Contents

Window Stations and Desktops

8/17/2022 • 2 minutes to read • Edit Online

Windows provides three main categories of objects: user interface, graphics device interface (GDI), and kernel. Kernel objects are securable, while user objects and GDI objects are not. Therefore, to provide additional security, user interface objects are managed using window stations and desktops, which themselves are securable objects.

For more information, see the following topics:

- About Window Stations and Desktops
- Window Station and Desktop Reference

About Window Stations and Desktops

8/17/2022 • 2 minutes to read • Edit Online

A *window station* is a securable object that is associated with a process, and contains a clipboard, an atom table, and one or more desktop objects.

A *desktop* is a securable object contained within a window station. A desktop has a logical display surface and contains user interface objects such as windows, menus, and hooks.

For more information, see the following topics:

- Window Stations
- Desktops
- Window Station and Desktop Creation
- Process Connection to a Window Station
- Thread Connection to a Desktop
- Window Station Security and Access Rights
- Desktop Security and Access Rights

Window Stations

8/17/2022 • 2 minutes to read • Edit Online

A *window station* contains a clipboard, an atom table, and one or more desktop objects. Each window station object is a securable object. When a window station is created, it is associated with the calling process and assigned to the current session.

The *interactive window station* is the only window station that can display a user interface or receive user input. It is assigned to the logon session of the interactive user, and contains the keyboard, mouse, and display device. It is always named "WinSta0". All other window stations are noninteractive, which means they cannot display a user interface or receive user input.

When a user logs on to a computer using Remote Desktop Services, a session is started for the user. Each session is associated with its own interactive window station named "WinSta0". For more information, see Remote Desktop Sessions.

For more information on window stations, see the following topics:

- Window Station and Desktop Creation
- Process Connection to a Window Station
- Window Station Security and Access Rights

Desktops

8/17/2022 • 2 minutes to read • Edit Online

A *desktop* has a logical display surface and contains user interface objects such as windows, menus, and hooks; it can be used to create and manage windows. Each desktop object is a securable object. When a desktop is created, it is associated with the current window station of the calling process and assigned to the calling thread.

Window messages can be sent only between processes that are on the same desktop. In addition, the hook procedure of a process running on a particular desktop can only receive messages intended for windows created in the same desktop.

The desktops associated with the interactive window station, Winsta0, can be made to display a user interface and receive user input, but only one of these desktops at a time is active. This active desktop, also known as the *input desktop*, is the one that is currently visible to the user and that receives user input. Applications can use the **OpenInputDesktop** function to get a handle to the input desktop. Applications that have the required access can use the **SwitchDesktop** function to specify a different input desktop.

By default, there are three desktops in the interactive window station: Default, ScreenSaver, and Winlogon.

The Default desktop is created when Winlogon starts the initial process as the logged-on user. At that point, the Default desktop becomes active, and it is used to interact with the user.

Whenever a secure screen saver activates, the system automatically switches to the ScreenSaver desktop, which protects the processes on the default desktop from unauthorized users. Unsecured screen savers run on Winsta0\Default.

The Winlogon desktop is active while a user logs on. The system switches to the default desktop when the shell indicates that it is ready to display something, or after thirty seconds, whichever comes first. During the user's session, the system switches to the Winlogon desktop when the user presses the CTRL+ALT+DEL key sequence, or when the User Account Control (UAC) dialog box is open.

Windows Server 2003 and Windows XP/2000: The UAC dialog box is not supported.

The Winlogon desktop's security descriptor allows access to a very restricted set of accounts, including the LocalSystem account. Applications generally do not carry any of these accounts' SIDs in their tokens and therefore cannot access the Winlogon desktop or switch to a different desktop while the Winlogon desktop is active.

For more information, see the following topics:

- Window Station and Desktop Creation
- Thread Connection to a Desktop
- Desktop Security and Access Rights

Window Station and Desktop Creation

8/17/2022 • 2 minutes to read • Edit Online

The system automatically creates the interactive window station. When an interactive user logs on, the system associates the interactive window station with the user logon session. The system also creates the default input desktop for the interactive window station (Winsta0\default). Processes started by the logged-on user are associated with the Winsta0\default desktop.

