Как минимум, сервис должен включать три компонента:

* A Main Entry point
* A Service Entry point
* A Service Control Handler

Сервис нужно создавать как консольное приложение, но не выполнять его.

Сервис инсталлируется с помощью команды sc.

Нам потребуются следующие глобальные объявления:

SERVICE\_STATUS g\_ServiceStatus = {0};

SERVICE\_STATUS\_HANDLE g\_StatusHandle = NULL;

HANDLE g\_ServiceStopEvent = INVALID\_HANDLE\_VALUE;

VOID WINAPI ServiceMain (DWORD argc, LPTSTR \*argv);

VOID WINAPI ServiceCtrlHandler (DWORD);

DWORD WINAPI ServiceWorkerThread (LPVOID lpParam);

#define SERVICE\_NAME \_T("My Sample Service")

Main Entry Point

Hide   Copy Code

int \_tmain (int argc, TCHAR \*argv[])

{

SERVICE\_TABLE\_ENTRY ServiceTable[] =

{

{SERVICE\_NAME, (LPSERVICE\_MAIN\_FUNCTION) ServiceMain},

{NULL, NULL}

};

if (StartServiceCtrlDispatcher (ServiceTable) == FALSE)

{

return GetLastError ();

}

return 0;

}

Service Entry Point

Hide   Shrink    Copy Code

VOID WINAPI ServiceMain (DWORD argc, LPTSTR \*argv)

{

DWORD Status = E\_FAIL;

*// Register our service control handler with the SCM*

g\_StatusHandle = RegisterServiceCtrlHandler (SERVICE\_NAME, ServiceCtrlHandler);

if (g\_StatusHandle == NULL)

{

goto EXIT;

}

*// Tell the service controller we are starting*

ZeroMemory (&g\_ServiceStatus, sizeof (g\_ServiceStatus));

g\_ServiceStatus.dwServiceType = SERVICE\_WIN32\_OWN\_PROCESS;

g\_ServiceStatus.dwControlsAccepted = 0;

g\_ServiceStatus.dwCurrentState = SERVICE\_START\_PENDING;

g\_ServiceStatus.dwWin32ExitCode = 0;

g\_ServiceStatus.dwServiceSpecificExitCode = 0;

g\_ServiceStatus.dwCheckPoint = 0;

if (SetServiceStatus (g\_StatusHandle , &g\_ServiceStatus) == FALSE)

{

OutputDebugString(\_T(

"My Sample Service: ServiceMain: SetServiceStatus returned error"));

}

*/\**

*\* Perform tasks necessary to start the service here*

*\*/*

*// Create a service stop event to wait on later*

g\_ServiceStopEvent = CreateEvent (NULL, TRUE, FALSE, NULL);

if (g\_ServiceStopEvent == NULL)

{

*// Error creating event*

*// Tell service controller we are stopped and exit*

g\_ServiceStatus.dwControlsAccepted = 0;

g\_ServiceStatus.dwCurrentState = SERVICE\_STOPPED;

g\_ServiceStatus.dwWin32ExitCode = GetLastError();

g\_ServiceStatus.dwCheckPoint = 1;

if (SetServiceStatus (g\_StatusHandle, &g\_ServiceStatus) == FALSE)

{

OutputDebugString(\_T(

"My Sample Service: ServiceMain: SetServiceStatus returned error"));

}

goto EXIT;

}

*// Tell the service controller we are started*

g\_ServiceStatus.dwControlsAccepted = SERVICE\_ACCEPT\_STOP;

g\_ServiceStatus.dwCurrentState = SERVICE\_RUNNING;

g\_ServiceStatus.dwWin32ExitCode = 0;

g\_ServiceStatus.dwCheckPoint = 0;

if (SetServiceStatus (g\_StatusHandle, &g\_ServiceStatus) == FALSE)

{

OutputDebugString(\_T(

"My Sample Service: ServiceMain: SetServiceStatus returned error"));

}

*// Start a thread that will perform the main task of the service*

HANDLE hThread = CreateThread (NULL, 0, ServiceWorkerThread, NULL, 0, NULL);

*// Wait until our worker thread exits signaling that the service needs to stop*

WaitForSingleObject (hThread, INFINITE);

*/\**

*\* Perform any cleanup tasks*

*\*/*

CloseHandle (g\_ServiceStopEvent);

*// Tell the service controller we are stopped*

g\_ServiceStatus.dwControlsAccepted = 0;

g\_ServiceStatus.dwCurrentState = SERVICE\_STOPPED;

g\_ServiceStatus.dwWin32ExitCode = 0;

g\_ServiceStatus.dwCheckPoint = 3;

if (SetServiceStatus (g\_StatusHandle, &g\_ServiceStatus) == FALSE)

{

OutputDebugString(\_T(

"My Sample Service: ServiceMain: SetServiceStatus returned error"));

}

EXIT:

return;

}

The Service Main Entry Point выполняет следующие задачи:

* **Register сервис хэндлер для обработки событий** Service Stop, Pause, Continue, Shutdown, etc control commands. Регистрируются через поле dwControlsAccepted структуры SERVICE\_STATUS как битовая маска.
* **Устанавливает Service Status**  в SERVICE\_PENDING и затем в SERVICE\_RUNNING. Устанавливает  SERVICE\_STATUS.dwControlsAccepted в 0 при изменении статуса на SERVICE\_STOPPED  или SERVICE\_PENDING.
* **Запускает рабочий поток сервиса**.

Service Control Handler

Hide   Shrink    Copy Code

VOID WINAPI ServiceCtrlHandler (DWORD CtrlCode)

{

switch (CtrlCode)

{

case SERVICE\_CONTROL\_STOP :

if (g\_ServiceStatus.dwCurrentState != SERVICE\_RUNNING)

break;

*/\**

*\* Perform tasks necessary to stop the service here*

*\*/*

g\_ServiceStatus.dwControlsAccepted = 0;

g\_ServiceStatus.dwCurrentState = SERVICE\_STOP\_PENDING;

g\_ServiceStatus.dwWin32ExitCode = 0;

g\_ServiceStatus.dwCheckPoint = 4;

if (SetServiceStatus (g\_StatusHandle, &g\_ServiceStatus) == FALSE)

{

OutputDebugString(\_T(

"My Sample Service: ServiceCtrlHandler: SetServiceStatus returned error"));

}

*// This will signal the worker thread to start shutting down*

SetEvent (g\_ServiceStopEvent);

break;

default:

break;

}

}

Здесь обработано событие  SERVICE\_CONTROL\_STOP. Можно обрабатывать другие события

 SERVICE\_CONTROL\_CONTINUE, SERVICE\_CONTROL\_INTERROGATE, SERVICE\_CONTROL\_PAUSE,SERVICE\_CONTROL\_SHUTDOWN.

Service Worker Thread

Hide   Copy Code

DWORD WINAPI ServiceWorkerThread (LPVOID lpParam)

{

*// Periodically check if the service has been requested to stop*

while (WaitForSingleObject(g\_ServiceStopEvent, 0) != WAIT\_OBJECT\_0)

{

*/\**

*\* Perform main service function here*

*\*/*

*// Simulate some work by sleeping*

Sleep(3000);

}

return ERROR\_SUCCESS;

}

C:\>sc create "My Sample Service" binPath= C:\SampleService.exe

.