Winlogon User Interface

Microsoft® Win32® Software Development Kit
for Microsoft® Windows®

Legal Information

Microsoft® Win32® Software Development Kit for Microsoft® Windows®

Winlogon User Interface for Windows NT

This document is an early release of the final specification. It is meant to specify and accompany software that is still in development. Some of the information in this documentation may be inaccurate or may not be an accurate representation of the functionality of the final specification or software. Microsoft assumes no responsibility for any damages that might occur either directly or indirectly from these inaccuracies.

Information in this document is subject to change without notice. Companies, names, and data used in examples herein are fictitious unless otherwise noted. No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of Microsoft Corporation.

Microsoft may have patents or pending patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. The furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property rights except as expressly provided in any written license agreement from Microsoft.

Microsoft, MS, MS-DOS, Win32, Windows, and Windows NT are registered trademarks of Microsoft Corporation in the U.S.A. and other countries.

About Winlogon User Interface

Winlogon is a component of Microsoft® Windows NT® that provides interactive, logon support. To allow for independent software vendor (ISV) and developer modifications to the interactive logon model, some aspects of Winlogon are replaceable. Specifically, the identification and authentication aspects of Winlogon are implemented in a replaceable dynamic-link library (DLL). This replaceable DLL is referred to as the Graphical Identification and Authentication DLL, or GINA. GINA lets developers implement smart-card, retinal-scan, or other authentication mechanisms in place of the standard Windows NT user name and password authentication.

This documentation provides a programmer's reference for anyone who needs to implement such a replacement DLL. The primary goal of this documentation is to describe what functionality can be achieved by the various components involved in interactive logon. (For example, what Winlogon takes care of and what a GINA developer must take care of.)

Those who use this documentation should have a firm knowledge of the Windows NT security architecture, especially with regard to tokens, authentication packages, and related matters. Any developer intending to write a replacement GINA is encouraged to work with Microsoft's Vendor Assistance program.

Note This technology is only applicable on Microsoft Windows NT.

Related Documentation

The following documents provide information related to the topics discussed in this documentation.

Windows NT Local Security Authority Authentication

This describes the authentication model supported by Windows NT. It also describes and defines the Local Security Authority (LSA) services and Authentication Package interfaces.

Windows NT Local Security Authority Protected Subsystem (LSA)

This describes the aspects of the Windows NT Local Security Authority that relate to security policy administration. Security policy of a system includes trust relationships with other machines and domains, assignment of privileges, audit generation control, system accessibility, and other similar topics.

Winlogon Terminology

Winlogon

WINLOGON.EXE is the executable image responsible for implementing interactive logon support. There are many aspects of interactive logon support, including an actual collection of identification and authentication information (as described under the term GINA), and a number of other functions such as window station and desktop management, screen-saver control, and multiple-network provider notifications.

GINA

This specification is used by developers who want to replace the component of Windows NT that performs identification and authentication of interactive users. This replaceable functionality is implemented as a dynamic-link library (DLL) which is loaded and called by WINLOGON.EXE. This DLL is referred to as the Graphical Identification and Authentication DLL, or GINA.

MSGINA.DLL

The standard GINA shipped with Windows NT is MSGINA.DLL.

Secure Attention Sequence (SAS)

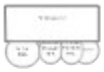
Winlogon uses a special sequence of events to recognize when a user wants to log on or perform other secure operations. This sequence of events is referred to as the secure attention sequence or SAS. The SAS provides a secure way for users to enter identification and authentication information. In this way, users are protected from password-collection programs and other flaws inherent in timeshare systems.

As an example, in earlier timeshare systems, users could walk up to an apparently unused terminal and press the ENTER key. The system would then prompt for username and password information. The problem with this was that the terminal could sometimes be in use and with a password collector program running which made the logon appear like a normal logon. However, in reality, the "Trojan horse" program would store away the user's password and then indicate that the user had entered an invalid password (or indicate some other reason why the logon failed). The end result would be that the user just gave his or her password to someone else.

In Windows NT, users can enter a secure attention sequence (SAS). This prompts Winlogon to switch to a secure desktop that no "Trojan horse" program has access to. The SAS for a standard Windows NT system is the CTRL+ALT+DEL key combination. Developers writing a replacement GINA are encouraged to achieve the same level of security with their SAS. By using a device such as a smart-card reader, the SAS could be the insertion or removal of a smart card. Developers can also choose to retain the CTRL+ALT+DEL key combination as the SAS.

Winlogon Structure

Winlogon has a main body that handles nonuser interface functions and those that are independent of authentication policy. This main program loads a GINA DLL that should include all authentication policy and is expected to perform all identification and authentication user interactions. In addition to the main body and GINA, Winlogon can load zero or more network providers that must perform secondary authentication. The following diagram illustrates the Winlogon structure.



Winlogon States

Winlogon serves as the process that authenticates and logs on the interactive user. Winlogon is in one of three states at any given time. These states are illustrated in the following diagram.

Logged-off State

When Winlogon is in the logged-off state, users are prompted to identify themselves and provide authentication information. If a user provides correct user account information and no restrictions prevent it, the user is logged on and a shell program (such as EXPLORER.EXE) is activated in the user's context. Winlogon changes into the logged-on state.

Logged-on State

When Winlogon is in the logged-on state, users can interact with the shell, activate additional applications, and generally perform their work. From the logged-on state, users can either stop all work and log off, or lock their workstations (leaving all work in place). If the user decides to log off, Winlogon will terminate all processes associated with that logon session and the window station will be available for another user. If, instead, the user decides to lock the workstation, then a secure desktop is displayed.

Workstation-locked State

When Winlogon is in the workstation-locked state, the secure desktop is displayed either until the user unlocks the workstation (by providing identification and authentication information that matches the information provided by the user who originally logged on), or until an administrator forces a logoff. If the workstation is unlocked, the user's typical desktop is again displayed and work can resume. If, however, an administrator unlocks the workstation (by providing the identification and authentication information of an administrator account), the logged-on user's processes are terminated and the workstation becomes available for another user.

As demonstrated, there are a number of different actions that can be performed in each of the Winlogon states. Some of these are necessary, some can be customized, and some are optional. There can also be other actions not provided by a standard Windows NT system that a GINA developer would like supported. For example, a high security system could automatically lock a workstation every 10 minutes and force users to reauthenticate themselves.

Window Stations and Desktops

A console/keyboard/mouse combination is represented in Windows NT as a window-station object, and within a window station several desktop objects can exist. A desktop is a collection of the windows that are visible at any one time.

In Windows NT, the window station object representing the physical screen, keyboard, and mouse is called *WinSta0*. Winlogon creates the following desktops in this window station:

Desktop	Description
Winlogon desktop	This is the desktop Winlogon uses (and GINAs should also use) for interactive identification and authentication, and other secure dialog boxes. Winlogon switches to this desktop automatically once it receives a secure attention sequence (SAS).
Application desktop	Each time a user successfully logs on, an application desktop is created for that logon session where all logged-on user activity will take place. This desktop is protected so no one but the system and the interactive logon session can access it. Note that only a particular *instance* of the logged-on user can access the desktop. If the interactive user also activates a process using the service controller, that service application will not be able to access the application desktop.
Screen saver desktop	This desktop is used when a screen saver is running. If a user is logged on, this desktop allows both system and the logon session access to the desktop. Otherwise, it allows only system access to the desktop.

When Winlogon initializes, it registers the SAS of CTRL+ALT+DEL with the system. Making this the first process ensures that no other application has hooked that key sequence. At this time, Winlogon creates three desktops: one for itself, one for the user's applications, and one for the screen saver.

As the owner of these desktops, Winlogon (and GINA) can switch the active, or visible, desktop to any of the three desktops. In general, GINA developers need not change the active thread desktop. Winlogon generally sets the desktop appropriately before dispatching to the GINA DLL. The description of each GINA dispatch routine clearly defines which desktop is established as the active thread desktop for that call.

Environment

A GINA DLL operates in the context of the Winlogon process and, as such, is invoked very early during the boot process. The DLL must follow rules so the integrity of the system is maintained, particularly in respect to interaction with the user.

Devices

The most common use of GINA is to communicate with an external device such as a smart-card reader. It is essential to set the start parameter for the device driver to system (WINNT.H: SERVICE_SYSTEM_START) to ensure that the driver is loaded by the time GINA is invoked.

Dialog Services

Winlogon implements two time-out functions. During any secure dialog, such as logon or unlocking a workstation, Winlogon can time-out the dialog boxes and return to the dialog process for the dialog.

Winlogon is also responsible for screen saver activation and termination. To enable Winlogon to perform these time-out functions, Winlogon provides a set of dialog services for GINAs. GINAs must use these services instead of the typical Win32® counterpart of these services. Otherwise, Winlogon could behave incorrectly and possibly cause system problems.

The Winlogon services provided for this purpose are:

Function	Description
WlxMessageBox	Similar to the Win32 **MessageBox** function.
WlxDialogBox	Similar to the Win32 **DialogBox** function.
WlxDialogBoxIndirect	Similar to the Win32 **DialogBoxIndirect** function.
WlxDialogBoxParam	Similar to the Win32 **DialogBoxParam** function.
WlxDialogBoxIndirectParam	Similar to the Win32 **DialogBoxIndirectParam** function.

GINA DLLs can also receive WLX_WM_SAS messages from Winlogon. These are sent to active dialog boxes if an SAS is received from either GINA or CTRL+ALT+DEL when Winlogon is so configured. This is useful if GINA is in the process of prompting for the matching PIN for a smart card and the card is removed from the smart-card reader. Winlogon provides WLX_DLG_SAS as an end code for use when that occurs.

Time-outs are also delivered in this manner. A WLX_WM_SAS message is sent with WLX_SAS_TYPE_SCRNSVR_TIMEOUT or WLX_SAS_TYPE_INPUT_TIMEOUT. The dialog box will end with an appropriate exit code. This lets GINA writers hook the time-out notifications.

GINA DLLs can have dialog boxes terminated by Winlogon with the code WLX_DLG_USER_LOGOFF. This indicates that the user has logged off during the running of the dialog box (for example, by calling the **ExitWindowsEx** function from another thread).

Messages to GINA

Winlogon will send messages to GINA while a dialog box is displayed. These are all encapsulated in the WLX_WM_SAS message as follows:

SAS Type in *wParam* Parameter	**Description**
WLX_SAS_TYPE_TIMEOUT	No user input was received within the specified time-out period.
WLX_SAS_TYPE_CTRL_ALT_DEL	A CTRL+ALT+DEL key sequence was received.
WLX_SAS_TYPE_SCRNSVR_TIMEOUT	The screen saver time-out period has expired, and a screen saver will be invoked.
WLX_SAS_TYPE_USER_LOGOFF	The user has logged off.