A process can use the **CreateWindowStation** function to create a new window station, and the **CreateDesktop** or **CreateDesktopEx** function to create a new desktop. The number of desktops that can be created is limited by the size of the system desktop heap. For more information, see **CreateDesktop**.

When a noninteractive process such as a service application attempts to connect to a window station and no window station exists for the process logon session, the system attempts to create a window station and desktop for the session. The name of the created window station is based on the logon session identifier, and the desktop is named default, as described here:

- If a service is running in the security context of the LocalSystem account but does not include the SERVICE_INTERACTIVE_PROCESS attribute, it uses the following window station and desktop: Service-0x0-3e7$\default. This window station is not interactive, so the service cannot display a user interface. In addition, processes created by the service cannot display a user interface.
- If the service is running in the security context of a user account, the name of the window station is based on the user SID Service-0x*Z1*-*Z2*$, where *Z1* is the high part of the logon SID and *Z2* is the low part of the logon SID. Because a SID is unique to the logon session, two services running in the same security context receive unique window stations. These window stations are not interactive.

The discretionary access control list (DACL) for the window station and desktop includes the following access rights for the service's user account:

Window Station:

WINSTA_ACCESSCLIPBOARD WINSTA_ACCESSGLOBALATOMS WINSTA_CREATEDESKTOP WINSTA_EXITWINDOWS WINSTA_READATTRIBUTES STANDARD_RIGHTS_REQUIRED

Desktop:

DESKTOP_CREATEMENU DESKTOP_CREATEWINDOW DESKTOP_ENUMERATE DESKTOP_HOOKCONTROL DESKTOP_READOBJECTS DESKTOP_WRITEOBJECTS STANDARD_RIGHTS_REQUIRED

Process Connection to a Window Station

8/17/2022 • 2 minutes to read • Edit Online

A process automatically establishes a connection to a window station and desktop when it first calls a USER32 or GDI32 function (other than the window station and desktop functions). The system determines the window station to which a process connects according to the following rules:

1. If the process has called the **SetProcessWindowStation** function, it connects to the window station specified in that call.
2. If the process did not call **SetProcessWindowStation**, it connects to the window station inherited from the parent process.
3. If the process did not call **SetProcessWindowStation** and did not inherit a window station, the system attempts to open for MAXIMUM_ALLOWED access and connect to a window station as follows:
   - If a window station name was specified in the **lpDesktop** member of the **STARTUPINFO** structure that was used when the process was created, the process connects to the specified window station.
   - Otherwise, if the process is running in the logon session of the interactive user, the process connects to the interactive window station.
   - If the process is running in a noninteractive logon session, the window station name is formed based on the logon session identifier and an attempt is made to open that window station. If the open operation fails because this window station does not exist, the system tries to create the window station and a default desktop.

The window station assigned during this connection process cannot be closed by calling the **CloseWindowStation** function.

When a process is connecting to a window station, the system searches the process's handle table for inherited handles. The system uses the first window station handle it finds. If you want a child process to connect to a particular inherited window station, you must ensure that only the desired handle is marked inheritable. If a child process inherits multiple window station handles, the results of the window station connection are undefined.

Handles to a window station that the system opens while connecting a process to a window station are not inheritable.

Related topics

Thread Connection to a Desktop

Thread Connection to a Desktop

8/17/2022 • 2 minutes to read • Edit Online

After a process connects to a window station, the system assigns a desktop to the thread making the connection. The system determines the desktop to assign to the thread according to the following rules:

1. If the thread has called the **SetThreadDesktop** function, it connects to the specified desktop.
2. If the thread did not call **SetThreadDesktop**, it connects to the desktop inherited from the parent process.
3. If the thread did not call **SetThreadDesktop** and did not inherit a desktop, the system attempts to open for MAXIMUM_ALLOWED access and connect to a desktop as follows:
   - If a desktop name was specified in the **lpDesktop** member of the **STARTUPINFO** structure that was used when the process was created, the thread connects to the specified desktop.
   - Otherwise, the thread connects to the default desktop of the window station to which the process connected.

