**Winlogon notification package. Теория и практика**

Роман Бурда

**Что это такое**

В операционных системах линии Windows NT диалоговая поддержка входа в систему реализована с помощью Winlogon. Он предоставляет целый ряд полезных интерфейсов. Наиболее известным является GINA (Graphical Identification and Authentication dll), который обычно используется, когда нужно изменить стандартный способ аутентификации. Например, реализовать возможность входа в систему по отпечаткам пальцев, сетчатке глаза или по электронной карточке. Менее известным, но не менее полезным является Winlogon notification package. Этот интерфейс реализует возможность обрабатывать события на вход и выход пользователей, включение и выключение операционной системы и некоторые другие. К сожалению, последний интерфейс реализован только в операционных системах Win2000 и выше.

**Теория**

Winlogon notification package – это dll-библиотеки, которые получают и обрабатывают события, сгенерированные Winlogon.

Logon – генерируется при входе пользователя в систему посредством Winlogon + GINA (через консоль, терминал и т. д.). Сообщения не будут приходить, если пользователь зашел посредством LogonUser.

Logoff – генерируется при выходе пользователя из системы (если он вошел в систему посредством Winlogon + GINA).

Startup – генерируется при запуске системы, а также при присоединении терминального клиента.

Shutdown – генерируется перед выключением системы, а также при отсоединении терминального клиента.

StartScreenSaver – генерируется при запуске screen saver-а. В обработчике этого события запрещено показывать интерфейс пользователя.

StopScreenSaver – генерируется при остановке screen saver-а. В обработчике этого события запрещено показывать интерфейс пользователя.

Lock – генерируется, когда пользователь блокирует рабочую станцию.

Unlock – генерируется, когда пользователь разблокировал рабочую станцию, или когда системный администратор снял блокировку, заставив пользователя выйти из системы.

StartShell – генерируется после того, как пользователь зашел в систему и сетевые соединения были установлены.

Для реализации Winlogon notification package необходимо следующее.

Сначала нужно создать динамическую библиотеку, которая будет импортировать набор функций-обработчиков событий. Прототип этих функций должен строиться по шаблону:

|  |
| --- |
| VOID FunctionName (PWLX\_NOTIFICATION\_INFO pInfo); |

Как видите, вся информация о данных события хранится в структуре WLX\_NOTIFICATION\_INFO.

|  |
| --- |
| typedef struct \_WLX\_NOTIFICATION\_INFO  {  ULONG Size ;  ULONG Flags ;  PWSTR UserName ;  PWSTR Domain ;  PWSTR WindowStation ;  HANDLE hToken ;  HDESK hDesktop ;  PFNMSGECALLBACK pStatusCallback;  } WLX\_NOTIFICATION\_INFO, \* PWLX\_NOTIFICATION\_INFO ; |

В этой структуре:

Size – определяет размер структуры в байтах;

Flags – поле зарезервировано и должно устанавливаться в 0;

UserName – строка, определяющая имя текущего пользователя. Если событие случилось до входа пользователя в систему, то это поле равно NULL;

Domain – строка, которая определяет название домена, в который вошел текущий пользователь. Если событие случилось до входа пользователя в систему, то это поле равно NULL;

WindowStation – определяет имя оконной станции, в которой текущий пользователь работает. Если событие случилось до входа пользователя в систему, то это поле равно NULL;

hToken – дескриптор token-а пользователя. Если событие случилось до входа пользователя в систему, то это поле равно NULL;

hDesktop – дескриптор Desktop’а для текущего события;

pStatusCallback – зарезервировано для внутреннего использования.