For time-outs and logoffs, Winlogon will end the dialog once the message has been sent. This message is sent so the dialog can respond in a useful manner (for example, by closing itself down if a logoff has occurred).

For input time-outs, the dialog is ended with the code WLX_DLG_INPUT_TIMEOUT. For screen saver time-outs, the dialog is ended with WLX_DLG_SCREEN_SAVER_TIMEOUT. For logoffs, the dialog is ended with WLX_DLG_USER_LOGOFF.

Screen Savers

Screen savers run on their own desktops, so they cannot interfere with a user's applications or with Winlogon's own windows. The screen saver type is loaded from the current user profile (HKEY_CURRENT_USER\Control Panel\Desktop). If the screen saver is marked as secure, Winlogon will place the workstation in the locked state after the screen saver runs.

User Profiles

In a standard Windows NT system, interactively logged-on users are given a *profile*. A profile is a registry hive that is tailored to a particular user. The profile is generally used to save user-specific information such as screen appearance (colors and border widths), mouse click speeds, whether there is a screen saver, and whether the screen saver is secure. This profile, referenced using the special registry key HKEY_CURRENT_USER, is loaded by Winlogon during the interactive boot process.

The interface between Winlogon and GINA DLLs includes information passed back from GINA that allows Winlogon to locate and load the logged-on user's profile.

Network Providers

You can configure a Windows NT system to support zero or more *network providers*. Each of these network providers can specify that it requires special interactive authentication processing. This capability allows installed networks to collect identification and authentication information specific to each network, yet allows them to collect it during normal logon and under the secure umbrella of Winlogon's context and desktop.

Winlogon calls network providers under a number of circumstances. Following a successful logon, Winlogon calls network providers so they can collect credentials and authenticate the user for their network. Winlogon also calls network providers when users change their passwords. This lets each user maintain a single password for use on all networks.

The **WLX_MPR_NOTIFY_INFO** structure, which is optional, provides this functionality and is used in a number of Winlogon and GINA functions. This structure includes the following members:

Member	Description
pszUserName	The account name of the logged-on user.
pszDomain	The domain name of the logged-on user. Not all authentication models have a domain concept (or its equivalent), so this member can be NULL.
pszPassword	When the user gave a clear-text password during authentication, providing it here lets other network providers use the same password (to achieve single logon) without prompting the user.
pszOldPassword	After a password change, providing the original password here as well as the new password in the **pszPassword** member, lets network providers upgrade their passwords without prompting the user.

If a GINA developer chooses not to provide this information to network providers for security reasons, or any other reason, this entire structure is optional. A NULL pointer can be provided requiring network providers to prompt for all their information.

Responsibilities and Features

Each component of the interactive logon process has a set of responsibilities. This section defines those responsibilities and indicates where you can implement features not contained in the standard version of Windows NT.

This section also describes the optional features of MSGINA.DLL. If you are developing a replacement GINA, you may want to implement some or all of these features as well.

Responsibilities of Winlogon

Winlogon has the following responsibilities:

- **Window Station and Desktop Protection**

  Winlogon sets the protection of the window station and corresponding desktops to ensure each is properly accessible. In general, this means that the local system has full access to these objects and that an interactively logged-on user (if there is one) has read access to the window station object and full access to the application desktop object.

- **Standard SAS Recognition**

  Winlogon has special hooks into the User32 server that allow it to monitor CTRL+ALT+DEL as a secure attention sequence. Winlogon makes this special relationship available to GINAs to use as their SAS, or as part of their SAS. That is, in general, GINAs should monitor SASes on their own. Any GINA that uses the standard CTRL+ALT+DEL key combination as its SAS (or one of several SASes it supports) should use the Winlogon support provided for this purpose.

- **SAS Routine Dispatching**

  When Winlogon encounters a standard SAS (if it is configured to monitor a standard SAS) or when an SAS is delivered to Winlogon by GINA, Winlogon sets the state accordingly, changes to the Winlogon desktop, and dispatches to one of the GINA's SAS processing routines.

- **User Profile Loading**

  When users log on, their user profiles are loaded into the registry. In this way, the user's processes can use the special registry key HKEY_LOCAL_USER. Winlogon does this automatically following successful logon, but before activation of the shell for the newly logged-on user.

- **Assignment of Security To User Shell**

  When a user logs on, GINA is responsible for creating one or more initial processes for that user (see Responsibilities of GINA). Winlogon provides a service for GINA in which to apply the newly logged-on user's security to these processes. Another option is that GINA can call the WIN32 function, **CreateProcessAsUser**, and let the system provide the service.

- **Screen Saver Control**

  Winlogon monitors keyboard and mouse activity to determine when to activate screen savers. Once the screen saver is activated, Winlogon continues to monitor keyboard and mouse activity to determine when to terminate the screen saver. If the screen saver is marked as secure, Winlogon treats the workstation as locked. When there is mouse or keyboard activity, Winlogon invokes GINA's **WlxDisplayLockedNotice** function and normally locked workstation behavior resumes. If the screen saver is not secure, any keyboard or mouse activity terminates the screen saver without notification to GINA.

- **Multiple Network Provider Support**

  Multiple networks installed on a Windows NT can be included in the authentication process and in password updating operations. This inclusion lets additional networks gather identification and authentication information all at one time during normal logon, using Winlogon's secure desktop. Some of the parameters required in the Winlogon services available to GINAs explicitly support these additional network providers.

Responsibilities of GINA

A GINA DLL has the following responsibilities:

- **SAS Monitoring**

  GINA is responsible for recognizing an SAS and monitoring for those events. Note that there can be more than one SAS and the definition of secure attention sequence can change over time. For example, there can be one set of secure attention sequences when Winlogon is in the logged-off state and another set when it is in the logged-on state.

  Winlogon services are provided to assist GINA in using the CTRL+ALT+DEL key combination as a secure attention sequence.

- **SAS Processing**

  One reason for modularizing Winlogon and making GINA replaceable is to provide alternative identification and authentication mechanisms. To do this, GINA must present *all* user interfaces resulting from the recognition of a secure attention sequence. When no user is logged on, GINA is responsible for collecting identification and authentication (as well as any other allowed, nonauthenticated functions). When a user is logged on, GINA is responsible for presenting whatever options or taking whatever actions are deemed appropriate. For example, in a system that includes a smart card, it may be appropriate to automatically lock the workstation if the user removes the smart card.

- **Shell Activation**

  When a user logs on, GINA is responsible for creating one or more initial processes for that user. (In this documentation, it is assumed that these initial processes present an interface to the user. However, the processes can actually be any processes and do not necessarily have to interact with the user.) These processes are referred to as the *user shell* or just the *shell*. As part of shell activation, GINA must assign the newly logged-on user's token to the processes, so the initial process(es) must be created suspended and the token assigned before the process(es) can run. Winlogon provides a service to assist GINA in assigning the token.

MSGINA.DLL Features

If you are writing a GINA to replace the Microsoft standard GINA (MSGINA), you may want to provide some or all of the MSGINA functionality. Following is a list of these features and a brief description of how they are controlled.

Registry key values control the availability or behavior of many of the MSGINA features. Unless otherwise noted, these key values belong to a registry key referred to as the Winlogon key and have value types of [REG_SZ]. The actual path of the Winlogon key is:

```
\HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\CurrentVersion\Winlogon
```

Having [REG_SZ] value types makes it convenient to query the key values using the **GetProfileString** and **GetProfileInt** functions.

- **Legal Notification Dialog Box**

  In many countries and states it is legal for anyone to walk up to a system, log on, and begin working unless there are notifications indicating that the system is intended for the private use of authorized users. Also, many users want company-specific messages displayed before normal logon. MSGINA uses two registry key values of the Winlogon key to implement such capabilities. If either key value is present and contains a non-null string, then a Legal Notice dialog box is displayed before the usual Welcome screen. The caption of the dialog box is taken from one of these key values, while its message is taken from the other.

  These key value names are:

Value	Description
LegalNoticeCaption	Any string you want displayed as the dialog caption.
LegalNoticeText	Any string you want displayed as the dialog body.

- **Don't Display Last Username**

  By default, the MSGINA logon screen displays the name of the last user to successfully log on to a computer. To alter this, MSGINA uses a registry key value of the Winlogon key. If this key value is present and contains a string value of 1, no user name is displayed in the logon dialog box.

- **AutoAdmin Logon**

  This feature allows a Windows NT system to log on a user automatically. This feature can be used in two ways:

  1. To enable a user to be automatically logged on exactly once. If the user logs off or the system is shut down, the user will not be automatically logged on afterward. The user must have a null password in order for this feature to be used.
  2. To enable a user to be logged on every time the system boots or the user logs off.

  This feature uses the following values of the Winlogon key:

Value	Description
AutoAdminLogon	"1" or any other string value.
DefaultUserName	The name of the user to automatically log on.
DefaultDomainName	The name of the domain that the user account is in.
DefaultPassword	The password of the user account, in clear text.

  If the AutoAdminLogon key value is present and contains a 1, an automatic logon will occur based on the DefaultPassword key value. The account being logged onto is specified using the DefaultUserName and DefaultDomainName key values. If present and non-null, the password in

DefaultPassword is also used.

If an automatic logon is being performed but the DefaultPassword key value is not present or is null, this is a one-time only autologon. The AutoAdminLogon key value will be set to contain a zero before the logon is attempted. This prevents any future autologons.

There is one additional caveat to autologon. If an autologon is indicated, MSGINA checks the state of the SHIFT key. If it is held down, it is assumed that the user wants to override autologon and provide identification and authentication information interactively. This is a critical feature when you are debugging a dedicated application. The SHIFT override of automatic logon is disabled if the **IgnoreShiftOverride** key value is present and has a value of 1.

- **Allow Unauthenticated Shutdown**

  You can configure MSGINA to include a *Shutdown* button in the logon dialog box. This lets users shut down the system without first logging on. Inclusion of this button is controlled by the following key value:

Value	Description
ShutdownWithoutLogon	"1" to include button; any other value to exclude button.

- **USERINIT.EXE Activation**

  USERINIT.EXE is an application activated by MSGINA at user logon time. It runs in the newly logged-on user's context and on the user's desktop. Its purpose is to set up the user's environment. This includes restoring net uses, establishing profile settings (such as fonts and screen colors), and running logon scripts, including those of network providers. It then activates the user shell program(s), which inherit the environment that USERINIT.EXE sets up. The shell programs that USERINIT.EXE activates are listed in the shell value under the Winlogon registry key.

  There can be more than one program listed under Winlogon, each separated by a comma. By default, Explorer is listed. If the shell registry value is not found or has no programs listed, Explorer is activated as the default shell.