The desktop assigned during this connection process cannot be closed by calling the **CloseDesktop** function.

When a process is connecting to a desktop, the system searches the process's handle table for inherited handles. The system uses the first desktop handle it finds. If you want a child process to connect to a particular inherited desktop, you must ensure that the only the desired handle is marked inheritable. If a child process inherits multiple desktop handles, the results of the desktop connection are undefined.

Handles to a desktop that the system opens while connecting a process to a desktop are not inheritable.

Related topics

Process Connection to a Window Station

Window Station Security and Access Rights

8/17/2022 • 2 minutes to read • Edit Online

Security enables you to control access to window station objects. For more information about security, see Access-Control Model.

You can specify a security descriptor for a window station object when you call the **CreateWindowStation** function. If you specify NULL, the window station gets a default security descriptor. The ACLs in the default security descriptor for a window station come from the primary or impersonation token of the creator.

To get or set the security descriptor of a window station object, call the **GetSecurityInfo** and **SetSecurityInfo** functions.

When you call the **OpenWindowStation** function, the system checks the requested access rights against the object's security descriptor.

The valid access rights for window station objects include the standard access rights and some object-specific access rights. The following table lists the standard access rights used by all objects.

VALUE	MEANING
DELETE (0x00010000L)	Required to delete the object.
READ_CONTROL (0x00020000L)	Required to read information in the security descriptor for the object, not including the information in the SACL. To read or write the SACL, you must request the ACCESS_SYSTEM_SECURITY access right. For more information, see SACL Access Right.
SYNCHRONIZE (0x00100000L)	Not supported for window station objects.
WRITE_DAC (0x00040000L)	Required to modify the DACL in the security descriptor for the object.
WRITE_OWNER (0x00080000L)	Required to change the owner in the security descriptor for the object.

The following table lists the object-specific access rights.

ACCESS RIGHT	DESCRIPTION
WINSTA_ALL_ACCESS (0x37F)	All possible access rights for the window station.
WINSTA_ACCESSCLIPBOARD (0x0004L)	Required to use the clipboard.
WINSTA_ACCESSGLOBALATOMS (0x0020L)	Required to manipulate global atoms.
WINSTA_CREATEDESKTOP (0x0008L)	Required to create new desktop objects on the window station.

ACCESS RIGHT	DESCRIPTION
WINSTA_ENUMDESKTOPS (0x0001L)	Required to enumerate existing desktop objects.
WINSTA_ENUMERATE (0x0100L)	Required for the window station to be enumerated.
WINSTA_EXITWINDOWS (0x0040L)	Required to successfully call the **ExitWindows** or **ExitWindowsEx** function. Window stations can be shared by users and this access type can prevent other users of a window station from logging off the window station owner.
WINSTA_READATTRIBUTES (0x0002L)	Required to read the attributes of a window station object. This attribute includes color settings and other global window station properties.
WINSTA_READSCREEN (0x0200L)	Required to access screen contents.
WINSTA_WRITEATTRIBUTES (0x0010L)	Required to modify the attributes of a window station object. The attributes include color settings and other global window station properties.

The following are the generic access rights for the interactive window station object, which is the window station assigned to the logon session of the interactive user.

