

pdh.h header

Article 01/24/2023

This header is used by Performance Counters. For more information, see:

- [Performance Counters](#)

pdh.h contains the following programming interfaces:

Functions

PdhAddCounterA
Adds the specified counter to the query. (ANSI)
PdhAddCounterW
Adds the specified counter to the query. (Unicode)
PdhAddEnglishCounterA
Adds the specified language-neutral counter to the query. (ANSI)
PdhAddEnglishCounterW
Adds the specified language-neutral counter to the query. (Unicode)
PdhBindInputDataSourceA
Binds one or more binary log files together for reading log data. (ANSI)
PdhBindInputDataSourceW
Binds one or more binary log files together for reading log data. (Unicode)
PdhBrowseCountersA
Displays a Browse Counters dialog box that the user can use to select one or more counters that they want to add to the query. To use handles to data sources, use the PdhBrowseCountersH function. (ANSI)

[PdhBrowseCountersHA](#)

Displays a Browse Counters dialog box that the user can use to select one or more counters that they want to add to the query. This function is identical to the PdhBrowseCounters function, except that it supports the use of handles to data sources. (ANSI)

[PdhBrowseCountersHW](#)

Displays a Browse Counters dialog box that the user can use to select one or more counters that they want to add to the query. This function is identical to the PdhBrowseCounters function, except that it supports the use of handles to data sources. (Unicode)

[PdhBrowseCountersW](#)

Displays a Browse Counters dialog box that the user can use to select one or more counters that they want to add to the query. To use handles to data sources, use the PdhBrowseCountersH function. (Unicode)

[PdhCalculateCounterFromRawValue](#)

Calculates the displayable value of two raw counter values.

[PdhCloseLog](#)

Closes the specified log file.

[PdhCloseQuery](#)

Closes all counters contained in the specified query, closes all handles related to the query, and frees all memory associated with the query.

[PdhCollectQueryData](#)

Collects the current raw data value for all counters in the specified query and updates the status code of each counter. (PdhCollectQueryData)

[PdhCollectQueryDataEx](#)

Uses a separate thread to collect the current raw data value for all counters in the specified query. The function then signals the application-defined event and waits the specified time interval before returning.

[PdhCollectQueryDataWithTime](#)

Collects the current raw data value for all counters in the specified query and updates the status code of each counter. (PdhCollectQueryDataWithTime)

[PdhComputeCounterStatistics](#)

Computes statistics for a counter from an array of raw values.

[PdhConnectMachineA](#)

Connects to the specified computer. (ANSI)

[PdhConnectMachineW](#)

Connects to the specified computer. (Unicode)

[PdhEnumLogSetNamesA](#)

Enumerates the names of the log sets within the DSN. (ANSI)

[PdhEnumLogSetNamesW](#)

Enumerates the names of the log sets within the DSN. (Unicode)

[PdhEnumMachinesA](#)

Returns a list of the computer names associated with counters in a log file. (PdhEnumMachinesA)

[PdhEnumMachinesHA](#)

Returns a list of the computer names associated with counters in a log file.
(PdhEnumMachinesHA)

[PdhEnumMachinesHW](#)

Returns a list of the computer names associated with counters in a log file.
(PdhEnumMachinesHW)

[PdhEnumMachinesW](#)

Returns a list of the computer names associated with counters in a log file. (PdhEnumMachinesW)

[PdhEnumObjectItemsA](#)

Returns the specified object's counter and instance names that exist on the specified computer or in the specified log file. To use handles to data sources, use the PdhEnumObjectItemsH function.
(ANSI)

[PdhEnumObjectItemsHA](#)

Returns the specified object's counter and instance names that exist on the specified computer or in the specified log file. This function is identical to the PdhEnumObjectItems function, except that it supports the use of handles to data sources. (ANSI)

[PdhEnumObjectItemsHW](#)

Returns the specified object's counter and instance names that exist on the specified computer or in the specified log file. This function is identical to the PdhEnumObjectItems function, except that it supports the use of handles to data sources. (Unicode)

[PdhEnumObjectItemsW](#)

Returns the specified object's counter and instance names that exist on the specified computer or in the specified log file. To use handles to data sources, use the PdhEnumObjectItemsH function. (Unicode)

[PdhEnumObjectsA](#)

Returns a list of objects available on the specified computer or in the specified log file. To use handles to data sources, use the PdhEnumObjectsH function. (ANSI)

[PdhEnumObjectsHA](#)

Returns a list of objects available on the specified computer or in the specified log file. This function is identical to PdhEnumObjects, except that it supports the use of handles to data sources. (ANSI)

[PdhEnumObjectsHW](#)

Returns a list of objects available on the specified computer or in the specified log file. This function is identical to PdhEnumObjects, except that it supports the use of handles to data sources. (Unicode)

[PdhEnumObjectsW](#)

Returns a list of objects available on the specified computer or in the specified log file. To use handles to data sources, use the PdhEnumObjectsH function. (Unicode)

[PdhExpandCounterPathA](#)

Examines the specified computer (or local computer if none is specified) for counters and instances of counters that match the wildcard strings in the counter path. (ANSI)

[PdhExpandCounterPathW](#)

Examines the specified computer (or local computer if none is specified) for counters and instances of counters that match the wildcard strings in the counter path. (Unicode)

[PdhExpandWildCardPathA](#)

Examines the specified computer or log file and returns those counter paths that match the given counter path which contains wildcard characters. To use handles to data sources, use the PdhExpandWildCardPathH function. (ANSI)

[PdhExpandWildCardPathHA](#)

Examines the specified computer or log file and returns those counter paths that match the given counter path which contains wildcard characters. This function is identical to the PdhExpandWildCardPath function, except that it supports the use of handles to data sources. (ANSI)

[PdhExpandWildCardPathHW](#)

Examines the specified computer or log file and returns those counter paths that match the given counter path which contains wildcard characters. This function is identical to the PdhExpandWildCardPath function, except that it supports the use of handles to data sources. (Unicode)

[PdhExpandWildCardPathW](#)

Examines the specified computer or log file and returns those counter paths that match the given counter path which contains wildcard characters. To use handles to data sources, use the PdhExpandWildCardPathH function. (Unicode)

[PdhFormatFromRawValue](#)

Computes a displayable value for the given raw counter values.

[PdhGetCounterInfoA](#)

Retrieves information about a counter, such as data size, counter type, path, and user-supplied data values. (ANSI)

[PdhGetCounterInfoW](#)

Retrieves information about a counter, such as data size, counter type, path, and user-supplied data values. (Unicode)

[PdhGetCounterTimeBase](#)

Returns the time base of the specified counter.

[PdhGetDataSourceTimeRangeA](#)

Determines the time range, number of entries and, if applicable, the size of the buffer containing the performance data from the specified input source. To use handles to data sources, use the PdhGetDataSourceTimeRangeH function. (ANSI)

[PdhGetDataSourceTimeRangeH](#)

Determines the time range, number of entries and, if applicable, the size of the buffer containing the performance data from the specified input source. This function is identical to the PdhGetDataSourceTimeRange function, except that it supports the use of handles to data sources.

[PdhGetDataSourceTimeRangeW](#)

Determines the time range, number of entries and, if applicable, the size of the buffer containing the performance data from the specified input source. To use handles to data sources, use the PdhGetDataSourceTimeRangeH function. (Unicode)

[PdhGetDefaultPerfCounterA](#)

Retrieves the name of the default counter for the specified object. This name can be used to set the initial counter selection in the Browse Counter dialog box. To use handles to data sources, use the PdhGetDefaultPerfCounterH function. (ANSI)

[PdhGetDefaultPerfCounterHA](#)

Retrieves the name of the default counter for the specified object. (ANSI)

[PdhGetDefaultPerfCounterHW](#)

Retrieves the name of the default counter for the specified object. (Unicode)

[PdhGetDefaultPerfCounterW](#)

Retrieves the name of the default counter for the specified object. This name can be used to set the initial counter selection in the Browse Counter dialog box. To use handles to data sources, use the PdhGetDefaultPerfCounterH function. (Unicode)

[PdhGetDefaultPerfObjectA](#)

Retrieves the name of the default object. This name can be used to set the initial object selection in the Browse Counter dialog box. To use handles to data sources, use the PdhGetDefaultPerfObjectH function. (ANSI)

[PdhGetDefaultPerfObjectHA](#)

Retrieves the name of the default object. (ANSI)

[PdhGetDefaultPerfObjectHW](#)

Retrieves the name of the default object. (Unicode)

[PdhGetDefaultPerfObjectW](#)

Retrieves the name of the default object. This name can be used to set the initial object selection in the Browse Counter dialog box. To use handles to data sources, use the PdhGetDefaultPerfObjectH function. (Unicode)

[PdhGetDllVersion](#)

Returns the version of the currently installed Pdh.dll file.

[PdhGetFormattedCounterArrayA](#)

Returns an array of formatted counter values. Use this function when you want to format the counter values of a counter that contains a wildcard character for the instance name. (ANSI)

[PdhGetFormattedCounterArrayW](#)

Returns an array of formatted counter values. Use this function when you want to format the counter values of a counter that contains a wildcard character for the instance name. (Unicode)

[PdhGetFormattedCounterValue](#)

Computes a displayable value for the specified counter.

[PdhGetLogFileSize](#)

Returns the size of the specified log file.

[PdhGetRawCounterArrayA](#)

Returns an array of raw values from the specified counter. Use this function when you want to retrieve the raw counter values of a counter that contains a wildcard character for the instance name. (ANSI)

[PdhGetRawCounterArrayW](#)

Returns an array of raw values from the specified counter. Use this function when you want to retrieve the raw counter values of a counter that contains a wildcard character for the instance name. (Unicode)

[PdhGetRawCounterValue](#)

Returns the current raw value of the counter.

[PdhIsRealTimeQuery](#)

Determines if the specified query is a real-time query.

[PdhLookupPerfIndexByNameA](#)

Returns the counter index corresponding to the specified counter name. (ANSI)

[PdhLookupPerfIndexByNameW](#)

Returns the counter index corresponding to the specified counter name. (Unicode)

[PdhLookupPerfNameByIndexA](#)

Returns the performance object name or counter name corresponding to the specified index. (ANSI)

[PdhLookupPerfNameByIndexW](#)

Returns the performance object name or counter name corresponding to the specified index. (Unicode)

[PdhMakeCounterPathA](#)

Creates a full counter path using the members specified in the PDH_COUNTER_PATH_ELEMENTS structure. (ANSI)

[PdhMakeCounterPathW](#)

Creates a full counter path using the members specified in the PDH_COUNTER_PATH_ELEMENTS structure. (Unicode)

[PdhOpenLogA](#)

Opens the specified log file for reading or writing. (ANSI)

[PdhOpenLogW](#)

Opens the specified log file for reading or writing. (Unicode)

[PdhOpenQueryA](#)

Creates a new query that is used to manage the collection of performance data. To use handles to data sources, use the PdhOpenQueryH function. (ANSI)

[PdhOpenQueryH](#)

Creates a new query that is used to manage the collection of performance data. This function is identical to the PdhOpenQuery function, except that it supports the use of handles to data sources.

[PdhOpenQueryW](#)

Creates a new query that is used to manage the collection of performance data. To use handles to data sources, use the PdhOpenQueryH function. (Unicode)

[PdhParseCounterPathA](#)

Parses the elements of the counter path and stores the results in the PDH_COUNTER_PATH_ELEMENTS structure. (ANSI)

[PdhParseCounterPathW](#)

Parses the elements of the counter path and stores the results in the PDH_COUNTER_PATH_ELEMENTS structure. (Unicode)

[PdhParseInstanceIdA](#)

Parses the elements of an instance string. (ANSI)

[PdhParseInstanceIdW](#)

Parses the elements of an instance string. (Unicode)

[PdhReadRawLogRecord](#)

Reads the information in the specified binary trace log file.

[PdhRemoveCounter](#)

Removes a counter from a query.

[PdhSelectDataSourceA](#)

Displays a dialog window that prompts the user to specify the source of the performance data. (ANSI)

[PdhSelectDataSourceW](#)

Displays a dialog window that prompts the user to specify the source of the performance data. (Unicode)

[PdhSetCounterScaleFactor](#)

Sets the scale factor that is applied to the calculated value of the specified counter when you request the formatted counter value. If the PDH_FMT_NOSCALE flag is set, then this scale factor is ignored.

[PdhSetDefaultRealTimeDataSource](#)

Specifies the source of the real-time data.

[PdhSetQueryTimeRange](#)

Limits the samples that you can read from a log file to those within the specified time range, inclusively.

[PdhUpdateLogA](#)

Collects counter data for the current query and writes the data to the log file. (ANSI)

[PdhUpdateLogFileCatalog](#)

Synchronizes the information in the log file catalog with the performance data in the log file.

[PdhUpdateLogW](#)

Collects counter data for the current query and writes the data to the log file. (Unicode)

[PdhValidatePathA](#)

Validates that the counter is present on the computer specified in the counter path. (ANSI)

[PdhValidatePathExA](#)

Validates that the specified counter is present on the computer or in the log file. (ANSI)

[PdhValidatePathExW](#)

Validates that the specified counter is present on the computer or in the log file. (Unicode)

[PdhValidatePathW](#)

Validates that the counter is present on the computer specified in the counter path. (Unicode)

Callback functions

[CounterPathCallBack](#)

Applications implement the CounterPathCallBack function to process the counter path strings returned by the Browse dialog box.

Structures

[PDH_BROWSE_DLG_CONFIG_A](#)

The PDH_BROWSE_DLG_CONFIG structure is used by the PdhBrowseCounters function to configure the Browse Performance Counters dialog box. (ANSI)

[PDH_BROWSE_DLG_CONFIG_HA](#)

The PDH_BROWSE_DLG_CONFIG_H structure is used by the PdhBrowseCountersH function to configure the Browse Performance Counters dialog box. (ANSI)

[PDH_BROWSE_DLG_CONFIG_HW](#)

The PDH_BROWSE_DLG_CONFIG_H structure is used by the PdhBrowseCountersH function to configure the Browse Performance Counters dialog box. (Unicode)

[PDH_BROWSE_DLG_CONFIG_W](#)

The PDH_BROWSE_DLG_CONFIG structure is used by the PdhBrowseCounters function to configure the Browse Performance Counters dialog box. (Unicode)

[PDH_COUNTER_INFO_A](#)

The PDH_COUNTER_INFO structure contains information describing the properties of a counter. This information also includes the counter path. (ANSI)

[PDH_COUNTER_INFO_W](#)

The PDH_COUNTER_INFO structure contains information describing the properties of a counter. This information also includes the counter path. (Unicode)

[PDH_COUNTER_PATH_ELEMENTS_A](#)

The PDH_COUNTER_PATH_ELEMENTS structure contains the components of a counter path. (ANSI)

[PDH_COUNTER_PATH_ELEMENTS_W](#)

The PDH_COUNTER_PATH_ELEMENTS structure contains the components of a counter path. (Unicode)

[PDH_DATA_ITEM_PATH_ELEMENTS_A](#)

The PDH_DATA_ITEM_PATH_ELEMENTS structure contains the path elements of a specific data item. (ANSI)

[PDH_DATA_ITEM_PATH_ELEMENTS_W](#)

The PDH_DATA_ITEM_PATH_ELEMENTS structure contains the path elements of a specific data item. (Unicode)

[PDH_FMT_COUNTERVALUE](#)

The PDH_FMT_COUNTERVALUE structure contains the computed value of the counter and its status.

[PDH_FMT_COUNTERVALUE_ITEM_A](#)

The PDH_FMT_COUNTERVALUE_ITEM structure contains the instance name and formatted value of a counter. (ANSI)

[PDH_FMT_COUNTERVALUE_ITEM_W](#)

The PDH_FMT_COUNTERVALUE_ITEM structure contains the instance name and formatted value of a counter. (Unicode)

[PDH_RAW_COUNTER](#)

The PDH_RAW_COUNTER structure returns the data as it was collected from the counter provider. No translation, formatting, or other interpretation is performed on the data.

[PDH_RAW_COUNTER_ITEM_A](#)

The PDH_RAW_COUNTER_ITEM structure contains the instance name and raw value of a counter. (ANSI)

[PDH_RAW_COUNTER_ITEM_W](#)

The PDH_RAW_COUNTER_ITEM structure contains the instance name and raw value of a counter. (Unicode)

[PDH_RAW_LOG_RECORD](#)

The PDH_RAW_LOG_RECORD structure contains information about a binary trace log file record.

[PDH_STATISTICS](#)

The PDH_STATISTICS structure contains the minimum, maximum, and mean values for an array of raw counters values.

[PDH_TIME_INFO](#)

The PDH_TIME_INFO structure contains information on time intervals as applied to the sampling of performance data.

Feedback

Was this page helpful?

 Yes

 No

[Get help at Microsoft Q&A](#)

CounterPathCallBack callback function (pdh.h)

Article02/22/2024

Applications implement the **CounterPathCallBack** function to process the counter path strings returned by the **Browse** dialog box.

Syntax

```
C++  
  
CounterPathCallBack Counterpathcallback;  
  
PDH_STATUS Counterpathcallback(  
    DWORD_PTR unnamedParam1  
)  
{...}
```

Parameters

unnamedParam1

User-defined value passed to the callback function by the **Browse** dialog box. You set this value in the **dwCallBackArg** member of the [PDH_BROWSE_DLG_CONFIG](#) structure.

Return value

Return **ERROR_SUCCESS** if the function succeeds.

If the function fails due to a transient error, you can return **PDH_RETRY** and PDH will call your callback immediately.

Otherwise, return an appropriate error code. The error code is passed back to the caller of [PdhBrowseCounters](#).

Remarks

The following members of the [PDH_BROWSE_DLG_CONFIG](#) structure are used to communicate with the callback function:

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	pdh.h

See also

[PDH_BROWSE_DLG_CONFIG](#)

[PdhBrowseCounters](#)

Feedback

Was this page helpful?

 Yes

 No

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

PDH_BROWSE_DLG_CONFIG_A structure (pdh.h)

Article07/27/2022

The **PDH_BROWSE_DLG_CONFIG** structure is used by the [PdhBrowseCounters](#) function to configure the **Browse Performance Counters** dialog box.

Syntax

C++

```
typedef struct _BrowseDlgConfig_A {
    DWORD          bIncludeInstanceIndex : 1;
    DWORD          bSingleCounterPerAdd : 1;
    DWORD          bSingleCounterPerDialog : 1;
    DWORD          bLocalCountersOnly : 1;
    DWORD          bWildCardInstances : 1;
    DWORD          bHideDetailBox : 1;
    DWORD          bInitializePath : 1;
    DWORD          bDisableMachineSelection : 1;
    DWORD          bIncludeCostlyObjects : 1;
    DWORD          bShowObjectBrowser : 1;
    DWORD          bReserved : 22;
    HWND           hWndOwner;
    LPSTR          szDataSource;
    LPSTR          szReturnPathBuffer;
    DWORD          cchReturnPathLength;
    CounterPathCallBack pCallBack;
    DWORD_PTR      dwCallBackArg;
    PDH_STATUS     CallBackStatus;
    DWORD          dwDefaultDetailLevel;
    LPSTR          szDialogBoxCaption;
} PDH_BROWSE_DLG_CONFIG_A, *PPDH_BROWSE_DLG_CONFIG_A;
```

Members

bIncludeInstanceIndex

If this flag is **TRUE**, the dialog box includes an index number for duplicate instance names. For example, if there are two cmd instances, the instance list will contain cmd and cmd#1. If this flag is **FALSE**, duplicate instance names will not contain an index number.

bSingleCounterPerAdd

If this flag is **TRUE**, the dialog returns only one counter. If this flag is **FALSE**, the dialog can return multiple selections, and wildcard selections are permitted. Selected counters are returned as a **MULTI_SZ** string.

bSingleCounterPerDialog

If this flag is **TRUE**, the dialog box uses an OK and Cancel button. The dialog returns when the user clicks either button. If this flag is **FALSE**, the dialog box uses an Add and Close button. The dialog box closes when the user clicks the Close button. The Add button can be clicked multiple times. The Add button overwrites the previously selected items with the currently selected items.

bLocalCountersOnly

If this flag is **TRUE**, the dialog box lets the user select counters only from the local computer (the path will not contain a computer name). If this flag is **FALSE**, the user can specify a computer from which to select counters. The computer name will prefix the counter path unless the user selects **Use local computer counters**.

bWildCardInstances

If this flag is **TRUE** and the user selects **All instances**, the counter path will include the wildcard character for the instance field.

If this flag is **FALSE**, and the user selects **All instances**, all the instances currently found for that object will be returned in a **MULTI_SZ** string.

bHideDetailBox

If this flag is **TRUE**, this removes **Detail level** from the dialog box so the user cannot change the detail level of the counters displayed in the dialog box. The detail level will be fixed to the value of the **dwDefaultDetailLevel** member.

If this flag is **FALSE**, this displays **Detail level** in the dialog box, allowing the user to change the detail level of the counters displayed.

Note that the counters displayed will be those whose detail level is less than or equal to the current detail level selection. Selecting a detail level of **Wizard** will display all counters and objects.

bInitializePath

If this flag is **TRUE**, the dialog highlights the counter and object specified in **szReturnPathBuffer** when the dialog box is first displayed, instead of using the default counter and object specified by the computer.

If this flag is **FALSE**, this selects the initial counter and object using the default counter and object information returned by the computer.

bDisableMachineSelection

If this flag is **TRUE**, the user cannot select a computer from **Select counters from computer**.

If this flag is **FALSE**, the user can select a computer from **Select counters from computer**. This is the default value. The list contains the local computer only unless you call the [PdhConnectMachine](#) to connect to other computers first.

bIncludeCostlyObjects

If this flag is **TRUE**, the counters list will also contain costly data—that is, data that requires a relatively large amount of processor time or memory overhead to collect.

If this flag is **FALSE**, the list will not contain costly counters. This is the default value.

bShowObjectBrowser

If this flag is **TRUE**, the dialog lists only performance objects. When the user selects an object, the dialog returns a counter path that includes the object and wildcard characters for the instance name and counter if the object is a multiple instance object. For example, if the "Process" object is selected, the dialog returns the string "\Process(*)*". If the object is a single instance object, the path contains a wildcard character for counter only. For example, "\System*". You can then pass the path to [PdhExpandWildCardPath](#) to retrieve a list of actual paths for the object.

bReserved

hWndOwner

Handle of the window to own the dialog. If **NULL**, the owner is the desktop.

szDataSource

Pointer to a **null**-terminated string that specifies the name of the log file from which the list of counters is retrieved. If **NULL**, the list of counters is retrieved from the local computer (or remote computer if specified).

szReturnPathBuffer

Pointer to a **MULTI_SZ** that contains the selected counter paths.

If **bInitializePath** is **TRUE**, you can use this member to specify a counter path whose components are used to highlight entries in computer, object, counter, and instance lists when the dialog is first displayed.

cchReturnPathLength

Size of the **szReturnPathBuffer** buffer, in **TCHARs**. If the callback function reallocates a new buffer, it must also update this value.

pCallBack

Pointer to the callback function that processes the user's selection. For more information, see [CounterPathCallBack](#).

dwCallBackArg

Caller-defined value that is passed to the callback function.

CallBackStatus

On entry to the callback function, this member contains the status of the path buffer. On exit, the callback function sets the status value resulting from processing.

If the buffer is too small to load the current selection, the dialog sets this value to **PDH_MORE_DATA**. If this value is **ERROR_SUCCESS**, then the **szReturnPathBuffer** member contains a valid counter path or counter path list.

If the callback function reallocates a new buffer, it should set this member to **PDH_RETRY** so that the dialog will try to load the buffer with the selected paths and call the callback function again.

If some other error occurred, then the callback function should return the appropriate PDH error status value.

dwDefaultDetailLevel

Default detail level to show in the **Detail level** list if **bHideDetailBox** is **FALSE**. If **bHideDetailBox** is **TRUE**, the dialog uses this value to filter the displayed performance counters and objects. You can specify one of the following values:

 Expand table

Detail level	Meaning
PERF_DETAIL_NOVICE	A novice user can understand the counter data.