- **Logged-On Security Options**

  When logged on, if a user enters an SAS, the user is presented with a security options screen. Among the options listed are:

  - Shut down the system
  - Log off
  - Change your password
  - Go to the task list
  - Lock the workstation

  An alternative GINA may provide similar options when an SAS is provided while a user is logged on.

Interaction Between Winlogon and GINA

To assist you in keeping track of how Winlogon and GINA DLLs work, here is the flow and response process:

Boot

Winlogon calls GINA's **WlxNegotiate** function.

Winlogon calls GINA's **WlxInitialize** function.

State of the workstation becomes "No one logged on."

No one logged on

(GINA monitoring devices for SAS).

GINA calls the **WlxSasNotify** function to indicate an SAS has been provided.

Winlogon delivers the SAS back to GINA by calling GINA's **WlxLoggedOutSas** function.

User logged on

GINA calls the **WlxSasNotify** function.

Winlogon delivers SAS back to GINA by calling GINA's **WlxLoggedOnSas** function.

User logged on, wants to lock computer

GINA calls the **WlxSasNotify** function.

Winlogon delivers the SAS back to GINA by calling GINA's **WlxLoggedOnSas** function.

GINA returns WLX_LOCKWINSTA.

User logged on, window station locked, wants to unlock computer

GINA calls the **WlxSasNotify** function.

Winlogon delivers the SAS back to GINA by calling GINA's **WlxWkstaLockedSas** function.

GINA returns WLX_UNLOCKWINSTA.

User logged on, program issues **ExitWindowsEx**

Winlogon calls GINA's **WlxLogoff** function.

User logged on, wants to log off using SAS

GINA calls the **WlxSasNotify** function.

Winlogon delivers the SAS back to GINA by calling GINA's **WlxLoggedOnSas** function.

GINA returns WLX_LOGOFFUSER.

Winlogon calls GINA's **WlxLogoff** function.

User logged on, wants to log off and shut down using **ExitWindowsEx**

Winlogon calls GINA's **WlxLogoff** function.

Winlogon calls GINA's **WlxShutdown** function.

User logged on, wants to log off and shut down using SAS

GINA calls the **WlxSasNotify** function.

Winlogon delivers SAS back to GINA by calling GINA's **WlxLoggedOnSas** function.

GINA returns WLX_LOGOFFANDSHUTDOWN.

Winlogon calls GINA's **WlxLogoff** function.

Winlogon calls GINA's **WlxShutdown** function.

Loading and Running a GINA DLL

Windows NT is shipped to load and execute the standard Microsoft GINA (MSGINA.DLL). To load a different GINA, you must alter the following registry key value:

Key Name:	\HKEY_LOCAL_MACHINE\Software\Microsoft\ Windows NT\CurrentVersion\Winlogon
Value Name:	GinaDLL
Value Type:	[REG_SZ]

If this key value is present and non-null, it should have a GINA DLL name. Winlogon will load and use that GINA DLL.

Building and Testing a GINA DLL

All functions, prototypes, structures, and constants are defined in the WINWLX.H header file.

To test a GINA DLL, use the WINLOGON.EXE from a checked version of the operating system which is available with the Windows NT DDK. The checked version of Winlogon supports the following features for debugging GINAs:

- You can use the following syntax to create a section in WIN.INI to specify Winlogon debugging flags:

```
[WinlogonDebug]
DebugFlags=Flag1[, Flag2...]
```

  The *DebugFlags* variable in a [WinlogonDebug] section can specify one or more of the following debugging flags:

Debugging flag	Description
Trace	Verbose trace information is sent to the debugger.
Init	Initialization.
SAS	Traces SAS notifications.
State	Traces state.
CoolSwitch	The CTRL+ALT+Shift+TAB key combination will cause a debug break in Winlogon.

- You can cause Winlogon to start in a debugger on the checked version from the Windows NT Device Driver Kit (DDK) by adding the following entry to the registry:

```
Machine:Software\Microsoft\Windows NT\CurrentVersion
    Image File Execution Options
        winlogon.exe
            Debugger = REG_SZ ntsd -d
```

  Note that you must use NTSD to debug Winlogon; you cannot use MSVC.

GINA DLL Interface Functions

A GINA DLL must export the following functions, which are called by Winlogon.

WlxActivateUserShell
WlxDisplayLockedNotice
WlxDisplaySASNotice
WlxInitialize
WlxIsLockOk
WlxIsLogoffOk
WlxLoggedOnSAS
WlxLoggedOutSAS
WlxLogoff
WlxNegotiate
WLXScreenSaverNotify
WlxShutdown
WlxStartApplication
WlxWkstaLockedSAS

Note Some of the above functions contain a description of the workstation state for that function. This description indicates what desktop GINA can expect to have set for its thread when the function call is made. In addition, the description also indicates whether the desktop is locked or unlocked. When it is locked, anyone, including GINA, is prevented from displaying another desktop. When it is unlocked, it permits the display of other desktops.

WlxActivateUserShell

The **WlxActivateUserShell** function is implemented by a replacement GINA DLL. Winlogon calls this function following a successful logon. Its purpose is to request that GINA activate the user shell program.

Note that the user shell should be activated in **WlxActivateUserShell** rather than in GINA's **WlxLoggedOutSAS** function. This gives Winlogon a chance to update its state, including setting workstation and desktop protections, before any logged-on user processes are allowed to run.

```
BOOL WlxActivateUserShell(
  PVOID pWlxContext,
  PWSTR pszDesktopName,
  PWSTR pszMprLogonScript,
  PVOID pEnvironment
);
```

Parameters

pWlxContext
(IN parameter) Context value associated with this window station. This is the context value that GINA returns when Winlogon calls the **WlxInitialize** function for this window station.

pszDesktopName
(IN parameter) Name of the desktop on which to start the shell. Pass this string to the Win32 **CreateProcess** or **CreateProcessAsUser** function through the *lpDesktop* member of the **STARTUPINFO** structure.

pszMprLogonScript
(IN parameter) Script names returned from the provider DLLs. Provider DLLs can return scripts to be executed during logon. GINA can reject these, but Winlogon will provide them if they are there.

pEnvironment
(IN parameter) Initial environment for the process. Winlogon creates this environment and hands it off to GINA. GINA can modify this environment before using it to initialize the user's shell.

Return Values

If the GINA DLL successfully started the shell processes, **WlxActivateUserShell** should return TRUE.

If the GINA DLL could not start the shell, **WlxActivateUserShell** should return FALSE. This indicates that Winlogon should terminate the logon session.

Remarks

Before calling the **WlxActivateUserShell** function, Winlogon sets the desktop and workstation state as follows:

Desktop	The current desktop is the Winlogon desktop.
Workstation	The desktop is not locked, thus allowing the display of other desktops.

Example

The following code shows a sample **WlxActivateUserShell** implementation:

```
WINAPI
WlxActivateUserShell(
    PVOID            pWlxContext,
    PWSTR            pszDesktopName,
    PWSTR            pszMprLogonScript,
    PVOID            pEnvironment)
{
    BOOL st;

    st = CreateProcessAsUser(hToken, ...);
    if (!st) {
        return (FALSE);    // Could not start shell.  Tell Winlogon to start
a logoff
 }
    return(TRUE);
}
```

See Also

WlxLoggedOutSAS, **WlxInitialize**

WlxDisplayLockedNotice

The **WlxDisplayLockedNotice** function is implemented by a replacement GINA DLL. Winlogon calls this function when the workstation is placed in the locked state. This lets GINA display information about the lock, such as who locked the workstation and when. GINA should display a dialog box that will be interrupted by a WLX_WM_SAS message, much like the **WlxDisplaySASNotice** function.

```
VOID WlxDisplayLockedNotice(
  PVOID
pWlxContext
);
```

Parameters

pWlxContext
(IN parameter) Context value associated with this window station. This is the context value that GINA returns when Winlogon calls the **WlxInitialize** function for this window station.

Return Values

None.

Remarks

Before calling the **WlxDisplaySASNotice** function, Winlogon sets the desktop and workstation state as follows:

Desktop	The current desktop is the Winlogon desktop.
Workstation	The desktop is locked, preventing the display of other desktops.

See Also

WlxDisplaySASNotice

WlxDisplaySASNotice

The **WlxDisplaySASNotice** function is implemented by a replacement GINA DLL. Winlogon calls this function when no user is logged on.

VOID WlxDisplaySASNotice(
  PVOID *pWlxContext*
);

Parameters

pWlxContext
(IN parameter) Context value associated with this window station. This is the context value that GINA returns when Winlogon calls the **WlxInitialize** function for this window station.

Return Values

None.

Remarks

Before calling the **WlxDisplaySASNotice** function, Winlogon sets the desktop and workstation state as follows:

Desktop	The current desktop is the Winlogon desktop.
Workstation	The desktop is locked, preventing the display of other desktops.

Example

The following code shows a sample **WlxDisplaySASNotice** implementation:

```
VOID
WINAPI
WlxDisplaySASNotice(
    PVOID    pWlxContext)
{
    PGINA_CONTEXT    pContext;

    pContext = (PGINA_CONTEXT) pWlxContext;
    pContext->pWlxFuncs->WlxDialogBox(    pContext->hWlx,
    pContext->hDllInstance,
     MAKEINTRESOURCE(IDD_WELCOME_DIALOG),
     NULL,
     WelcomeDlgProc);
}
```

See Also

WlxInitialize

WlxInitialize

The **WlxInitialize** function is implemented by a replacement GINA DLL. Winlogon calls this function once for each window station present on the computer.

```
BOOL WlxInitialize(
  LPWSTR lpWinsta,
  HANDLE hWlx,
  PVOID pvReserved,
  PVOID pWinlogonFunctions,
  PVOID *pWlxContext
);
```

Parameters

lpWinsta
Points to the name of the window station being initialized.

hWlx
Handle to Winlogon. GINA must provide this handle in subsequent Winlogon calls related to the specified window station.

pvReserved
Reserved.

pWinlogonFunctions
Receives a pointer to a Winlogon function dispatch table. The contents of the table is dependent on the GINA DLL version returned from the **WlxNegotiate** call. The table does not change, so the GINA DLL can reference the table instead of copying it.

pWlxContext
(OUT parameter) This parameter lets GINA return a 32-bit context value that will be provided in all future calls related to this window station. Generally, the value returned will be similar to a pointer to a context structure allocated by GINA for this window station.

Return Values

If the GINA DLL is successfully initialized, **WlxInitialize** should return TRUE.

If the GINA DLL is not successfully initialized, **WlxInitialize** should return FALSE. In this case, the system will not boot.

Remarks

The **WlxInitialize** function is called once for each window station present on the computer. This lets the DLL initialize itself, including obtaining addresses of Winlogon support functions used by this DLL.