ACCESS RIGHT	DESCRIPTION
GENERIC_READ	STANDARD_RIGHTS_READ WINSTA_ENUMDESKTOPS WINSTA_ENUMERATE WINSTA_READATTRIBUTES WINSTA_READSCREEN
GENERIC_WRITE	STANDARD_RIGHTS_WRITE WINSTA_ACCESSCLIPBOARD WINSTA_CREATEDESKTOP WINSTA_WRITEATTRIBUTES
GENERIC_EXECUTE	STANDARD_RIGHTS_EXECUTE WINSTA_ACCESSGLOBALATOMS WINSTA_EXITWINDOWS
GENERIC_ALL	STANDARD_RIGHTS_REQUIRED WINSTA_ACCESSCLIPBOARD WINSTA_ACCESSGLOBALATOMS WINSTA_CREATEDESKTOP WINSTA_ENUMDESKTOPS WINSTA_ENUMERATE WINSTA_EXITWINDOWS WINSTA_READATTRIBUTES WINSTA_READSCREEN WINSTA_WRITEATTRIBUTES

The following are the generic access rights for a noninteractive window station object. The system assigns noninteractive window stations to all logon sessions other than that of the interactive user.

ACCESS RIGHT	DESCRIPTION
GENERIC_READ	STANDARD_RIGHTS_READ WINSTA_ENUMDESKTOPS WINSTA_ENUMERATE WINSTA_READATTRIBUTES
GENERIC_WRITE	STANDARD_RIGHTS_WRITE WINSTA_ACCESSCLIPBOARD WINSTA_CREATEDESKTOP
GENERIC_EXECUTE	STANDARD_RIGHTS_EXECUTE WINSTA_ACCESSGLOBALATOMS WINSTA_EXITWINDOWS
GENERIC_ALL	STANDARD_RIGHTS_REQUIRED WINSTA_ACCESSCLIPBOARD WINSTA_ACCESSGLOBALATOMS WINSTA_CREATEDESKTOP WINSTA_ENUMDESKTOPS WINSTA_ENUMERATE WINSTA_EXITWINDOWS WINSTA_READATTRIBUTES

You can request the ACCESS_SYSTEM_SECURITY access right to a window station object if you want to read or write the object's SACL. For more information, see Access-Control Lists (ACLs) and SACL Access Right.

Desktop Security and Access Rights

8/17/2022 • 2 minutes to read • Edit Online

Security enables you to control access to desktop objects. For more information about security, see Access-Control Model.

You can specify a security descriptor for a desktop object when you call the **CreateDesktop** or **CreateDesktopEx** function. If you specify NULL, the desktop gets a default security descriptor. The ACLs in the default security descriptor for a desktop come from its parent window station.

To get or set the security descriptor of a window station object, call the **GetSecurityInfo** and **SetSecurityInfo** functions.

When you call the **OpenDesktop** or **OpenInputDesktop** function, the system checks the requested access rights against the object's security descriptor.

The valid access rights for desktop objects include the standard access rights and some object-specific access rights. The following table lists the standard access rights used by all objects.

VALUE	MEANING
DELETE (0x00010000L)	Required to delete the object.
READ_CONTROL (0x00020000L)	Required to read information in the security descriptor for the object, not including the information in the SACL. To read or write the SACL, you must request the ACCESS_SYSTEM_SECURITY access right. For more information, see SACL Access Right.
SYNCHRONIZE (0x00100000L)	Not supported for desktop objects.
WRITE_DAC (0x00040000L)	Required to modify the DACL in the security descriptor for the object.
WRITE_OWNER (0x00080000L)	Required to change the owner in the security descriptor for the object.

The following table lists the object-specific access rights.

ACCESS RIGHT	DESCRIPTION
DESKTOP_CREATEMENU (0x0004L)	Required to create a menu on the desktop.
DESKTOP_CREATEWINDOW (0x0002L)	Required to create a window on the desktop.
DESKTOP_ENUMERATE (0x0040L)	Required for the desktop to be enumerated.
DESKTOP_HOOKCONTROL (0x0008L)	Required to establish any of the window hooks.
DESKTOP_JOURNALPLAYBACK (0x0020L)	Required to perform journal playback on a desktop.

ACCESS RIGHT	DESCRIPTION
DESKTOP_JOURNALRECORD (0x0010L)	Required to perform journal recording on a desktop.
DESKTOP_READOBJECTS (0x0001L)	Required to read objects on the desktop.
DESKTOP_SWITCHDESKTOP (0x0100L)	Required to activate the desktop using the **SwitchDesktop** function.
DESKTOP_WRITEOBJECTS (0x0080L)	Required to write objects on the desktop.

