

Créer un service Windows en PureBasic

par LEFEVRE Franck ([Accueil](#))

Date de publication : 19 août 2008

Dernière mise à jour :

Ce tutoriel a pour objectif de vous apprendre à créer un service Windows en PureBasic.

I - Introduction.....	3
II - Pré-requis.....	4
III - Les principales fonctions.....	5
III-1 - La fonction Main().....	5
III-2 - La fonction de log : WriteToLog().....	5
III-3 - Le thread principal : Service_MainLoop().....	5
III-4 - La fonction par défaut : Service_Default().....	6
IV - Installer & Désinstaller un service.....	7
V - Service & Statut.....	8
V-1 - Démarrer un service.....	8
V-2 - Changer l'état du service.....	8
V-3 - Gérer le changement d'état du service.....	9
V-4 - Demander l'état du service.....	10
VI - Compilation & Exemple.....	11
VII - Conclusion.....	12
VIII - Remerciements.....	13

I - Introduction

Equivalent du démon sous Unix, un service est un processus qui s'exécute en arrière-plan. L'utilisateur ne peut donc pas interagir avec lui. Il se présente sous la forme d'un exécutable qui peut recevoir des actions comme de démarrer, de s'arrêter ou de se mettre en pause.

De nombreux services sont des composants de Windows. Par exemple, l'un d'entre eux est le Pare-Feu Windows qui correspond bien à la définition donnée précédemment.

Afin de voir l'ensemble des services géré par votre machine, ouvrez une invite de commandes et tapez "services.msc". Vous devriez arriver sur une fenêtre de ce type :

Gestionnaire de Services


De ce gestionnaire de services, vous pouvez manipuler les services : gérer leur exécution (Arrêt, (Re)Démarrage, Pause), leur configuration (paramètres de démarrage, compte utilisé) et accéder à leurs dépendances.

Le gestionnaire de contrôle de services - alias Service Control Manager ou SCM - gère l'ensemble des services présents sur votre ordinateur. Le SCM maintient à jour une base de données des services installés via la base de registres.

Chaque service correspond dès lors à un enregistrement incluant :

- le nom
- la désignation
- le type de démarrage (automatique ou à la demande)
- le type de service
- l'état courant
- les codes de contrôles autorisés en entrée
- un pointeur vers la liste des dépendances.

But de notre exemple : Nous allons développer un service qui vérifiera toutes les secondes la mémoire disponible et l'enregistrera dans un fichier log. Ce fichier log stockera aussi les changements d'état de notre service et les accès aux différentes fonctions afin de voir comment fonctionne un service.

 *Le service va écrire toutes les secondes dans le fichier de log. Par conséquent, la taille va rapidement augmenter. Merci de surveiller sa taille.*

II - Pré-requis

Au début du code, certaines constantes sont à initialiser. Elles seront utiles dans différentes procédures.

- `#MyService_LogFile` : Chemin et nom du fichier de log
- `#MyService_Name` : Nom du service
- `#MyService_AppName` : Nom de l'exécutable
- `#MyService_DisplayName` : Libellé du service
- `#MyService_Description` : Description du service

III - Les principales fonctions

III-1 - La fonction Main()

Cette fonction est la fonction principale qui sera lancée à n'importe quel démarrage du programme. En fonction du paramètre qu'elle aura reçu, elle exécutera telle ou telle fonction du service. Ainsi, le paramètre "-s" démarrera le service. Dans le cas où il n'y aurait pas de paramètre, la fonction Service_Default() sera exécutée.

```

Procedure Main()
    Select ProgramParameter(0)
        Case "-i", "install"
            Service_Install()
        Case "-d", "delete"
            Service_Delete()
        Case "-s", "start"
            Service_Start()
        Case "-k", "kill", "stop"
            Service_Stop()
        Default
            Service_Default(#MyService_Name)
    EndSelect
EndProcedure
    
```

III-2 - La fonction de log : WriteToLog()

La fonction WriteToLog() sera utile car elle permettra d'un simple appel de fonction de pouvoir ajouter une ligne dans le fichier de log prédéfini via la constante #MyService_LogFile.

```

Procedure WriteToLog(entry.s)
    Protected hFile.l
    hFile = OpenFile(#PB_Any, #MyService_LogFile)
    If hFile = #Null
        ProcedureReturn #False
    EndIf
    FileSeek(hFile, Lof(hFile))
    WriteStringN(hFile, entry)
    CloseFile(hFile)
    ProcedureReturn #True
EndProcedure
    
```

III-3 - Le thread principal : Service_MainLoop()

La fonction Service_MainLoop() a pour but d'être le thread qui tournera en continu tant que le service sera en cours. Ainsi à son premier lancement, il permettra d'enregistrer la fonction qui interceptera les requêtes de contrôle pour le service via la fonction RegisterServiceCtrlHandler(). Puis de définir l'état actuel du service en actif, soit #SERVICE_RUNNING.