The DLL can return a context pointer that will be passed in all future interactions from Winlogon to GINA. This lets GINA keep a global context associated with the specified window station.

Note Windows NT supports only one window station, called *Winsta0*. Additional physical window stations may be supported in future releases.

Before calling the **WlxInitialize** function, Winlogon sets the desktop and workstation state as follows:

Desktop	The current desktop is the Winlogon desktop.
Workstation	The desktop is locked, preventing the display of another desktop.

Example

The following code shows a sample **WlxInitialize** implementation:

```
static PWLX_DISPATCH_VERSION_1_0  pWinlogon;
BOOL
WlxInitialize(
    LPWSTR        lpWinsta,
    HANDLE        hWlx,
    PVOID         pWinlogonFunctions,
    PVOID        *pContext)
{
    PGINA_CONTEXT pGinaContext;

    // allocate a context block for this window station
    AllocateAndInitGinaContext( &pGinaContext, hWlx );
    (*pContext) = pGinaContext;

    // Save the pointer to the Winlogon dispatch table
    pWinlogon = (PWLX_DISPATCH_VERSION_1_0)pWinlogonFunctions;

    // Any other initialization that the DLL needs, e.g. open a card
    // reader device

    // Optionally, at this point, call WlxSASNotify, or return

    return(TRUE);
}
```

See Also

WlxNegotiate

WlxIsLockOk

The **WlxIsLockOk** function is implemented by a replacement GINA DLL. Winlogon calls this function before locking the workstation if, for example, a screen saver is marked as secure.

BOOL WlxIsLockOk(
  PVOID
pWlxContext
);

Parameters

pWlxContext
(IN parameter) Context value associated with this window station. This is the context value that GINA returns when Winlogon calls the **WlxInitialize** function for this window station.

Return Values

If it is acceptable to lock the workstation, **WlxIsLockOk** should return TRUE.

If it is not acceptable to lock the workstation, **WlxIsLockOk** should return FALSE.

See Also

WlxInitialize

WlxIsLogoffOk

The **WlxIsLogoffOk** function is implemented by a replacement GINA DLL. Winlogon calls this function when the user has initiated a logoff operation (for example by calling the **ExitWindowsEx** function). GINA can determine whether the logoff attempt should be allowed.

```
BOOL WlxIsLogoffOk(
  PVOID
pWlxContext
);
```

Parameters

pWlxContext
(IN parameter) Context value associated with this window station. This is the context value that GINA returns when Winlogon calls the **WlxInitialize** function for this window station.

Return Values

If it is acceptable to log off the user, **WlxIsLogoffOk** should return TRUE.

If it is not acceptable to log off the user, **WlxIsLogoffOk** should return FALSE.

See Also

WlxInitialize

WlxLoggedOnSAS

The **WlxLoggedOnSAS** function is implemented by a replacement GINA DLL. Winlogon calls this function when a secure SAS event is received and a user is logged on and the workstation is unlocked. An SAS in this situation indicates that the user needs to contact the security system.

Note that an SAS in this situation is distinguished from an SAS when the workstation is locked.

```
int WlxLoggedOnSAS(
  PVOID pWlxContext,
  DWORD dwSasType,
  PVOID pReserved
);
```

Parameters

pWlxContext
(IN parameter) Context value associated with this window station. This is the context value that GINA returns when Winlogon calls the **WlxInitialize** function for this window station.

dwSasType
(IN parameter) Indicates the type of SAS that occurred. Values from zero to WLX_SAS_TYPE_MAX_MSFT_VALUE are reserved to define Microsoft standard SAS types. GINA developers can use values greater than WLX_SAS_TYPE_MAX_MSFT_VALUE to define additional SAS types. The following values are predefined:

Value	Description
WLX_SAS_TYPE_CTRL_ALT_DEL	Indicates that the user has typed the standard CTRL+ALT+DEL SAS.

pReserved
(IN parameter) Reserved.

Return Values

The function should return one of the following values:

Return Values	Description
WLX_SAS_ACTION_NONE	Return to the default desktop.
WLX_SAS_ACTION_LOCK_WKSTA	Lock the workstation and wait for next SAS.
WLX_SAS_ACTION_LOGOFF	Log the user off the workstation.
WLX_SAS_ACTION_SHUTDOWN	Log the user off and shut down the computer.
WLX_SAS_ACTION_SHUTDOWN_REBOOT	Shut down and reboot the computer.
WLX_SAS_ACTION_SHUTDOWN_POWER_OFF	Shut down and turn off the computer, if hardware allows.
WLX_SAS_ACTION_PWD_CHANGED	Indicates that the user changed his or her password. Notify network providers.
WLX_SAS_ACTION_TASKLIST	Invoke the task list.

Remarks

This function is generally used when the logged-on user wants to shut down, log out, or lock the workstation. The extension DLL can lock the workstation by returning WLX_LOCK_WKSTA. Winlogon locks the workstation and calls **WlxWkstaLockedSAS** the next time an SAS is received.

The extension DLL can use the profile to determine what information is needed about the system.

Before calling your **WlxLoggedOnSAS** function, Winlogon sets the desktop and workstation state as follows:

Desktop	The current desktop is the Winlogon desktop.
Workstation	The desktop is locked, preventing the display of other desktops.

Example

The following code shows a sample **WlxLoggedOnSAS** implementation:

```
WINAPI
WlxLoggedOnSAS(
      PVOID                  pWlxContext,
      DWORD                  dwSasType,
      PVOID                  pReserved)
{
      int      Ret;
      switch (

      Ret = MyLoggedOnSasDialog( pWlxContext );
      switch (Ret)
      {
            case      IDCANCEL:
                  return (WLX_SAS_ACTION_NONE);

            case      MY_ID_LOCKWINSTA:
                  return (WLX_SAS_ACTION_LOCK_WKSTA);

            case      MY_ID_CHANGEPWD:
                  // run a password change dialog

            case      MY_ID_TASKLIST:
                  return(WLX_SAS_ACTION_TASKLIST );

            case      MY_ID_SHUTDOWN:
                  return(WLX_SAS_ACTION_SHUTDOWN);

            case      MY_ID_LOGOFF:
                  return(WLX_LOGOFF);
      }
}
```

See Also

WlxInitialize, **WlxWkstaLockedSAS**

WlxLoggedOutSAS

The **WlxLoggedOutSAS** function is implemented by a replacement GINA DLL. Winlogon calls this function when a secure attention sequence (SAS) event is received and no user is logged on. This indicates that a logon attempt should be made.

```
int WlxLoggedOutSAS(
  PVOID pWlxContext,
  DWORD dwSasType,
  PLUID pAuthenticationId,
  PSID pLogonSid,
  PDWORD pdwOptions,
  PHANDLE phToken,
  PWLX_MPR_NOTIFY_INFO pMprNotifyInfo,
  PVOID *pProfile
);
```

Parameters

pWlxContext

(IN parameter) Context value associated with this window station. This is the context value that GINA returns when Winlogon calls the **WlxInitialize** function for this window station.

dwSasType

(IN parameter) Indicates the type of SAS that occurred. Values from zero to WLX_SAS_TYPE_MAX_MSFT_VALUE are reserved to define Microsoft standard SAS types. GINA developers can use values greater than WLX_SAS_TYPE_MAX_MSFT_VALUE to define additional SAS types. The following values are predefined:

Value	Description
WLX_SAS_TYPE_CTRL_ALT_DEL	Indicates that the user typed the standard CTRL+ALT+DEL SAS.

pAuthenticationId

(OUT parameter) The authentication identifier associated with the current logon session. You can get this value by using the **GetTokenInformation** function to retrieve a **TOKEN_STATISTICS** structure for the token returned by the **LogonUser** function.

pLogonSid

(IN OUT parameter) This parameter contains a pointer to a security identifier (SID). The SID is unique to the current logon session. Winlogon uses this SID to change the protection on the window station and application desktop so the newly logged-on user can access them.

Winlogon provides a SID. You can also get the SID by using the **GetTokenInformation** function to retrieve a **TOKEN_GROUPS** structure for the token returned by the **LogonUser** function. To do this, search the array returned in the **TOKEN_GROUPS** structure for the group with the SE_GROUP_LOGON_ID attribute.

pdwOptions

(OUT parameter) Points to a 32-bit variable that receives a set of logon options. The following option is defined:

Value	Description
WLX_LOGON_OPT_NO_PROFILE	Indicates that Winlogon must *not* load a profile for the logged-on user. Either the GINA DLL will take care of this activity, or the user does not need a profile.

phToken

(OUT parameter) Points to a handle variable. If the logon operation was successful, you must set this handle to a token that represents the logged-on user. Use the **LogonUser** function to get this token. When the user logs off, Winlogon closes this handle and then calls the **WlxLogoff** function. If you will need this handle during your **WlxLogoff**, make a duplicate of the handle before returning it to Winlogon.

pMprNotifyInfo

(OUT parameter) This parameter contains a pointer to a structure for returning password information to other network providers. GINA is not required to return this information. If GINA returns password information, it should fill in the pointers in the structure. Any NULL field in the structure will be ignored by Winlogon. GINA should use the **LocalAlloc** function to allocate the strings individually; Winlogon will then free them.

pProfile

(OUT parameter) Upon return from a successful authentication, the *pProfile* parameter must point to one of the WLX_PROFILE_xxx structures. The first **DWORD** in the profile structure is used to indicate which of the WLX_PROFILE_xxx structures is being returned. The information in this structure is used by Winlogon to load the logged-on user's profile. This structure and any strings or buffers pointed to from within this structure are freed by Winlogon when no longer needed.

Return Values

The function should return one of the following values:

WLX_SAS_ACTION_LOGON	A user has logged on.
WLX_SAS_ACTION_NONE	A logon attempt was unsuccessful or canceled.
WLX_SAS_ACTION_SHUTDOWN	The user requested that the system be shut down.

Remarks

Before calling your **WlxLoggedOutSAS** function, Winlogon sets the desktop and workstation state as follows:

Desktop	The current desktop is the Winlogon desktop.
Workstation	The desktop is locked, preventing the display of other desktops.