PERF_DETAIL_ADVANCED	The counter data is provided for advanced users.
PERF_DETAIL_EXPERT	The counter data is provided for expert users.
PERF_DETAIL_WIZARD	The counter data is provided for system designers.

szDialogBoxCaption

Pointer to a **null**-terminated string that specifies the optional caption to display in the caption bar of the dialog box. If this member is **NULL**, the caption will be **Browse Performance Counters**.

Remarks

Each time the [Add](#) button is clicked, the **szReturnPathBuffer** buffer contains the selected counter and the **pCallBack** callback function is called. The callback function should call the [PdhAddCounter](#) function for each counter in the buffer.

Requirements

 Expand table

Requirement	Value
Minimum supported client	Windows XP [desktop apps only]
Minimum supported server	Windows Server 2003 [desktop apps only]
Header	pdh.h

See also

[CounterPathCallBack](#)

[PdhAddCounter](#)

[PdhBrowseCounters](#)

Feedback

Was this page helpful?

 Yes

 No

Provide product feedback  | Get help at Microsoft Q&A

PDH_BROWSE_DLG_CONFIG_HA structure (pdh.h)

Article07/27/2022

The **PDH_BROWSE_DLG_CONFIG_H** structure is used by the [PdhBrowseCountersH](#) function to configure the **Browse Performance Counters** dialog box.

Syntax

C++

```
typedef struct _BrowseDlgConfig_HA {
    DWORD          bIncludeInstanceIndex : 1;
    DWORD          bSingleCounterPerAdd : 1;
    DWORD          bSingleCounterPerDialog : 1;
    DWORD          bLocalCountersOnly : 1;
    DWORD          bWildCardInstances : 1;
    DWORD          bHideDetailBox : 1;
    DWORD          bInitializePath : 1;
    DWORD          bDisableMachineSelection : 1;
    DWORD          bIncludeCostlyObjects : 1;
    DWORD          bShowObjectBrowser : 1;
    DWORD          bReserved : 22;
    HWND           hWndOwner;
    PDH_HLOG       hDataSource;
    LPSTR          szReturnPathBuffer;
    DWORD          cchReturnPathLength;
    CounterPathCallBack pCallBack;
    DWORD_PTR      dwCallBackArg;
    PDH_STATUS     CallBackStatus;
    DWORD          dwDefaultDetailLevel;
    LPSTR          szDialogBoxCaption;
} PDH_BROWSE_DLG_CONFIG_HA, *PPDH_BROWSE_DLG_CONFIG_HA;
```

Members

bIncludeInstanceIndex

If this flag is **TRUE**, the dialog box includes an index number for duplicate instance names. For example, if there are two cmd instances, the instance list will contain cmd and cmd#1. If this flag is **FALSE**, duplicate instance names will not contain an index number.

bSingleCounterPerAdd

If this flag is **TRUE**, the dialog returns only one counter. If this flag is **FALSE**, the dialog can return multiple selections, and wildcard selections are permitted. Selected counters are returned as a **MULTI_SZ** string.

bSingleCounterPerDialog

If this flag is **TRUE**, the dialog box uses an OK and Cancel button. The dialog returns when the user clicks either button. If this flag is **FALSE**, the dialog box uses an Add and Close button. The dialog box closes when the user clicks the Close button. The Add button can be clicked multiple times. The Add button overwrites the previously selected items with the currently selected items.

bLocalCountersOnly

If this flag is **TRUE**, the dialog box lets the user select counters only from the local computer (the path will not contain a computer name). If this flag is **FALSE**, the user can specify a computer from which to select counters. The computer name will prefix the counter path unless the user selects **Use local computer counters**.

bWildCardInstances

If this flag is **TRUE** and the user selects **All instances**, the counter path will include the wildcard character for the instance field.

If this flag is **FALSE**, and the user selects **All instances**, all the instances currently found for that object will be returned in a **MULTI_SZ** string.

bHideDetailBox

If this flag is **TRUE**, this removes **Detail level** from the dialog box so the user cannot change the detail level of the counters displayed in the dialog box. The detail level will be fixed to the value of the **dwDefaultDetailLevel** member.

If this flag is **FALSE**, this displays **Detail level** in the dialog box, allowing the user to change the detail level of the counters displayed.

Note that the counters displayed will be those whose detail level is less than or equal to the current detail level selection. Selecting a detail level of **Wizard** will display all counters and objects.

bInitializePath

If this flag is **TRUE**, the dialog highlights the counter and object specified in **szReturnPathBuffer** when the dialog box is first displayed, instead of using the default counter and object specified by the computer.

If this flag is **FALSE**, this selects the initial counter and object using the default counter and object information returned by the computer.

`bDisableMachineSelection`

If this flag is **TRUE**, the user cannot select a computer from **Select counters from computer**.

If this flag is **FALSE**, the user can select a computer from **Select counters from computer**. This is the default value. The list contains the local computer only unless you call the [PdhConnectMachine](#) to connect to other computers first.

`bIncludeCostlyObjects`

If this flag is **TRUE**, the counters list will also contain costly data—that is, data that requires a relatively large amount of processor time or memory overhead to collect.

If this flag is **FALSE**, the list will not contain costly counters. This is the default value.

`bShowObjectBrowser`

If this flag is **TRUE**, the dialog lists only performance objects. When the user selects an object, the dialog returns a counter path that includes the object and wildcard characters for the instance name and counter if the object is a multiple instance object. For example, if the "Process" object is selected, the dialog returns the string "\Process(*)*". If the object is a single instance object, the path contains a wildcard character for counter only. For example, "\System*". You can then pass the path to [PdhExpandWildCardPath](#) to retrieve a list of actual paths for the object.

`bReserved`

`hWndOwner`

Handle of the window to own the dialog. If **NULL**, the owner is the desktop.

`hDataSource`

Handle to a data source returned by the [PdhBindInputDataSource](#) function.

`szReturnPathBuffer`

Pointer to a **MULTI_SZ** that contains the selected counter paths.

If **bInitializePath** is **TRUE**, you can use this member to specify a counter path whose components are used to highlight entries in computer, object, counter, and instance lists when the dialog is first displayed.

cchReturnPathLength

Size of the **szReturnPathBuffer** buffer, in TCHARs. If the callback function reallocates a new buffer, it must also update this value.

pCallBack

Pointer to the callback function that processes the user's selection. For more information, see [CounterPathCallBack](#).

dwCallBackArg

Caller-defined value that is passed to the callback function.

CallBackStatus

On entry to the callback function, this member contains the status of the path buffer. On exit, the callback function sets the status value resulting from processing.

If the buffer is too small to load the current selection, the dialog sets this value to PDH_MORE_DATA. If this value is ERROR_SUCCESS, then the **szReturnPathBuffer** member contains a valid counter path or counter path list.

If the callback function reallocates a new buffer, it should set this member to PDH_RETRY so that the dialog will try to load the buffer with the selected paths and call the callback function again.

If some other error occurred, then the callback function should return the appropriate PDH error status value.

dwDefaultDetailLevel

Default detail level to show in the **Detail level** list if **bHideDetailBox** is FALSE. If **bHideDetailBox** is TRUE, the dialog uses this value to filter the displayed performance counters and objects. You can specify one of the following values:

 Expand table

Detail level	Meaning
PERF_DETAIL_NOVICE	A novice user can understand the counter data.
PERF_DETAIL_ADVANCED	The counter data is provided for advanced users.
PERF_DETAIL_EXPERT	The counter data is provided for expert users.
PERF_DETAIL_WIZARD	The counter data is provided for system designers.

szDialogBoxCaption

Pointer to a **null**-terminated string that specifies the optional caption to display in the caption bar of the dialog box. If this member is **NULL**, the caption will be **Browse Performance Counters**.

Remarks

Each time the [Add](#) button is clicked, the **szReturnPathBuffer** buffer contains the selected counter and the **pCallBack** callback function is called. The callback function should call the [PdhAddCounter](#) function for each counter in the buffer.

ⓘ Note

The pdh.h header defines PDH_BROWSE_DLG_CONFIG_H 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]
Header	pdh.h

See also

[CounterPathCallBack](#)

[PdhAddCounter](#)

[PdhBindInputDataSource](#)

Feedback

Was this page helpful?

 Yes

 No

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

PDH_BROWSE_DLG_CONFIG_HW structure (pdh.h)

Article07/27/2022

The **PDH_BROWSE_DLG_CONFIG_H** structure is used by the [PdhBrowseCountersH](#) function to configure the **Browse Performance Counters** dialog box.

Syntax

C++

```
typedef struct _BrowseDlgConfig_HW {
    DWORD          bIncludeInstanceIndex : 1;
    DWORD          bSingleCounterPerAdd : 1;
    DWORD          bSingleCounterPerDialog : 1;
    DWORD          bLocalCountersOnly : 1;
    DWORD          bWildCardInstances : 1;
    DWORD          bHideDetailBox : 1;
    DWORD          bInitializePath : 1;
    DWORD          bDisableMachineSelection : 1;
    DWORD          bIncludeCostlyObjects : 1;
    DWORD          bShowObjectBrowser : 1;
    DWORD          bReserved : 22;
    HWND           hWndOwner;
    PDH_HLOG       hDataSource;
    LPWSTR         szReturnPathBuffer;
    DWORD          cchReturnPathLength;
    CounterPathCallBack pCallBack;
    DWORD_PTR      dwCallBackArg;
    PDH_STATUS     CallBackStatus;
    DWORD          dwDefaultDetailLevel;
    LPWSTR         szDialogBoxCaption;
} PDH_BROWSE_DLG_CONFIG_HW, *PPDH_BROWSE_DLG_CONFIG_HW;
```

Members

bIncludeInstanceIndex

If this flag is **TRUE**, the dialog box includes an index number for duplicate instance names. For example, if there are two cmd instances, the instance list will contain cmd and cmd#1. If this flag is **FALSE**, duplicate instance names will not contain an index number.

bSingleCounterPerAdd

If this flag is **TRUE**, the dialog returns only one counter. If this flag is **FALSE**, the dialog can return multiple selections, and wildcard selections are permitted. Selected counters are returned as a **MULTI_SZ** string.

bSingleCounterPerDialog

If this flag is **TRUE**, the dialog box uses an OK and Cancel button. The dialog returns when the user clicks either button. If this flag is **FALSE**, the dialog box uses an Add and Close button. The dialog box closes when the user clicks the Close button. The Add button can be clicked multiple times. The Add button overwrites the previously selected items with the currently selected items.

bLocalCountersOnly

If this flag is **TRUE**, the dialog box lets the user select counters only from the local computer (the path will not contain a computer name). If this flag is **FALSE**, the user can specify a computer from which to select counters. The computer name will prefix the counter path unless the user selects **Use local computer counters**.

bWildCardInstances

If this flag is **TRUE** and the user selects **All instances**, the counter path will include the wildcard character for the instance field.

If this flag is **FALSE**, and the user selects **All instances**, all the instances currently found for that object will be returned in a **MULTI_SZ** string.

bHideDetailBox

If this flag is **TRUE**, this removes **Detail level** from the dialog box so the user cannot change the detail level of the counters displayed in the dialog box. The detail level will be fixed to the value of the **dwDefaultDetailLevel** member.

If this flag is **FALSE**, this displays **Detail level** in the dialog box, allowing the user to change the detail level of the counters displayed.

Note that the counters displayed will be those whose detail level is less than or equal to the current detail level selection. Selecting a detail level of **Wizard** will display all counters and objects.

bInitializePath

If this flag is **TRUE**, the dialog highlights the counter and object specified in **szReturnPathBuffer** when the dialog box is first displayed, instead of using the default counter and object specified by the computer.

If this flag is **FALSE**, this selects the initial counter and object using the default counter and object information returned by the computer.

bDisableMachineSelection

If this flag is **TRUE**, the user cannot select a computer from **Select counters from computer**.

If this flag is **FALSE**, the user can select a computer from **Select counters from computer**. This is the default value. The list contains the local computer only unless you call the [PdhConnectMachine](#) to connect to other computers first.

bIncludeCostlyObjects

If this flag is **TRUE**, the counters list will also contain costly data—that is, data that requires a relatively large amount of processor time or memory overhead to collect.

If this flag is **FALSE**, the list will not contain costly counters. This is the default value.

bShowObjectBrowser

If this flag is **TRUE**, the dialog lists only performance objects. When the user selects an object, the dialog returns a counter path that includes the object and wildcard characters for the instance name and counter if the object is a multiple instance object. For example, if the "Process" object is selected, the dialog returns the string "\Process(*)*". If the object is a single instance object, the path contains a wildcard character for counter only. For example, "\System*". You can then pass the path to [PdhExpandWildCardPath](#) to retrieve a list of actual paths for the object.

bReserved

hWndOwner

Handle of the window to own the dialog. If **NULL**, the owner is the desktop.

hDataSource

Handle to a data source returned by the [PdhBindInputDataSource](#) function.

szReturnPathBuffer

Pointer to a **MULTI_SZ** that contains the selected counter paths.

If **bInitializePath** is **TRUE**, you can use this member to specify a counter path whose components are used to highlight entries in computer, object, counter, and instance lists when the dialog is first displayed.

cchReturnPathLength

Size of the **szReturnPathBuffer** buffer, in TCHARs. If the callback function reallocates a new buffer, it must also update this value.

pCallBack

Pointer to the callback function that processes the user's selection. For more information, see [CounterPathCallBack](#).

dwCallBackArg

Caller-defined value that is passed to the callback function.

CallBackStatus

On entry to the callback function, this member contains the status of the path buffer. On exit, the callback function sets the status value resulting from processing.

If the buffer is too small to load the current selection, the dialog sets this value to PDH_MORE_DATA. If this value is ERROR_SUCCESS, then the **szReturnPathBuffer** member contains a valid counter path or counter path list.

If the callback function reallocates a new buffer, it should set this member to PDH_RETRY so that the dialog will try to load the buffer with the selected paths and call the callback function again.

If some other error occurred, then the callback function should return the appropriate PDH error status value.

dwDefaultDetailLevel

Default detail level to show in the **Detail level** list if **bHideDetailBox** is FALSE. If **bHideDetailBox** is TRUE, the dialog uses this value to filter the displayed performance counters and objects. You can specify one of the following values:

Detail level	Meaning
PERF_DETAIL_NOVICE	A novice user can understand the counter data.
PERF_DETAIL_ADVANCED	The counter data is provided for advanced users.
PERF_DETAIL_EXPERT	The counter data is provided for expert users.
PERF_DETAIL_WIZARD	The counter data is provided for system designers.

szDialogBoxCaption

Pointer to a **null**-terminated string that specifies the optional caption to display in the caption bar of the dialog box. If this member is **NULL**, the caption will be **Browse Performance Counters**.

Remarks

Each time the [Add](#) button is clicked, the **szReturnPathBuffer** buffer contains the selected counter and the **pCallBack** callback function is called. The callback function should call the [PdhAddCounter](#) function for each counter in the buffer.

Note

The pdh.h header defines PDH_BROWSE_DLG_CONFIG_H 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 is not encoding-neutral can lead to mismatches that result in compilation or runtime errors. For more information, see [Conventions for Function Prototypes](#).

Requirements

Minimum supported client	Windows XP [desktop apps only]
Minimum supported server	Windows Server 2003 [desktop apps only]
Header	pdh.h

See also

[CounterPathCallBack](#)

[PdhAddCounter](#)

[PdhBindInputDataSource](#)

[PdhBrowseCountersH](#)

Feedback

Was this page helpful?

 Yes

 No

Get help at Microsoft Q&A

PDH_BROWSE_DLG_CONFIG_W structure (pdh.h)

Article07/27/2022

The **PDH_BROWSE_DLG_CONFIG** structure is used by the [PdhBrowseCounters](#) function to configure the **Browse Performance Counters** dialog box.

Syntax

C++

```
typedef struct _BrowseDlgConfig_W {
    DWORD          bIncludeInstanceIndex : 1;
    DWORD          bSingleCounterPerAdd : 1;
    DWORD          bSingleCounterPerDialog : 1;
    DWORD          bLocalCountersOnly : 1;
    DWORD          bWildCardInstances : 1;
    DWORD          bHideDetailBox : 1;
    DWORD          bInitializePath : 1;
    DWORD          bDisableMachineSelection : 1;
    DWORD          bIncludeCostlyObjects : 1;
    DWORD          bShowObjectBrowser : 1;
    DWORD          bReserved : 22;
    HWND           hWndOwner;
    LPWSTR         szDataSource;
    LPWSTR         szReturnPathBuffer;
    DWORD          cchReturnPathLength;
    CounterPathCallBack pCallBack;
    DWORD_PTR      dwCallBackArg;
    PDH_STATUS     CallBackStatus;
    DWORD          dwDefaultDetailLevel;
    LPWSTR         szDialogBoxCaption;
} PDH_BROWSE_DLG_CONFIG_W, *PPDH_BROWSE_DLG_CONFIG_W;
```

Members

bIncludeInstanceIndex

If this flag is **TRUE**, the dialog box includes an index number for duplicate instance names. For example, if there are two cmd instances, the instance list will contain cmd and cmd#1. If this flag is **FALSE**, duplicate instance names will not contain an index number.

bSingleCounterPerAdd

If this flag is **TRUE**, the dialog returns only one counter. If this flag is **FALSE**, the dialog can return multiple selections, and wildcard selections are permitted. Selected counters are returned as a **MULTI_SZ** string.

bSingleCounterPerDialog

If this flag is **TRUE**, the dialog box uses an OK and Cancel button. The dialog returns when the user clicks either button. If this flag is **FALSE**, the dialog box uses an Add and Close button. The dialog box closes when the user clicks the Close button. The Add button can be clicked multiple times. The Add button overwrites the previously selected items with the currently selected items.

bLocalCountersOnly

If this flag is **TRUE**, the dialog box lets the user select counters only from the local computer (the path will not contain a computer name). If this flag is **FALSE**, the user can specify a computer from which to select counters. The computer name will prefix the counter path unless the user selects **Use local computer counters**.

bWildCardInstances

If this flag is **TRUE** and the user selects **All instances**, the counter path will include the wildcard character for the instance field.

If this flag is **FALSE**, and the user selects **All instances**, all the instances currently found for that object will be returned in a **MULTI_SZ** string.

bHideDetailBox

If this flag is **TRUE**, this removes **Detail level** from the dialog box so the user cannot change the detail level of the counters displayed in the dialog box. The detail level will be fixed to the value of the **dwDefaultDetailLevel** member.

If this flag is **FALSE**, this displays **Detail level** in the dialog box, allowing the user to change the detail level of the counters displayed.

Note that the counters displayed will be those whose detail level is less than or equal to the current detail level selection. Selecting a detail level of **Wizard** will display all counters and objects.

bInitializePath

If this flag is **TRUE**, the dialog highlights the counter and object specified in **szReturnPathBuffer** when the dialog box is first displayed, instead of using the default counter and object specified by the computer.

If this flag is **FALSE**, this selects the initial counter and object using the default counter and object information returned by the computer.

bDisableMachineSelection

If this flag is **TRUE**, the user cannot select a computer from **Select counters from computer**.

If this flag is **FALSE**, the user can select a computer from **Select counters from computer**. This is the default value. The list contains the local computer only unless you call the [PdhConnectMachine](#) to connect to other computers first.

bIncludeCostlyObjects

If this flag is **TRUE**, the counters list will also contain costly data—that is, data that requires a relatively large amount of processor time or memory overhead to collect.

If this flag is **FALSE**, the list will not contain costly counters. This is the default value.

bShowObjectBrowser

If this flag is **TRUE**, the dialog lists only performance objects. When the user selects an object, the dialog returns a counter path that includes the object and wildcard characters for the instance name and counter if the object is a multiple instance object. For example, if the "Process" object is selected, the dialog returns the string "\Process(*)*". If the object is a single instance object, the path contains a wildcard character for counter only. For example, "\System*". You can then pass the path to [PdhExpandWildCardPath](#) to retrieve a list of actual paths for the object.

bReserved

hWndOwner

Handle of the window to own the dialog. If **NULL**, the owner is the desktop.

szDataSource

Pointer to a **null**-terminated string that specifies the name of the log file from which the list of counters is retrieved. If **NULL**, the list of counters is retrieved from the local computer (or remote computer if specified).

szReturnPathBuffer

Pointer to a **MULTI_SZ** that contains the selected counter paths.

If **bInitializePath** is **TRUE**, you can use this member to specify a counter path whose components are used to highlight entries in computer, object, counter, and instance lists when the dialog is first displayed.

cchReturnPathLength

Size of the **szReturnPathBuffer** buffer, in **TCHARs**. If the callback function reallocates a new buffer, it must also update this value.

pCallBack

Pointer to the callback function that processes the user's selection. For more information, see [CounterPathCallBack](#).

dwCallBackArg

Caller-defined value that is passed to the callback function.

CallBackStatus

On entry to the callback function, this member contains the status of the path buffer. On exit, the callback function sets the status value resulting from processing.

If the buffer is too small to load the current selection, the dialog sets this value to **PDH_MORE_DATA**. If this value is **ERROR_SUCCESS**, then the **szReturnPathBuffer** member contains a valid counter path or counter path list.

If the callback function reallocates a new buffer, it should set this member to **PDH_RETRY** so that the dialog will try to load the buffer with the selected paths and call the callback function again.

If some other error occurred, then the callback function should return the appropriate PDH error status value.

dwDefaultDetailLevel

Default detail level to show in the **Detail level** list if **bHideDetailBox** is **FALSE**. If **bHideDetailBox** is **TRUE**, the dialog uses this value to filter the displayed performance counters and objects. You can specify one of the following values:

Detail level	Meaning
PERF_DETAIL_NOVICE	A novice user can understand the counter data.
PERF_DETAIL_ADVANCED	The counter data is provided for advanced users.
PERF_DETAIL_EXPERT	The counter data is provided for expert users.

szDialogBoxCaption

Pointer to a **null**-terminated string that specifies the optional caption to display in the caption bar of the dialog box. If this member is **NULL**, the caption will be **Browse Performance Counters**.

Remarks

Each time the [Add](#) button is clicked, the **szReturnPathBuffer** buffer contains the selected counter and the **pCallBack** callback function is called. The callback function should call the [PdhAddCounter](#) function for each counter in the buffer.

Requirements

Minimum supported client	Windows XP [desktop apps only]
Minimum supported server	Windows Server 2003 [desktop apps only]
Header	pdh.h

See also

[CounterPathCallBack](#)

[PdhAddCounter](#)

[PdhBrowseCounters](#)

Feedback

Was this page helpful?

 Yes

 No

[Get help at Microsoft Q&A](#)

PDH_COUNTER_INFO_A structure (pdh.h)

Article09/01/2022

The PDH_COUNTER_INFO structure contains information describing the properties of a counter. This information also includes the counter path.

Syntax

C++

```
typedef struct _PDH_COUNTER_INFO_A {
    DWORD      dwLength;
    DWORD      dwType;
    DWORD      CVersion;
    DWORD      CStatus;
    LONG       lScale;
    LONG       lDefaultScale;
    DWORD_PTR  dwUserData;
    DWORD_PTR  dwQueryUserData;
    LPSTR      szFullPath;
    union {
        PDH_DATA_ITEM_PATH_ELEMENTS_A DataItemPath;
        PDH_COUNTER_PATH_ELEMENTS_A   CounterPath;
        struct {
            LPSTR szMachineName;
            LPSTR szObjectName;
            LPSTR szInstanceName;
            LPSTR szParentInstance;
            DWORD dwInstanceIndex;
            LPSTR szCounterName;
        };
    };
    LPSTR      szExplainText;
    DWORD      DataBuffer[1];
} PDH_COUNTER_INFO_A, *PPDH_COUNTER_INFO_A;
```

Members

dwLength

Size of the structure, including the appended strings, in bytes.

dwType

Counter type. For a list of counter types, see the Counter Types section of the [Windows Server 2003 Deployment Kit](#). The counter type constants are defined in Winperf.h.

`CVersion`

Counter version information. Not used.

`CStatus`

Counter status that indicates if the counter value is valid. For a list of possible values, see [Checking PDH Interface Return Values](#).

