



 Filter by title

- CredUIConfirmCredentialsA function
- CredUIConfirmCredentialsW function
- CredUIParseUserNameA function
- CredUIParseUserNameW function
- CredUIPromptForCredentialsA function**
- CredUIPromptForWindowsCredentialsA function
- CredUIPromptForWindowsCredentialsW function
- CredUIReadSSOCredW function
- CredUIStoreSSOCredW function
- CredUnmarshalCredentialA function
- CredUnmarshalCredentialW function
- CredUnPackAuthenticationBufferA function
- CredUnPackAuthenticationBufferW function
- CredUnprotectA function
- CredUnprotectW function
- CredWriteA function
- CredWriteDomainCredentialsA function
- CredWriteDomainCredentialsW function
- CredWriteW function
- USERNAME_TARGET_CREDENTIAL_INFO structure
- > Wincrypt.h
- > Winnetwk.h
- > Winnt.h
- > Winreg.h
- > Winsafer.h
- > Winscard.h
- > Winsvc.h
- > Winternl.h
- > Wintrust.h
- > Winuser.h
- > Winwx.h
- > Xenroll.h

 **Download PDF**

CredUIPromptForCredentialsA function (wincred.h)

Article • 02/09/2023

 [Feedback](#)

In this article

- [Syntax](#)
- [Parameters](#)
- [Return value](#)
- [Remarks](#)
- [Show 2 more](#)

The **CredUIPromptForCredentials** function creates and displays a configurable dialog box that accepts credentials information from a user.

Applications that target Windows Vista or Windows Server 2008 should call [CredUIPromptForWindowsCredentials](#) instead of this function, for the following reasons:

- [CredUIPromptForWindowsCredentials](#) is consistent with the current Windows user interface.
- [CredUIPromptForWindowsCredentials](#) is more extensible, allowing integration of additional authentication mechanisms such as biometrics and smart cards.
- [CredUIPromptForWindowsCredentials](#) is compliant with the Common Criteria specification.

Syntax

C++ Copy

```
CREDUIAPI DWORD CredUIPromptForCredentialsA(  
    [in, optional] PCREDUI_INFOA pUiInfo,  
    [in]           PCSTR          pszTargetName,  
    [in]           PCtxtHandle    pContext,  
    [in, optional] DWORD          dwAuthError,  
    [in, out]      PSTR           pszUserName,  
    [in]           ULONG          ulUserNameBufferSize,  
    [in, out]      PSTR           pszPassword,  
    [in]           ULONG          ulPasswordBufferSize,  
    [in, out]      BOOL           *save,  
    [in]           DWORD          dwFlags  
);
```

Parameters

[in, optional] pUiInfo

A pointer to a [CREDUI_INFO](#) structure that contains information for customizing the appearance of the dialog box.

[in] pszTargetName

A pointer to a null-terminated string that contains the name of the target for the credentials, typically a server name. For Distributed File System (DFS) connections, this string is of the form *ServerName\ShareName*. This parameter is used to identify target information when storing and retrieving credentials.

[in] pContext

This parameter is reserved for future use. It must be **NULL**.

[in, optional] dwAuthError

Specifies why the credential dialog box is needed. A caller can pass this Windows error parameter, returned by another authentication call, to allow the dialog box to accommodate certain errors. For example, if the password expired status code is passed, the dialog box could prompt the user to change the password on the account.

[in, out] pszUserName

A pointer to a null-terminated string that contains the user name for the credentials. If a nonzero-length string is passed, the *UserName* option of the dialog box is prefilled with the string. In the case of credentials other than *UserName/Password*, a marshaled format of the credential can be passed in. This string is created by calling [CredMarshalCredential](#).

This function copies the user-supplied name to this buffer, copying a maximum of *ulUserNameMaxChars* characters. This format can be converted to *UserName/Password* format by using [CredUIParseUsername](#). A marshaled format can be passed directly to a [security support provider](#) (SSP).

If the CREDUI_FLAGS_DO_NOT_PERSIST flag is not specified, the value returned in this parameter is of a form that should not be inspected, printed, or persisted other than passing it to [CredUIParseUsername](#). The subsequent results of **CredUIParseUsername** can be passed only to a client-side authentication function such as [WNetAddConnection](#) or an SSP function.