После создания динамической библиотеки нужно указать Winlogon, где ее можно найти, и какие события она будет обрабатывать. Вся информация размещается в системном реестре в ключе

HKEY\_LOCAL\_MACHINE\Software\Microsoft\Windows NT\CurrentVersion\Winlogon\Notify

Полный доступ к нему имеют только пользователи с правами локального администратора. Обыкновенные пользователи не смогут создавать или изменять подключи и/или значения.

|  |
| --- |
| СОВЕТ  Стоит отметить, что этого ключа не существует в ОС Win9x, но существует недокументированный аналог – HKEY\_LOCAL\_MACHINE\System\CurrentControlSet\control\MPRServices. С помощью этого ключа можно указать процессу mpr.exe (этот процес является некоторым аналогом Winlogon для Win9x, поскольку также отвечает за вход пользователей в систему) загружать в свое адресное пространство необходимую динамическую библиотеку. Таким образом, теоретически можно получить доступ к событиям входа и выхода пользователя из системы. Для указания библиотеки достаточно создать подключ с произвольным именем, и уже там указать следующие значения. ‘DLLName’ (REG\_SZ)– имя динамической библиотеки или путь к ней, ‘EntryPoint’ (REG\_SZ)– точка входа в библиотеку. Ею может быть название любой экспортируемой функции. ‘StackSize’ (REG\_DWORD) – размер стека обычно устанавливается в 0. После перезагрузки системы указанная библиотека будет подгружена в адресное пространство mpr.exe. Поскольку в Win9x доступ к системному реестру имеют все, этот ключ довольно часто используется троянами. Ведь для них это своего рода автозагрузка. Возможно, именно поэтому Microsoft не документировала этот механизм. |

Для того чтобы подписать динамическую библиотеку на события Winlogon, необходимо создать подключ с произвольным именем, после чего указать значения в нем:

DllName (REG\_EXPAND\_SZ) – имя библиотеки, содержащей notification package, например Notify.dll;

Impersonate (REG\_DWORD) – указывает, должен ли быть имперсонирован контекст безопасности вошедшего пользователя, когда Winlogon вызывает функцию-обработчик. (1 – должен быть имперсонирован, 0 – не должен).

Asynchronous (REG\_DWORD) – указывает, должен ли процесс Winlogon создавать отдельный поток для вызова функции-обработчика (1 – создавать отдельный поток, 0 – не создавать). Фактически, это значение указывает, как будет работать Winlogon с обработчиком – асинхронно или синхронно.

Следующие значения являются необязательными. Они определяют названия функций, которые будут обрабатывать одноименные события. То есть существует возможность подписаться не на все события, а только на некоторые. Все значения имеют тип REG\_SZ. В таблице 1 приведены их названия и возможные значения.

|  |  |
| --- | --- |
| Имя значения | Пример |
| Logon | WLEventLogon |
| Logoff | WLEventLogoff |
| Startup | WLEventStartup |
| Shutdown | WLEventShutdown |
| StartScreenSaver | WLEventStartScreenSaver |
| StopScreenSaver | WLEventStopScreenSaver |
| Lock | WLEventLock |
| Unlock | WLEventUnlock |
| StartShell | WLEventStartShell |

Чтобы изменения вступили в силу, нужно перегрузить компьютер. Это объясняется тем, что Winlogon загружает dll перед загрузкой системы.

Практика

Создадим в Visual Studio пример простой программы, ведущей логи сообщений Winlogon.

|  |
| --- |
| ПРИМЕЧАНИЕ  Демонстрационный проект был создан и скомпилирован в Visual Studio 6.0. |

Создадим проект Win32 Dynamic-Link Library с именем Notify. Укажем визарду разновидность проекта – A simple DLL project. Итак, создан пустой проект. Укажем используемые заголовочные файлы.

|  |
| --- |
| #include <Winwlx.h>  #include <stdio.h> |

Первый заголовочный файл хранит в себе прототип WLX\_NOTIFICATION\_INFO. А второй пригодится в процессе создания логов.