`lScale`

Scale factor to use when computing the displayable value of the counter. The scale factor is a power of ten. The valid range of this parameter is PDH_MIN_SCALE (-7) (the returned value is the actual value times 10^{-7}) to PDH_MAX_SCALE (+7) (the returned value is the actual value times 10^{+7}). A value of zero will set the scale to one, so that the actual value is returned

`lDefaultScale`

Default scale factor as suggested by the counter's provider.

`dwUserData`

The value passed in the *dwUserData* parameter when calling [PdhAddCounter](#).

`dwQueryUserData`

The value passed in the *dwUserData* parameter when calling [PdhOpenQuery](#).

`szFullPath`

Null-terminated string that specifies the full counter path. The string follows this structure in memory.

`DataItemPath`

A [PDH_DATA_ITEM_PATH_ELEMENTS](#) structure. Not used.

`CounterPath`

A [PDH_COUNTER_PATH_ELEMENTS](#) structure.

`szMachineName`

Null-terminated string that contains the name of the computer specified in the counter path. Is **NULL**, if the path does not specify a computer. The string follows this structure in memory.

`szObjectName`

Null-terminated string that contains the name of the performance object specified in the counter path. The string follows this structure in memory.

`szInstanceName`

Null-terminated string that contains the name of the object instance specified in the counter path. Is **NULL**, if the path does not specify an instance. The string follows this structure in memory.

`szParentInstance`

Null-terminated string that contains the name of the parent instance specified in the counter path. Is **NULL**, if the path does not specify a parent instance. The string follows this structure in memory.

`dwInstanceIndex`

Instance index specified in the counter path. Is 0, if the path does not specify an instance index.

`szCounterName`

Null-terminated string that contains the counter name. The string follows this structure in memory.

`szExplainText`

Help text that describes the counter. Is **NULL** if the source is a log file.

`DataBuffer[1]`

Start of the string data that is appended to the structure.

Remarks

When you allocate memory for this structure, allocate enough memory for the member strings, such as `szCounterName`, that are appended to the end of this structure.

Requirements

Minimum supported client	Windows XP [desktop apps only]
Minimum supported server	Windows Server 2003 [desktop apps only]
Header	pdh.h

See also

[PDH_COUNTER_PATH_ELEMENTS](#)

[PDH_DATA_ITEM_PATH_ELEMENTS](#)

[PdhGetCounterInfo](#)

Feedback

Was this page helpful?

 Yes

 No

[Get help at Microsoft Q&A](#)

PDH_COUNTER_INFO_W structure (pdh.h)

Article09/01/2022

The PDH_COUNTER_INFO structure contains information describing the properties of a counter. This information also includes the counter path.

Syntax

C++

```
typedef struct _PDH_COUNTER_INFO_W {
    DWORD      dwLength;
    DWORD      dwType;
    DWORD      CVersion;
    DWORD      CStatus;
    LONG       lScale;
    LONG       lDefaultScale;
    DWORD_PTR  dwUserData;
    DWORD_PTR  dwQueryUserData;
    LPWSTR     szFullPath;
    union {
        PDH_DATA_ITEM_PATH_ELEMENTS_W DataItemPath;
        PDH_COUNTER_PATH_ELEMENTS_W   CounterPath;
        struct {
            LPWSTR szMachineName;
            LPWSTR szObjectName;
            LPWSTR szInstanceName;
            LPWSTR szParentInstance;
            DWORD   dwInstanceIndex;
            LPWSTR szCounterName;
        };
    };
    LPWSTR     szExplainText;
    DWORD      DataBuffer[1];
} PDH_COUNTER_INFO_W, *PPDH_COUNTER_INFO_W;
```

Members

dwLength

Size of the structure, including the appended strings, in bytes.

dwType

Counter type. For a list of counter types, see the Counter Types section of the [Windows Server 2003 Deployment Kit](#). The counter type constants are defined in Winperf.h.

`CVersion`

Counter version information. Not used.

`CStatus`

Counter status that indicates if the counter value is valid. For a list of possible values, see [Checking PDH Interface Return Values](#).

`lScale`

Scale factor to use when computing the displayable value of the counter. The scale factor is a power of ten. The valid range of this parameter is PDH_MIN_SCALE (-7) (the returned value is the actual value times 10^{-7}) to PDH_MAX_SCALE (+7) (the returned value is the actual value times 10^{+7}). A value of zero will set the scale to one, so that the actual value is returned

`lDefaultScale`

Default scale factor as suggested by the counter's provider.

`dwUserData`

The value passed in the *dwUserData* parameter when calling [PdhAddCounter](#).

`dwQueryUserData`

The value passed in the *dwUserData* parameter when calling [PdhOpenQuery](#).

`szFullPath`

Null-terminated string that specifies the full counter path. The string follows this structure in memory.

`DataItemPath`

A [PDH_DATA_ITEM_PATH_ELEMENTS](#) structure. Not used.

`CounterPath`

A [PDH_COUNTER_PATH_ELEMENTS](#) structure.

`szMachineName`

Null-terminated string that contains the name of the computer specified in the counter path. Is **NULL**, if the path does not specify a computer. The string follows this structure in memory.

`szObjectName`

Null-terminated string that contains the name of the performance object specified in the counter path. The string follows this structure in memory.

`szInstanceName`

Null-terminated string that contains the name of the object instance specified in the counter path. Is **NULL**, if the path does not specify an instance. The string follows this structure in memory.

`szParentInstance`

Null-terminated string that contains the name of the parent instance specified in the counter path. Is **NULL**, if the path does not specify a parent instance. The string follows this structure in memory.

`dwInstanceIndex`

Instance index specified in the counter path. Is 0, if the path does not specify an instance index.

`szCounterName`

Null-terminated string that contains the counter name. The string follows this structure in memory.

`szExplainText`

Help text that describes the counter. Is **NULL** if the source is a log file.

`DataBuffer[1]`

Start of the string data that is appended to the structure.

Remarks

When you allocate memory for this structure, allocate enough memory for the member strings, such as `szCounterName`, that are appended to the end of this structure.

Requirements

Minimum supported client	Windows XP [desktop apps only]
Minimum supported server	Windows Server 2003 [desktop apps only]
Header	pdh.h

See also

[PDH_COUNTER_PATH_ELEMENTS](#)

[PDH_DATA_ITEM_PATH_ELEMENTS](#)

[PdhGetCounterInfo](#)

Feedback

Was this page helpful?

 Yes

 No

[Get help at Microsoft Q&A](#)

PDH_COUNTER_PATH_ELEMENTS_A structure (pdh.h)

Article07/27/2022

The **PDH_COUNTER_PATH_ELEMENTS** structure contains the components of a counter path.

Syntax

C++

```
typedef struct _PDH_COUNTER_PATH_ELEMENTS_A {
    LPSTR szMachineName;
    LPSTR szObjectName;
    LPSTR szInstanceName;
    LPSTR szParentInstance;
    DWORD dwInstanceIndex;
    LPSTR szCounterName;
} PDH_COUNTER_PATH_ELEMENTS_A, *PPDH_COUNTER_PATH_ELEMENTS_A;
```

Members

szMachineName

Pointer to a null-terminated string that specifies the computer name.

szObjectName

Pointer to a null-terminated string that specifies the object name.

szInstanceName

Pointer to a null-terminated string that specifies the instance name. Can contain a wildcard character.

szParentInstance

Pointer to a null-terminated string that specifies the parent instance name. Can contain a wildcard character.

dwInstanceIndex

Index used to uniquely identify duplicate instance names.

szCounterName

Pointer to a null-terminated string that specifies the counter name.

Remarks

This structure is used by [PdhMakeCounterPath](#) to create a counter path and by [PdhParseCounterPath](#) to parse a counter path.

When you allocate memory for this structure, allocate enough memory for the member strings, such as **szCounterName**, that are appended to the end of this structure.

Requirements

Minimum supported client	Windows XP [desktop apps only]
Minimum supported server	Windows Server 2003 [desktop apps only]
Header	pdh.h

See also

[PdhMakeCounterPath](#)

[PdhParseCounterPath](#)

Feedback

Was this page helpful?

 Yes

 No

[Get help at Microsoft Q&A](#)

PDH_COUNTER_PATH_ELEMENTS_W structure (pdh.h)

Article07/27/2022

The **PDH_COUNTER_PATH_ELEMENTS** structure contains the components of a counter path.

Syntax

C++

```
typedef struct _PDH_COUNTER_PATH_ELEMENTS_W {
    LPWSTR szMachineName;
    LPWSTR szObjectName;
    LPWSTR szInstanceName;
    LPWSTR szParentInstance;
    DWORD   dwInstanceIndex;
    LPWSTR szCounterName;
} PDH_COUNTER_PATH_ELEMENTS_W, *PPDH_COUNTER_PATH_ELEMENTS_W;
```

Members

szMachineName

Pointer to a null-terminated string that specifies the computer name.

szObjectName

Pointer to a null-terminated string that specifies the object name.

szInstanceName

Pointer to a null-terminated string that specifies the instance name. Can contain a wildcard character.

szParentInstance

Pointer to a null-terminated string that specifies the parent instance name. Can contain a wildcard character.

dwInstanceIndex

Index used to uniquely identify duplicate instance names.

szCounterName

Pointer to a null-terminated string that specifies the counter name.

Remarks

This structure is used by [PdhMakeCounterPath](#) to create a counter path and by [PdhParseCounterPath](#) to parse a counter path.

When you allocate memory for this structure, allocate enough memory for the member strings, such as **szCounterName**, that are appended to the end of this structure.

Requirements

Minimum supported client	Windows XP [desktop apps only]
Minimum supported server	Windows Server 2003 [desktop apps only]
Header	pdh.h

See also

[PdhMakeCounterPath](#)

[PdhParseCounterPath](#)

Feedback

Was this page helpful?

 Yes

 No

[Get help at Microsoft Q&A](#)

PDH_DATA_ITEM_PATH_ELEMENTS_A structure (pdh.h)

Article07/27/2022

The **PDH_DATA_ITEM_PATH_ELEMENTS** structure contains the path elements of a specific data item.

Syntax

C++

```
typedef struct _PDH_DATA_ITEM_PATH_ELEMENTS_A {
    LPSTR szMachineName;
    GUID ObjectGUID;
    DWORD dwItemId;
    LPSTR szInstanceName;
} PDH_DATA_ITEM_PATH_ELEMENTS_A, *PPDH_DATA_ITEM_PATH_ELEMENTS_A;
```

Members

szMachineName

Pointer to a null-terminated string that specifies the name of the computer where the data item resides.

ObjectGUID

GUID of the object where the data item resides.

dwItemId

Identifier of the data item.

szInstanceName

Pointer to a null-terminated string that specifies the name of the data item instance.

Requirements

Minimum supported client	Windows XP [desktop apps only]
Minimum supported server	Windows Server 2003 [desktop apps only]
Header	pdh.h

See also

[PDH_COUNTER_INFO](#)

Feedback

Was this page helpful?



[Get help at Microsoft Q&A](#)

PDH_DATA_ITEM_PATH_ELEMENTS_W structure (pdh.h)

Article07/27/2022

The **PDH_DATA_ITEM_PATH_ELEMENTS** structure contains the path elements of a specific data item.

Syntax

C++

```
typedef struct _PDH_DATA_ITEM_PATH_ELEMENTS_W {
    LPWSTR szMachineName;
    GUID    ObjectGUID;
    DWORD   dwItemId;
    LPWSTR  szInstanceName;
} PDH_DATA_ITEM_PATH_ELEMENTS_W, *PPDH_DATA_ITEM_PATH_ELEMENTS_W;
```

Members

szMachineName

Pointer to a null-terminated string that specifies the name of the computer where the data item resides.

ObjectGUID

GUID of the object where the data item resides.

dwItemId

Identifier of the data item.

szInstanceName

Pointer to a null-terminated string that specifies the name of the data item instance.

Requirements

Minimum supported client	Windows XP [desktop apps only]
Minimum supported server	Windows Server 2003 [desktop apps only]
Header	pdh.h

See also

[PDH_COUNTER_INFO](#)

Feedback

Was this page helpful?



[Get help at Microsoft Q&A](#)

PDH_FMT_COUNTERVALUE structure (pdh.h)

Article 10/05/2021

The **PDH_FMT_COUNTERVALUE** structure contains the computed value of the counter and its status.

Syntax

C++

```
typedef struct _PDH_FMT_COUNTERVALUE {
    DWORD CStatus;
    union {
        LONG    longValue;
        double  doubleValue;
        LONGLONG largeValue;
        LPCSTR   AnsiStringValue;
        LPCWSTR  WideStringValue;
    };
} PDH_FMT_COUNTERVALUE, *PPDH_FMT_COUNTERVALUE;
```

Members

CStatus

Counter status that indicates if the counter value is valid. Check this member before using the data in a calculation or displaying its value. For a list of possible values, see [Checking PDH Interface Return Values](#).

longValue

The computed counter value as a **LONG**.

doubleValue

The computed counter value as a **DOUBLE**.

largeValue

The computed counter value as a **LONGLONG**.

AnsiStringValue

The computed counter value as a **LPCSTR**. Not supported.

`WideStringValue`

The computed counter value as a **LPCWSTR**. Not supported.

Remarks

You specify the data type of the computed counter value when you call [PdhGetFormattedCounterValue](#) or [PdhCalculateCounterFromRawValue](#) to compute the counter's value.

Requirements

Minimum supported client	Windows XP [desktop apps only]
Minimum supported server	Windows Server 2003 [desktop apps only]
Header	pdh.h

See also

[PdhCalculateCounterFromRawValue](#)

[PdhGetFormattedCounterValue](#)

Feedback

Was this page helpful?

 Yes

 No

[Get help at Microsoft Q&A](#)

PDH_FMT_COUNTERVALUE_ITEM_A structure (pdh.h)

Article07/27/2022

The **PDH_FMT_COUNTERVALUE_ITEM** structure contains the instance name and formatted value of a counter.

Syntax

C++

```
typedef struct _PDH_FMT_COUNTERVALUE_ITEM_A {
    LPSTR             szName;
    PDH_FMT_COUNTERVALUE FmtValue;
} PDH_FMT_COUNTERVALUE_ITEM_A, *PPDH_FMT_COUNTERVALUE_ITEM_A;
```

Members

szName

Pointer to a null-terminated string that specifies the instance name of the counter. The string is appended to the end of this structure.

FmtValue

A [PDH_FMT_COUNTERVALUE](#) structure that contains the counter value of the instance.

Requirements

Minimum supported client	Windows XP [desktop apps only]
Minimum supported server	Windows Server 2003 [desktop apps only]
Header	pdh.h

See also

[PDH_FMT_COUNTERVALUE](#)

Feedback

Was this page helpful?

 Yes

 No

Get help at Microsoft Q&A

PDH_FMT_COUNTERVALUE_ITEM_W structure (pdh.h)

Article07/27/2022

The **PDH_FMT_COUNTERVALUE_ITEM** structure contains the instance name and formatted value of a counter.

Syntax

C++

```
typedef struct _PDH_FMT_COUNTERVALUE_ITEM_W {
    LPWSTR             szName;
    PDH_FMT_COUNTERVALUE FmtValue;
} PDH_FMT_COUNTERVALUE_ITEM_W, *PPDH_FMT_COUNTERVALUE_ITEM_W;
```

Members

szName

Pointer to a null-terminated string that specifies the instance name of the counter. The string is appended to the end of this structure.

FmtValue

A [PDH_FMT_COUNTERVALUE](#) structure that contains the counter value of the instance.

Requirements

Minimum supported client	Windows XP [desktop apps only]
Minimum supported server	Windows Server 2003 [desktop apps only]
Header	pdh.h

See also

[PDH_FMT_COUNTERVALUE](#)

Feedback

Was this page helpful?

 Yes

 No

Get help at Microsoft Q&A

PDH_RAW_COUNTER structure (pdh.h)

Article10/05/2021

The PDH_RAW_COUNTER structure returns the data as it was collected from the counter provider. No translation, formatting, or other interpretation is performed on the data.

Syntax

C++

```
typedef struct _PDH_RAW_COUNTER {
    DWORD    CStatus;
    FILETIME TimeStamp;
    LONGLONG FirstValue;
    LONGLONG SecondValue;
    DWORD    MultiCount;
} PDH_RAW_COUNTER, *PPDH_RAW_COUNTER;
```

Members

CStatus

Counter status that indicates if the counter value is valid. Check this member before using the data in a calculation or displaying its value. For a list of possible values, see [Checking PDH Interface Return Values](#).

TimeStamp

Local time for when the data was collected, in [FILETIME](#) format.

FirstValue

First raw counter value.

SecondValue

Second raw counter value. Rate counters require two values in order to compute a displayable value.

MultiCount

If the counter type contains the [PERF_MULTI_COUNTER](#) flag, this member contains the additional counter data used in the calculation. For example, the

PERF_100NSEC_MULTI_TIMER counter type contains the PERF_MULTI_COUNTER flag.

Requirements

Minimum supported client	Windows XP [desktop apps only]
Minimum supported server	Windows Server 2003 [desktop apps only]
Header	pdh.h

See also

[PdhCalculateCounterFromRawValue](#)

[PdhComputeCounterStatistics](#)

[PdhGetRawCounterValue](#)

Feedback

Was this page helpful?

 Yes

 No

[Get help at Microsoft Q&A](#)

PDH_RAW_COUNTER_ITEM_A structure (pdh.h)

Article07/27/2022

The **PDH_RAW_COUNTER_ITEM** structure contains the instance name and raw value of a counter.

Syntax

C++

```
typedef struct _PDH_RAW_COUNTER_ITEM_A {
    LPSTR          szName;
    PDH_RAW_COUNTER RawValue;
} PDH_RAW_COUNTER_ITEM_A, *PPDH_RAW_COUNTER_ITEM_A;
```

Members

szName

Pointer to a null-terminated string that specifies the instance name of the counter. The string is appended to the end of this structure.

RawValue

A [PDH_RAW_COUNTER](#) structure that contains the raw counter value of the instance.

Requirements

Minimum supported client	Windows XP [desktop apps only]
Minimum supported server	Windows Server 2003 [desktop apps only]
Header	pdh.h

See also

[PDH_RAW_COUNTER](#)

Feedback

Was this page helpful?

 Yes

 No

Get help at Microsoft Q&A

PDH_RAW_COUNTER_ITEM_W structure (pdh.h)

Article07/27/2022

The **PDH_RAW_COUNTER_ITEM** structure contains the instance name and raw value of a counter.

Syntax

C++

```
typedef struct _PDH_RAW_COUNTER_ITEM_W {
    LPWSTR          szName;
    PDH_RAW_COUNTER RawValue;
} PDH_RAW_COUNTER_ITEM_W, *PPDH_RAW_COUNTER_ITEM_W;
```

Members

szName

Pointer to a null-terminated string that specifies the instance name of the counter. The string is appended to the end of this structure.

RawValue

A [PDH_RAW_COUNTER](#) structure that contains the raw counter value of the instance.

Requirements

Minimum supported client	Windows XP [desktop apps only]
Minimum supported server	Windows Server 2003 [desktop apps only]
Header	pdh.h

See also

[PDH_RAW_COUNTER](#)

Feedback

Was this page helpful?

 Yes

 No

Get help at Microsoft Q&A

PDH_RAW_LOG_RECORD structure (pdh.h)

Article09/01/2022

The **PDH_RAW_LOG_RECORD** structure contains information about a binary trace log file record.

Syntax

C++

```
typedef struct _PDH_RAW_LOG_RECORD {
    DWORD dwStructureSize;
    DWORD dwRecordType;
    DWORD dwItems;
    UCHAR RawBytes[1];
} PDH_RAW_LOG_RECORD, *PPDH_RAW_LOG_RECORD;
```

Members

`dwStructureSize`

Size of this structure, in bytes. The size includes this structure and the **RawBytes** appended to the end of this structure.

`dwRecordType`

Type of record. This member can be one of the following values.

Value	Meaning
<code>PDH_LOG_TYPE_BINARY</code>	A binary trace format record
<code>PDH_LOG_TYPE_CSV</code>	A comma-separated-value format record
<code>PDH_LOG_TYPE_PERFMON</code>	A Perfmon format record
<code>PDH_LOG_TYPE_SQL</code>	A SQL format record
<code>PDH_LOG_TYPE_TSV</code>	A tab-separated-value format record

`dwItems`

Size of the RawBytes data.

RawBytes[1]

Binary record.

Requirements

Minimum supported client	Windows XP [desktop apps only]
Minimum supported server	Windows Server 2003 [desktop apps only]
Header	pdh.h

See also

[PdhReadRawLogRecord](#)

Feedback

Was this page helpful?

 Yes

 No

[Get help at Microsoft Q&A](#)

PDH_STATISTICS structure (pdh.h)

Article 10/05/2021

The PDH_STATISTICS structure contains the minimum, maximum, and mean values for an array of raw counters values.

Syntax

C++

```
typedef struct _PDH_STATISTICS {
    DWORD             dwFormat;
    DWORD             count;
    PDH_FMT_COUNTERVALUE min;
    PDH_FMT_COUNTERVALUE max;
    PDH_FMT_COUNTERVALUE mean;
} PDH_STATISTICS, *PPDH_STATISTICS;
```

Members

`dwFormat`

Format of the data. The format is specified in the *dwFormat* when calling [PdhComputeCounterStatistics](#).

`count`

Number of values in the array.

`min`

Minimum of the values.

`max`

Maximum of the values.

`mean`

Mean of the values.

Requirements

Minimum supported client	Windows XP [desktop apps only]
Minimum supported server	Windows Server 2003 [desktop apps only]
Header	pdh.h

See also

[PdhComputeCounterStatistics](#)

Feedback

Was this page helpful?



[Get help at Microsoft Q&A](#)

PDH_TIME_INFO structure (pdh.h)

Article 10/05/2021

The PDH_TIME_INFO structure contains information on time intervals as applied to the sampling of performance data.

Syntax

C++

```
typedef struct _PDH_TIME_INFO {
    LONGLONG StartTime;
    LONGLONG EndTime;
    DWORD     SampleCount;
} PDH_TIME_INFO, *PPDH_TIME_INFO;
```

Members

StartTime

Starting time of the sample interval, in local [FILETIME](#) format.

EndTime

Ending time of the sample interval, in local [FILETIME](#) format.

SampleCount

Number of samples collected during the interval.

Requirements

Minimum supported client	Windows XP [desktop apps only]
Minimum supported server	Windows Server 2003 [desktop apps only]
Header	pdh.h

See also

Feedback

Was this page helpful?

 Yes

 No

Get help at Microsoft Q&A

PdhAddCounterA function (pdh.h)

Article 02/09/2023

Adds the specified counter to the query.

Syntax

C++

```
PDH_FUNCTION PdhAddCounterA(
    [in] PDH_HQUERY hQuery,
    [in] LPCSTR     szFullCounterPath,
    [in] DWORD_PTR   dwUserData,
    [out] PDH_HCOUNTER *phCounter
);
```

Parameters

[in] hQuery

Handle to the query to which you want to add the counter. This handle is returned by the [PdhOpenQuery](#) function.

[in] szFullCounterPath

Null-terminated string that contains the counter path. For details on the format of a counter path, see [Specifying a Counter Path](#). The maximum length of a counter path is PDH_MAX_COUNTER_PATH.

[in] dwUserData

User-defined value. This value becomes part of the counter information. To retrieve this value later, call the [PdhGetCounterInfo](#) function and access the dwUserData member of the [PDH_COUNTER_INFO](#) structure.

[out] phCounter

Handle to the counter that was added to the query. You may need to reference this handle in subsequent calls.

Return value

Return ERROR_SUCCESS if the function succeeds.

If the function fails, the return value is a [system error code](#) or a [PDH error code](#). The following are possible values.

 [Expand table](#)

Return code	Description
PDH_CSTATUS_BAD_COUNTERNAME	The counter path could not be parsed or interpreted.
PDH_CSTATUS_NO_COUNTER	Unable to find the specified counter on the computer or in the log file.
PDH_CSTATUS_NO_COUNTERNAME	The counter path is empty.
PDH_CSTATUS_NO_MACHINE	The path did not contain a computer name, and the function was unable to retrieve the local computer name.
PDH_CSTATUS_NO_OBJECT	Unable to find the specified object on the computer or in the log file.
PDH_FUNCTION_NOT_FOUND	Unable to determine the calculation function to use for this counter.
PDH_INVALID_ARGUMENT	One or more arguments are not valid.
PDH_INVALID_HANDLE	The query handle is not valid.
PDH_MEMORY_ALLOCATION_FAILURE	Unable to allocate memory required to complete the function.