Après cela, c'est dans cette boucle principale que l'on implémente les actions que l'on désire exécuter. Dans notre cas, nous enregistrons dans le fichier de log toutes les 1000ms la mémoire disponible.

```

Procedure Service_MainLoop()
    Protected hError.l
    Protected memory.MEMORYSTATUS

    WriteToLog("Service_MainLoop() > Start")
    With ServiceStatus
        \dwServiceType           = #SERVICE_WIN32_OWN_PROCESS
        \dwCurrentState         = #SERVICE_START_PENDING
        \dwControlsAccepted
    = #SERVICE_ACCEPT_STOP | #SERVICE_ACCEPT_SHUTDOWN | #SERVICE_ACCEPT_PAUSE_CONTINUE
        \dwWin32ExitCode        = 0
        \dwServiceSpecificExitCode = 0
    
```

```

        \dwCheckPoint          = 0
        \dwWaitHint           = 0
    EndWith
    hStatus = RegisterServiceCtrlHandler_(#MyService_Name, @Service_CtrlHandler())
    If hStatus = 0
        WriteToLog("Registering Control Handler failed")
        ProcedureReturn
    EndIf
    SetServiceStatus_(hStatus, @ServiceStatus)

; Faire l'initialisation ici

ServiceStatus\dwCurrentState = #SERVICE_RUNNING
SetServiceStatus_(hStatus, @ServiceStatus)
; La boucle principale
While ServiceStatus\dwCurrentState = #SERVICE_RUNNING
    Protected buffer.s{16}
    GlobalMemoryStatus_(@memory)
    buffer = Str(memory\dwAvailPhys)
    result = WriteToLog("Mem>"+buffer)
    If result = #False
        ServiceStatus\dwCurrentState = #SERVICE_STOPPED
        ServiceStatus\dwWin32ExitCode= -1
        SetServiceStatus_(hStatus, @ServiceStatus)
        ProcedureReturn #False
    EndIf
    Sleep_(1000)
    Service_UpdateStatus()
Wend
WriteToLog("Service_MainLoop() > End")
EndProcedure
    
```

III-4 - La fonction par défaut : Service_Default()

La fonction par défaut est spéciale. Elle a pour but de connecter le thread principal au SCM. Ainsi toutes les actions intervenant au SCM seront répercutées vers notre thread principal défini ci-dessus.

```

Procedure Service_Default(Name.s)
    Protected ServiceTable.SERVICE_TABLE_ENTRY
    WriteToLog("Service_Default() > Start")
    With ServiceTable
        \lpServiceName = @Name
        \lpServiceProc = @Service_MainLoop()
    EndWith
    ret = StartServiceCtrlDispatcher_(@ServiceTable)
    WriteToLog("Service_Default() > End")
EndProcedure
    
```

IV - Installer & Désinstaller un service

L'inscription et la désinscription du service se déroulent sur le même procédé. Il vous faut tout d'abord ouvrir une connexion au SCM via la fonction `OpenSCManager`. Dans le cas de l'installation, nous utilisons la fonction `CreateService` pour créer le service en lui donnant en paramètre :

- le pointeur du SCM
- le nom du service
- le libellé du service
- les accès désirés
- le type de service
- le type de démarrage : `#SERVICE_AUTO_START` pour un démarrage automatique avec le système ou `#SERVICE_DEMAND_START` pour un démarrage à la demande
- le niveau d'erreur en cas d'échec de démarrage
- le chemin du service
- le groupe d'ordre de chargement
- la valeur unique d'étiquette pour un service appartenant au groupe d'ordre de chargement indiqué précédemment
- le pointeur vers le tableau de dépendances
- le nom du compte utilisateur lançant le service
- le mot de passe relatif au compte