У нас уже есть функция DllMain, любезно предоставленная визардом. Оставим ее без изменений. Напишем функцию создания логов. Поскольку структура WLX\_NOTIFICATION\_INFO содержит в себе Юникод-строки, будет удобно выводить данные в текстовый файл типа Юникод.

|  |
| --- |
| void WriteLog(PWSTR pStrEventName,  PWLX\_NOTIFICATION\_INFO pInfo)  {  // Первый параметр функции WriteLog будет  // содержать описание события,  // а второй - сами данные.  // Задаем название log-файла  WCHAR fName[] = L"C:\\log.txt";  WCHAR buf[ 1024 ];  // Готовим строку для логов. Стоит отметить что если бы  // был обьявлен макрос STRICT, вывод hDesktop  // необходимо было бы делать по другому.  swprintf(buf, L"Event: %s, User: %s, Domain: %s,"  L"Window station: %s, User token 0x%p,"  L"Desktop 0x%p\r\n", pStrEventName,  pInfo->UserName, pInfo->Domain, pInfo->WindowStation,  pInfo->hToken, pInfo->hDesktop);  // Открываем log-файл для записи.  HANDLE hFile = CreateFileW(fName, GENERIC\_WRITE,  FILE\_SHARE\_READ, NULL, OPEN\_EXISTING,  FILE\_ATTRIBUTE\_NORMAL, NULL);  DWORD dwWritten;  // Если такого файла нет, или случилась какая-то другая  // ошибка, пробуем создать новый файл.  if (hFile == INVALID\_HANDLE\_VALUE)  {  hFile = CreateFileW(fName, GENERIC\_WRITE,  FILE\_SHARE\_READ, NULL, CREATE\_NEW,  FILE\_ATTRIBUTE\_NORMAL, NULL);  if (hFile == INVALID\_HANDLE\_VALUE)  {  // Если файл не создался, сообщаем пользователю  // системным звуком о возникновении ошибки.  ::MessageBeep(MB\_ICONASTERISK);  return;  }  WCHAR data = 0xFEFF;  // Указываем, что созданный файл будет  // включать текст типа Юникод.  if (!WriteFile(hFile, &data, sizeof(data),  &dwWritten, NULL))  ::MessageBeep(MB\_ICONASTERISK);  }  //Поскольку открытый файл может быть не пустым,  //переходим в его конец  if (SetFilePointer(hFile, 0, NULL, FILE\_END) ==  INVALID\_SET\_FILE\_POINTER)  {  ::MessageBeep(MB\_ICONASTERISK);  return;  }  //Записываем строку лога в файл  if (!WriteFile(hFile, buf, wcslen(buf) \* sizeof(WCHAR),  &dwWritten, NULL))  ::MessageBeep(MB\_ICONASTERISK);  //Заканчиваем работу с файлом  CloseHandle(hFile);  } |

Теперь осталось написать функции-обработчики событий. Ради удобства назовем их так же, как показано в таблице 1.

|  |
| --- |
| extern "C" void \_\_stdcall  WLEventLogon(PWLX\_NOTIFICATION\_INFO pInfo)  {  WriteLog(L"user logon", pInfo);  }  extern "C" void \_\_stdcall  WLEventLogoff(PWLX\_NOTIFICATION\_INFO pInfo)  {  WriteLog(L"user logoff", pInfo);  }  extern "C" void \_\_stdcall  WLEventStartup(PWLX\_NOTIFICATION\_INFO pInfo)  {  WriteLog(L"system startup", pInfo);  }  extern "C" void \_\_stdcall  WLEventShutdown(PWLX\_NOTIFICATION\_INFO pInfo)  {  WriteLog(L"system shutdown", pInfo);  }  extern "C" void \_\_stdcall  WLEventStartScreenSaver(PWLX\_NOTIFICATION\_INFO pInfo)  {  WriteLog(L"screen saver started", pInfo);  }  extern "C" void \_\_stdcall  WLEventStopScreenSaver(PWLX\_NOTIFICATION\_INFO pInfo)  {  WriteLog(L"screen saver stopped", pInfo);  }  extern "C" void \_\_stdcall  WLEventLock(PWLX\_NOTIFICATION\_INFO pInfo)  {  WriteLog(L"Workstation locked", pInfo);  }  extern "C" void \_\_stdcall  WLEventUnlock(PWLX\_NOTIFICATION\_INFO pInfo)  {  WriteLog(L"Workstation unlocked", pInfo);  }  extern "C" void \_\_stdcall  WLEventStartShell(PWLX\_NOTIFICATION\_INFO pInfo)  {  WriteLog(L"User already loged on and network"  L" resouces are avaliable", pInfo);  } |