Remarks

If the counter path contains a wildcard character, all counter names matching the wildcard character are added to the query.

If a counter instance is specified that does not yet exist, **PdhAddCounter** does not report an error condition. Instead, it returns ERROR_SUCCESS. The reason for this behavior is that it is not known whether a nonexistent counter instance has been specified or whether one will exist but has not yet been created.

To remove the counter from the query, use the [PdhRemoveCounter](#) function.

Examples

For an example, see [Browsing Performance Counters](#) or [Reading Performance Data from a Log File](#).

 **Note**

The pdh.h header defines PdhAddCounter 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 is 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	pdh.h
Library	Pdh.lib
DLL	Pdh.dll

See also

[PdhAddEnglishCounter](#)

[PdhBrowseCounters](#)

[PdhMakeCounterPath](#)

[PdhOpenQuery](#)

[PdhRemoveCounter](#)

Feedback

Was this page helpful?

 Yes

 No

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

PdhAddCounterW function (pdh.h)

Article 02/09/2023

Adds the specified counter to the query.

Syntax

C++

```
PDH_FUNCTION PdhAddCounterW(
    [in] PDH_HQUERY hQuery,
    [in] LPCWSTR szFullCounterPath,
    [in] DWORD_PTR dwUserData,
    [out] PDH_HCOUNTER *phCounter
);
```

Parameters

[in] hQuery

Handle to the query to which you want to add the counter. This handle is returned by the [PdhOpenQuery](#) function.

[in] szFullCounterPath

Null-terminated string that contains the counter path. For details on the format of a counter path, see [Specifying a Counter Path](#). The maximum length of a counter path is PDH_MAX_COUNTER_PATH.

[in] dwUserData

User-defined value. This value becomes part of the counter information. To retrieve this value later, call the [PdhGetCounterInfo](#) function and access the dwUserData member of the [PDH_COUNTER_INFO](#) structure.

[out] phCounter

Handle to the counter that was added to the query. You may need to reference this handle in subsequent calls.

Return value

Return ERROR_SUCCESS if the function succeeds.

If the function fails, the return value is a [system error code](#) or a [PDH error code](#). The following are possible values.

Return code	Description
PDH_CSTATUS_BAD_COUNTERNAME	The counter path could not be parsed or interpreted.
PDH_CSTATUS_NO_COUNTER	Unable to find the specified counter on the computer or in the log file.
PDH_CSTATUS_NO_COUNTERNAME	The counter path is empty.
PDH_CSTATUS_NO_MACHINE	The path did not contain a computer name, and the function was unable to retrieve the local computer name.
PDH_CSTATUS_NO_OBJECT	Unable to find the specified object on the computer or in the log file.
PDH_FUNCTION_NOT_FOUND	Unable to determine the calculation function to use for this counter.
PDH_INVALID_ARGUMENT	One or more arguments are not valid.
PDH_INVALID_HANDLE	The query handle is not valid.
PDH_MEMORY_ALLOCATION_FAILURE	Unable to allocate memory required to complete the function.

Remarks

If the counter path contains a wildcard character, all counter names matching the wildcard character are added to the query.

If a counter instance is specified that does not yet exist, [PdhAddCounter](#) does not report an error condition. Instead, it returns ERROR_SUCCESS. The reason for this behavior is that it is not known whether a nonexistent counter instance has been specified or whether one will exist but has not yet been created.

To remove the counter from the query, use the [PdhRemoveCounter](#) function.

Examples

For an example, see [Browsing Performance Counters](#) or [Reading Performance Data from a Log File](#).

Note

The pdh.h header defines PdhAddCounter 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 is not encoding-neutral can lead to mismatches that result in compilation or runtime errors. For more information, see [Conventions for Function Prototypes](#).

Requirements

Minimum supported client	Windows XP [desktop apps only]
Minimum supported server	Windows Server 2003 [desktop apps only]
Target Platform	Windows
Header	pdh.h
Library	Pdh.lib
DLL	Pdh.dll

See also

[PdhAddEnglishCounter](#)

[PdhBrowseCounters](#)

[PdhMakeCounterPath](#)

[PdhOpenQuery](#)

[PdhRemoveCounter](#)

Feedback

Was this page helpful?

 Yes

 No

[Get help at Microsoft Q&A](#)

PdhAddEnglishCounterA function (pdh.h)

Article 02/09/2023

Adds the specified language-neutral counter to the query.

Syntax

C++

```
PDH_FUNCTION PdhAddEnglishCounterA(
    [in] PDH_HQUERY hQuery,
    [in] LPCSTR szFullCounterPath,
    [in] DWORD_PTR dwUserData,
    [out] PDH_HCOUNTER *phCounter
);
```

Parameters

[in] hQuery

Handle to the query to which you want to add the counter. This handle is returned by the [PdhOpenQuery](#) function.

[in] szFullCounterPath

Null-terminated string that contains the counter path. For details on the format of a counter path, see [Specifying a Counter Path](#). The maximum length of a counter path is PDH_MAX_COUNTER_PATH.

[in] dwUserData

User-defined value. This value becomes part of the counter information. To retrieve this value later, call the [PdhGetCounterInfo](#) function and access the dwQueryUserData member of the [PDH_COUNTER_INFO](#) structure.

[out] phCounter

Handle to the counter that was added to the query. You may need to reference this handle in subsequent calls.

Return value

Return ERROR_SUCCESS if the function succeeds.

If the function fails, the return value is a [system error code](#) or a [PDH error code](#). The following are possible values.

 [Expand table](#)

Return code	Description
PDH_CSTATUS_BAD_COUNTERNAME	The counter path could not be parsed or interpreted.
PDH_CSTATUS_NO_COUNTER	Unable to find the specified counter on the computer or in the log file.
PDH_CSTATUS_NO_COUNTERNAME	The counter path is empty.
PDH_CSTATUS_NO_MACHINE	The path did not contain a computer name and the function was unable to retrieve the local computer name.
PDH_CSTATUS_NO_OBJECT	Unable to find the specified object on the computer or in the log file.
PDH_FUNCTION_NOT_FOUND	Unable to determine the calculation function to use for this counter.
PDH_INVALID_ARGUMENT	One or more arguments are not valid.
PDH_INVALID_HANDLE	The query handle is not valid.
PDH_MEMORY_ALLOCATION_FAILURE	Unable to allocate memory required to complete the function.

Remarks

This function provides a language-neutral way to add performance counters to the query. In contrast, the counter path that you specify in the [PdhAddCounter](#) function must be localized.

If a counter instance is specified that does not yet exist, [PdhAddEnglishCounter](#) does not report an error condition. Instead, it returns ERROR_SUCCESS. The reason for this behavior is that it is not known whether a nonexistent counter instance has been specified or whether one will exist but has not yet been created.

To remove the counter from the query, use the [PdhRemoveCounter](#) function.

Note If the counter path contains a wildcard character, the non-wildcard portions of the path will be localized, but wildcards will not be expanded before adding the localized counter path to the query. In this case, you will need to use the following procedure to add all matching counter names to the query.

1. Make a query
2. Use **PdhAddEnglishCounter** with the string containing wildcards
3. Use **PdhGetCounterInfo** on the counter handle returned by **PdhAddEnglishCounter** to get a localized full path (*szFullPath*.) This string still contains wildcards, but the non-wildcard parts are now localized.
4. Use **PdhExpandWildCardPath** to expand the wildcards.
5. Use **PdhAddCounter** on each of the resulting paths

 **Note**

The pdh.h header defines **PdhAddEnglishCounter** 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 is 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 Vista [desktop apps only]
Minimum supported server	Windows Server 2008 [desktop apps only]
Target Platform	Windows
Header	pdh.h
Library	Pdh.lib
DLL	Pdh.dll

See also

[PdhAddCounter](#)

[PdhBrowseCounters](#)

[PdhMakeCounterPath](#)

[PdhOpenQuery](#)

[PdhRemoveCounter](#)

Feedback

Was this page helpful?

 Yes

 No

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

PdhAddEnglishCounterW function (pdh.h)

Article 02/09/2023

Adds the specified language-neutral counter to the query.

Syntax

C++

```
PDH_FUNCTION PdhAddEnglishCounterW(
    [in] PDH_HQUERY hQuery,
    [in] LPCWSTR szFullCounterPath,
    [in] DWORD_PTR dwUserData,
    [out] PDH_HCOUNTER *phCounter
);
```

Parameters

[in] hQuery

Handle to the query to which you want to add the counter. This handle is returned by the [PdhOpenQuery](#) function.

[in] szFullCounterPath

Null-terminated string that contains the counter path. For details on the format of a counter path, see [Specifying a Counter Path](#). The maximum length of a counter path is PDH_MAX_COUNTER_PATH.

[in] dwUserData

User-defined value. This value becomes part of the counter information. To retrieve this value later, call the [PdhGetCounterInfo](#) function and access the dwQueryUserData member of the [PDH_COUNTER_INFO](#) structure.

[out] phCounter

Handle to the counter that was added to the query. You may need to reference this handle in subsequent calls.

Return value

Return ERROR_SUCCESS if the function succeeds.

If the function fails, the return value is a [system error code](#) or a [PDH error code](#). The following are possible values.

Return code	Description
PDH_CSTATUS_BAD_COUNTERNAME	The counter path could not be parsed or interpreted.
PDH_CSTATUS_NO_COUNTER	Unable to find the specified counter on the computer or in the log file.
PDH_CSTATUS_NO_COUNTERNAME	The counter path is empty.
PDH_CSTATUS_NO_MACHINE	The path did not contain a computer name and the function was unable to retrieve the local computer name.
PDH_CSTATUS_NO_OBJECT	Unable to find the specified object on the computer or in the log file.
PDH_FUNCTION_NOT_FOUND	Unable to determine the calculation function to use for this counter.
PDH_INVALID_ARGUMENT	One or more arguments are not valid.
PDH_INVALID_HANDLE	The query handle is not valid.
PDH_MEMORY_ALLOCATION_FAILURE	Unable to allocate memory required to complete the function.

Remarks

This function provides a language-neutral way to add performance counters to the query. In contrast, the counter path that you specify in the [PdhAddCounter](#) function must be localized.

If a counter instance is specified that does not yet exist, [PdhAddEnglishCounter](#) does not report an error condition. Instead, it returns ERROR_SUCCESS. The reason for this behavior is that it is not known whether a nonexistent counter instance has been specified or whether one will exist but has not yet been created.

To remove the counter from the query, use the [PdhRemoveCounter](#) function.

Note If the counter path contains a wildcard character, the non-wildcard portions of the path will be localized, but wildcards will not be expanded before adding the localized counter path to the query. In this case, you will need to use the following procedure to add all matching counter names to the query.

1. Make a query
2. Use **PdhAddEnglishCounter** with the string containing wildcards
3. Use **PdhGetCounterInfo** on the counter handle returned by **PdhAddEnglishCounter** to get a localized full path (*szFullPath*.) This string still contains wildcards, but the non-wildcard parts are now localized.
4. Use **PdhExpandWildCardPath** to expand the wildcards.
5. Use **PdhAddCounter** on each of the resulting paths

 **Note**

The pdh.h header defines **PdhAddEnglishCounter** 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 is not encoding-neutral can lead to mismatches that result in compilation or runtime errors. For more information, see [Conventions for Function Prototypes](#).

Requirements

Minimum supported client	Windows Vista [desktop apps only]
Minimum supported server	Windows Server 2008 [desktop apps only]
Target Platform	Windows
Header	pdh.h
Library	Pdh.lib
DLL	Pdh.dll

See also

[PdhAddCounter](#)

[PdhBrowseCounters](#)

[PdhMakeCounterPath](#)

[PdhOpenQuery](#)

[PdhRemoveCounter](#)

Feedback

Was this page helpful?

 Yes

 No

[Get help at Microsoft Q&A](#)

PdhBindInputDataSourceA function (pdh.h)

Article 02/22/2024

Binds one or more binary log files together for reading log data.

Syntax

C++

```
PDH_FUNCTION PdhBindInputDataSourceA(
    [out] PDH_HLOG *phDataSource,
    [in]  LPCSTR   LogFileNameList
);
```

Parameters

[out] phDataSource

Handle to the bound data sources.

[in] LogFileNameList

Null-terminated string that contains one or more binary log files to bind together. Terminate each log file name with a **null**-terminator character and the list with one additional **null**-terminator character. The log file names can contain absolute or relative paths. You cannot specify more than 32 log files.

If **NULL**, the source is a real-time data source.

Return value

Returns **ERROR_SUCCESS** if the function succeeds.

If the function fails, the return value is a [system error code](#) or a [PDH error code](#).

Remarks

This function is used with the PDH functions that require a handle to a data source. For a list of these functions, see [See Also](#).

You cannot specify more than one comma-delimited (CSV) or tab-delimited (TSV) file. The list can contain only one type of file—you cannot combine multiple file types.

To close the bound log files, call the [PdhCloseLog](#) function using the log handle.

 **Note**

The pdh.h header defines PdhBindInputDataSource 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 is 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	pdh.h
Library	Pdh.lib
DLL	Pdh.dll

See also

[PdhBrowseCountersH](#)

[PdhEnumMachinesH](#)

[PdhEnumObjectItemsH](#)

[PdhEnumObjectsH](#)

[PdhExpandWildCardPathH](#)

[PdhGetDataSourceTimeRangeH](#)

[PdhGetDefaultPerfCounterH](#)

[PdhGetDefaultPerfObjectH](#)

[PdhOpenQueryH](#)

Feedback

Was this page helpful?

 Yes

 No

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

PdhBindInputDataSourceW function (pdh.h)

Article 02/09/2023

Binds one or more binary log files together for reading log data.

Syntax

C++

```
PDH_FUNCTION PdhBindInputDataSourceW(
    [out] PDH_HLOG *phDataSource,
    [in]  LPCWSTR  LogFileNameList
);
```

Parameters

[out] phDataSource

Handle to the bound data sources.

[in] LogFileNameList

Null-terminated string that contains one or more binary log files to bind together. Terminate each log file name with a **null**-terminator character and the list with one additional **null**-terminator character. The log file names can contain absolute or relative paths. You cannot specify more than 32 log files.

If **NULL**, the source is a real-time data source.

Return value

Returns **ERROR_SUCCESS** if the function succeeds.

If the function fails, the return value is a [system error code](#) or a [PDH error code](#).

Remarks

This function is used with the PDH functions that require a handle to a data source. For a list of these functions, see [See Also](#).

You cannot specify more than one comma-delimited (CSV) or tab-delimited (TSV) file. The list can contain only one type of file—you cannot combine multiple file types.

To close the bound log files, call the [PdhCloseLog](#) function using the log handle.

 **Note**

The pdh.h header defines PdhBindInputDataSource 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 is not encoding-neutral can lead to mismatches that result in compilation or runtime errors. For more information, see [Conventions for Function Prototypes](#).

Requirements

Minimum supported client	Windows XP [desktop apps only]
Minimum supported server	Windows Server 2003 [desktop apps only]
Target Platform	Windows
Header	pdh.h
Library	Pdh.lib
DLL	Pdh.dll

See also

[PdhBrowseCountersH](#)

[PdhEnumMachinesH](#)

[PdhEnumObjectItemsH](#)

[PdhEnumObjectsH](#)

[PdhExpandWildCardPathH](#)

[PdhGetDataSourceTimeRangeH](#)

[PdhGetDefaultPerfCounterH](#)

[PdhGetDefaultPerfObjectH](#)

[PdhOpenQueryH](#)

Feedback

Was this page helpful?

 Yes

 No

[Get help at Microsoft Q&A](#)

PdhBrowseCountersA function (pdh.h)

Article02/22/2024

Displays a **Browse Counters** dialog box that the user can use to select one or more counters that they want to add to the query.

To use handles to data sources, use the [PdhBrowseCountersH](#) function.

Syntax

C++

```
PDH_FUNCTION PdhBrowseCountersA(  
    [in] PPDH_BROWSE_DLG_CONFIG_A pBrowseDlgData  
) ;
```

Parameters

[in] pBrowseDlgData

A [PDH_BROWSE_DLG_CONFIG](#) structure that specifies the behavior of the dialog box.

Return value

If the function succeeds, it returns [ERROR_SUCCESS](#).

If the function fails, the return value is a [system error code](#) or a [PDH error code](#).

Remarks

Note that the dialog box can return [PDH_DIALOG_CANCELLED](#) if [bSingleCounterPerDialog](#) is **FALSE** and the user clicks the **Close** button, so your error handling would have to account for this.

For information on using this function, see [Browsing Counters](#).

Examples

For an example, see [Browsing Performance Counters](#).

ⓘ Note

The pdh.h header defines PdhBrowseCounters 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 is 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	pdh.h
Library	Pdh.lib
DLL	Pdh.dll

See also

[CounterPathCallBack](#)

[PDH_BROWSE_DLG_CONFIG](#)

[PdhBrowseCountersH](#)

Feedback

Was this page helpful?

 Yes

 No

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

PdhBrowseCountersHA function (pdh.h)

Article02/22/2024

Displays a **Browse Counters** dialog box that the user can use to select one or more counters that they want to add to the query.

This function is identical to the [PdhBrowseCounters](#) function, except that it supports the use of handles to data sources.

Syntax

C++

```
PDH_FUNCTION PdhBrowseCountersHA(  
    [in] PPDH_BROWSE_DLG_CONFIG_HA pBrowseDlgData  
) ;
```

Parameters

[in] pBrowseDlgData

A [PDH_BROWSE_DLG_CONFIG_H](#) structure that specifies the behavior of the dialog box.

Return value

If the function succeeds, it returns `ERROR_SUCCESS`.

If the function fails, the return value is a [system error code](#) or a [PDH error code](#).

Remarks

Note that the dialog box can return `PDH_DIALOG_CANCELLED` if `bSingleCounterPerDialog` is `FALSE` and the user clicks the **Close** button, so your error handling would have to account for this.

ⓘ Note

The `pdh.h` header defines `PdhBrowseCountersH` 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 is 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	pdh.h
Library	Pdh.lib
DLL	Pdh.dll

See also

[PDH_BROWSE_DLG_CONFIG_H](#)

[PdhBindInputDataSource](#)

[PdhEnumMachinesH](#)

[PdhEnumObjectItemsH](#)

[PdhEnumObjectsH](#)

[PdhExpandWildCardPathH](#)

[PdhGetDataSourceTimeRangeH](#)

[PdhGetDefaultPerfCounterH](#)

[PdhGetDefaultPerfObjectH](#)

[PdhOpenQueryH](#)

Feedback

Was this page helpful?

 Yes

 No

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

PdhBrowseCountersHW function (pdh.h)

Article 02/09/2023

Displays a **Browse Counters** dialog box that the user can use to select one or more counters that they want to add to the query.

This function is identical to the [PdhBrowseCounters](#) function, except that it supports the use of handles to data sources.

Syntax

C++

```
PDH_FUNCTION PdhBrowseCountersHW(
    [in] PPDH_BROWSE_DLG_CONFIG_HW pBrowseDlgData
);
```

Parameters

[in] `pBrowseDlgData`

A [PDH_BROWSE_DLG_CONFIG_H](#) structure that specifies the behavior of the dialog box.

Return value

If the function succeeds, it returns `ERROR_SUCCESS`.

If the function fails, the return value is a [system error code](#) or a [PDH error code](#).

Remarks

Note that the dialog box can return `PDH_DIALOG_CANCELLED` if `bSingleCounterPerDialog` is `FALSE` and the user clicks the **Close** button, so your error handling would have to account for this.

ⓘ Note

The pdh.h header defines PdhBrowseCountersH 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 is not encoding-neutral can lead to mismatches that result in compilation or runtime errors. For more information, see [Conventions for Function Prototypes](#).

Requirements

Minimum supported client	Windows XP [desktop apps only]
Minimum supported server	Windows Server 2003 [desktop apps only]
Target Platform	Windows
Header	pdh.h
Library	Pdh.lib
DLL	Pdh.dll

See also

[PDH_BROWSE_DLG_CONFIG_H](#)

[PdhBindInputDataSource](#)

[PdhEnumMachinesH](#)

[PdhEnumObjectItemsH](#)

[PdhEnumObjectsH](#)

[PdhExpandWildCardPathH](#)

[PdhGetDataSourceTimeRangeH](#)

[PdhGetDefaultPerfCounterH](#)

[PdhGetDefaultPerfObjectH](#)

[PdhOpenQueryH](#)

Feedback

Was this page helpful?

 Yes

 No

[Get help at Microsoft Q&A](#)

PdhBrowseCountersW function (pdh.h)

Article02/09/2023

Displays a **Browse Counters** dialog box that the user can use to select one or more counters that they want to add to the query.

To use handles to data sources, use the [PdhBrowseCountersH](#) function.

Syntax

C++

```
PDH_FUNCTION PdhBrowseCountersW(
    [in] PPDH_BROWSE_DLG_CONFIG_W pBrowseDlgData
);
```

Parameters

[in] `pBrowseDlgData`

A [PDH_BROWSE_DLG_CONFIG](#) structure that specifies the behavior of the dialog box.

Return value

If the function succeeds, it returns `ERROR_SUCCESS`.

If the function fails, the return value is a [system error code](#) or a [PDH error code](#).

Remarks

Note that the dialog box can return `PDH_DIALOG_CANCELLED` if `bSingleCounterPerDialog` is `FALSE` and the user clicks the **Close** button, so your error handling would have to account for this.

For information on using this function, see [Browsing Counters](#).

Examples

For an example, see [Browsing Performance Counters](#).

Note

The pdh.h header defines PdhBrowseCounters 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 is not encoding-neutral can lead to mismatches that result in compilation or runtime errors. For more information, see [Conventions for Function Prototypes](#).

Requirements

Minimum supported client	Windows XP [desktop apps only]
Minimum supported server	Windows Server 2003 [desktop apps only]
Target Platform	Windows
Header	pdh.h
Library	Pdh.lib
DLL	Pdh.dll

See also

[CounterPathCallBack](#)

[PDH_BROWSE_DLG_CONFIG](#)

[PdhBrowseCountersH](#)

Feedback

Was this page helpful?

 Yes

 No

[Get help at Microsoft Q&A](#)

PdhCalculateCounterFromRawValue function (pdh.h)

Article 10/13/2021

Calculates the displayable value of two raw counter values.

Syntax

C++

```
PDH_FUNCTION PdhCalculateCounterFromRawValue(
    [in] PDH_HCOUNTER           hCounter,
    [in] DWORD                  dwFormat,
    [in] PPDH_RAW_COUNTER       rawValue1,
    [in] PPDH_RAW_COUNTER       rawValue2,
    [out] PPDH_FMT_COUNTERVALUE fmtValue
);
```

Parameters

[in] hCounter

Handle to the counter to calculate. The function uses information from the counter to determine how to calculate the value. This handle is returned by the [PdhAddCounter](#) function.

[in] dwFormat

Determines the data type of the calculated value. Specify one of the following values.

[+] Expand table

Value	Meaning
PDH_FMT_DOUBLE	Return the calculated value as a double-precision floating point real.
PDH_FMT_LARGE	Return the calculated value as a 64-bit integer.
PDH_FMT_LONG	Return the calculated value as a long integer.

You can use the bitwise inclusive OR operator (|) to combine the data type with one of the following scaling factors.

[+] Expand table

Value	Meaning
PDH_FMT_NOSCALE	Do not apply the counter's scaling factor in the calculation.
PDH_FMT_NOCAP100	Counter values greater than 100 (for example, counter values measuring the processor load on multiprocessor computers) will not be reset to 100. The default behavior is that counter values are capped at a value of 100.
PDH_FMT_1000	Multiply the final value by 1,000.

[in] rawValue1

Raw counter value used to compute the displayable counter value. For details, see the [PDH_RAW_COUNTER](#) structure.

[in] rawValue2

Raw counter value used to compute the displayable counter value. For details, see [PDH_RAW_COUNTER](#). Some counters (for example, rate counters) require two raw values to calculate a displayable value. If the counter type does not require a second value, set this parameter to **NULL**. This value must be the older of the two raw values.