If no domain or server is specified as part of this parameter, the value of the **pszTargetName** parameter is used as the domain to form a *DomainName\UserName* pair. On output, this parameter receives a string that contains that *DomainName\UserName* pair.

[in] ulUserNameBufferSize

The maximum number of characters that can be copied to *pszUserName* including the terminating null character.

Note CREDUI_MAX_USERNAME_LENGTH does not include the terminating null character.

[in, out] pszPassword

A pointer to a null-terminated string that contains the password for the credentials. If a nonzero-length string is specified for *pszPassword*, the password option of the dialog box will be prefilled with the string.

This function copies the user-supplied password to this buffer, copying a maximum of *ulPasswordMaxChars* characters. If the CREDUI_FLAGS_DO_NOT_PERSIST flag is not specified, the value returned in this parameter is of a form that should not be inspected, printed, or persisted other than passing it to a client-side authentication function such as [WNetAddConnection](#) or an SSP function.

When you have finished using the password, clear the password from memory by calling the [SecureZeroMemory](#) function. For more information about protecting passwords, see [Handling Passwords](#).

[in] ulPasswordBufferSize

The maximum number of characters that can be copied to *pszPassword* including the terminating null character.

Note CREDUI_MAX_PASSWORD_LENGTH does not include the terminating null character.

[in, out] save

A pointer to a **BOOL** that specifies the initial state of the **Save** check box and receives the state of the **Save** check box after the user has responded to the dialog box. If this value is not **NULL** and **CredUIPromptForCredentials** returns NO_ERROR, then *pfSave* returns the state of the **Save** check box when the user chose **OK** in the dialog box.


If the CREDUI_FLAGS_PERSIST flag is specified, the **Save** check box is not displayed, but is considered to be selected.

If the CREDUI_FLAGS_DO_NOT_PERSIST flag is specified and CREDUI_FLAGS_SHOW_SAVE_CHECK_BOX is not specified, the **Save** check box is not displayed, but is considered to be cleared.

An application that needs to use CredUI to prompt the user for credentials, but does not need the credential management services provided by the credential manager, can use *pfSave* to receive the state of the **Save** check box after the user closes the dialog box. To do this, the caller must specify CREDUI_FLAGS_DO_NOT_PERSIST and CREDUI_FLAGS_SHOW_SAVE_CHECK_BOX in *dwFlags*. When CREDUI_FLAGS_DO_NOT_PERSIST and CREDUI_FLAGS_SHOW_SAVE_CHECK_BOX are set, the application is responsible for examining **pfSave* after the function returns, and if **pfSave* is **TRUE**, then the application must take the appropriate action to save the user credentials within the resources of the application.

[in] dwFlags

A **DWORD** value that specifies special behavior for this function. This value can be a bitwise-OR combination of zero or more of the following values.


 Expand table

| Value | Meaning |
|--|---|
| CREDUI_FLAGS_ALWAYS_SHOW_UI 0x00080 | Specifies that a user interface will be shown even if the credentials can be returned from an existing credential in credential manager. This flag is permitted only if CREDUI_FLAGS_GENERIC_CREDENTIALS is also specified. |
| CREDUI_FLAGS_COMPLETE_USERNAME 0x00800 | Populate the combo box with the prompt for a user name. |
| CREDUI_FLAGS_DO_NOT_PERSIST 0x00002 | Do not store credentials or display check boxes. You can pass CREDUI_FLAGS_SHOW_SAVE_CHECK_BOX with this flag to display the Save check box only, and the result is returned in the <i>pfSave</i> output parameter. |
| CREDUI_FLAGS_EXCLUDE_CERTIFICATES 0x00008 | Populate the combo box with user name/password only. Do not display certificates or smart cards in the combo box. |
| CREDUI_FLAGS_EXPECT_CONFIRMATION 0x20000 | Specifies that the caller will call CredUIConfirmCredentials after checking to |