Теперь необходимо экспортировать эти функции для других приложений. Для этого создадим файл Notify.def и добавим его в проект. В единственной секции EXPORTS укажем экспортируемые функции

|  |
| --- |
| EXPORTS  WLEventLogon  WLEventLogoff  WLEventStartup  WLEventShutdown  WLEventStartScreenSaver  WLEventStopScreenSaver  WLEventLock  WLEventUnlock  WLEventStartShell |

После успешной компиляции необходимо подписаться на события Winlogon.

|  |
| --- |
| ПРЕДУПРЕЖДЕНИЕ  Необходимо помнить, что dll исполняется в адресном пространстве процесса, который ее вызывает. Если dll спровоцирует исключительную ситуацию, тогда, если не обработать ошибку, работа процесса будет прекращена. Процесс Winlogon.exe очень чувствителен к таким вещам. При возникновении ошибки он покажет синий экран смерти, а в случае наличия соответствующих настроек перегрузит систему. Поэтому настойчиво рекомендуется тестировать программу на виртуальном компьютере. |

Для этого зайдем в реестр и отыщем ключ:

HKEY\_LOCAL\_MACHINE\Software\Microsoft\Windows NT\CurrentVersion\Winlogon\Notify

Создадим в нем подключ MyNotify и значения, как показано на рисунке ниже.

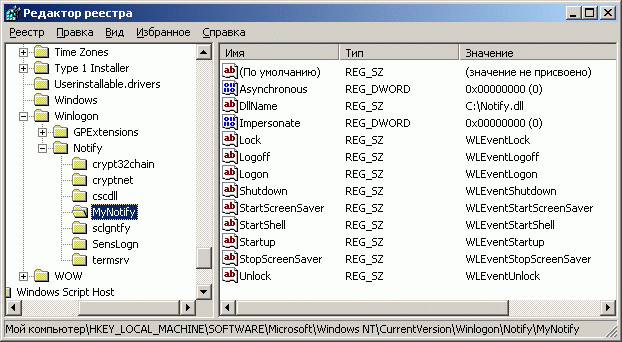


Рисунок 1.

Для удобства поместим файл Notify.dll по указанному пути. После перезагрузки системы приложение начнет создавать логи.

|  |
| --- |
| ПРИМЕЧАНИЕ  Если логи не создаются, проверьте, не используется ли Notify.dll другим процессом (если файл не удается переименовать – значит, кто-то его использует). Если Notify.dll используется, а логов нет, скорее всего, неправильно задано одно из значений в названиях функций-обработчиков, или такие функции не экспортируются динамической библиотекой. Если Notify.dll не используется, проверьте, правильно ли задан ключ DllName с названием библиотеки. Возможно, библиотека не зарегистрирована в системе, или не может быть загружена. |

**Заключение**

Winlogon notification package может пригодиться в совершенно неожиданных местах. Например, если из системной службы необходимо получить имена пользователей, работающих в системе. Все, что нужно – это написать и зарегистрировать в системе dll, которая по запросу передаст данные в службу (или сама будет передавать эти данные в ответ на некоторые нужные события).

Надеюсь, что данная статья расширит знания читателя об интерфейсах Winlogon, а также поможет использовать полученные знания на практике.