[out] fmtValue

A [PDH_FMT_COUNTERVALUE](#) structure that receives the calculated counter value.

Return value

If the function succeeds, it returns **ERROR_SUCCESS**.

If the function fails, the return value is a [system error code](#) or a [PDH error code](#). The following are possible values.

[+] Expand table

Return code	Description
PDH_INVALID_ARGUMENT	An argument is not correct or is incorrectly formatted.

PDH_INVALID_HANDLE

The counter handle is not valid.

Remarks

To retrieve the current raw counter value from the query, call the [PdhGetRawCounterValue](#) function.

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	pdh.h
Library	Pdh.lib
DLL	Pdh.dll

See also

[PDH_FMT_COUNTERVALUE](#)

[PDH_RAW_COUNTER](#)

[PdhGetFormattedCounterValue](#)

[PdhGetRawCounterValue](#)

[PdhSetCounterScaleFactor](#)

Feedback

Was this page helpful?

 Yes

 No

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

PdhCloseLog function (pdh.h)

Article02/22/2024

Closes the specified log file.

Syntax

C++

```
PDH_FUNCTION PdhCloseLog(
    [in] PDH_HLOG hLog,
    [in] DWORD     dwFlags
);
```

Parameters

[in] hLog

Handle to the log file to be closed. This handle is returned by the [PdhOpenLog](#) function.

[in] dwFlags

You can specify the following flag.

 Expand table

Value	Meaning
PDH_FLAGS_CLOSE_QUERY	Closes the query associated with the specified log file handle. See the <i>hQuery</i> parameter of PdhOpenLog .

Return value

If the function succeeds, it returns `ERROR_SUCCESS` and closes and deletes the query.

If the function fails, the return value is a [system error code](#) or a [PDH error code](#). The following is a possible value.

 Expand table

Return code	Description
-------------	-------------

PDH_INVALID_HANDLE

The log file handle is not valid.

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	pdh.h
Library	Pdh.lib
DLL	Pdh.dll

See also

[PdhBindInputDataSource](#)

[PdhOpenLog](#)

Feedback

Was this page helpful?

 Yes

 No

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

PdhCloseQuery function (pdh.h)

Article02/22/2024

Closes all counters contained in the specified query, closes all handles related to the query, and frees all memory associated with the query.

Syntax

C++

```
PDH_FUNCTION PdhCloseQuery(  
    [in] PDH_HQUERY hQuery  
);
```

Parameters

[in] hQuery

Handle to the query to close. This handle is returned by the [PdhOpenQuery](#) function.

Return value

If the function succeeds, it returns ERROR_SUCCESS. Otherwise, the function returns a [system error code](#) or a [PDH error code](#).

The following is a possible value.

[+] Expand table

Return code	Description
PDH_INVALID_HANDLE	The query handle is not valid.

Remarks

Do not use the counter handles associated with this query after calling this function.

The following shows the syntax if calling this function from Visual Basic.

syntax

```
PdhCloseQuery(  
    ByVal QueryHandle as Long  
)  
as Long
```

Examples

For an example, see [Browsing Performance Counters](#) or [Reading Performance Data from a Log File](#).

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	pdh.h
Library	Pdh.lib
DLL	Pdh.dll

See also

[PdhOpenQuery](#)

Feedback

Was this page helpful?

 Yes

 No

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

PdhCollectQueryData function (pdh.h)

Article07/27/2022

Collects the current raw data value for all counters in the specified query and updates the status code of each counter.

Syntax

C++

```
PDH_FUNCTION PdhCollectQueryData(
    [in, out] PDH_HQUERY hQuery
);
```

Parameters

[in, out] hQuery

Handle of the query for which you want to collect data. The [PdhOpenQuery](#) function returns this handle.

Return value

If the function succeeds, it returns ERROR_SUCCESS. Otherwise, the function returns a [system error code](#) or a [PDH error code](#).

The following are possible values.

[] Expand table

Return code	Description
PDH_INVALID_HANDLE	The query handle is not valid.
PDH_NO_DATA	The query does not currently contain any counters. The query may not contain data because the user is not running with an elevated token (see Limited User Access Support).

Remarks

Call this function when you want to collect counter data for the counters in the query. PDH stores the raw counter values for the current and previous collection.

If you want to retrieve the current raw counter value, call the [PdhGetRawCounterValue](#) function. If you want to compute a displayable value for the counter value, call the [PdhGetFormattedCounterValue](#) function. If the counter path contains a wildcard for the instance name, instead call the [PdhGetRawCounterArray](#) and [PdhGetFormattedCounterArray](#) functions, respectively.

When [PdhCollectQueryData](#) is called for data from one counter instance only and the counter instance does not exist, the function returns PDH_NO_DATA. However, if data from more than one counter is queried, [PdhCollectQueryData](#) may return ERROR_SUCCESS even if one of the counter instances does not yet exist. This is because it is not known if the specified counter instance does not exist, or if it will exist but has not yet been created. In this case, call [PdhGetRawCounterValue](#) or [PdhGetFormattedCounterValue](#) for each of the counter instances of interest to determine whether they exist.

The following shows the syntax if calling this function from Visual Basic.

syntax

```
PdhCollectQueryData(  
    ByVal QueryHandle As Long  
)  
As Long
```

Examples

For an example, see [Browsing Performance Counters](#) or [Reading Performance Data from a Log File](#).

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

Requirement	Value
Header	pdh.h
Library	Pdh.lib
DLL	Pdh.dll

See also

[PdhGetFormattedCounterValue](#)

[PdhGetRawCounterValue](#)

Feedback

Was this page helpful?

 Yes

 No

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

PdhCollectQueryDataEx function (pdh.h)

Article10/13/2021

Uses a separate thread to collect the current raw data value for all counters in the specified query. The function then signals the application-defined event and waits the specified time interval before returning.

Syntax

C++

```
PDH_FUNCTION PdhCollectQueryDataEx(
    [in] PDH_HQUERY hQuery,
    [in] DWORD      dwIntervalTime,
    [in] HANDLE     hNewDataEvent
);
```

Parameters

[in] hQuery

Handle of the query. The query identifies the counters that you want to collect. The [PdhOpenQuery](#) function returns this handle.

[in] dwIntervalTime

Time interval to wait, in seconds.

[in] hNewDataEvent

Handle to the event that you want PDH to signal after the time interval expires. To create an event object, call the [CreateEvent](#) function.

Return value

If the function succeeds, it returns ERROR_SUCCESS.

If the function fails, the return value is a [system error code](#) or a [PDH error code](#). The following are possible values.

 Expand table

Return code	Description
PDH_INVALID_HANDLE	The query handle is not valid.
PDH_NO_DATA	The query does not currently have any counters.

Remarks

PDH terminates the thread when you call the [PdhCloseQuery](#) function. If you call [PdhCollectQueryDataEx](#) more than once, each subsequent call terminates the thread from the previous call and then starts a new thread.

When [PdhCollectQueryDataEx](#) is called for data from one counter instance only and the counter instance does not exist, the function returns PDH_NO_DATA. However, if data from more than one counter is queried, [PdhCollectQueryDataEx](#) may return ERROR_SUCCESS even if one of the counter instances does not yet exist. This is because it is not known if the specified counter instance does not exist, or if it will exist but has not yet been created. In this case, call [PdhGetRawCounterValue](#) or [PdhGetFormattedCounterValue](#) for each of the counter instances of interest to determine whether they exist.

PDH stores the raw counter values for the current and previous collection. If you want to retrieve the current raw counter value, call the [PdhGetRawCounterValue](#) function. If you want to compute a displayable value for the counter value, call the [PdhGetFormattedCounterValue](#). If the counter path contains a wildcard for the instance name, instead call the [PdhGetRawCounterArray](#) and [PdhGetFormattedCounterArray](#) functions, respectively.

Examples

The following example shows how to use this function.

C++

```
#include <windows.h>
#include <stdio.h>
#include <stdlib.h>
#include <pdh.h>

#pragma comment(lib, "pdh.lib")

CONST PWSTR COUNTER_NAME      = L"\Processor(0)\% Processor Time";
CONST ULONG SAMPLE_COUNT      = 10;
CONST ULONG SAMPLE_INTERVAL   = 2;
```

```
void wmain(void)
{
    PDH_STATUS Status;
    HANDLE Event = NULL;
    PDH_HQUERY Query = NULL;
    PDH_HCOUNTER Counter;
    ULONG WaitResult;
    ULONG CounterType;
    PDH_FMT_COUNTERVALUE DisplayValue;

    Status = PdhOpenQuery(NULL, 0, &Query);
    if (Status != ERROR_SUCCESS)
    {
        wprintf(L"\nPdhOpenQuery failed with status 0x%x.", Status);
        goto Cleanup;
    }

    Status = PdhAddCounter(Query, COUNTER_NAME, 0, &Counter);
    if (Status != ERROR_SUCCESS)
    {
        wprintf(L"\nPdhAddCounter failed with 0x%x.", Status);
        goto Cleanup;
    }

    //
    // Calculating the formatted value of some counters requires access to
    // the
    // value of a previous sample. Make this call to get the first sample
    // value
    // populated, to be used later for calculating the next sample.
    //

    Status = PdhCollectQueryData(Query);
    if (Status != ERROR_SUCCESS)
    {
        wprintf(L"\nPdhCollectQueryData failed with status 0x%x.", Status);
        goto Cleanup;
    }

    //
    // This will create a separate thread that will collect raw counter data
    // every 2 seconds and set the supplied Event.
    //

    Event = CreateEvent(NULL, FALSE, FALSE, L"MyEvent");
    if (Event == NULL)
    {
        wprintf(L"\nCreateEvent failed with status 0x%x.", GetLastError());
        goto Cleanup;
    }

    Status = PdhCollectQueryDataEx(Query, SAMPLE_INTERVAL, Event);
    if (Status != ERROR_SUCCESS)
    {
```

```

        wprintf(L"\nPdhCollectQueryDataEx failed with status 0x%x.", 
Status);
        goto Cleanup;
    }

    //
    // Collect and format 10 samples, 2 seconds apart.
    //

    for (ULONG i = 0; i < SAMPLE_COUNT; i++)
    {
        WaitResult = WaitForSingleObject(Event, INFINITE);

        if (WaitResult == WAIT_OBJECT_0)
        {
            Status = PdhGetFormattedCounterValue(Counter, PDH_FMT_DOUBLE,
&CounterType, &DisplayValue);

            if (Status == ERROR_SUCCESS)
            {
                wprintf(L"\nCounter Value: %.20g",
DisplayValue.doubleValue);
            }
            else
            {
                wprintf(L"\nPdhGetFormattedCounterValue failed with status
0x%x.", Status);
                goto Cleanup;
            }
        }
        else if (WaitResult == WAIT_FAILED)
        {
            wprintf(L"\nWaitForSingleObject failed with status 0x%x.",
GetLastError());
            goto Cleanup;
        }
    }

Cleanup:

    if (Event)
    {
        CloseHandle(Event);
    }

    //
    // This will close both the Query handle and all associated Counter
handles
    // returned by PdhAddCounter.
    //

    if (Query)
    {
        PdhCloseQuery(Query);
    }

```



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	pdh.h
Library	Pdh.lib
DLL	Pdh.dll

See also

[CreateEvent](#)

[PdhCollectQueryData](#)

[PdhOpenQuery](#)

Feedback

Was this page helpful?

thumb up Yes

thumb down No

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

PdhCollectQueryDataWithTime function (pdh.h)

Article07/27/2022

Collects the current raw data value for all counters in the specified query and updates the status code of each counter.

Syntax

C++

```
PDH_FUNCTION PdhCollectQueryDataWithTime(
    [in, out] PDH_HQUERY hQuery,
    [out]      LONGLONG *pllTimeStamp
);
```

Parameters

[in, out] hQuery

Handle of the query for which you want to collect data. The [PdhOpenQuery](#) function returns this handle.

[out] pllTimeStamp

Time stamp when the first counter value in the query was retrieved. The time is specified as FILETIME.

Return value

If the function succeeds, it returns ERROR_SUCCESS. Otherwise, the function returns a [system error code](#) or a [PDH error code](#).

The following are possible values.

[] Expand table

Return code	Description
PDH_INVALID_HANDLE	The query handle is not valid.

PDH_NO_DATA

The query does not currently have any counters.

Remarks

Call this function when you want to collect counter data for the counters in the query. PDH stores the raw counter values for the current and previous collection.

If you want to retrieve the current raw counter value, call the [PdhGetRawCounterValue](#) function. If you want to compute a displayable value for the counter value, call the [PdhGetFormattedCounterValue](#). If the counter path contains a wildcard for the instance name, instead call the [PdhGetRawCounterArray](#) and [PdhGetFormattedCounterArray](#) functions, respectively.

When [PdhCollectQueryDataEx](#) is called for data from one counter instance only, and the counter instance does not exist, the function returns PDH_NO_DATA. However, if data from more than one counter is queried, [PdhCollectQueryDataEx](#) may return ERROR_SUCCESS even if one of the counter instances does not yet exist. This is because it is not known if the specified counter instance does not exist, or if it will exist but has not yet been created. In this case, call the [PdhGetRawCounterValue](#) or [PdhGetFormattedCounterValue](#) function for each of the counter instances of interest to determine whether they exist.

Requirements

[] Expand table

Requirement	Value
Minimum supported client	Windows Vista [desktop apps only]
Minimum supported server	Windows Server 2008 [desktop apps only]
Target Platform	Windows
Header	pdh.h
Library	Pdh.lib
DLL	Pdh.dll

See also

[PdhCollectQueryData](#)

[PdhGetFormattedCounterValue](#)

[PdhGetRawCounterValue](#)

[PdhOpenQuery](#)

Feedback

Was this page helpful?

 Yes

 No

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

PdhComputeCounterStatistics function (pdh.h)

Article 10/13/2021

Computes statistics for a counter from an array of raw values.

Syntax

C++

```
PDH_FUNCTION PdhComputeCounterStatistics(
    [in] PDH_HCOUNTER hCounter,
    [in] DWORD dwFormat,
    [in] DWORD dwFirstEntry,
    [in] DWORD dwNumEntries,
    [in] PPDH_RAW_COUNTER lpRawValueArray,
    [out] PPDH_STATISTICS data
);
```

Parameters

[in] hCounter

Handle of the counter for which you want to compute statistics. The [PdhAddCounter](#) function returns this handle.

[in] dwFormat

Determines the data type of the formatted value. Specify one of the following values.

[] Expand table

Value	Meaning
PDH_FMT_DOUBLE	Return the calculated value as a double-precision floating point real.
PDH_FMT_LARGE	Return the calculated value as a 64-bit integer.
PDH_FMT_LONG	Return the calculated value as a long integer.

You can use the bitwise inclusive OR operator (`|`) to combine the data type with one of the following scaling factors.

[Expand table

Value	Meaning
<code>PDH_FMT_NOSCALE</code>	Do not apply the counter's scaling factors in the calculation.
<code>PDH_FMT_NOCAP100</code>	Counter values greater than 100 (for example, counter values measuring the processor load on multiprocessor computers) will not be reset to 100. The default behavior is that counter values are capped at a value of 100.
<code>PDH_FMT_1000</code>	Multiply the final value by 1,000.

`[in] dwFirstEntry`

Zero-based index of the first raw counter value to use to begin the calculations. The index value must point to the oldest entry in the buffer. The function starts at this entry and scans through the buffer, wrapping at the last entry back to the beginning of the buffer and up to the `dwFirstEntry-1` entry, which is assumed to be the newest or most recent data.

`[in] dwNumEntries`

Number of raw counter values in the `lpRawValueArray` buffer.

`[in] lpRawValueArray`

Array of `PDH_RAW_COUNTER` structures that contain `dwNumEntries` entries.

`[out] data`

A `PDH_STATISTICS` structure that receives the counter statistics.

Return value

If the function succeeds, it returns `ERROR_SUCCESS`.

If the function fails, the return value is a `system error code` or a `PDH error code`. The following are possible values.

[Expand table

Return code	Description
PDH_INVALID_ARGUMENT	An argument is not correct or is incorrectly formatted.
PDH_INVALID_HANDLE	The counter handle is not valid.

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	pdh.h
Library	Pdh.lib
DLL	Pdh.dll

See also

[PDH_RAW_COUNTER](#)

[PDH_STATISTICS](#)

[PdhCalculateCounterFromRawValue](#)

[PdhGetRawCounterValue](#)

[PdhSetCounterScaleFactor](#)

Feedback

Was this page helpful?

 Yes

 No

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

PdhConnectMachineA function (pdh.h)

Article02/22/2024

Connects to the specified computer.

Syntax

C++

```
PDH_FUNCTION PdhConnectMachineA(  
    [in] LPCSTR szMachineName  
);
```

Parameters

[in] szMachineName

Null-terminated string that specifies the name of the computer to connect to. If **NULL**, PDH connects to the local computer.

Return value

If the function succeeds, it returns **ERROR_SUCCESS**.

If the function fails, the return value is a [system error code](#) or a [PDH error code](#). The following are possible values.

[+] Expand table

Return code	Description
PDH_CSTATUS_NO_MACHINE	Unable to connect to the specified computer. Could be caused by the computer not being on, not supporting PDH, not being connected to the network, or having the permissions set on the registry that prevent remote connections or remote performance monitoring by the user.
PDH_MEMORY_ALLOCATION_FAILURE	Unable to allocate a dynamic memory block. Occurs when there is a serious memory shortage in the system due to too many applications running on the system or an insufficient memory paging file.

Remarks

Typically, applications do not call this function and instead the connection is made when the application adds the counter to the query.

However, you can use this function if you want to include more than the local computer in the **Select counters from computer** list on the **Browse Counters** dialog box. For details, see the [PDH_BROWSE_DLG_CONFIG](#) structure.

ⓘ Note

The pdh.h header defines PdhConnectMachine 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 is 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	pdh.h
Library	Pdh.lib
DLL	Pdh.dll

See also

[PdhEnumMachines](#)

Feedback

Was this page helpful?

 Yes

 No

Provide product feedback  | Get help at Microsoft Q&A

PdhConnectMachineW function (pdh.h)

Article 02/09/2023

Connects to the specified computer.

Syntax

C++

```
PDH_FUNCTION PdhConnectMachineW(
    [in] LPCWSTR szMachineName
);
```

Parameters

[in] szMachineName

Null-terminated string that specifies the name of the computer to connect to. If **NULL**, PDH connects to the local computer.

Return value

If the function succeeds, it returns **ERROR_SUCCESS**.

If the function fails, the return value is a [system error code](#) or a [PDH error code](#). The following are possible values.

Return code	Description
PDH_CSTATUS_NO_MACHINE	Unable to connect to the specified computer. Could be caused by the computer not being on, not supporting PDH, not being connected to the network, or having the permissions set on the registry that prevent remote connections or remote performance monitoring by the user.
PDH_MEMORY_ALLOCATION_FAILURE	Unable to allocate a dynamic memory block. Occurs when there is a serious memory shortage in the system due to too many applications running on the system or an insufficient memory paging file.

Remarks

Typically, applications do not call this function and instead the connection is made when the application adds the counter to the query.

However, you can use this function if you want to include more than the local computer in the **Select counters from computer** list on the **Browse Counters** dialog box. For details, see the [PDH_BROWSE_DLG_CONFIG](#) structure.

ⓘ Note

The pdh.h header defines PdhConnectMachine 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 is not encoding-neutral can lead to mismatches that result in compilation or runtime errors. For more information, see [Conventions for Function Prototypes](#).

Requirements

Minimum supported client	Windows XP [desktop apps only]
Minimum supported server	Windows Server 2003 [desktop apps only]
Target Platform	Windows
Header	pdh.h
Library	Pdh.lib
DLL	Pdh.dll

See also

[PdhEnumMachines](#)

Feedback

Was this page helpful?

Yes

No

Get help at Microsoft Q&A

PdhEnumLogSetNamesA function (pdh.h)

Article 02/22/2024

Enumerates the names of the log sets within the DSN.

Syntax

C++

```
PDH_FUNCTION PdhEnumLogSetNamesA(
    [in]      LPCSTR  szDataSource,
    [out]     PZZSTR  mszDataSetNameList,
    [in, out] LPDWORD pcchBufferLength
);
```

Parameters

[in] `szDataSource`

Null-terminated string that specifies the DSN.

[out] `mszDataSetNameList`

Caller-allocated buffer that receives the list of **null**-terminated log set names. The list is terminated with a **null**-terminator character. Set to **NULL** if the *pcchBufferLength* parameter is zero.

[in, out] `pcchBufferLength`

Size of the *mszLogSetNameList* buffer, in **TCHARs**. If zero on input, the function returns **PDH_MORE_DATA** and sets this parameter to the required buffer size. If the buffer is larger than the required size, the function sets this parameter to the actual size of the buffer that was used. If the specified size on input is greater than zero but less than the required size, you should not rely on the returned size to reallocate the buffer.

Return value

If the function succeeds, it returns **ERROR_SUCCESS**.

If the function fails, the return value is a [system error code](#) or a [PDH error code](#). The following are possible values.

[+] [Expand table](#)

Return code	Description
PDH_MORE_DATA	The size of the <i>mszLogSetNameList</i> buffer is too small to contain all the data. This return value is expected if <i>pcchBufferLength</i> is zero on input. If the specified size on input is greater than zero but less than the required size, you should not rely on the returned size to reallocate the buffer.
PDH_INVALID_ARGUMENT	A parameter is not valid. For example, on some releases you could receive this error if the specified size on input is greater than zero but less than the required size.

Remarks

You should call this function twice, the first time to get the required buffer size (set *mszLogSetNameList* to **NULL** and *pcchBufferLength* to 0), and the second time to get the data.

Note

The pdh.h header defines PdhEnumLogSetNames 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 is 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

Requirement	Value
Header	pdh.h
Library	Pdh.lib
DLL	Pdh.dll

Feedback

Was this page helpful?

 Yes

 No

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

PdhEnumLogSetNamesW function (pdh.h)

Article 02/09/2023

Enumerates the names of the log sets within the DSN.

Syntax

C++

```
PDH_FUNCTION PdhEnumLogSetNamesW(
    [in]      LPCWSTR szDataSource,
    [out]     PZWSTR mszDataSetNameList,
    [in, out] LPDWORD pcchBufferLength
);
```

Parameters

[in] szDataSource

Null-terminated string that specifies the DSN.

[out] mszDataSetNameList

Caller-allocated buffer that receives the list of **null**-terminated log set names. The list is terminated with a **null**-terminator character. Set to **NULL** if the *pcchBufferLength* parameter is zero.

[in, out] pcchBufferLength

Size of the *mszLogSetNameList* buffer, in **TCHARs**. If zero on input, the function returns **PDH_MORE_DATA** and sets this parameter to the required buffer size. If the buffer is larger than the required size, the function sets this parameter to the actual size of the buffer that was used. If the specified size on input is greater than zero but less than the required size, you should not rely on the returned size to reallocate the buffer.

Return value

If the function succeeds, it returns **ERROR_SUCCESS**.

If the function fails, the return value is a [system error code](#) or a [PDH error code](#). The following are possible values.

Return code	Description
PDH_MORE_DATA	The size of the <i>mszLogSetNameList</i> buffer is too small to contain all the data. This return value is expected if <i>pcchBufferLength</i> is zero on input. If the specified size on input is greater than zero but less than the required size, you should not rely on the returned size to reallocate the buffer.
PDH_INVALID_ARGUMENT	A parameter is not valid. For example, on some releases you could receive this error if the specified size on input is greater than zero but less than the required size.

Remarks

You should call this function twice, the first time to get the required buffer size (set *mszLogSetNameList* to **NULL** and *pcchBufferLength* to 0), and the second time to get the data.

Note

The pdh.h header defines PdhEnumLogSetNames 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 is not encoding-neutral can lead to mismatches that result in compilation or runtime errors. For more information, see [Conventions for Function Prototypes](#).

Requirements

Minimum supported client	Windows XP [desktop apps only]
Minimum supported server	Windows Server 2003 [desktop apps only]
Target Platform	Windows
Header	pdh.h
Library	Pdh.lib

DLL

Pdh.dll

Feedback

Was this page helpful?