Example

The following code shows a sample **WlxLoggedOutSAS** implementation:

```
#define    LOCAL_GROUP_COUNT (1)
WINAPI
WlxLoggedOutSAS(
       PVOID                     pWlxContext,
       DWORD                     dwSasType,
       PLUID                     pAuthenticationId,
       PSID                      pLogonSid,
       PDWORD                    pdwOptions,
       PHANDLE                   phToken,
       PWLX_MPR_NOTIFY_INFO      pMprNotifyInfo,
       PVOID                     pProfile)
{
       HANDLE                    hToken;
       NTSTATUS                  Status;
       INT                       LengthLocalGroups;
       PTOKEN_GROUPS       LocalGroups;

       //
       // Gather credentials, e.g. from card reader
       //

       if (!LogonUser( ..., &hToken))
       {
              return(WLX_SAS_ACTION_NONE);
       }

       *phToken = hToken;
                              // Otherwise, return NULL and MPR apps will not
get any credentials
       return (WLX_SAS_ACTION_LOGON);
}
```

See Also

WlxInitialize

WlxLogoff

The **WlxLogoff** function is implemented by a replacement GINA DLL. Winlogon calls this function to notify GINA of a logoff operation on this workstation. No action is necessary.

VOID WlxLogoff(
  PVOID
pWlxContext
);

Parameters

pWlxContext
(IN parameter) Context value associated with this window station. This is the context value that GINA returns when Winlogon calls the **WlxInitialize** function for this window station.

Return Values

None.

Remarks

Before calling the **WlxLogoff** function, Winlogon sets the desktop and workstation state as follows:

Desktop	The current desktop is the Winlogon desktop.
Workstation	The desktop is locked, preventing the display of other desktops.

Example

The following code shows a sample **WlxLogoff** implementation:

```
VOID
WlxLogoff(
    PVOID            pWlxContext)
{
    //
    // Logoff action, if any:
    //
    // for example, spit out the smart card from the smart-card reader...
}
```

See Also

WlxInitialize

WlxNegotiate

The **WlxNegotiate** function is implemented by a replacement GINA DLL. This is the first call made by Winlogon to the GINA DLL. Winlogon and a GINA DLL can use **WlxNegotiate** to determine the version of the interface for which each was written.

BOOL WlxNegotiate(
  DWORD *dwWinlogonVersion*,
  PDWORD *pdwDllVersion*
);

Parameters

dwWinlogonVersion
Version supported by Winlogon.

pdwDllVersion
Version supported by the GINA DLL. This version must be no greater than the version indicated in *dwWinlogonVersion.* This return value establishes which dispatch table will be passed to GINA in subsequent **WlxInitialize** calls.

Return Values

If your GINA DLL can operate with the Winlogon version specified by *dwWinlogonVersion*, **WlxNegotiate** should return TRUE. In this case, Winlogon will continue to initialize.

If your GINA DLL cannot operate with the Winlogon version specified by *dwWinlogonVersion*, **WlxNegotiate** should return FALSE. In this case, Winlogon and the system will not boot.

Remarks

Before calling the **WlxNegotiate** function, Winlogon sets the desktop and workstation state as follows:

Desktop	The current desktop is the Winlogon desktop.
Workstation	The desktop is locked, preventing the display of other desktops.

Example

The following code shows a sample **WlxNegotiate** implementation:

```
WINAPI
WlxNegotiate(
    DWORD        dwWinlogonVersion,
    DWORD        *pdwDllVersion)
{
    if (dwWinlogonVersion < WLX_CURRENT_VERSION)
        // panic: should never happen.
        return(FALSE);

    *pdwDllVersion = WLX_CURRENT_VERSION;
    return(TRUE)
}
```

See Also

WlxInitialize

WlxScreenSaverNotify

The **WlxScreenSaverNotify** function is implemented by a replacement GINA DLL. Winlogon calls this function immediately before a screen saver is activated. This gives the GINA DLL an opportunity to affect the screen saver operation.

VOID WlxScreenSaverNotify(
  PVOID *pWlxContext*,
  BOOL *pfSecure*
);

Parameters

pWlxContext
(IN parameter) Context value associated with this window station. This is the context value that GINA returns when Winlogon calls the **WlxInitialize** function for this window station.

pfSecure
(IN OUT parameter) Indicates on entry whether the current screen saver will be treated as secure (for example, the window station switched to the locked state). On return, Winlogon will use the value that is returned.

Return Values

If the screen saver should be executed, **WlxScreenSaverNotify** should return TRUE.

If the screen saver should not be executed, **WlxScreenSaverNotify** should return FALSE.

Remarks

Before calling your **WlxScreenSaverNofity** function, Winlogon sets the desktop and workstation state as follows:

Desktop	The current desktop is the Winlogon desktop.
Workstation	The desktop is not locked, thus allowing the display of other desktops.

The **WlxScreenSaverNotify** function supercedes the **WlxIsLockOk** function. If the GINA DLL does not export this function, the default behavior is:

```
BOOL
DefaultScreenSaverNofify(
    PVOID        pWlxContext,
    BOOL    *        pfSecure)
{
    if (*pfSecure)
    {
            *pfSecure = WlxIsLockOk();
    }

    return( TRUE );
}
```

Example

The following code shows a sample **WlxScreenSaverNotify** implementation:

```
BOOL
WlxScreenSaverNofify(
    PVOID        pWlxContext,
```

```
    BOOL     *        pfSecure)
{
    //
    // Make all screen savers secure:
    //

    *pfSecure = TRUE;

    return( TRUE );
}
```

See Also

WlxInitialize

WlxShutdown

The **WlxShutdown** function is implemented by a replacement GINA DLL. Winlogon calls this function just before shutting down so GINA can perform any shutdown tasks, such as ejecting a smart card from a reader. The user has already logged off and the **WlxLogoff** function has been called.

```
VOID WlxShutdown(
  PVOID pWlxContext,
  DWORD ShutdownType
);
```

Parameters

pWlxContext

(IN parameter) Context value associated with this window station. This is the context value that GINA returns when Winlogon calls the **WlxInitialize** function for this window station.

ShutdownType

(IN parameter) Indicates the type of shutdown. One of the following values is specified:

Value	Description
WLX_SAS_ACTION_SHUTDOWN	Log the user off and shut down the computer.
WLX_SAS_ACTION_SHUTDOWN_REBOOT	Shut down and reboot the computer.
WLX_SAS_ACTION_SHUTDOWN_POWER_OFF	Shut down and turn off the computer, if the hardware allows.

Return Values

None.

Remarks

Before calling your **WlxShutdown** function, Winlogon sets the desktop and workstation state as follows:

Desktop	If the user is logged on and the workstation is not locked, the current desktop is the application desktop. Otherwise, it is the Winlogon desktop.
Workstation	If the current desktop is the application desktop, the desktop is not locked, thus allowing the display of other desktops. If the current desktop is the Winlogon desktop, the desktop is locked, preventing the display of another desktop.

Example

The following code shows a sample **WlxShutdown** implementation:

```
VOID
 WlxShutdown(
      PVOID                  pWlxContext,
      DWORD                  ShutdownType)
{
      PGINA_CONTEXT      pGinaContext = (PGINA_CONTEXT)pWlxContext;
      CloseHandle(GinaContext->hCardReaderDevice);
      return;
}
```

See Also

WlxInitialize, **WlxLogoff**

WlxStartApplication

The **WlxStartApplication** function is implemented by a replacement GINA DLL. Winlogon calls this function when the system needs an application started in the user's context. In Windows NT 4.0 release, this can occur for two reasons: Explorer has terminated unexpectedly and needs to be restarted, or the extended task manager needs to run. GINA can overide this behavior, if appropriate, through the **WlxStartApplication** function.

BOOL WlxStartApplication(
  PVOID *pWlxContext*,
  PWSTR *pszDesktopName*,
  PVOID *pEnvironment*,
  PWSTR *pszCmdLine*
);

Parameters

pWlxContext
(IN parameter) Context value associated with this window station. This is the context value that GINA returns when Winlogon calls the **WlxInitialize** function for this window station.

pszDesktopName
(IN parameter) Name of the desktop on which to start the shell. Pass this string to the Win32 **CreateProcess** or **CreateProcessAsUser** function through the *lpDesktop* member of the **STARTUPINFO** structure.

pEnvironment
(IN parameter) Initial environment for the process. Winlogon creates this environment and hands it off to GINA. GINA can modify this environment before using it to initialize the user's shell.

pszCmdLine
(IN parameter) Program to execute.

Return Values

If the application started, **WlxScreenSaverNotify** should return TRUE.

If the application did not start, **WlxIsLogoffOk** should return FALSE.

Remarks

Before calling the **WlxScreenSaverNofity** function, Winlogon sets the desktop and workstation state as follows:

Desktop	The current desktop is the Winlogon desktop.
Workstation	The desktop is not locked, thus allowing the display of other desktops

The **WlxStartApplication** function is new. If it is not exported by GINA, then Winlogon will execute the process.

Example

The following code shows a sample **WlxStartApplication** implementation:

```
BOOL
WlxStartApplication(
         PVOID                  pWlxContext,
         PWSTR                  pszDesktopName,
         PVOID                  pEnvironment,
         PWSTR                  pszCmdLine)
```

```
{
        CreateProcess(NULL, pszCmdLine,...);
}
```

See Also

WlxInitialize

WlxWkstaLockedSAS

The **WlxWkstaLockedSAS** function is implemented by a replacement GINA DLL. Winlogon calls this function when it receives a secure attention sequence (SAS) and the workstation is locked. GINA can return indicating the workstation is to remain locked, the workstation is to be unlocked, or the logged-on user is being forced to log off (which leaves the workstation locked until the logoff is completed).

```
int WlxWkstaLockedSAS(
  PVOID pWlxContext,
  DWORD dwSasType
);
```

Parameters

pWlxContext

(IN parameter) Context value associated with this window station. This is the context value that GINA returns when Winlogon calls the **WlxInitialize** function for this window station.

dwSasType

(IN parameter) Indicates the type of SAS that occurred. Values from zero to WLX_SAS_TYPE_MAX_MSFT_VALUE are reserved to define Microsoft standard SAS types. GINA developers can use values greater than WLX_SAS_TYPE_MAX_MSFT_VALUE to define additional SAS types. The following values are predefined:

Value	Description
WLX_SAS_TYPE_CTRL_ALT_DEL	Indicates that the user has typed the standard CTRL+ALT+DEL SAS.
WLX_SAS_TYPE_SCRNSVR_TIMEOUT	Indicates that keyboard/mouse inactivity has lead to screen saver activation. The GINA DLL specifies whether this constitutes a workstation locking event.
WLX_SAS_TYPE_SCRNSVR_ACTIVITY	Indicates that keyboard or mouse activity occurred while a secure screen saver was active.

Return Values

The function should return one of the following values:

Return Values	Description
WLX_SAS_ACTION_NONE	Workstation remains locked.
WLX_SAS_ACTION_UNLOCK_WKSTA	Unlock the workstation.
WLX_SAS_ACTION_FORCE_LOGOFF	Force the user to log off.

Remarks

Before calling the **WlxWkstaLockedSAS** function, Winlogon sets the desktop and workstation state as follows:

Desktop	The current desktop is the Winlogon desktop.
Workstation	The desktop is locked, preventing the display of another desktop.