The following are the generic access rights for a desktop object contained in the interactive window station of the user's logon session.

ACCESS RIGHT	DESCRIPTION
GENERIC_READ	DESKTOP_ENUMERATE DESKTOP_READOBJECTS STANDARD_RIGHTS_READ
GENERIC_WRITE	DESKTOP_CREATEMENU DESKTOP_CREATEWINDOW DESKTOP_HOOKCONTROL DESKTOP_JOURNALPLAYBACK DESKTOP_JOURNALRECORD DESKTOP_WRITEOBJECTS STANDARD_RIGHTS_WRITE
GENERIC_EXECUTE	DESKTOP_SWITCHDESKTOP STANDARD_RIGHTS_EXECUTE
GENERIC_ALL	DESKTOP_CREATEMENU DESKTOP_CREATEWINDOW DESKTOP_ENUMERATE DESKTOP_HOOKCONTROL DESKTOP_JOURNALPLAYBACK DESKTOP_JOURNALRECORD DESKTOP_READOBJECTS DESKTOP_SWITCHDESKTOP DESKTOP_WRITEOBJECTS STANDARD_RIGHTS_REQUIRED

You can request the ACCESS_SYSTEM_SECURITY access right to a desktop object if you want to read or write the object's SACL. For more information, see Access-Control Lists (ACLs) and SACL Access Right.

Window Station and Desktop Reference

8/17/2022 • 2 minutes to read • Edit Online

The following elements are used with window stations and desktops:

- Window Station and Desktop Functions
- Window Station and Desktop Structures

Window Station and Desktop Functions

8/17/2022 • 2 minutes to read • Edit Online

Applications can use the following functions with window station objects.

FUNCTION	DESCRIPTION
CloseWindowStation	Closes an open window station handle.
CreateWindowStation	Creates a window station object, associates it with the current process, and assigns it to the current session.
EnumWindowStations	Enumerates all window stations in the current session.
GetProcessWindowStation	Retrieves a handle to the current window station for the calling process.
GetUserObjectInformation	Retrieves information about the specified window station or desktop object.
GetUserObjectSecurity	Retrieves security information for the specified window station or desktop object.
OpenWindowStation	Opens the specified window station.
SetProcessWindowStation	Assigns the specified window station to the calling process.
SetUserObjectInformation	Sets information about the specified window station or desktop object.
SetUserObjectSecurity	Sets security information for the specified window station or desktop object.

Applications can use the following functions with desktop objects.

FUNCTION	DESCRIPTION
CloseDesktop	Closes an open handle to a desktop object.
CreateDesktop	Creates a new desktop, associates it with the current window station of the calling process, and assigns it to the calling thread.
CreateDesktopEx	Creates a new desktop, associates it with the current window station of the calling process, and assigns it to the calling thread.
EnumDesktops	Enumerates all desktops associated with the current window station of the calling process.

FUNCTION	DESCRIPTION
EnumDesktopWindows	Enumerates all top-level windows associated with the specified desktop.
GetThreadDesktop	Retrieves a handle to the desktop assigned to the specified thread.
GetUserObjectInformation	Gets information about a window station or desktop object.
GetUserObjectSecurity	Gets security information for a window station or desktop object.
OpenDesktop	Opens the specified desktop object.
OpenInputDesktop	Opens the desktop that receives user input.
SetThreadDesktop	Assigns the specified desktop to the calling thread.
SetUserObjectInformation	Sets information about a window station or desktop object.
SetUserObjectSecurity	Sets security information for a window station or desktop object.
SwitchDesktop	Makes a desktop visible and activates it. This enables the desktop to receive input from the user.

Window Station and Desktop Structures

8/17/2022 • 2 minutes to read • Edit Online

The following structure is used with window stations and desktops:

- USEROBJECTFLAGS