[Get help at Microsoft Q&A](#)

PdhEnumMachinesA function (pdh.h)

Article02/09/2023

Returns a list of the computer names associated with counters in a log file. The computer names were either specified when adding counters to the query or when calling the [PdhConnectMachine](#) function. The computers listed include those that are currently connected and online, in addition to those that are offline or not returning performance data.

To use handles to data sources, use the [PdhEnumMachinesH](#) function.

Syntax

C++

```
PDH_FUNCTION PdhEnumMachinesA(
    [in]      LPCSTR  szDataSource,
    [out]     PZZSTR  mszMachineList,
    [in, out] LPDWORD pcchBufferSize
);
```

Parameters

[in] `szDataSource`

Null-terminated string that specifies the name of a log file. The function enumerates the names of the computers whose counter data is in the log file. If **NULL**, the function enumerates the list of computers that were specified when adding counters to a real time query or when calling the [PdhConnectMachine](#) function.

[out] `mszMachineList`

Caller-allocated buffer to receive the list of null-terminated strings that contain the computer names. The list is terminated with two null-terminator characters. Set to **NULL** if `pcchBufferSize` is zero.

[in, out] `pcchBufferSize`

Size of the `mszMachineNameList` buffer, in TCHARs. If zero on input, the function returns **PDH_MORE_DATA** and sets this parameter to the required buffer size. If the buffer is larger than the required size, the function sets this parameter to the actual size of the

buffer that was used. If the specified size on input is greater than zero but less than the required size, you should not rely on the returned size to reallocate the buffer.

Return value

If the function succeeds, it returns ERROR_SUCCESS.

If the function fails, the return value is a [system error code](#) or a [PDH error code](#). The following are possible values.

[+] Expand table

Return code	Description
PDH_MORE_DATA	The <i>mszMachineNameList</i> buffer is too small to contain all the data. This return value is expected if <i>pcchBufferLength</i> is zero on input. If the specified size on input is greater than zero but less than the required size, you should not rely on the returned size to reallocate the buffer.
PDH_INVALID_ARGUMENT	A parameter is not valid. For example, on some releases you could receive this error if the specified size on input is greater than zero but less than the required size.

Remarks

You should call this function twice, the first time to get the required buffer size (set *mszMachineNameList* to **NULL** and *pcchBufferLength* to 0), and the second time to get the data.

! Note

The pdh.h header defines PdhEnumMachines 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 is not encoding-neutral can lead to mismatches that result in compilation or runtime errors. For more information, see [Conventions for Function Prototypes](#).

Requirements

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

See also

[PdhConnectMachine](#)

[PdhEnumMachinesH](#)

Feedback

Was this page helpful?

 Yes

 No

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

PdhEnumMachinesHA function (pdh.h)

Article02/22/2024

Returns a list of the computer names associated with counters in a log file. The computer names were either specified when adding counters to the query or when calling the [PdhConnectMachine](#) function. The computers listed include those that are currently connected and online, in addition to those that are offline or not returning performance data.

This function is identical to the [PdhEnumMachines](#) function, except that it supports the use of handles to data sources.

Syntax

C++

```
PDH_FUNCTION PdhEnumMachinesHA(
    [in]      PDH_HLOG hDataSource,
    [out]     PZZSTR   mszMachineList,
    [in, out] LPDWORD  pcchBufferSize
);
```

Parameters

[in] `hDataSource`

Handle to a data source returned by the [PdhBindInputDataSource](#) function.

[out] `mszMachineList`

Caller-allocated buffer to receive the list of **null**-terminated strings that contain the computer names. The list is terminated with two **null**-terminator characters. Set to **NULL** if `pcchBufferSize` is zero.

[in, out] `pcchBufferSize`

Size of the `mszMachineNameList` buffer, in TCHARs. If zero on input, the function returns **PDH_MORE_DATA** and sets this parameter to the required buffer size. If the buffer is larger than the required size, the function sets this parameter to the actual size of the buffer that was used. If the specified size on input is greater than zero but less than the required size, you should not rely on the returned size to reallocate the buffer.

Return value

If the function succeeds, it returns ERROR_SUCCESS.

If the function fails, the return value is a [system error code](#) or a [PDH error code](#). The following are possible values.

[+] Expand table

Return code	Description
PDH_MORE_DATA	The <i>mszMachineNameList</i> buffer is too small to contain all the data. This return value is expected if <i>pcchBufferLength</i> is zero on input. If the specified size on input is greater than zero but less than the required size, you should not rely on the returned size to reallocate the buffer.
PDH_INVALID_ARGUMENT	A parameter is not valid. For example, on some releases you could receive this error if the specified size on input is greater than zero but less than the required size.

Remarks

You should call this function twice, the first time to get the required buffer size (set *mszMachineNameList* to **NULL** and *pcchBufferLength* to 0), and the second time to get the data.

! Note

The pdh.h header defines PdhEnumMachinesH 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	pdh.h
Library	Pdh.lib
DLL	Pdh.dll

See also

[PdhBindInputDataSource](#)

[PdhConnectMachine](#)

Feedback

Was this page helpful?

 Yes

 No

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

PdhEnumMachinesHW function (pdh.h)

Article02/09/2023

Returns a list of the computer names associated with counters in a log file. The computer names were either specified when adding counters to the query or when calling the [PdhConnectMachine](#) function. The computers listed include those that are currently connected and online, in addition to those that are offline or not returning performance data.

This function is identical to the [PdhEnumMachines](#) function, except that it supports the use of handles to data sources.

Syntax

C++

```
PDH_FUNCTION PdhEnumMachinesHW(
    [in]      PDH_HLOG hDataSource,
    [out]     PZWSTR mszMachineList,
    [in, out] LPDWORD pcchBufferSize
);
```

Parameters

[in] `hDataSource`

Handle to a data source returned by the [PdhBindInputDataSource](#) function.

[out] `mszMachineList`

Caller-allocated buffer to receive the list of **null**-terminated strings that contain the computer names. The list is terminated with two **null**-terminator characters. Set to **NULL** if `pcchBufferSize` is zero.

[in, out] `pcchBufferSize`

Size of the `mszMachineNameList` buffer, in TCHARs. If zero on input, the function returns **PDH_MORE_DATA** and sets this parameter to the required buffer size. If the buffer is larger than the required size, the function sets this parameter to the actual size of the buffer that was used. If the specified size on input is greater than zero but less than the required size, you should not rely on the returned size to reallocate the buffer.

Return value

If the function succeeds, it returns ERROR_SUCCESS.

If the function fails, the return value is a [system error code](#) or a [PDH error code](#). The following are possible values.

Return code	Description
PDH_MORE_DATA	The <i>mszMachineNameList</i> buffer is too small to contain all the data. This return value is expected if <i>pcchBufferLength</i> is zero on input. If the specified size on input is greater than zero but less than the required size, you should not rely on the returned size to reallocate the buffer.
PDH_INVALID_ARGUMENT	A parameter is not valid. For example, on some releases you could receive this error if the specified size on input is greater than zero but less than the required size.

Remarks

You should call this function twice, the first time to get the required buffer size (set *mszMachineNameList* to **NULL** and *pcchBufferLength* to 0), and the second time to get the data.

Note

The pdh.h header defines PdhEnumMachinesH 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 is not encoding-neutral can lead to mismatches that result in compilation or runtime errors. For more information, see [Conventions for Function Prototypes](#).

Requirements

Minimum supported client	Windows XP [desktop apps only]
Minimum supported server	Windows Server 2003 [desktop apps only]
Target Platform	Windows

Header	pdh.h
Library	Pdh.lib
DLL	Pdh.dll

See also

[PdhBindInputDataSource](#)

[PdhConnectMachine](#)

Feedback

Was this page helpful?



[Get help at Microsoft Q&A](#)

PdhEnumMachinesW function (pdh.h)

Article02/09/2023

Returns a list of the computer names associated with counters in a log file. The computer names were either specified when adding counters to the query or when calling the [PdhConnectMachine](#) function. The computers listed include those that are currently connected and online, in addition to those that are offline or not returning performance data.

To use handles to data sources, use the [PdhEnumMachinesH](#) function.

Syntax

C++

```
PDH_FUNCTION PdhEnumMachinesW(
    [in]      LPCWSTR szDataSource,
    [out]     PZZWSTR mszMachineList,
    [in, out] LPDWORD pcchBufferSize
);
```

Parameters

[in] `szDataSource`

Null-terminated string that specifies the name of a log file. The function enumerates the names of the computers whose counter data is in the log file. If **NULL**, the function enumerates the list of computers that were specified when adding counters to a real time query or when calling the [PdhConnectMachine](#) function.

[out] `mszMachineList`

Caller-allocated buffer to receive the list of null-terminated strings that contain the computer names. The list is terminated with two null-terminator characters. Set to **NULL** if `pcchBufferSize` is zero.

[in, out] `pcchBufferSize`

Size of the `mszMachineNameList` buffer, in TCHARs. If zero on input, the function returns **PDH_MORE_DATA** and sets this parameter to the required buffer size. If the buffer is larger than the required size, the function sets this parameter to the actual size of the

buffer that was used. If the specified size on input is greater than zero but less than the required size, you should not rely on the returned size to reallocate the buffer.

Return value

If the function succeeds, it returns ERROR_SUCCESS.

If the function fails, the return value is a [system error code](#) or a [PDH error code](#). The following are possible values.

Return code	Description
PDH_MORE_DATA	The <i>mszMachineNameList</i> buffer is too small to contain all the data. This return value is expected if <i>pcchBufferLength</i> is zero on input. If the specified size on input is greater than zero but less than the required size, you should not rely on the returned size to reallocate the buffer.
PDH_INVALID_ARGUMENT	A parameter is not valid. For example, on some releases you could receive this error if the specified size on input is greater than zero but less than the required size.

Remarks

You should call this function twice, the first time to get the required buffer size (set *mszMachineNameList* to **NULL** and *pcchBufferLength* to 0), and the second time to get the data.

Note

The pdh.h header defines PdhEnumMachines 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 is not encoding-neutral can lead to mismatches that result in compilation or runtime errors. For more information, see [Conventions for Function Prototypes](#).

Requirements

Minimum supported client	Windows XP [desktop apps only]
Minimum supported server	Windows Server 2003 [desktop apps only]
Target Platform	Windows
Header	pdh.h
Library	Pdh.lib
DLL	Pdh.dll

See also

[PdhConnectMachine](#)

[PdhEnumMachinesH](#)

Feedback

Was this page helpful?

 Yes

 No

[Get help at Microsoft Q&A](#)

PdhEnumObjectItemsA function (pdh.h)

Article 02/09/2023

Returns the specified object's counter and instance names that exist on the specified computer or in the specified log file.

To use handles to data sources, use the [PdhEnumObjectItemsH](#) function.

Syntax

C++

```
PDH_FUNCTION PdhEnumObjectItemsA(
    [in]      LPCSTR  szDataSource,
    [in]      LPCSTR  szMachineName,
    [in]      LPCSTR  szObjectName,
    [out]     PZZSTR  mszCounterList,
    [in, out] LPDWORD  pcchCounterListLength,
    [out]     PZZSTR  mszInstanceList,
    [in, out] LPDWORD  pcchInstanceListLength,
    [in]      DWORD   dwDetailLevel,
    [in]      DWORD   dwFlags
);
```

Parameters

[in] *szDataSource*

Null-terminated string that specifies the name of the log file used to enumerate the counter and instance names. If **NULL**, the function uses the computer specified in

the *szMachineName* parameter to enumerate the names.

[in] *szMachineName*

Null-terminated string that specifies the name of the computer that contains the counter and instance names that you want to enumerate.

Include the leading slashes in the computer name, for example, \computername.

If the *szDataSource* parameter is **NULL**, you can set *szMachineName* to **NULL** to specify the local computer.

[in] *szObjectName*

Null-terminated string that specifies the name of the object whose counter and instance names you want to enumerate.

[out] `mszCounterList`

Caller-allocated buffer that receives a list of **null**-terminated counter names provided by the specified object. The list contains unique counter names. The list is terminated by two **NUL** characters. Set to **NULL** if the *pcchCounterListLength* parameter is zero.

[in, out] `pcchCounterListLength`

Size of the *mszCounterList* buffer, in **TCHARs**. If zero on input and the object exists, the function returns **PDH_MORE_DATA** and sets this parameter to the required buffer size. If the buffer is larger than the required size, the function sets this parameter to the actual size of the buffer that was used. If the specified size on input is greater than zero but less than the required size, you should not rely on the returned size to reallocate the buffer.

[out] `mszInstanceList`

Caller-allocated buffer that receives a list of **null**-terminated instance names provided by the specified object. The list contains unique instance names. The list is terminated by two **NUL** characters. Set to **NULL** if *pcchInstanceListLength* is zero.

[in, out] `pcchInstanceListLength`

Size of the *mszInstanceList* buffer, in **TCHARs**. If zero on input and the object exists, the function returns **PDH_MORE_DATA** and sets this parameter to the required buffer size. If the buffer is larger than the required size, the function sets this parameter to the actual size of the buffer that was used. If the specified size on input is greater than zero but less than the required size, you should not rely on the returned size to reallocate the buffer.

If the specified object does not support variable instances, then the returned value will be zero. If the specified object does support variable instances, but does not currently have any instances, then the value returned is 2, which is the size of an empty **MULTI_SZ** list string.

[in] `dwDetailLevel`

Detail level of the performance items to return. All items that are of the specified detail level or less will be returned (the levels are listed in increasing order). This parameter can be one of the following values.

Value	Meaning
PERF_DETAIL_NOVICE	Novice user level of detail.
PERF_DETAIL_ADVANCED	Advanced user level of detail.
PERF_DETAIL_EXPERT	Expert user level of detail.
PERF_DETAIL_WIZARD	System designer level of detail.

[in] dwFlags

This parameter must be zero.

Return value

If the function succeeds, it returns ERROR_SUCCESS.

If the function fails, the return value is a [system error code](#) or a [PDH error code](#). The following are possible values.

Return code	Description
PDH_MORE_DATA	One of the buffers is too small to contain the list of names. This return value is expected if <i>pcchCounterListLength</i> or <i>pcchInstanceListLength</i> is zero on input. If the specified size on input is greater than zero but less than the required size, you should not rely on the returned size to reallocate the buffer.
PDH_INVALID_ARGUMENT	A parameter is not valid. For example, on some releases you could receive this error if the specified size on input is greater than zero but less than the required size.
PDH_MEMORY_ALLOCATION_FAILURE	Unable to allocate memory to support this function.
PDH_CSTATUS_NO_MACHINE	The specified computer is offline or unavailable.
PDH_CSTATUS_NO_OBJECT	The specified object could not be found on the specified computer or in the specified log file.

Remarks

You should call this function twice, the first time to get the required buffer size (set the buffers to **NULL** and the sizes to 0), and the second time to get the data.

Consecutive calls to this function will return identical lists of counters and instances, because **PdhEnumObjectItems** will always query the list of performance objects defined by the last call to [PdhEnumObjects](#) or [PdhEnumObjectItems](#). To refresh the list of performance objects, call **PdhEnumObjects** with a *bRefresh* flag value of **TRUE** before calling **PdhEnumObjectItems** again.

The order of the instance and counter names is undetermined.

Examples

For an example, see [Enumerating Process Objects](#).

ⓘ Note

The pdh.h header defines **PdhEnumObjectItems** 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 is 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	pdh.h
Library	Pdh.lib
DLL	Pdh.dll

See also

[PdhEnumObjectItemsH](#)

[PdhEnumObjects](#)

Feedback

Was this page helpful?

 Yes

 No

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

PdhEnumObjectItemsHA function (pdh.h)

Article 02/09/2023

Returns the specified object's counter and instance names that exist on the specified computer or in the specified log file.

This function is identical to the [PdhEnumObjectItems](#) function, except that it supports the use of handles to data sources.

Syntax

C++

```
PDH_FUNCTION PdhEnumObjectItemsHA(
    [in]      PDH_HLOG hDataSource,
    [in]      LPCSTR   szMachineName,
    [in]      LPCSTR   szObjectName,
    [out]     PZZSTR   mszCounterList,
    [in, out] LPDWORD  pcchCounterListLength,
    [out]     PZZSTR   mszInstanceList,
    [in, out] LPDWORD  pcchInstanceListLength,
    [in]      DWORD    dwDetailLevel,
    [in]      DWORD    dwFlags
);
```

Parameters

[in] `hDataSource`

Handle to a data source returned by the [PdhBindInputDataSource](#) function.

[in] `szMachineName`

Null-terminated string that specifies the name of the computer that contains the counter and instance names that you want to enumerate.

Include the leading slashes in the computer name, for example, \computername.

If the `szDataSource` parameter is **NULL**, you can set `szMachineName` to **NULL** to specify the local computer.

[in] `szObjectName`

Null-terminated string that specifies the name of the object whose counter and instance names you want to enumerate.

[out] `mszCounterList`

Caller-allocated buffer that receives a list of **null**-terminated counter names provided by the specified object. The list contains unique counter names. The list is terminated by two **NUL** characters. Set to **NULL** if the *pcchCounterListLength* parameter is zero.

[in, out] `pcchCounterListLength`

Size of the *mszCounterList* buffer, in **TCHARs**. If zero on input and the object exists, the function returns **PDH_MORE_DATA** and sets this parameter to the required buffer size. If the buffer is larger than the required size, the function sets this parameter to the actual size of the buffer that was used. If the specified size on input is greater than zero but less than the required size, you should not rely on the returned size to reallocate the buffer.

[out] `mszInstanceList`

Caller-allocated buffer that receives a list of **null**-terminated instance names provided by the specified object. The list contains unique instance names. The list is terminated by two **NUL** characters. Set to **NULL** if the *pcchInstanceListLength* parameter is zero.

[in, out] `pcchInstanceListLength`

Size of the *mszInstanceList* buffer, in **TCHARs**. If zero on input and the object exists, the function returns **PDH_MORE_DATA** and sets this parameter to the required buffer size. If the buffer is larger than the required size, the function sets this parameter to the actual size of the buffer that was used. If the specified size on input is greater than zero but less than the required size, you should not rely on the returned size to reallocate the buffer.

If the specified object does not support variable instances, then the returned value will be zero. If the specified object does support variable instances, but does not currently have any instances, then the value returned is 2, which is the size of an empty **MULTI_SZ** list string.

[in] `dwDetailLevel`

Detail level of the performance items to return. All items that are of the specified detail level or less will be returned (the levels are listed in increasing order). This parameter can be one of the following values.

Value	Meaning
PERF_DETAIL_NOVICE	Novice user level of detail.
PERF_DETAIL_ADVANCED	Advanced user level of detail.
PERF_DETAIL_EXPERT	Expert user level of detail.
PERF_DETAIL_WIZARD	System designer level of detail.

[in] dwFlags

This parameter must be zero.

Return value

If the function succeeds, it returns ERROR_SUCCESS.

If the function fails, the return value is a [system error code](#) or a [PDH error code](#). The following are possible values.

Return code	Description
PDH_MORE_DATA	One of the buffers is too small to contain the list of names. This return value is expected if <i>pcchCounterListLength</i> or <i>pcchInstanceListLength</i> is zero on input. If the specified size on input is greater than zero but less than the required size, you should not rely on the returned size to reallocate the buffer.
PDH_INVALID_ARGUMENT	A parameter is not valid. For example, on some releases you could receive this error if the specified size on input is greater than zero but less than the required size.
PDH_MEMORY_ALLOCATION_FAILURE	Unable to allocate memory to support this function.
PDH_CSTATUS_NO_MACHINE	The specified computer is offline or unavailable.
PDH_CSTATUS_NO_OBJECT	The specified object could not be found on the specified computer or in the specified log file.

Remarks

You should call this function twice, the first time to get the required buffer size (set the buffers to **NULL** and the sizes to 0), and the second time to get the data.

Consecutive calls to this function will return identical lists of counters and instances, because **PdhEnumObjectItemsH** will always query the list of performance objects defined by the last call to [PdhEnumObjectsH](#) or [PdhEnumObjectItemsH](#). To refresh the list of performance objects, call [PdhEnumObjectsH](#) with a *bRefresh* flag value of **TRUE** before calling [PdhEnumObjectItemsH](#) again.

The order of the instance and counter names is undetermined.

Note

The pdh.h header defines [PdhEnumObjectItemsH](#) 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 is 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	pdh.h
Library	Pdh.lib
DLL	Pdh.dll

See also

[PdhBindInputDataSource](#)

[PdhEnumObjectsH](#)

Feedback

Was this page helpful?

 Yes

 No

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

PdhEnumObjectItemsHW function (pdh.h)

Article 02/09/2023

Returns the specified object's counter and instance names that exist on the specified computer or in the specified log file.

This function is identical to the [PdhEnumObjectItems](#) function, except that it supports the use of handles to data sources.

Syntax

C++

```
PDH_FUNCTION PdhEnumObjectItemsHW(
    [in]      PDH_HLOG hDataSource,
    [in]      LPCWSTR szMachineName,
    [in]      LPCWSTR szObjectName,
    [out]     PZTWSTR mszCounterList,
    [in, out] LPDWORD pcchCounterListLength,
    [out]     PZTWSTR mszInstanceList,
    [in, out] LPDWORD pcchInstanceListLength,
    [in]      DWORD dwDetailLevel,
    [in]      DWORD dwFlags
);
```

Parameters

[in] `hDataSource`

Handle to a data source returned by the [PdhBindInputDataSource](#) function.

[in] `szMachineName`

Null-terminated string that specifies the name of the computer that contains the counter and instance names that you want to enumerate.

Include the leading slashes in the computer name, for example, \computername.

If the `szDataSource` parameter is **NULL**, you can set `szMachineName` to **NULL** to specify the local computer.

[in] `szObjectName`

Null-terminated string that specifies the name of the object whose counter and instance names you want to enumerate.

[out] `mszCounterList`

Caller-allocated buffer that receives a list of **null**-terminated counter names provided by the specified object. The list contains unique counter names. The list is terminated by two **NUL** characters. Set to **NULL** if the *pcchCounterListLength* parameter is zero.

[in, out] `pcchCounterListLength`

Size of the *mszCounterList* buffer, in **TCHARs**. If zero on input and the object exists, the function returns **PDH_MORE_DATA** and sets this parameter to the required buffer size. If the buffer is larger than the required size, the function sets this parameter to the actual size of the buffer that was used. If the specified size on input is greater than zero but less than the required size, you should not rely on the returned size to reallocate the buffer.

[out] `mszInstanceList`

Caller-allocated buffer that receives a list of **null**-terminated instance names provided by the specified object. The list contains unique instance names. The list is terminated by two **NUL** characters. Set to **NULL** if the *pcchInstanceListLength* parameter is zero.

[in, out] `pcchInstanceListLength`

Size of the *mszInstanceList* buffer, in **TCHARs**. If zero on input and the object exists, the function returns **PDH_MORE_DATA** and sets this parameter to the required buffer size. If the buffer is larger than the required size, the function sets this parameter to the actual size of the buffer that was used. If the specified size on input is greater than zero but less than the required size, you should not rely on the returned size to reallocate the buffer.

If the specified object does not support variable instances, then the returned value will be zero. If the specified object does support variable instances, but does not currently have any instances, then the value returned is 2, which is the size of an empty **MULTI_SZ** list string.

[in] `dwDetailLevel`

Detail level of the performance items to return. All items that are of the specified detail level or less will be returned (the levels are listed in increasing order). This parameter can be one of the following values.

Value	Meaning
PERF_DETAIL_NOVICE	Novice user level of detail.
PERF_DETAIL_ADVANCED	Advanced user level of detail.
PERF_DETAIL_EXPERT	Expert user level of detail.
PERF_DETAIL_WIZARD	System designer level of detail.

[in] dwFlags

This parameter must be zero.

Return value

If the function succeeds, it returns ERROR_SUCCESS.

If the function fails, the return value is a [system error code](#) or a [PDH error code](#). The following are possible values.