Example

The following code shows a sample **WlxWkstaLockedSAS** implementation:

```
WINAPI
WlxWkstaLockedSAS(
            PVOID                                   pWlxContext)
{
            //
            //  Validate the user's credentials again (for example, read from
smart-
            //  card reader)
            //

            if (valid credentials)
            {
                        return(WLX_UNLOCK_WKSTA);
            }

            if (administrative override)
            {
                        return(WLX_FORCE_LOGOFF);
            }

            return(WLX_NO_ACTION);
}
```

See Also

WlxInitialize

Winlogon Functions For Use By GINAs

Winlogon exports the following support functions to assist GINA DLLs. Note that, as with all GINA services, these functions are Unicode only.

GINA DLLs are required to use Wlx dialog functions instead of the generic Win32 functions for handling dialog calls. This is necessary for the Winlogon time-outs to operate correctly.

WlxAssignShellProtection
WlxChangePasswordNotify
WlxChangePasswordNotifyEx
WlxCreateUserDesktop
WlxDialogBox
WlxDialogBoxIndirect
WlxDialogBoxIndirectParam
WlxDialogBoxParam
WlxGetSourceDesktop
WlxMessageBox
WlxSasNotify
WlxSetContextPointer
WlxSetReturnDesktop
WlxSetTimeout
WlxSwitchDesktopToUser
WlxSwitchDesktopToWinlogon
WlxUseCtrlAltDel

WlxAssignShellProtection

The **WlxAssignShellProtection** function allows a GINA DLL to have Winlogon assign protection to the shell program of a newly logged-on user. The shell process should be created suspended, then the **WlxAssignShellProtection** function should be called to apply the correct protection to the shell process.

int WlxAssignShellProtection(
  HANDLE *hWlx*,
  HANDLE *hToken*,
  HANDLE *hProcess*,
  HANDLE *hThread*
);

Parameters

hWlx
(IN parameter) Winlogon handle provided to GINA in the **WlxInitialize** call.

hToken
(IN parameter) Handle to a primary token. This token will be used as the primary token for the process referred to in the following Remarks section. The token must be open for TOKEN_DUPLICATE access.

hProcess
(IN parameter) Handle to the process to modify. The process must be created in the suspended state, and this should be the handle returned in the **PROCESS_INFORMATION** structure.

hThread
(IN parameter) Handle to the initial thread of the process.

Return Values

The function returns any errors encountered while trying to assign protection.

Remarks

After the **WlxAssignShellProtection** function returns, the caller:

1. Makes a Primary Token duplicate of the token.
2. Changes the protection on the token so the new user can access it.
3. Changes the protection on the new shell process so the new user can access it.
4. Assigns the duplicate token as the primary token of the shell process.

If there are multiple top-level user shell processes, GINA must make a call for each one.

The **WlxAssignShellProtection** function is an optional call provided for the convenience of GINA developers.

See Also

WlxInitialize

WlxChangePasswordNotify

The **WlxChangePasswordNotify** function should be called by GINA DLLs that have changed a password. This allows the other network providers on the computer to update their passwords as well. (This function has been superceded by the **WlxChangePasswordNotifyEx** function.)

int WlxChangePasswordNotify(
  HANDLE *hWlx*,
  PWLX_MPR_NOTIFY_INFO *pMprInfo*,
  DWORD *dwChangeInfo*
);

Parameters

hWlx
(IN parameter) Winlogon handle provided to GINA in the **WlxInitialize** call.

pMprInfo
(IN parameter) Points to a **WLX_MPR_NOTIFY_INFO** structure that contains Multiple Provider Router (MPR) information. All pointers to memory in this structure will be freed by using the **LocalFree** function. Password strings will be filled with zeros, then freed.

dwChangeInfo
Change the information flags from Network Provider API.

See Also

WlxInitialize, **WLX_MPR_NOTIFY_INFO**

WlxChangePasswordNotifyEx

The **WlxChangePasswordNotifyEx** function lets the GINA DLL optionally specify the particular network provider to notify of a password change. In this way, a GINA DLL can pass a change password request through to a specific network provider.

```
int WlxChangePasswordNotifyEx(
  HANDLE hWlx,
  PWLX_MPR_NOTIFY_INFO pMprInfo,
  DWORD dwChangeInfo,
  PWSTR ProviderName,
  PVOID Reserved
);
```

Parameters

hWlx
(IN parameter) Winlogon handle provided to GINA in the **WlxInitialize** call.

pMprInfo
(IN parameter) Points to a **WLX_MPR_NOTIFY_INFO** structure that contains Multiple Provider Router (MPR) information. All pointers to memory in this structure will be freed by using the **LocalFree** function. Password strings will be filled with zeros, then freed.

dwChangeInfo
Change the information flags from Network Provider API.

ProviderName
(IN parameter) Name of a specific network provider, or NULL to allow the system to notify them all.

Reserved
(IN parameter) Reserved. Must be zero.

Return Values

If the **WlxChangePasswordNotifyEx** function succeeds, zero is returned. Any other value indicates an error.

See Also

WlxInitialize, **WLX_MPR_NOTIFY_INFO**

WlxCreateUserDesktop

The **WlxCreateUserDesktop** function is provided to GINA DLLS so they can create alternate desktops for the user.

```
int WlxCreateUserDesktop(
  HANDLE hWlx,
  HANDLE hToken,
  DWORD Flags,
  PWSTR pszDesktopName,
  PWLX_DESKTOP *ppDesktop
);
```

Parameters

hWlx
(IN parameter) Winlogon handle provided to GINA in the **WlxInitialize** call.

hToken
(IN parameter) Handle to the token of the user for whom the desktop is being created.

Flags
(IN parameter) Flags indicating how to create the desktop. These are the flag values:

Value	Description
WLX_CREATE_INSTANCE_ONLY	Indicates that only this instance of the user has access.
WLX_CREATE_USER	Indicates that any instance of this user has access.

pszDesktopName
(IN parameter) Name of the desktop.

ppDesktop
(OUT parameter) If successful, receives a pointer to a **WLX_DESKTOP** structure for the new desktop that is suitable for passing later into **WlxSetReturnDesktop**.

Remarks

If a handle to the desktop is provided, Winlogon will duplicate the handle. If no handle is provided, Winlogon will attempt to open the desktop named in the *pDesktop* parameter. If the provided desktop is not valid, or is Winlogon or ScreenSaver, the call will fail.

See Also

WlxInitialize, **WlxSetReturnDesktop**

WlxDialogBox

The **WlxDialogBox** function duplicates the Win32 **DialogBox** function, except that it allows Winlogon to time-out the dialog box.

```
int WlxDialogBox(
  HANDLE hWlx,
  HANDLE hInst,
  LPWSTR lpszTemplate,
  HWND hwndOwner,
  DLGPROC dlgprc
);
```

Parameters

hWlx
(IN parameter) Winlogon handle provided to GINA in the **WlxInitialize** call.

hInst
Identifies an instance of the module whose executable file contains the dialog box template.

lpszTemplate
Identifies the dialog box template. This parameter is either the address of a null-terminated character string that specifies the name of the dialog box template, or an integer value that specifies the resource identifier of the dialog box template. If the parameter specifies a resource identifier, its high-order word must be zero and its low-order word must contain the identifier. You can use the **MAKEINTRESOURCE** macro to create this value.

hwndOwner
Identifies the window that owns the dialog box.

dlgprc
Points to the dialog box procedure. For more information about the dialog box procedure, see the description of the **DialogProc** callback function in the Win32 SDK.

Return Values

If the function succeeds, the return value is the *nResult* parameter given in the call to the **EndDialog** function used to terminate the dialog box.

If the function fails, the return value is -1.

Remarks

For more information, see the description of the **DialogBox** function in the Win32 SDK.

See Also

WlxInitialize

WlxDialogBoxIndirect

The **WlxDialogBoxIndirect** function duplicates the Win32 **DialogBoxIndirect** function, except that it allows Winlogon to time-out the dialog box.

```
int WlxDialogBoxIndirect(
  HANDLE hWlx,
  HANDLE hInst,
  LPCDLGTEMPLATE hDialogTemplate,
  HWND hwndOwner,
  DLGPROC dlgprc
);
```

Parameters

hWlx
(IN parameter) Winlogon handle provided to GINA in the **WlxInitialize** call.

hInst
Identifies the instance of the module that creates the dialog box.

hDialogTemplate
Specifies the address of a global memory object that contains a dialog box template used to create the dialog box. The template is in the form of a **DLGTEMPLATE** structure followed by one or more **DLGITEMTEMPLATE** structures. For a full description of these structures, see the Win32 SDK.

hwndOwner
Identifies the window that owns the dialog box.

dlgprc
Points to the dialog box procedure. For more information about the dialog box procedure, see the description of the **DialogProc** callback function in the Win32 SDK.

Return Values

If the function succeeds, the return value is the *nResult* parameter given in the call to the **EndDialog** function used to terminate the dialog box.

If the function fails, the return value is -1.

Remarks

For more information, see the description of the **DialogBoxIndirect** function in the Win32 SDK.

See Also

WlxInitialize

WlxDialogBoxIndirectParam

The **WlxDialogBoxIndirectParam** function duplicates the Win32 **DialogBoxIndirectParam** function, except that it allows Winlogon to time-out the dialog box.

```
int WlxDialogBoxIndirectParam(
  HANDLE hWlx,
  HANDLE hInst,
  LPCDLGTEMPLATE hDialogTemplate,
  HWND hwndOwner,
  DLGPROC dlgprc,
  LPARAM dwInitParam
);
```

Parameters

hWlx
(IN parameter) Winlogon handle provided to GINA in the **WlxInitialize** call.

hInst
Identifies the instance of the module that creates the dialog box.

hDialogTemplate
Specifies the address of a global memory object that contains a dialog box template used to create the dialog box. The template is in the form of a **DLGTEMPLATE** structure followed by one or more **DLGITEMTEMPLATE** structures. For a full description of these structures, see the Win32 SDK.

hwndOwner
Identifies the window that owns the dialog box.

dlgprc
Points to the dialog box procedure. For more information about the dialog box procedure, see the description of the **DialogProc** callback function in the Win32 SDK.

dwInitParam
Specifies the value to pass to the dialog box in the *lParam* parameter of the WM_INITDIALOG message.

Return Values

If the function succeeds, the return value is the *nResult* parameter given in the call to the **EndDialog** function used to terminate the dialog box.

If the function fails, the return value is -1.

Remarks

For more information, see the description of the **DialogBoxIndirectParam** function in the Win32 SDK.