Dans le cas de la désinstallation, nous utilisons la fonction `DeleteService` qui prend simplement en paramètre le pointeur ouvert via la fonction `OpenService` sur le service désiré. Après tout cela, nous fermons les pointeurs du service et du SCM.

```

;@desc : Installe le service #MyService_Name dans le ServiceControlManager
Procedure Service_Install()
    Protected sDir.s
    Protected hSCManager.l, hService.l
    Protected SD.SERVICE_DESCRIPTION
    WriteToLog("Service_Install() > Start")
    sDir = GetCurrentDirectory() + #MyService_AppName
    hSCManager = OpenSCManager_(#Null, #Null, #SC_MANAGER_ALL_ACCESS)
    ; nous créons le service
    hService =
    CreateService_(hSCManager, #MyService_Name, #MyService_DisplayName, #SERVICE_ALL_ACCESS,
    #SERVICE_WIN32_OWN_PROCESS, #SERVICE_DEMAND_START, #SERVICE_ERROR_NORMAL,
    sDir, #Null, #Null, #Null, #Null, #Null)
    ; nous définissons sa description pour le SCM
    SD\lpDescription = #MyService_Description
    ChangeServiceConfig2_(hService, #SERVICE_CONFIG_DESCRIPTION, @SD)
    CloseServiceHandle_(hService)

    WriteToLog("Monitoring installé.")
    WriteToLog("Service_Install() > End")
EndProcedure
;@desc : Désinstalle le service #MyService_Name du ServiceControlManager
Procedure Service_Delete()
    Protected hSCManager.l, hServ.l
    WriteToLog("Service_Delete() > Start")
    hSCManager = OpenSCManager_(#Null, #Null, #SC_MANAGER_ALL_ACCESS)
    hServ = OpenService_(hSCManager, #MyService_Name, #SERVICE_ALL_ACCESS)
    DeleteService_(hServ)
    CloseServiceHandle_(hServ)
    CloseServiceHandle_(hSCManager)

    WriteToLog("Monitoring désinstallé.")
    WriteToLog("Service_Delete() > End")
EndProcedure
    
```

V - Service & Statut

V-1 - Démarrer un service

Démarrer un service utilise la même technique que l'installation ou la désinstallation du service. En dehors de la fonction principale utilisée qui est StartService.

Cette étape est différente que de simplement changer le statut. Pourquoi ? Parce que démarrer le service lance en arrière-plan le démarrage de la fonction principale. Alors que changer le statut n'a de simple incidence que le changement d'état.

```

;@desc : Changement d'état : START
Procedure Service_Start()
    Protected hSCManager.l, hServ.l
    WriteToLog("Service_Start() > Start")
    hSCManager = OpenSCManager_(#Null, #Null, #SC_MANAGER_ALL_ACCESS)
    hServ = OpenService_(hSCManager, #MyService_Name, #SERVICE_ALL_ACCESS)
    StartService_(hServ, 0, #Null)
    WriteToLog("Monitoring démarré.")
    CloseServiceHandle_(hServ)
    CloseServiceHandle_(hSCManager)
    WriteToLog("Service_Start() > End")
EndProcedure
    
```

V-2 - Changer l'état du service

Changer l'état du service utilise toujours la même technique qui est d'abord de se connecter au SCM puis au service souhaité.

Ensuite, on utilise la fonction ControlService pour envoyer au service l'état souhaité parmi ceux-ci disponibles :

- **#SERVICE_CONTROL_CONTINUE** : cet état permet de relancer un service en pause
- **#SERVICE_CONTROL_INTERROGATE** : cet état notifie le service qu'il doit déclarer son état actuel au SCM
- **#SERVICE_CONTROL_PARAMCHANGE** : cet état notifie le service que ses paramètres de démarrage ont changés
- **#SERVICE_CONTROL_PAUSE** : cet état permet de mettre en pause un service actif
- **#SERVICE_CONTROL_STOP** : cet état permet d'arrêter un service

```

;@desc : Changement d'état : STOP
Procedure Service_Stop()
    Protected hSCManager.l, hServ.l
    WriteToLog("Service_Stop() > Start")
    hSCManager = OpenSCManager_(#Null, #Null, #SC_MANAGER_ALL_ACCESS)
    hServ = OpenService_(hSCManager, #MyService_Name, #SERVICE_ALL_ACCESS)
    ControlService_(hServ, #SERVICE_CONTROL_STOP, @ServiceStatus)
    WriteToLog("Monitoring arrêté.")
    CloseServiceHandle_(hServ)
    CloseServiceHandle_(hSCManager)
    WriteToLog("Service_Stop() > End")
EndProcedure

;@desc : Changement d'état : PAUSE > Start
Procedure Service_Pause()
    Protected hSCManager.l, hServ.l
    WriteToLog("Service_Pause() > Start")
    hSCManager = OpenSCManager_(#Null, #Null, #SC_MANAGER_ALL_ACCESS)
    hServ = OpenService_(hSCManager, #MyService_Name, #SERVICE_ALL_ACCESS)
    ControlService_(hServ, #SERVICE_CONTROL_PAUSE, @ServiceStatus)
    WriteToLog("Monitoring suspendu.")
    CloseServiceHandle_(hServ)
    CloseServiceHandle_(hSCManager)
    WriteToLog("Service_Pause() > End")
EndProcedure

;@desc : Changement d'état : PAUSE > Stop
    
```

```

Procedure Service_Continue()
    Protected hSCManager.l, hServ.l
    WriteToLog("Service_Continue() > Start")
    hSCManager = OpenSCManager_(#Null, #Null, #SC_MANAGER_ALL_ACCESS)
    hServ      = OpenService_(hSCManager, #MyService_Name, #SERVICE_ALL_ACCESS)
    ControlService_(hServ, #SERVICE_CONTROL_CONTINUE, @ServiceStatus)
    WriteToLog("Monitoring repris.")
    CloseServiceHandle_(hServ)
    CloseServiceHandle_(hSCManager)
    WriteToLog("Service_Continue() > End")
EndProcedure
    
```