Return code	Description
PDH_MORE_DATA	One of the buffers is too small to contain the list of names. This return value is expected if <i>pcchCounterListLength</i> or <i>pcchInstanceListLength</i> is zero on input. If the specified size on input is greater than zero but less than the required size, you should not rely on the returned size to reallocate the buffer.
PDH_INVALID_ARGUMENT	A parameter is not valid. For example, on some releases you could receive this error if the specified size on input is greater than zero but less than the required size.
PDH_MEMORY_ALLOCATION_FAILURE	Unable to allocate memory to support this function.
PDH_CSTATUS_NO_MACHINE	The specified computer is offline or unavailable.
PDH_CSTATUS_NO_OBJECT	The specified object could not be found on the specified computer or in the specified log file.

Remarks

You should call this function twice, the first time to get the required buffer size (set the buffers to **NULL** and the sizes to 0), and the second time to get the data.

Consecutive calls to this function will return identical lists of counters and instances, because **PdhEnumObjectItemsH** will always query the list of performance objects defined by the last call to [PdhEnumObjectsH](#) or [PdhEnumObjectItemsH](#). To refresh the list of performance objects, call **PdhEnumObjectsH** with a *bRefresh* flag value of **TRUE** before calling **PdhEnumObjectItemsH** again.

The order of the instance and counter names is undetermined.

ⓘ Note

The pdh.h header defines **PdhEnumObjectItemsH** 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 is not encoding-neutral can lead to mismatches that result in compilation or runtime errors. For more information, see [Conventions for Function Prototypes](#).

Requirements

Minimum supported client	Windows XP [desktop apps only]
Minimum supported server	Windows Server 2003 [desktop apps only]
Target Platform	Windows
Header	pdh.h
Library	Pdh.lib
DLL	Pdh.dll

See also

[PdhBindInputDataSource](#)

[PdhEnumObjectsH](#)

Feedback

Was this page helpful?

Yes

No

Get help at Microsoft Q&A

PdhEnumObjectItemsW function (pdh.h)

Article02/09/2023

Returns the specified object's counter and instance names that exist on the specified computer or in the specified log file.

To use handles to data sources, use the [PdhEnumObjectItemsH](#) function.

Syntax

C++

```
PDH_FUNCTION PdhEnumObjectItemsW(
    [in]      LPCWSTR szDataSource,
    [in]      LPCWSTR szMachineName,
    [in]      LPCWSTR szObjectName,
    [out]     PZZWSTR mszCounterList,
    [in, out] LPDWORD  pcchCounterListLength,
    [out]     PZZWSTR mszInstanceList,
    [in, out] LPDWORD  pcchInstanceListLength,
    [in]      DWORD    dwDetailLevel,
    [in]      DWORD    dwFlags
);
```

Parameters

[in] *szDataSource*

Null-terminated string that specifies the name of the log file used to enumerate the counter and instance names. If **NULL**, the function uses the computer specified in

the *szMachineName* parameter to enumerate the names.

[in] *szMachineName*

Null-terminated string that specifies the name of the computer that contains the counter and instance names that you want to enumerate.

Include the leading slashes in the computer name, for example, \computername.

If the *szDataSource* parameter is **NULL**, you can set *szMachineName* to **NULL** to specify the local computer.

[in] *szObjectName*

Null-terminated string that specifies the name of the object whose counter and instance names you want to enumerate.

[out] `mszCounterList`

Caller-allocated buffer that receives a list of **null**-terminated counter names provided by the specified object. The list contains unique counter names. The list is terminated by two **NUL** characters. Set to **NULL** if the *pcchCounterListLength* parameter is zero.

[in, out] `pcchCounterListLength`

Size of the *mszCounterList* buffer, in **TCHARs**. If zero on input and the object exists, the function returns **PDH_MORE_DATA** and sets this parameter to the required buffer size. If the buffer is larger than the required size, the function sets this parameter to the actual size of the buffer that was used. If the specified size on input is greater than zero but less than the required size, you should not rely on the returned size to reallocate the buffer.

[out] `mszInstanceList`

Caller-allocated buffer that receives a list of **null**-terminated instance names provided by the specified object. The list contains unique instance names. The list is terminated by two **NUL** characters. Set to **NULL** if *pcchInstanceListLength* is zero.

[in, out] `pcchInstanceListLength`

Size of the *mszInstanceList* buffer, in **TCHARs**. If zero on input and the object exists, the function returns **PDH_MORE_DATA** and sets this parameter to the required buffer size. If the buffer is larger than the required size, the function sets this parameter to the actual size of the buffer that was used. If the specified size on input is greater than zero but less than the required size, you should not rely on the returned size to reallocate the buffer.

If the specified object does not support variable instances, then the returned value will be zero. If the specified object does support variable instances, but does not currently have any instances, then the value returned is 2, which is the size of an empty **MULTI_SZ** list string.

[in] `dwDetailLevel`

Detail level of the performance items to return. All items that are of the specified detail level or less will be returned (the levels are listed in increasing order). This parameter can be one of the following values.

Value	Meaning
PERF_DETAIL_NOVICE	Novice user level of detail.
PERF_DETAIL_ADVANCED	Advanced user level of detail.
PERF_DETAIL_EXPERT	Expert user level of detail.
PERF_DETAIL_WIZARD	System designer level of detail.

[in] dwFlags

This parameter must be zero.

Return value

If the function succeeds, it returns ERROR_SUCCESS.

If the function fails, the return value is a [system error code](#) or a [PDH error code](#). The following are possible values.

Return code	Description
PDH_MORE_DATA	One of the buffers is too small to contain the list of names. This return value is expected if <i>pcchCounterListLength</i> or <i>pcchInstanceListLength</i> is zero on input. If the specified size on input is greater than zero but less than the required size, you should not rely on the returned size to reallocate the buffer.
PDH_INVALID_ARGUMENT	A parameter is not valid. For example, on some releases you could receive this error if the specified size on input is greater than zero but less than the required size.
PDH_MEMORY_ALLOCATION_FAILURE	Unable to allocate memory to support this function.
PDH_CSTATUS_NO_MACHINE	The specified computer is offline or unavailable.
PDH_CSTATUS_NO_OBJECT	The specified object could not be found on the specified computer or in the specified log file.

Remarks

You should call this function twice, the first time to get the required buffer size (set the buffers to **NULL** and the sizes to 0), and the second time to get the data.

Consecutive calls to this function will return identical lists of counters and instances, because **PdhEnumObjectItems** will always query the list of performance objects defined by the last call to [PdhEnumObjects](#) or [PdhEnumObjectItems](#). To refresh the list of performance objects, call [PdhEnumObjects](#) with a *bRefresh* flag value of **TRUE** before calling [PdhEnumObjectItems](#) again.

The order of the instance and counter names is undetermined.

Examples

For an example, see [Enumerating Process Objects](#).

ⓘ Note

The pdh.h header defines PdhEnumObjectItems 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 is not encoding-neutral can lead to mismatches that result in compilation or runtime errors. For more information, see [Conventions for Function Prototypes](#).

Requirements

Minimum supported client	Windows XP [desktop apps only]
Minimum supported server	Windows Server 2003 [desktop apps only]
Target Platform	Windows
Header	pdh.h
Library	Pdh.lib
DLL	Pdh.dll

See also

[PdhEnumObjectItemsH](#)

[PdhEnumObjects](#)

Feedback

Was this page helpful?

 Yes

 No

[Get help at Microsoft Q&A](#)

PdhEnumObjectsA function (pdh.h)

Article02/09/2023

Returns a list of objects available on the specified computer or in the specified log file.

To use handles to data sources, use the [PdhEnumObjectsH](#) function.

Syntax

C++

```
PDH_FUNCTION PdhEnumObjectsA(
    [in]      LPCSTR  szDataSource,
    [in]      LPCSTR  szMachineName,
    [out]     PZZSTR  mszObjectList,
    [in, out] LPDWORD  pcchBufferSize,
    [in]      DWORD   dwDetailLevel,
    [in]      BOOL    bRefresh
);
```

Parameters

[in] *szDataSource*

Null-terminated string that specifies the name of the log file used to enumerate the performance objects. If **NULL**, the function uses the computer specified in

the *szMachineName* parameter to enumerate the names.

[in] *szMachineName*

Null-terminated string that specifies the name of the computer used to enumerate the performance objects.

Include the leading slashes in the computer name, for example, \computername.

If the *szDataSource* parameter is **NULL**, you can set *szMachineName* to **NULL** to specify the local computer.

[out] *mszObjectList*

Caller-allocated buffer that receives the list of object names. Each object name in this list is terminated by a **null** character. The list is terminated with two **null**-terminator

characters. Set to **NULL** if the *pcchBufferLength* parameter is zero.

[in, out] pcchBufferSize

Size of the *mszObjectList* buffer, in TCHARs. If zero on input, the function returns PDH_MORE_DATA and sets this parameter to the required buffer size. If the buffer is larger than the required size, the function sets this parameter to the actual size of the buffer that was used. If the specified size on input is greater than zero but less than the required size, you should not rely on the returned size to reallocate the buffer.

Windows XP: Add one to the required buffer size.

[in] dwDetailLevel

Detail level of the performance items to return. All items that are of the specified detail level or less will be returned (the levels are listed in increasing order). This parameter can be one of the following values.

 Expand table

Value	Meaning
PERF_DETAIL_NOVICE	Novice user level of detail.
PERF_DETAIL_ADVANCED	Advanced user level of detail.
PERF_DETAIL_EXPERT	Expert user level of detail.
PERF_DETAIL_WIZARD	System designer level of detail.

[in] bRefresh

Indicates if the cached object list should be automatically refreshed. Specify one of the following values.

If you call this function twice, once to get the size of the list and a second time to get the actual list, set this parameter to **TRUE** on the first call and **FALSE** on the second call. If both calls are **TRUE**, the second call may also return PDH_MORE_DATA because the object data may have changed between calls.

 Expand table

Value	Meaning
TRUE	The object cache is automatically refreshed before the objects are returned.

FALSE

Do not automatically refresh the cache.

Return value

If the function succeeds, it returns ERROR_SUCCESS.

If the function fails, the return value is a [system error code](#) or a [PDH error code](#). The following are possible values.

[+] Expand table

Return code	Description
PDH_MORE_DATA	The <i>mszObjectList</i> buffer is too small to hold the list of objects. This return value is expected if <i>pcchBufferLength</i> is zero on input. If the specified size on input is greater than zero but less than the required size, you should not rely on the returned size to reallocate the buffer.
PDH_CSTATUS_NO_MACHINE	The specified computer is offline or unavailable.
PDH_CSTATUS_NO_OBJECT	The specified object could not be found.
PDH_INVALID_ARGUMENT	A parameter is not valid. For example, on some releases you could receive this error if the specified size on input is greater than zero but less than the required size.

Remarks

You should call this function twice, the first time to get the required buffer size (set *mszObjectList* to **NULL** and *pcchBufferLength* to 0), and the second time to get the data.

ⓘ Note

The pdh.h header defines PdhEnumObjects 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 is 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	pdh.h
Library	Pdh.lib
DLL	Pdh.dll

See also

[PdhEnumObjectItems](#)

[PdhEnumObjectsH](#)

Feedback

Was this page helpful?

 Yes

 No

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

PdhEnumObjectsHA function (pdh.h)

Article 02/09/2023

Returns a list of objects available on the specified computer or in the specified log file.

This function is identical to [PdhEnumObjects](#), except that it supports the use of handles to data sources.

Syntax

C++

```
PDH_FUNCTION PdhEnumObjectsHA(
    [in]      PDH_HLOG hDataSource,
    [in]      LPCSTR   szMachineName,
    [out]     PZZSTR   mszObjectList,
    [in, out] LPDWORD  pcchBufferSize,
    [in]      DWORD    dwDetailLevel,
    [in]      BOOL     bRefresh
);
```

Parameters

[in] `hDataSource`

Handle to a data source returned by the [PdhBindInputDataSource](#) function.

[in] `szMachineName`

Null-terminated string that specifies the name of the computer used to enumerate the performance objects.

Include the leading slashes in the computer name, for example, \computername.

If `szDataSource` is **NULL**, you can set `szMachineName` to **NULL** to specify the local computer.

[out] `mszObjectList`

Caller-allocated buffer that receives the list of object names. Each object name in this list is terminated by a **null** character. The list is terminated with two **null**-terminator characters. Set to **NULL** if `pcchBufferLength` is zero.

[in, out] pcchBufferSize

Size of the *mszObjectList* buffer, in TCHARs. If zero on input, the function returns PDH_MORE_DATA and sets this parameter to the required buffer size. If the buffer is larger than the required size, the function sets this parameter to the actual size of the buffer that was used. If the specified size on input is greater than zero but less than the required size, you should not rely on the returned size to reallocate the buffer.

Windows XP: Add one to the required buffer size.

[in] dwDetailLevel

Detail level of the performance items to return. All items that are of the specified detail level or less will be returned (the levels are listed in increasing order). This parameter can be one of the following values.

[Expand table](#)

Value	Meaning
PERF_DETAIL_NOVICE	Novice user level of detail.
PERF_DETAIL_ADVANCED	Advanced user level of detail.
PERF_DETAIL_EXPERT	Expert user level of detail.
PERF_DETAIL_WIZARD	System designer level of detail.

[in] bRefresh

Indicates if the cached object list should be automatically refreshed. Specify one of the following values.

If you call this function twice, once to get the size of the list and a second time to get the actual list, set this parameter to **TRUE** on the first call and **FALSE** on the second call. If both calls are **TRUE**, the second call may also return PDH_MORE_DATA because the object data may have changed between calls.

[Expand table](#)

Value	Meaning
TRUE	The object cache is automatically refreshed before the objects are returned.
FALSE	Do not automatically refresh the cache.

Return value

If the function succeeds, it returns ERROR_SUCCESS.

If the function fails, the return value is a [system error code](#) or a [PDH error code](#). The following are possible values.

[+] [Expand table](#)

Return code	Description
PDH_MORE_DATA	The <i>mszObjectList</i> buffer is too small to hold the list of objects. This return value is expected if <i>pcchBufferLength</i> is zero on input. If the specified size on input is greater than zero but less than the required size, you should not rely on the returned size to reallocate the buffer.
PDH_CSTATUS_NO_MACHINE	The specified computer is offline or unavailable.
PDH_CSTATUS_NO_OBJECT	The specified object could not be found.
PDH_INVALID_ARGUMENT	A parameter is not valid. For example, on some releases you could receive this error if the specified size on input is greater than zero but less than the required size.

Remarks

You should call this function twice, the first time to get the required buffer size (set *mszObjectList* to **NULL** and *pcchBufferLength* to 0), and the second time to get the data.

Note

The pdh.h header defines PdhEnumObjectsH 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 is 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	pdh.h
Library	Pdh.lib
DLL	Pdh.dll

See also

[PdhBindInputDataSource](#)

[PdhEnumObjectItemsH](#)

Feedback

Was this page helpful?

 Yes

 No

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

PdhEnumObjectsHW function (pdh.h)

Article 02/09/2023

Returns a list of objects available on the specified computer or in the specified log file.

This function is identical to [PdhEnumObjects](#), except that it supports the use of handles to data sources.

Syntax

C++

```
PDH_FUNCTION PdhEnumObjectsHW(
    [in]      PDH_HLOG hDataSource,
    [in]      LPCWSTR szMachineName,
    [out]     PZWSTR mszObjectList,
    [in, out] LPDWORD pcchBufferSize,
    [in]      DWORD   dwDetailLevel,
    [in]      BOOL    bRefresh
);
```

Parameters

[in] `hDataSource`

Handle to a data source returned by the [PdhBindInputDataSource](#) function.

[in] `szMachineName`

Null-terminated string that specifies the name of the computer used to enumerate the performance objects.

Include the leading slashes in the computer name, for example, \computername.

If `szDataSource` is **NULL**, you can set `szMachineName` to **NULL** to specify the local computer.

[out] `mszObjectList`

Caller-allocated buffer that receives the list of object names. Each object name in this list is terminated by a **null** character. The list is terminated with two **null**-terminator characters. Set to **NULL** if `pcchBufferLength` is zero.

[in, out] pcchBufferSize

Size of the *mszObjectList* buffer, in TCHARs. If zero on input, the function returns PDH_MORE_DATA and sets this parameter to the required buffer size. If the buffer is larger than the required size, the function sets this parameter to the actual size of the buffer that was used. If the specified size on input is greater than zero but less than the required size, you should not rely on the returned size to reallocate the buffer.

Windows XP: Add one to the required buffer size.

[in] dwDetailLevel

Detail level of the performance items to return. All items that are of the specified detail level or less will be returned (the levels are listed in increasing order). This parameter can be one of the following values.

Value	Meaning
PERF_DETAIL_NOVICE	Novice user level of detail.
PERF_DETAIL_ADVANCED	Advanced user level of detail.
PERF_DETAIL_EXPERT	Expert user level of detail.
PERF_DETAIL_WIZARD	System designer level of detail.

[in] bRefresh

Indicates if the cached object list should be automatically refreshed. Specify one of the following values.

If you call this function twice, once to get the size of the list and a second time to get the actual list, set this parameter to **TRUE** on the first call and **FALSE** on the second call. If both calls are **TRUE**, the second call may also return PDH_MORE_DATA because the object data may have changed between calls.

Value	Meaning
TRUE	The object cache is automatically refreshed before the objects are returned.
FALSE	Do not automatically refresh the cache.

Return value

If the function succeeds, it returns ERROR_SUCCESS.

If the function fails, the return value is a [system error code](#) or a [PDH error code](#). The following are possible values.

Return code	Description
PDH_MORE_DATA	The <i>mszObjectList</i> buffer is too small to hold the list of objects. This return value is expected if <i>pcchBufferLength</i> is zero on input. If the specified size on input is greater than zero but less than the required size, you should not rely on the returned size to reallocate the buffer.
PDH_CSTATUS_NO_MACHINE	The specified computer is offline or unavailable.
PDH_CSTATUS_NO_OBJECT	The specified object could not be found.
PDH_INVALID_ARGUMENT	A parameter is not valid. For example, on some releases you could receive this error if the specified size on input is greater than zero but less than the required size.

Remarks

You should call this function twice, the first time to get the required buffer size (set *mszObjectList* to **NULL** and *pcchBufferLength* to 0), and the second time to get the data.

ⓘ Note

The pdh.h header defines PdhEnumObjectsH 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

Minimum supported client	Windows XP [desktop apps only]
Minimum supported server	Windows Server 2003 [desktop apps only]
Target Platform	Windows
Header	pdh.h

Library	Pdh.lib
DLL	Pdh.dll

See also

[PdhBindInputDataSource](#)

[PdhEnumObjectItemsH](#)

Feedback

Was this page helpful?



[Get help at Microsoft Q&A](#)

PdhEnumObjectsW function (pdh.h)

Article02/09/2023

Returns a list of objects available on the specified computer or in the specified log file.

To use handles to data sources, use the [PdhEnumObjectsH](#) function.

Syntax

C++

```
PDH_FUNCTION PdhEnumObjectsW(
    [in]      LPCWSTR szDataSource,
    [in]      LPCWSTR szMachineName,
    [out]     PZZWSTR mszObjectList,
    [in, out] LPDWORD pcchBufferSize,
    [in]      DWORD   dwDetailLevel,
    [in]      BOOL    bRefresh
);
```

Parameters

[in] *szDataSource*

Null-terminated string that specifies the name of the log file used to enumerate the performance objects. If **NULL**, the function uses the computer specified in

the *szMachineName* parameter to enumerate the names.

[in] *szMachineName*

Null-terminated string that specifies the name of the computer used to enumerate the performance objects.

Include the leading slashes in the computer name, for example, \computername.

If the *szDataSource* parameter is **NULL**, you can set *szMachineName* to **NULL** to specify the local computer.

[out] *mszObjectList*

Caller-allocated buffer that receives the list of object names. Each object name in this list is terminated by a **null** character. The list is terminated with two **null**-terminator

characters. Set to **NULL** if the *pcchBufferLength* parameter is zero.

[in, out] pcchBufferSize

Size of the *mszObjectList* buffer, in TCHARs. If zero on input, the function returns PDH_MORE_DATA and sets this parameter to the required buffer size. If the buffer is larger than the required size, the function sets this parameter to the actual size of the buffer that was used. If the specified size on input is greater than zero but less than the required size, you should not rely on the returned size to reallocate the buffer.

Windows XP: Add one to the required buffer size.

[in] dwDetailLevel

Detail level of the performance items to return. All items that are of the specified detail level or less will be returned (the levels are listed in increasing order). This parameter can be one of the following values.

Value	Meaning
PERF_DETAIL_NOVICE	Novice user level of detail.
PERF_DETAIL_ADVANCED	Advanced user level of detail.
PERF_DETAIL_EXPERT	Expert user level of detail.
PERF_DETAIL_WIZARD	System designer level of detail.

[in] bRefresh

Indicates if the cached object list should be automatically refreshed. Specify one of the following values.

If you call this function twice, once to get the size of the list and a second time to get the actual list, set this parameter to **TRUE** on the first call and **FALSE** on the second call. If both calls are **TRUE**, the second call may also return PDH_MORE_DATA because the object data may have changed between calls.

Value	Meaning
TRUE	The object cache is automatically refreshed before the objects are returned.
FALSE	Do not automatically refresh the cache.

Return value

If the function succeeds, it returns ERROR_SUCCESS.

If the function fails, the return value is a [system error code](#) or a [PDH error code](#). The following are possible values.

Return code	Description
PDH_MORE_DATA	The <i>mszObjectList</i> buffer is too small to hold the list of objects. This return value is expected if <i>pcchBufferLength</i> is zero on input. If the specified size on input is greater than zero but less than the required size, you should not rely on the returned size to reallocate the buffer.
PDH_CSTATUS_NO_MACHINE	The specified computer is offline or unavailable.
PDH_CSTATUS_NO_OBJECT	The specified object could not be found.
PDH_INVALID_ARGUMENT	A parameter is not valid. For example, on some releases you could receive this error if the specified size on input is greater than zero but less than the required size.

Remarks

You should call this function twice, the first time to get the required buffer size (set *mszObjectList* to **NULL** and *pcchBufferLength* to 0), and the second time to get the data.

ⓘ Note

The pdh.h header defines PdhEnumObjects 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 is not encoding-neutral can lead to mismatches that result in compilation or runtime errors. For more information, see [Conventions for Function Prototypes](#).

Requirements

Minimum supported client	Windows XP [desktop apps only]
Minimum supported server	Windows Server 2003 [desktop apps only]
Target Platform	Windows

Header	pdh.h
Library	Pdh.lib
DLL	Pdh.dll

See also

[PdhEnumObjectItems](#)

[PdhEnumObjectsH](#)

Feedback

Was this page helpful?

 Yes

 No

[Get help at Microsoft Q&A](#)

PdhExpandCounterPathA function (pdh.h)

Article02/09/2023

Examines the specified computer (or local computer if none is specified) for counters and instances of counters that match the wildcard strings in the counter path.

Note This function is superseded by the **PdhExpandWildCardPath** function.

Syntax

C++

```
PDH_FUNCTION PdhExpandCounterPathA(
    [in]      LPCSTR szWildCardPath,
    [out]     PZZSTR mszExpandedPathList,
    [in, out] LPDWORD pcchPathListLength
);
```

Parameters

[in] `szWildCardPath`

Null-terminated string that contains the counter path to expand. The function searches the computer specified in the path for matches. If the path does not specify a computer, the function searches the local computer. The maximum length of a counter path is `PDH_MAX_COUNTER_PATH`.

[out] `mszExpandedPathList`

Caller-allocated buffer that receives the list of expanded counter paths that match the wildcard specification in `szWildCardPath`. Each counter path in this list is terminated by a null character. The list is terminated with two **NUL** characters. Set to **NUL** if `pcchPathListLength` is zero.

[in, out] `pcchPathListLength`

Size of the *mszExpandedPathList* buffer, in TCHARs. If zero on input, the function returns PDH_MORE_DATA and sets this parameter to the required buffer size. If the buffer is larger than the required size, the function sets this parameter to the actual size of the buffer that was used. If the specified size on input is greater than zero but less than the required size, you should not rely on the returned size to reallocate the buffer.

Note You must add one to the required size on Windows XP.

Return value

If the function succeeds, it returns ERROR_SUCCESS.

If the function fails, the return value is a [system error code](#) or a [PDH error code](#).