| | |
|---|---|
| | determine whether the returned credentials are actually valid. This mechanism ensures that credentials that are not valid are not saved to the credential manager. Specify this flag in all cases unless CREDUI_FLAGS_DO_NOT_PERSIST is specified. |
| CREDUI_FLAGS_GENERIC_CREDENTIALS 0x40000 | Consider the credentials entered by the user to be generic credentials. |
| CREDUI_FLAGS_INCORRECT_PASSWORD 0x00001 | Notify the user of insufficient credentials by displaying the "Logon unsuccessful" balloon tip. |
| CREDUI_FLAGS_KEEP_USERNAME 0x100000 | Do not allow the user to change the supplied user name. |
| CREDUI_FLAGS_PASSWORD_ONLY_OK 0x00200 | Populate the combo box with the password only. Do not allow a user name to be entered. |
| CREDUI_FLAGS_PERSIST 0x01000 | Do not show the Save check box, but the credential is saved as though the box were shown and selected. |
| CREDUI_FLAGS_REQUEST_ADMINISTRATOR 0x00004 | Populate the combo box with local administrators only. Windows XP Home Edition: This flag will filter out the well-known Administrator account. |
| CREDUI_FLAGS_REQUIRE_CERTIFICATE 0x00010 | Populate the combo box with certificates and smart cards only. Do not allow a user name to be entered. |
| CREDUI_FLAGS_REQUIRE_SMARTCARD 0x00100 | Populate the combo box with certificates or smart cards only. Do not allow a user name to be entered. |
| CREDUI_FLAGS_SERVER_CREDENTIAL 0x04000 | This flag is meaningful only in locating a matching credential to prefill the dialog box, should authentication fail. When this flag is specified, wildcard credentials will not be matched. It has no effect when writing a credential. CredUI does not create credentials that contain wildcard characters. Any found were either created explicitly by the user or created programmatically, as happens when a RAS connection is made. |
| CREDUI_FLAGS_SHOW_SAVE_CHECK_BOX 0x00040 | If the check box is selected, show the Save check box and return TRUE in the <i>pfSave</i> output parameter, otherwise, return FALSE . Check box uses the value in <i>pfSave</i> by default. |
| CREDUI_FLAGS_USERNAME_TARGET_CREDENTIALS 0x80000 | The credential is a "runas" credential. The <i>TargetName</i> parameter specifies the name of the command or program being run. It is used for prompting purposes only. |
| CREDUI_FLAGS_VALIDATE_USERNAME 0x00400 | Check that the user name is valid. |

Return value

The return value is a **DWORD** and can be one of the following values.

 Expand table

| Value | Description |
|-----------------|---|
| ERROR_CANCELLED | User chose Cancel . The <i>pszUserName</i> and <i>pszPassword</i> parameters have not changed. |

| | |
|-----------------------------|---|
| ERROR_INVALID_FLAGS | This status is returned for any of the flag configurations that are not valid. |
| ERROR_INVALID_PARAMETER | <p>Either <i>pszTargetName</i> is NULL, the empty string, or longer than CREDUI_MAX_DOMAIN_LENGTH, or <i>pUiInfo</i> is not NULL and the CredUI_INFO structure pointed to did not meet one of the following requirements:</p> <ul style="list-style-type: none">• The cbSize member must be one.• If the hbmBanner member is not NULL, it must be of type OBJ_BITMAP.• If the pszMessageText member is not NULL, it must not be greater than CREDUI_MAX_MESSAGE_LENGTH.• If the pszCaptionText member is not NULL, it must not be greater than CREDUI_MAX_CAPTION_LENGTH. |
| ERROR_NO_SUCH_LOGON_SESSION | The credential manager cannot be used. Typically, this error is handled by calling CredUIPromptForCredentials and passing in the CREDUI_FLAGS_DO_NOT_PERSIST flag. |
| NO_ERROR | User chose OK . The <i>pszUserName</i> , <i>pszPassword</i> , and <i>pfSave</i> parameters will return the values documented earlier. |

Remarks

The **CredUIPromptForCredentials** function creates and displays an application modal dialog box. After the dialog box is displayed and until it is closed by the user or application, no other window in the application can become active.

The flags CREDUI_FLAGS_REQUIRE_SMARTCARD, CREDUI_FLAGS_REQUIRE_CERTIFICATE, and CREDUI_FLAGS_EXCLUDE_CERTIFICATE are mutually exclusive.

If CREDUI_FLAGS_DO_NOT_PERSIST is specified, either *pszTargetName* must be specified or *uiInfo->pszMessageText* and *uiInfo->pszCaption* must be specified.

The flags CREDUI_FLAG_USERNAME_TARGET_CREDENTIALS and CREDUI_FLAGS_GENERIC_CREDENTIALS are mutually exclusive. If neither is specified, the credential is a domain credential.