V-3 - Gérer le changement d'état du service

Cette étape est utile car elle permet d'intercepter les changements d'état du service. Ainsi en fonction du changement d'état auquel on arrive, on peut demander au service de faire telle ou telle action. Pour l'initialiser, cela se passe dans la boucle principale (fonction Service_MainLoop()) avec la fonction RegisterServiceCtrlHandler qui permet de définir la fonction pour intercepter le changement d'état.

```

Procedure Service_CtrlHandler(CtrlRequest.l)
    WriteToLog("Service_CtrlHandler() > Start")
    Select CtrlRequest
    Case #SERVICE_CONTROL_CONTINUE
    ;{
        WriteToLog("Monitoring resumed.")
        With ServiceStatus
            \dwCurrentState = #SERVICE_RUNNING
        EndWith
    ;}
    Case #SERVICE_CONTROL_INTERROGATE
    ;{
        WriteToLog("Monitoring reported its current status information to the service control manager.")
    ;}
    Case #SERVICE_CONTROL_PAUSE
    ;{
        WriteToLog("Monitoring paused.")
        With ServiceStatus
            \dwCurrentState = #SERVICE_PAUSED
        EndWith
    ;}
    Case #SERVICE_CONTROL_STOP
    ;{
        WriteToLog("Monitoring stopped.")
        With ServiceStatus
            \dwWin32ExitCode = 0
            \dwCurrentState = #SERVICE_STOPPED
        EndWith
    ;}
    Case #SERVICE_CONTROL_SHUTDOWN
    ;{
        WriteToLog("Monitoring shutdowned.")
        With ServiceStatus
            \dwWin32ExitCode = 0
            \dwCurrentState = #SERVICE_STOPPED
        EndWith
    ;}
    Default
    ;{
        WriteToLog("CtrlRequest Unknown = "+Str(CtrlRequest))
    ;}
    EndSelect
    ; Report current status
    SetServiceStatus_(hStatus, @ServiceStatus)
    WriteToLog("Service_CtrlHandler() > End")
EndProcedure
    
```

V-4 - Demander l'état du service

Il est possible qu'à un quelconque moment, vous ayez besoin de demander l'état du service. Dans ce cas, il faut utiliser la fonction `QueryServiceStatus` qui utilise l'un des paramètres comme valeur de retour de l'état du service. Dans l'exemple ci-dessus, la fonction `Service_UpdateStatus()` est utilisé dans notre boucle principale pour mettre à jour l'état du service à chaque tour de boucle.

```
;@desc : Actualiser l'état du service
Procedure Service_UpdateStatus()
    Protected hSCManager.l, hServ.l
    hSCManager = OpenSCManager_(#Null, #Null, #SC_MANAGER_ALL_ACCESS)
    hServ      = OpenService_(hSCManager, #MyService_Name, #SERVICE_ALL_ACCESS)
    QueryServiceStatus_(hServ,@ServiceStatus)
    CloseServiceHandle_(hServ)
    CloseServiceHandle_(hSCManager)
EndProcedure
```

VI - Compilation & Exemple

Téléchargez la source du tutoriel :

[Source](#) **Source du tutoriel**

Ensuite, soit on compile la source, soit on utilise l'exécutable fourni. Dès lors, on utilise une invite de commandes MS-DOS pour travailler avec ce service.

On se place dans le dossier de la source :

```
C:\Documents and Settings\usr>cd Source
```

On installe le service :

```
C:\Documents and Settings\usr\Source>service -i
```

On démarre le service :

```
C:\Documents and Settings\usr\Source>service -s
```

On arrête le service :

```
C:\Documents and Settings\usr\Source>service -k
```

On désinstalle le service :

```
C:\Documents and Settings\usr\Source>service -d
```

VII - Conclusion

Voilà nous arrivons à la fin de cet article.

Dans cet article, on a appris à créer un service, c'est-à-dire installer, désinstaller, démarrer et changer l'état d'un service et tout cela en PureBasic. Comme on a pu s'en rendre compte, on utilise assez souvent la même technique pour accéder à un service et le modifier. Ce travail n'étant qu'une piste pour débiter dans les services Windows : toutes les possibilités des services s'offrent à vous.

J'espère que cet article permettra aux gens d'avoir un nouveau point de vue sur PureBasic.

Et maintenant à vos claviers !

Références :

-  **MSDN : Services**

VIII - Remerciements

Merci à **buchs** et **Comtois** pour la relecture.
Enfin je dis un grand merci à l'équipe de DVP pour leur travail.