See Also

WlxInitialize

WlxDialogBoxParam

The **WlxDialogBoxParam** function duplicates the Win32 **DialogBoxParam** function, except that it allows Winlogon to time-out the dialog box.

```
int WlxDialogBoxParam(
  HANDLE hWlx,
  HANDLE hInst,
  LPWSTR lpszTemplate,
  HWND hwndOwner,
  DLGPROC dlgprc,
  LPARAM dwInitParam
);
```

Parameters

hWlx
(IN parameter) Winlogon handle provided to GINA in the **WlxInitialize** call.

hInst
Identifies an instance of the module whose executable file contains the dialog box template.

lpszTemplate
Identifies the dialog box template. This parameter is either the address of a null-terminated character string that specifies the name of the dialog box template, or an integer value that specifies the resource identifier of the dialog box template. If the parameter specifies a resource identifier, its high-order word must be zero and its low-order word must contain the identifier. You can use the **MAKEINTRESOURCE** macro to create this value.

hwndOwner
Identifies the window that owns the dialog box.

dlgprc
Points to the dialog box procedure. For more information about the dialog box procedure, see the description of the **DialogProc** callback function in the Win32 SDK.

dwInitParam
Specifies the value to pass to the dialog box in the *lParam* parameter of the WM_INITDIALOG message.

Return Values

If the function succeeds, the return value is the value of the *nResult* parameter given in the call to the **EndDialog** function used to terminate the dialog box.

If the function fails, the return value is -1.

Remarks

For more information, see the description of the **DialogBoxParam** function in the Win32 SDK.

See Also

WlxInitialize

WlxGetSourceDesktop

The **WlxGetSourceDesktop** function is used by the GINA DLL to determine the name and handle of the desktop that was active prior to Winlogon switching to the Winlogon desktop. GINA DLLs can use this to modify their behavior, depending on the originating desktop.

```
BOOL WlxGetSourceDesktop(
  HANDLE hWlx,
  PWLX_DESKTOP *ppDesktop
);
```

Parameters

hWlx
(IN parameter) Winlogon handle provided to GINA in the **WlxInitialize** call.

ppDesktop
(OUT parameter) Receives a pointer to a **WLX_DESKTOP** structure containing necessary information describing the desktop. This pointer can be freed with **LocalFree**.

Return Values

If the **WlxGetSourceDesktop** function succeeds, TRUE is returned. If the function fails, then FALSE is returned.

Remarks

None.

See Also

WlxInitialize

WlxMessageBox

The **WlxMessageBox** function duplicates the Win32 **MessageBox** function, except that it allows Winlogon to time-out the dialog box.

int WlxMessageBox(
  HANDLE *hWlx*,
  HWND *hwndOwner*,
  LPWST *lpszText*,
  LPWSTR *lpszTitle*,
  UINT *fuStyle*
);

Parameters

hWlx
(IN parameter) Winlogon handle provided to GINA in the **WlxInitialize** call.

hwndOwner
Identifies the owner window of the message box to be created. If this parameter is NULL, the message box has no owner window.

lpszText
Points to a null-terminated string containing the message to be displayed.

lpszTitle
Points to a null-terminated string used for the dialog box title. If this parameter is NULL, the default title Error is used.

fuStyle
Specifies the contents and behavior of the dialog box. This parameter can be a combination of the following values:

Value	Description
MB_ABORTRETRYIGNORE	The message box contains three push buttons: Abort, Retry, and Ignore.
MB_APPLMODAL	The user must respond to the message box before continuing work in the window identified by the *hWnd* parameter. However, the user can move to the windows of other applications and work in those windows. Depending on the hieracrchy of windows in the application, the user may be able to move to other windows within the application. All child windows of the parent of the message box are automatically disabled, but pop-up windows are not. MB_APPLMODAL is the default value if neither MB_SYSTEMMODAL nor MB_TASKMODAL is specified.
MB_DEFAULT_DESKTOP_ONLY	The desktop currently receiving input must be a default desktop;

	otherwise, the function fails. A default desktop is one that an application runs on after the user has logged on.
MB_DEFBUTTON1	The first button is the default button. Note that the first button is always the default unless MB_DEFBUTTON2 or MB_DEFBUTTON3 is specified.
MB_DEFBUTTON2	The second button is a default button.
MB_DEFBUTTON3	The third button is a default button.
MB_DEFBUTTON4	The fourth button is a default button.
MB_ICONASTERISK	An icon consisting of a lowercase letter *i* in a circle appears in the message box.
MB_ICONEXCLAMATION	An exclamation-point icon appears in the message box.
MB_ICONHAND	A stop-sign icon appears in the message box.
MB_ICONINFORMATION	An icon consisting of a lowercase letter *i* in a circle appears in the message box.
MB_ICONQUESTION	A question-mark icon appears in the message box.
MB_ICONSTOP	A stop-sign icon appears in the message box.
MB_OK	The message box contains one push button: OK.
MB_OKCANCEL	The message box contains two push buttons: OK and Cancel.
MB_RETRYCANCEL	The message box contains two push buttons: Retry and Cancel.
MB_SERVICE_NOTIFICATION	The caller is a service notifying the user of an event. The function brings up a message box on the current active desktop, even if there is no user logged on to the computer.
MB_SETFOREGROUND	The message box becomes the foreground window. Internally, Windows calls the **SetForegroundWindow** function for the message box.
MB_SYSTEMMODAL	All applications are suspended until the user responds to the message box. Unless the application specifies MB_ICONHAND, the message box does not become modal until

	after it is created. Consequently, the owner window and other windows continue to receive messages resulting from its activation. Use system-modal message boxes to notify the user of serious, potentially damaging errors that require immediate attention (for example, running out of memory).
MB_TASKMODAL	Same as MB_APPLMODAL except that all the top-level windows belonging to the current task are disabled if the *hWnd* parameter is NULL. Use this flag when the calling application or library does not have a window handle available, but still needs to prevent input to other windows in the current application without suspending other applications.
MB_YESNO	The message box contains two push buttons: Yes and No.
MB_YESNOCANCEL	The message box contains three push buttons: Yes, No, and Cancel.

Return Values

The return value is zero if there is not enough memory to create the message box.

If the function succeeds, the return value is one of the following menu item values returned by the dialog box:

Value	Description
IDABORT	Abort button was selected.
IDCANCEL	Cancel button was selected.
IDIGNORE	Ignore button was selected.
IDNO	No button was selected.
IDOK	OK button was selected.
IDRETRY	Retry button was selected.
IDYES	Yes button was selected.

If a message box has a Cancel button, the function returns the IDCANCEL value if either the ESC key is pressed or the Cancel button is selected. If the message box has no Cancel button, pressing ESC has no effect.

Remarks

For more information, see the description of the **MessageBox** function in the Win32 SDK.

See Also

WlxInitialize

WlxSasNotify

The **WlxSasNotify** function lets a GINA DLL notify Winlogon of a secure attention sequence event.

```
VOID WlxSasNotify(
  HANDLE hWlx,
  DWORD dwSasType
);
```

Parameters

hWlx

(IN parameter) Winlogon handle provided to GINA in the **WlxInitialize** call.

dwSasType

(IN parameter.) Indicates the type of SAS that occurred. This value will be delivered to the GINA SAS service routine Winlogon calls (**WlxLoggedOutSAS**, **WlxLoggedOnSAS**, or **WlxWkstaLockedSAS**).

Values from zero to WLX_SAS_TYPE_MAX_MSFT_VALUE are reserved to define Microsoft standard SAS types. GINA developers can use values greater than WLX_SAS_TYPE_MAX_MSFT_VALUE to define additional SAS types. The following values are predefined:

Value	Description
WLX_SAS_TYPE_CTRL_ALT_DEL	Indicates that the user has typed the CTRL+ALT+DEL SAS.

Return Values

None.

See Also

WlxInitialize, **WlxLoggedOnSAS**, **WlxLoggedOutSAS**, **WlxWkstaLockedSAS**

WlxSetContextPointer

The **WlxSetContextPointer** function lets a GINA DLL specify the context pointer passed by Winlogon as the first parameter to all the GINA functions. By using **WlxSetContextPointer**, GINA can specify a new context pointer to update the one returned by the **WlxInitialize** function.

If a GINA DLL wants to call **WlxSasNotify** during the processing of the **WlxInitialize** function, it should first call **WlxSetContextPointer** to associate any context with GINA.

```
VOID WlxSetContextPointer(
  HANDLE hWlx,
  PVOID pWlxContext
);
```

Parameters

hWlx
(IN parameter) Winlogon handle provided to GINA in the **WlxInitialize** call.

pWlxContext
(IN parameter) Context pointer that GINA associates with.

Return Values

None.

See Also

WlxInitialize, **WlxSasNotify**

WlxSetReturnDesktop

The **WlxSetReturnDesktop** function is used by the GINA DLL to specify the desktop to which Winlogon will switch once the current SAS event is completed. This call is valid only during the **WlxLoggedOnSAS** or **WlxWkstaLockedSAS** call. Attempts to call this function at other times will return an error.

```
int WlxSetReturnDesktop(
  HANDLE hWlx,
  PWLX_DESKTOP pDesktop
);
```

Parameters

hWlx
(IN parameter) Winlogon handle provided to GINA in the **WlxInitialize** call.

pDesktop
(IN parameter) Pointer to desktop information about the return desktop.

Return Values

None.

Remarks

If a handle to the desktop is provided, Winlogon will duplicate the handle. If no handle is provided, Winlogon will attempt to open the desktop named in the *pDesktop* parameter. If the provided desktop is not valid, or is Winlogon or ScreenSaver, the call will fail.

See Also

WlxInitialize

WlxSetTimeout

The **WlxSetTimeout** function lets a GINA DLL change the time-out associated with a dialog box. The default time-out is two minutes.

BOOL WlxSetTimeout(
  HANDLE *hWlx***,**
  DWORD *dwTimeout*
);

Parameters

hWlx
  Handle assigned by Winlogon during initialization.

dwTimeout
  Requested time-out, in seconds.

Return Values

If the new time-out was accepted, the return value is TRUE.

If the new time-out was not accepted, the return value is FALSE.

WlxSwitchDesktopToUser

The **WlxSwitchDesktopToUser** function lets the GINA DLL switch between visual desktops. This function is valid only for the currently operating thread.

```
int WlxSwitchDesktopToUser(
  HANDLE hWlx
);
```

Parameters

hWlx
(IN parameter) Winlogon handle provided to GINA in the **WlxInitialize** call.

See Also

WlxInitialize

WlxSwitchDesktopToWinlogon

The **WlxSwitchDesktopToWinlogon** function lets the GINA DLL switch between visual desktops. This function is valid only for the currently operating thread.

```
int WlxSwitchDesktopToWinlogon(
  HANDLE   hWlx
);
```

Parameters

hWlx
(IN parameter) Winlogon handle provided to GINA in the **WlxInitialize** call.

