR2
    StrCmpS $Bit 64 0 +4
    StrCmp "${_FSR}" /DISABLEFSR 0 +3
    ExecDos::Exec /TOSTACK /DISABLEFSR `"${SC}" start "${_SVC}"`
    Goto +2
    ExecDos::Exec /TOSTACK `"${SC}" start "${_SVC}"`
    Pop ${_ERR1}
    Pop ${_ERR2}
!macroend
```

## Service :: Stop

```
##=
#= Service::Stop
#
# USAGE:
# ${Service::Stop} "NAME" /DISABLEFSR $0 $1
#
#    ::Stop      = Sends a STOP control request to a service.
#    NAME        = The Service name
#    /DISABLEFSR = Disables redirection if x64. Use "" to skip.
#    $0          = Return after call
#    $1          =   ''    ''    ''
#
!define Service::Stop `!insertmacro _Service::Stop`
!macro _Service::Stop _SVC _FSR _ERR1 _ERR2
    StrCmpS $Bit 64 0 +4
    StrCmp "${_FSR}" /DISABLEFSR 0 +3
    ExecDos::Exec /TOSTACK /DISABLEFSR `"${SC}" stop "${_SVC}"`
    Goto +2
    ExecDos::Exec /TOSTACK `"${SC}" stop "${_SVC}"`
    Pop ${_ERR1}
    Pop ${_ERR2}
!macroend
```

## Service :: Supprimer

```
##=
#= Service::Remove
#
# USAGE:
# ${Service::Remove} "NAME" /DISABLEFSR $0 $1
#
#    ::Remove    = Deletes a service entry from the registry.
#    NAME        = The Service name
#    /DISABLEFSR = Disables redirection if x64. Use "" to skip.
#    $0          = Return after call
```

```
#    $1          =   ''    ''    ''
#
# Be sure to stop a service first if it's running.
#
!define Service::Remove `!insertmacro _Service::Remove`
!macro _Service::Remove _SVC _FSR _ERR1 _ERR2
    StrCmpS $Bit 64 0 +4
    StrCmp "${_FSR}" /DISABLEFSR 0 +3
    ExecDos::Exec /TOSTACK /DISABLEFSR `"${SC}" delete "${_SVC}"`
    Goto +2
    ExecDos::Exec /TOSTACK `"${SC}" delete "${_SVC}"`
    Pop ${_ERR1}
    Pop ${_ERR2}
!macroend
```

Lire Prestations de service en ligne: https://riptutorial.com/fr/nsis/topic/10184/prestations-de-service

# Crédits

| S. No | Chapitres | Contributeurs |
|---|---|---|
| 1 | Démarrer avec nsis | Anders, Community, knipp, Roland Bär, wintersolider |
| 2 | Enregistrement de bibliothèques (RegDLL) | demon.devin |
| 3 | Point net | demon.devin |
| 4 | Prestations de service | demon.devin |