An X509 certificate must have an [enhanced key usage](#) (EKU) value of **szOID_KP_SMARTCARD_LOGON** (1.3.6.1.4.1.311.20.2.2) to be displayed.

Windows XP: This EKU value is not required to display X509 certificates.

If CREDUI_FLAGS_GENERIC_CREDENTIALS is not specified or CREDUI_FLAGS_COMPLETE_USERNAME is specified, the typed name is *syntax checked*. Syntax checking applies the same rules as applied by [CredUIParseUserName](#). If the typed name is not valid, the user is prompted for a valid one. If the domain portion of the typed name is missing, one will be supplied based on the target name.

If CREDUI_FLAGS_GENERIC_CREDENTIALS is specified and CREDUI_FLAGS_VALIDATE_USERNAME is also specified, the typed name is syntax checked. If the typed name is not valid, the user is prompted for a valid one.

If CREDUI_FLAGS_GENERIC_CREDENTIALS is specified and neither CREDUI_FLAGS_COMPLETE_USERNAME nor CREDUI_FLAGS_VALIDATE_USERNAME is specified, the typed name is not syntax checked in any way.

If neither CREDUI_FLAGS_PERSIST nor CREDUI_FLAGS_DO_NOT_PERSIST is set, the **Save** check box is shown, and it controls whether the credential is saved.

Calling Modes

- The caller will attempt to access the target resource, call credui (passing a description of the target resource and the failure status), call [CredUIParseUserName](#), access the target

resource again, and then call [CredUIConfirmCredentials](#).

- The caller can prompt for credentials without accessing any resources by passing CREDUI_FLAGS_DO_NOT_PERSIST.
- For generic credentials, there is no authentication package. Therefore, the application needs to access the credential to do the authentication. Prompt for credentials, not passing CREDUI_FLAGS_ALWAYS_SHOW_UI before the first authentication. The user interface will appear only if there is no credential in the credential manager. On all subsequent messages from within the application, CREDUI_FLAGS_ALWAYS_SHOW_UI will be passed because the credential in the credential manager is clearly not valid for that resource.

Target Information

Target Information is information about the location of the resource to be accessed. For a list of all potential target names for a resource, call [CredGetTargetInfo](#). **CredGetTargetInfo** returns information that was cached by the Negotiate, NTLM, or Kerberos authentication package when one of those packages was used to authenticate to the named target. **CredGetTargetInfo** returns some or all of the following names for the target:

- NetBIOS server name of the computer
- DNS server name of the computer
- NetBIOS domain name of the domain the computer belongs to
- DNS domain name of the domain the computer belongs to
- DNS tree name of the tree the computer belongs to
- Name of the package that collected the information

Any piece of this information can be missing if the information does not apply to the target computer. For instance, a computer that is a member of a workgroup does not have a NetBIOS domain name.

Credentials are stored in the credential manager based on target name. Each target name is stored as generally as possible without colliding with credentials already stored in the credential manager. Because credentials are stored by target name, a particular user can only have one credential per target stored in the credential manager.

ⓘ Note

The wincred.h header defines CredUIPromptForCredentials as an alias which automatically selects the ANSI or Unicode version of this function based on the definition of the UNICODE preprocessor constant. Mixing usage of the encoding-neutral alias with code that not encoding-neutral can lead to mismatches that result in compilation or runtime errors. For more information, see [Conventions for Function Prototypes](#).

Requirements

[↗](#) Expand table

| Requirement | Value |
|--------------------------|---|
| Minimum supported client | Windows XP [desktop apps only] |
| Minimum supported server | Windows Server 2003 [desktop apps only] |
| Target Platform | Windows |
| Header | wincred.h |
| Library | Credui.lib |
| DLL | Credui.dll |

See also

- CredGetTargetInfo
- CredMarshalCredential
- CredUIConfirmCredentials
- CredUIParseUserName
- CredUIPromptForWindowsCredentials
- CredUI_INFO
- WNetAddConnection

Feedback

Was this page helpful?

Yes

No

[Provide product feedback](#) | [Get help at Microsoft Q&A](#)

Additional resources

Events

Nov 20, 12 AM - Nov 22, 12 AM

Gain the competitive edge you need with powerful AI and Cloud solutions by attending Microsoft Ignite online.
[Register now](#)