See Also

WlxInitialize

WlxUseCtrlAltDel

The **WlxUseCtrlAltDel** function lets GINA tell Winlogon to use the standard CTRL+ALT+DEL key combination as a secure attention sequence (SAS). If a GINA DLL uses this function, it is not required to use the **WlxSasNotify** function. However, if GINA has other SASes in addition to CTRL+ALT+DEL, it must use **WlxSasNotify** to deliver those additional SASes.

VOID WlxUseCtrlAltDel(
  HANDLE *hWlx*
);

Parameters

hWlx
(IN parameter) Winlogon handle provided to GINA in the **WlxInitialize** call.

Return Values

None.

See Also

WlxInitialize, **WlxSasNotify**

Winlogon Structures

Use the following structures to implement GINA features:

WLX_DESKTOP
WLX_DISPATCH_VERSION_1_0
WLX_DISPATCH_VERSION_1_1
WLX_LOGON_OPT_xxx
WLX_MPR_NOTIFY_INFO
WLX_PROFILE_V1_0
WLX_PROFILE_V2_0
WLX_SAS_TYPE_XXX

WLX_DESKTOP

The **WLX_DESKTOP** structure and flags are used to pass desktop information between the GINA DLL and Winlogon.

```
typedef struct _win32_WLX_DESKTOP {
    DWORD    Size;
    DWORD    Flags;
    HDESK    hDesktop;
    PWSTR    pszDesktopName;
} WLX_DESKTOP;
```

Members

Size
Size of the structure. This must be set to *sizeof*(WLX_DESKTOP).

Flags
Flags indicating the valid fields. These are:

Value	Description
WLX_DESKTOP_NAME	Indicates that the *pszDesktopName* field is valid.
WLX_DESKTOP_HANDLE	Indicates that the *hDesktop* field is valid.

hDesktop
Handle returned from **CreateDesktop** or **OpenDesktop**.

pszDesktopName
Name of the desktop. See the **WlxGetSourceDesktop** or **WlxCreateUserDesktop** function for details on the allocation of this string.

WLX_DISPATCH_VERSION_1_0

The **WLX_DISPATCH_VERSION_1_0** structure defines the format of the version 1.0 Winlogon function dispatch table passed to the GINA DLL in the **WlxInitialize** call.

The routine type definitions used in this structure are defined in WINWLX.H to match the routine descriptions presented in this documentation.

```
typedef struct _win32_DISPATCH_VERSION_1_0 {
    PWLX_USE_CTRL_ALT_DEL            WlxUseCtrlAltDel;
    PWLX_SET_CONTEXT_POINTER         WlxSetContextPointer;
    PWLX_SAS_NOTIFY                  WlxSasNotify;
    PWLX_SET_TIMEOUT                 WlxSetTimeout;
    PWLX_ASSIGN_SHELL_PROTECTION     WlxAssignShellProtection
    PWLX_MESSAGE_BOX                 WlxMessageBox;
    PWLX_DIALOG_BOX                  WlxDialogBox
    PWLX_DIALOG_BOX_PARAM            WlxDialogBoxParam;
    PWLX_DIALOG_BOX_INDIRECT         WlxDialogBoxIndirect;
    PWLX_DIALOG_BOX_INDIRECT_PARAM   WlxDialogBoxIndirectParam;
    PWLX_SWITCH_DESKTOP_TO_USER      WlxSwitchDesktopToUser;
    PWLX_SWITCH_DESKTOP_TO_WINLOGON  WlxSwitchDesktopToWinlogon;
    PWLX_CHANGE_PASSWORD_NOTIFY      WlxChangePasswordNotify;
} WLX_DISPATCH_VERSION_1_0;
```

See Also

WlxInitialize

WLX_DISPATCH_VERSION_1_1

The **WLX_DISPATCH_VERSION_1_1** structure defines the format of the version 1.1 Winlogon function dispatch table passed to the GINA DLL in the **WlxInitialize** call.

The routine type definitions used in this structure are defined in WINWLX.H to match the routine descriptions presented in this documentation.

```
typedef struct _win32_DISPATCH_VERSION_1_1 {
    PWLX_USE_CTRL_ALT_DEL            WlxUseCtrlAltDel;
    PWLX_SET_CONTEXT_POINTER         WlxSetContextPointer;
    PWLX_SAS_NOTIFY                  WlxSasNotify;
    PWLX_SET_TIMEOUT                 WlxSetTimeout;
    PWLX_ASSIGN_SHELL_PROTECTION     WlxAssignShellProtection
    PWLX_MESSAGE_BOX                 WlxMessageBox;
    PWLX_DIALOG_BOX                  WlxDialogBox
    PWLX_DIALOG_BOX_PARAM            WlxDialogBoxParam;
    PWLX_DIALOG_BOX_INDIRECT         WlxDialogBoxIndirect;
    PWLX_DIALOG_BOX_INDIRECT_PARAM   WlxDialogBoxIndirectParam;
    PWLX_SWITCH_DESKTOP_TO_USER      WlxSwitchDesktopToUser;
    PWLX_SWITCH_DESKTOP_TO_WINLOGON  WlxSwitchDesktopToWinlogon;
    PWLX_CHANGE_PASSWORD_NOTIFY      WlxChangePasswordNotify;
    PWLX_GET_SOURCE_DESKTOP          WlxGetSourceDesktop;
    PWLX_SET_RETURN_DESKTOP          WlxSetReturnDesktop;
    PWLX_CREATE_USER_DESKTOP         WlxCreateUserDesktop;
} WLX_DISPATCH_VERSION_1_1;
```

See Also

WlxInitialize

WLX_LOGON_OPT_xxx

Upon successful logon, the GINA DLL can specify the following option to Winlogon (through the *dwOptions* parameter of the **WlxLoggedOutSAS** function).

WLX_LOGON_OPT_NO_PROFILE
: When set, this option specifies that Winlogon must *not* load a profile for the logged-on user. Either the GINA DLL will take care of this activity, or the user does not need a profile.

WLX_MPR_NOTIFY_INFO

The **WLX_MPR_NOTIFY_INFO** structure is returned from a GINA DLL following successful authentication. Winlogon uses this information to provide network providers with identification and authentication information already collected. Winlogon is responsible for freeing both the main structure and all string and other buffers pointed to from within the structure.

```
typedef struct _win32_MPR_NOTIFY_INFO {
    PWSTR    pszUserName;
    PWSTR    pszDomain;
    PWSTR    pszPassword;
    PWSTR    pszOldPassword;
} WLX_MPR_NOTIFY_INFO;
```

Members

pszUserName
The name of the account logged onto (for example, "joeu"). The string pointed to by this member must be separately allocated and will be separately deallocated by Winlogon.

pszDomain
The name of the domain used for logon. The string pointed to by this member must be separately allocated and will be separately deallocated by Winlogon.

pszPassword
Clear-text password of the user account. If the **OldPassword** member is non-null, the **pszPassword** member contains the new password in a password change operation. The string pointed to by this field must be separately allocated and will be separately deallocated by Winlogon.

pszOldPassword
Clear-text old password of the user account whose password has just been changed. The **Password** member contains the new password. The string pointed to by this member must be separately allocated and will be separately deallocated by Winlogon.

PROFILE TYPES

GINA DLLs are expected to return account information to Winlogon following a succcessful logon. This information allows Winlogon to support profile loading and supplemental network providers.

In order for GINA DLLs to return different sets of profile information over time, the first DWORD of each profile structure must contain a type identifier. The following constants are defined profile type identifiers:

Profile Types	Description
WLX_PROFILE_TYPE_V1_0	The standard profile for V1_0.
WLX_PROFILE_TYPE_V2_0	The advanced profile for Windows NT 4.0 and later.

WLX_PROFILE_V1_0

The **WLX_PROFILE_V1_0** structure is returned from a GINA DLL following authentication. Winlogon uses the profile path to load the newly logged-on user's profile. Winlogon is responsible for freeing both the main structure and all string and other buffers pointed to from within the structure.

```
typedef struct _win32_PROFILE_V1_0 {
    DWORD    dwType;
    PWSTR    pszProfile;
} WLX_PROFILE_V1_0;
```

Members

dwType
Must be set to WLX_PROFILE_TYPE_V1_0.

pszProfile
Profile path (for example, "%SystemRoot%\system32\config\JoeU001"). The string pointed to by this member must be separately allocated and will be separately deallocated by Winlogon.

WLX_PROFILE_V2_0

The **WLX_PROFILE_V2_0** structure is returned from a GINA DLL following authentication. It contains information in addition to the information provided in V1_0 for setting up the initial environment.

```
typedef struct _win32_PROFILE_V2_0 {
    DWORD    dwType;
    PWSTR    pszProfile;
    PWSTR    pszPolicy;
    PWSTR    pszNetworkDefaultUserProfile;
    PWSTR    pszServerName;
    PWSTR    pszEnvironment;
} WLX_PROFILE_V2_0;
```

Members

dwType
Must be set to WLX_PROFILE_TYPE_V2_0.

pszProfile
Profile path (for example, "%SystemRoot%\system32\config\JoeU001") or a network path such as "\\server\share\profiles\floating\JoeU.usr." This string must be separately allocated and will be freed by Winlogon.

pszPolicy
Path to the policy file that will be applied to the user logging on. This string must be separately allocated and will be freed by Winlogon. It can also be NULL.

pszNetworkDefaultUserProfile
If a new profile will be created, this is the path to the default profile to use. This string must be separately allocated and will be freed by Winlogon. It can also be NULL.

pszServerName
Name of the server that validated the logon. This server will be used to enumerate the global groups of which the user is a member. This string must be separately allocated and will be freed by Winlogon. It can also be NULL.

pszEnvironment
Default environment variables to include during the construction of the user's environment. The environment is a series of null-terminated strings of the form:

Variable=Value or
variable=%other variable% or
variable=other variable% (additional text is optional)

For example:
logonServer=\\pdc
homePath=%logonServer%\shar

WLX_SAS_TYPE_XXX

Following are the secure attention sequence (SAS) types.

All values from zero to 127 are reserved for Microsoft definition. Values above 127 are reserved for cutomer definition. These values are passed to routines that have a *dwSasType* parameter.

Value	Description
WLX_SAS_TYPE_CTRL_ALT_DEL	Indicates that the standard CTRL+ALT+DEL SAS has been entered.
WLX_SAS_TYPE_SCRNSVR_TIMEOUT	Indicates that keyboard/mouse inactivity has led to a screen saver activation.
WLX_SAS_TYPE_TIMEOUT	Indicates that an input time-out has occurred.