 Expand table

Return code	Description
PDH_MORE_DATA	The <i>mszExpandedPathList</i> buffer is too small to contain the list of paths. This return value is expected if <i>pcchPathListLength</i> is zero on input. If the specified size on input is greater than zero but less than the required size, you should not rely on the returned size to reallocate the buffer.
PDH_INVALID_ARGUMENT	A parameter is not valid. For example, on some releases you could receive this error if the specified size on input is greater than zero but less than the required size.
PDH_MEMORY_ALLOCATION_FAILURE	Unable to allocate memory to support this function.

Remarks

You should call this function twice, the first time to get the required buffer size (set *mszExpandedPathList* to **NULL** and *pcchPathListLength* to 0), and the second time to get the data.

The general counter path format is as follows:

\computer\object(parent/instance#index)\counter

The parent, instance, index, and counter components of the counter path may contain either a valid name or a wildcard character. The computer, parent, instance, and index components are not necessary for all counters.

The counter paths that you must use is determined by the counter itself. For example, the LogicalDisk object has an instance index, so you must provide the #index, or a wildcard. Therefore, you could use the following format:

```
\LogicalDisk(/#*)*
```

In comparison, the Process object does not require an instance index. Therefore, you could use the following format:

```
\Process(*)\ID Process
```

The following is a list of the possible formats:

- \\computer\object(parent/instance#index)\counter
- \\computer\object(parent/instance)\counter
- \\computer\object(instance#index)\counter
- \\computer\object(instance)\counter
- \\computer\object\counter
- \object(parent/instance#index)\counter
- \object(parent/instance)\counter
- \object(instance#index)\counter
- \object(instance)\counter
- \object\counter

If a wildcard character is specified in the parent name, all instances of the specified object that match the specified instance and counter fields will be returned.

If a wildcard character is specified in the instance name, all instances of the specified object and parent object will be returned if all instance names corresponding to the specified index match the wildcard character.

If a wildcard character is specified in the counter name, all counters of the specified object are returned.

Partial counter path string matches (for example, "pro*") are not supported.

Examples

The following example demonstrates how to this function.

C++

```
#include <windows.h>
#include <stdio.h>
#include <stdlib.h>
#include <pdh.h>
#include <pdhmsg.h>

#pragma comment(lib, "pdh.lib")

CONST PWSTR WILDCARD_PATH = L"\Processor(*)\*";

void wmain(void)
{
    PDH_STATUS Status;
    PWSTR EndOfPaths;
    PWSTR Paths = NULL;
    DWORD BufferSize = 0;

    Status = PdhExpandCounterPath(WILDCARD_PATH, Paths, &BufferSize);

    while (Status == PDH_MORE_DATA)
    {
        Paths = (PWSTR)malloc(BufferSize * sizeof(WCHAR));
        Status = PdhExpandCounterPath(WILDCARD_PATH, Paths, &BufferSize);
    }

    if (Status != ERROR_SUCCESS)
    {
        wprintf(L"\nPdhExpandCounterPath failed with status 0x%x", Status);
        goto Cleanup;
    }

    if (Paths == NULL)
    {
        wprintf(L"\nThe counter path %s cannot be expanded.", WILDCARD_PATH);
        goto Cleanup;
    }

    EndOfPaths = Paths + BufferSize;

    // On Vista and later operating systems, the buffer is terminated with
    // two null-terminator characters; however, on earlier systems, the buffer
    // is not terminated with two null-terminator characters. This covers both
    // cases.
    for (PWSTR p = Paths; ((p != EndOfPaths) && (*p != L'\0')); p +=
wcslen(p) + 1)
    {
        wprintf(L"\n%s", p);
    }
}
```

```
Cleanup:  
    if (Paths)  
    {  
        free(Paths);  
    }  
}
```

ⓘ Note

The pdh.h header defines PdhExpandCounterPath 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 is 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	pdh.h
Library	Pdh.lib
DLL	Pdh.dll

See also

[PdhExpandWildCardPath](#)

[PdhMakeCounterPath](#)

Feedback

Was this page helpful?

 Yes

 No

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

PdhExpandCounterPathW function (pdh.h)

Article02/09/2023

Examines the specified computer (or local computer if none is specified) for counters and instances of counters that match the wildcard strings in the counter path.

Note This function is superseded by the **PdhExpandWildCardPath** function.

Syntax

C++

```
PDH_FUNCTION PdhExpandCounterPathW(
    [in]      LPCWSTR szWildCardPath,
    [out]     PZWSTR mszExpandedPathList,
    [in, out] LPDWORD pcchPathListLength
);
```

Parameters

[in] `szWildCardPath`

Null-terminated string that contains the counter path to expand. The function searches the computer specified in the path for matches. If the path does not specify a computer, the function searches the local computer. The maximum length of a counter path is `PDH_MAX_COUNTER_PATH`.

[out] `mszExpandedPathList`

Caller-allocated buffer that receives the list of expanded counter paths that match the wildcard specification in `szWildCardPath`. Each counter path in this list is terminated by a null character. The list is terminated with two `NUL` characters. Set to `NUL` if `pcchPathListLength` is zero.

[in, out] `pcchPathListLength`

Size of the *mszExpandedPathList* buffer, in TCHARs. If zero on input, the function returns PDH_MORE_DATA and sets this parameter to the required buffer size. If the buffer is larger than the required size, the function sets this parameter to the actual size of the buffer that was used. If the specified size on input is greater than zero but less than the required size, you should not rely on the returned size to reallocate the buffer.

Note You must add one to the required size on Windows XP.

Return value

If the function succeeds, it returns ERROR_SUCCESS.

If the function fails, the return value is a [system error code](#) or a [PDH error code](#).

Return code	Description
PDH_MORE_DATA	The <i>mszExpandedPathList</i> buffer is too small to contain the list of paths. This return value is expected if <i>pcchPathListLength</i> is zero on input. If the specified size on input is greater than zero but less than the required size, you should not rely on the returned size to reallocate the buffer.
PDH_INVALID_ARGUMENT	A parameter is not valid. For example, on some releases you could receive this error if the specified size on input is greater than zero but less than the required size.
PDH_MEMORY_ALLOCATION_FAILURE	Unable to allocate memory to support this function.

Remarks

You should call this function twice, the first time to get the required buffer size (set *mszExpandedPathList* to **NULL** and *pcchPathListLength* to 0), and the second time to get the data.

The general counter path format is as follows:

\computer\object(parent-instance#index)\counter

The parent, instance, index, and counter components of the counter path may contain either a valid name or a wildcard character. The computer, parent, instance, and index components are not necessary for all counters.

The counter paths that you must use is determined by the counter itself. For example, the LogicalDisk object has an instance index, so you must provide the #index, or a wildcard. Therefore, you could use the following format:

\LogicalDisk(/#*)*

In comparison, the Process object does not require an instance index. Therefore, you could use the following format:

\Process(*)\ID Process

The following is a list of the possible formats:

- \\computer\object(parent/instance#index)\counter
- \\computer\object(parent/instance)\counter
- \\computer\object(instance#index)\counter
- \\computer\object(instance)\counter
- \\computer\object\counter
- \object(parent/instance#index)\counter
- \object(parent/instance)\counter
- \object(instance#index)\counter
- \object(instance)\counter
- \object\counter

If a wildcard character is specified in the parent name, all instances of the specified object that match the specified instance and counter fields will be returned.

If a wildcard character is specified in the instance name, all instances of the specified object and parent object will be returned if all instance names corresponding to the specified index match the wildcard character.

If a wildcard character is specified in the counter name, all counters of the specified object are returned.

Partial counter path string matches (for example, "pro*") are not supported.

Examples

The following example demonstrates how to this function.

C++

```
#include <windows.h>
#include <stdio.h>
```

```
#include <stdlib.h>
#include <pdh.h>
#include <pdhmsg.h>

#pragma comment(lib, "pdh.lib")

CONST PWSTR WILDCARD_PATH = L"\Processor(*)\*";

void wmain(void)
{
    PDH_STATUS Status;
    PWSTR EndOfPaths;
    PWSTR Paths = NULL;
    DWORD BufferSize = 0;

    Status = PdhExpandCounterPath(WILDCARD_PATH, Paths, &BufferSize);

    while (Status == PDH_MORE_DATA)
    {
        Paths = (PWSTR)malloc(BufferSize * sizeof(WCHAR));
        Status = PdhExpandCounterPath(WILDCARD_PATH, Paths, &BufferSize);
    }

    if (Status != ERROR_SUCCESS)
    {
        wprintf(L"\nPdhExpandCounterPath failed with status 0x%x", Status);
        goto Cleanup;
    }

    if (Paths == NULL)
    {
        wprintf(L"\nThe counter path %s cannot be expanded.", WILDCARD_PATH);
        goto Cleanup;
    }

    EndOfPaths = Paths + BufferSize;

    // On Vista and later operating systems, the buffer is terminated with
    // two null-terminator characters; however, on earlier systems, the buffer
    // is not terminated with two null-terminator characters. This covers both
    // cases.
    for (PWSTR p = Paths; ((p != EndOfPaths) && (*p != L'\0')); p +=
        wcslen(p) + 1)
    {
        wprintf(L"\n%s", p);
    }

Cleanup:
    if (Paths)
    {
        free(Paths);
    }
}
```

}

ⓘ Note

The pdh.h header defines PdhExpandCounterPath 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 is not encoding-neutral can lead to mismatches that result in compilation or runtime errors. For more information, see [Conventions for Function Prototypes](#).

Requirements

Minimum supported client	Windows XP [desktop apps only]
Minimum supported server	Windows Server 2003 [desktop apps only]
Target Platform	Windows
Header	pdh.h
Library	Pdh.lib
DLL	Pdh.dll

See also

[PdhExpandWildCardPath](#)

[PdhMakeCounterPath](#)

Feedback

Was this page helpful?

 Yes

 No

[Get help at Microsoft Q&A](#)

PdhExpandWildCardPathA function (pdh.h)

Article02/09/2023

Examines the specified computer or log file and returns those counter paths that match the given counter path which contains wildcard characters.

To use handles to data sources, use the [PdhExpandWildCardPathH](#) function.

Syntax

C++

```
PDH_FUNCTION PdhExpandWildCardPathA(
    [in]      LPCSTR  szDataSource,
    [in]      LPCSTR  szWildCardPath,
    [out]     PZZSTR  mszExpandedPathList,
    [in, out] LPDWORD pcchPathListLength,
    [in]      DWORD   dwFlags
);
```

Parameters

[in] `szDataSource`

Null-terminated string that contains the name of a log file. The function uses the performance objects and counters defined in the log file to expand the path specified in the `szWildCardPath` parameter.

If **NULL**, the function searches the computer specified in `szWildCardPath`.

[in] `szWildCardPath`

Null-terminated string that specifies the counter path to expand. The maximum length of a counter path is PDH_MAX_COUNTER_PATH.

If the `szDataSource` parameter is **NULL**, the function searches the computer specified in the path for matches. If the path does not specify a computer, the function searches the local computer.

[out] `mszExpandedPathList`

Caller-allocated buffer that receives a list of **null**-terminated counter paths that match the wildcard specification in the *szWildCardPath*. The list is terminated by two **NULL** characters. Set to **NULL** if *pcchPathListLength* is zero.

[in, out] *pcchPathListLength*

Size of the *mszExpandedPathList* buffer, in **TCHARs**. If zero on input and the object exists, the function returns **PDH_MORE_DATA** and sets this parameter to the required buffer size. If the buffer is larger than the required size, the function sets this parameter to the actual size of the buffer that was used. If the specified size on input is greater than zero but less than the required size, you should not rely on the returned size to reallocate the buffer.

Note You must add one to the required size on Windows XP.

[in] *dwFlags*

Flags that indicate which wildcard characters not to expand. You can specify one or more flags.

[+] Expand table

Value	Meaning
PDH_NOEXPANDCOUNTERS	Do not expand the counter name if the path contains a wildcard character for counter name.
PDH_NOEXPANDINSTANCES	Do not expand the instance name if the path contains a wildcard character for parent instance, instance name, or instance index.
PDH_REFRESHCOUNTERS	Refresh the counter list.

Return value

If the function succeeds, it returns **ERROR_SUCCESS**.

If the function fails, the return value is a [system error code](#) or a [PDH error code](#).

[+] Expand table

Return code	Description
-------------	-------------

PDH_MORE_DATA	The <i>mszExpandedPathList</i> buffer is not large enough to contain the list of paths. This return value is expected if <i>pcchPathListLength</i> is zero on input. If the specified size on input is greater than zero but less than the required size, you should not rely on the returned size to reallocate the buffer.
PDH_INVALID_ARGUMENT	A parameter is not valid. For example, on some releases you could receive this error if the specified size on input is greater than zero but less than the required size.
PDH_INVALID_PATH	The specified object does not contain an instance.
PDH_MEMORY_ALLOCATION_FAILURE	Unable to allocate memory to support this function.
PDH_CSTATUS_NO_OBJECT	Unable to find the specified object on the computer or in the log file.

Remarks

You should call this function twice, the first time to get the required buffer size (set *mszExpandedPathList* to **NULL** and *pcchPathListLength* to 0), and the second time to get the data.

PdhExpandWildCardPath differs from [PdhExpandCounterPath](#) in the following ways:

1. Lets you control which wildcard characters are expanded.
2. The contents of a log file can be used as the source of counter names.

The general counter path format is as follows:

\computer\object(parent/instance#index)\counter

The parent, instance, index, and counter components of the counter path may contain either a valid name or a wildcard character. The computer, parent, instance, and index components are not necessary for all counters.

The following is a list of the possible formats:

- \\computer\object(parent/instance#index)\counter
- \\computer\object(parent/instance)\counter
- \\computer\object(instance#index)\counter
- \\computer\object(instance)\counter
- \\computer\object\counter
- \object(parent/instance#index)\counter
- \object(parent/instance)\counter

- \object(instance#index)\counter
- \object(instance)\counter
- \object\counter

Use an asterisk (*) as the wildcard character, for example, \object(*)\counter.

If a wildcard character is specified in the parent name, all instances of the specified object that match the specified instance and counter fields will be returned. For example, \object(*/instance)\counter.

If a wildcard character is specified in the instance name, all instances of the specified object and parent object will be returned if all instance names corresponding to the specified index match the wildcard character. For example, \object(parent/*)\counter. If the object does not contain an instance, an error occurs.

If a wildcard character is specified in the counter name, all counters of the specified object are returned.

Partial counter path string matches (for example, "pro*") are supported.

Prior to Windows Vista: Partial wildcard matches are not supported.

ⓘ Note

The pdh.h header defines PdhExpandWildCardPath 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 is 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	pdh.h

Requirement	Value
Library	Pdh.lib
DLL	Pdh.dll

See also

[PdhEnumObjectItems](#)

[PdhEnumObjects](#)

[PdhExpandCounterPath](#)

[PdhExpandWildCardPathH](#)

Feedback

Was this page helpful?

 Yes

 No

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

PdhExpandWildCardPathHA function (pdh.h)

Article02/09/2023

Examines the specified computer or log file and returns those counter paths that match the given counter path which contains wildcard characters.

This function is identical to the [PdhExpandWildCardPath](#) function, except that it supports the use of handles to data sources.

Syntax

C++

```
PDH_FUNCTION PdhExpandWildCardPathHA(
    [in]      PDH_HLOG hDataSource,
    [in]      LPCSTR   szWildCardPath,
    [out]     PZZSTR   mszExpandedPathList,
    [in, out] LPDWORD  pcchPathListLength,
    [in]      DWORD    dwFlags
);
```

Parameters

[in] `hDataSource`

Handle to a data source returned by the [PdhBindInputDataSource](#) function.

[in] `szWildCardPath`

Null-terminated string that specifies the counter path to expand. The maximum length of a counter path is PDH_MAX_COUNTER_PATH.

If `hDataSource` is a real time data source, the function searches the computer specified in the path for matches. If the path does not specify a computer, the function searches the local computer.

[out] `mszExpandedPathList`

Caller-allocated buffer that receives a list of null-terminated counter paths that match the wildcard specification in the `szWildCardPath`. The list is terminated by two **NUL** characters. Set to **NUL** if `pcchPathListLength` is zero.

[in, out] pcchPathListLength

Size of the *mszExpandedPathList* buffer, in TCHARs. If zero on input and the object exists, the function returns PDH_MORE_DATA and sets this parameter to the required buffer size. If the buffer is larger than the required size, the function sets this parameter to the actual size of the buffer that was used. If the specified size on input is greater than zero but less than the required size, you should not rely on the returned size to reallocate the buffer.

Note You must add one to the required size on Windows XP.

[in] dwFlags

Flags that indicate which wildcard characters not to expand. You can specify one or more flags.

[[Expand table](#)]

Value	Meaning
PDH_NOEXPANDCOUNTERS	Do not expand the counter name if the path contains a wildcard character for counter name.
PDH_NOEXPANDINSTANCES	Do not expand the instance name if the path contains a wildcard character for parent instance, instance name, or instance index.
PDH_REFRESHCOUNTERS	Refresh the counter list.

Return value

If the function succeeds, it returns ERROR_SUCCESS.

If the function fails, the return value is a [system error code](#) or a [PDH error code](#).

[[Expand table](#)]

Return code	Description
PDH_MORE_DATA	The <i>mszExpandedPathList</i> buffer is not large enough to contain the list of paths. This return value is expected if <i>pcchPathListLength</i> is zero on input. If the specified size on input is greater than zero but less than the required

	size, you should not rely on the returned size to reallocate the buffer.
PDH_INVALID_ARGUMENT	A parameter is not valid. For example, on some releases you could receive this error if the specified size on input is greater than zero but less than the required size.
PDH_MEMORY_ALLOCATION_FAILURE	Unable to allocate memory to support this function.
PDH_CSTATUS_NO_OBJECT	Unable to find the specified object on the computer or in the log file.

Remarks

You should call this function twice, the first time to get the required buffer size (set *mszExpandedPathList* to **NULL** and *pcchPathListLength* to 0), and the second time to get the data.

PdhExpandWildCardPathH differs from [PdhExpandCounterPath](#) in the following ways:

1. Lets you control which wildcard characters are expanded.
2. The contents of a log file can be used as the source of counter names.

The general counter path format is as follows:

\computer\object(parent/instance#index)\counter

The parent, instance, index, and counter components of the counter path may contain either a valid name or a wildcard character. The computer, parent, instance, and index components are not necessary for all counters.

The following is a list of the possible formats:

- \\computer\object(parent/instance#index)\counter
- \\computer\object(parent/instance)\counter
- \\computer\object(instance#index)\counter
- \\computer\object(instance)\counter
- \\computer\object\counter
- \object(parent/instance#index)\counter
- \object(parent/instance)\counter
- \object(instance#index)\counter
- \object(instance)\counter
- \object\counter

Use an asterisk (*) as the wildcard character, for example, \object(*)\counter.

If a wildcard character is specified in the parent name, all instances of the specified object that match the specified instance and counter fields will be returned. For example, \object(*\instance)\counter.

If a wildcard character is specified in the instance name, all instances of the specified object and parent object will be returned if all instance names corresponding to the specified index match the wildcard character. For example, \object(parent*)\counter.

If a wildcard character is specified in the counter name, all counters of the specified object are returned.

Partial counter path string matches (for example, "pro*") are supported.

Prior to Windows Vista: Partial wildcard matches are not supported.

 **Note**

The pdh.h header defines PdhExpandWildCardPathH 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 is 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	pdh.h
Library	Pdh.lib
DLL	Pdh.dll

See also

[PdhBindInputDataSourceA function](#) [PdhEnumObjectItemsHA function](#)

[PdhEnumObjectsHA function](#) [PdhExpandCounterPathA function](#)

Feedback

Was this page helpful?

 Yes

 No

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

PdhExpandWildCardPathHW function (pdh.h)

Article02/09/2023

Examines the specified computer or log file and returns those counter paths that match the given counter path which contains wildcard characters.

This function is identical to the [PdhExpandWildCardPath](#) function, except that it supports the use of handles to data sources.

Syntax

C++

```
PDH_FUNCTION PdhExpandWildCardPathHW(
    [in]      PDH_HLOG hDataSource,
    [in]      LPCWSTR szWildCardPath,
    [out]     PZWSTR  mszExpandedPathList,
    [in, out] LPDWORD pcchPathListLength,
    [in]      DWORD   dwFlags
);
```

Parameters

[in] `hDataSource`

Handle to a data source returned by the [PdhBindInputDataSource](#) function.

[in] `szWildCardPath`

Null-terminated string that specifies the counter path to expand. The maximum length of a counter path is PDH_MAX_COUNTER_PATH.

If `hDataSource` is a real time data source, the function searches the computer specified in the path for matches. If the path does not specify a computer, the function searches the local computer.

[out] `mszExpandedPathList`

Caller-allocated buffer that receives a list of null-terminated counter paths that match the wildcard specification in the `szWildCardPath`. The list is terminated by two **NUL** characters. Set to **NUL** if `pcchPathListLength` is zero.

[in, out] pcchPathListLength

Size of the *mszExpandedPathList* buffer, in TCHARs. If zero on input and the object exists, the function returns PDH_MORE_DATA and sets this parameter to the required buffer size. If the buffer is larger than the required size, the function sets this parameter to the actual size of the buffer that was used. If the specified size on input is greater than zero but less than the required size, you should not rely on the returned size to reallocate the buffer.

Note You must add one to the required size on Windows XP.

[in] dwFlags

Flags that indicate which wildcard characters not to expand. You can specify one or more flags.

Value	Meaning
PDH_NOEXPANDCOUNTERS	Do not expand the counter name if the path contains a wildcard character for counter name.
PDH_NOEXPANDINSTANCES	Do not expand the instance name if the path contains a wildcard character for parent instance, instance name, or instance index.
PDH_REFRESHCOUNTERS	Refresh the counter list.

Return value

If the function succeeds, it returns ERROR_SUCCESS.

If the function fails, the return value is a [system error code](#) or a [PDH error code](#).

Return code	Description
PDH_MORE_DATA	The <i>mszExpandedPathList</i> buffer is not large enough to contain the list of paths. This return value is expected if <i>pcchPathListLength</i> is zero on input. If the specified size on input is greater than zero but less than the required size, you should not rely on the returned size to reallocate the buffer.
PDH_INVALID_ARGUMENT	A parameter is not valid. For example, on some releases you could receive this error if the specified size on input

	is greater than zero but less than the required size.
PDH_MEMORY_ALLOCATION_FAILURE	Unable to allocate memory to support this function.
PDH_CSTATUS_NO_OBJECT	Unable to find the specified object on the computer or in the log file.

Remarks

You should call this function twice, the first time to get the required buffer size (set *mszExpandedPathList* to **NULL** and *pcchPathListLength* to 0), and the second time to get the data.

PdhExpandWildCardPathH differs from [PdhExpandCounterPath](#) in the following ways:

1. Lets you control which wildcard characters are expanded.
2. The contents of a log file can be used as the source of counter names.

The general counter path format is as follows:

\computer\object(parent/instance#index)\counter

The parent, instance, index, and counter components of the counter path may contain either a valid name or a wildcard character. The computer, parent, instance, and index components are not necessary for all counters.

The following is a list of the possible formats:

- \\computer\object(parent/instance#index)\counter
- \\computer\object(parent/instance)\counter
- \\computer\object(instance#index)\counter
- \\computer\object(instance)\counter
- \\computer\object\counter
- \object(parent/instance#index)\counter
- \object(parent/instance)\counter
- \object(instance#index)\counter
- \object(instance)\counter
- \object\counter

Use an asterisk (*) as the wildcard character, for example, \object(*)\counter.

If a wildcard character is specified in the parent name, all instances of the specified object that match the specified instance and counter fields will be returned. For example, \object(*/instance)\counter.

If a wildcard character is specified in the instance name, all instances of the specified object and parent object will be returned if all instance names corresponding to the specified index match the wildcard character. For example, \object(parent/*)\counter.

If a wildcard character is specified in the counter name, all counters of the specified object are returned.

Partial counter path string matches (for example, "pro*") are supported.

Prior to Windows Vista: Partial wildcard matches are not supported.

 **Note**

The pdh.h header defines PdhExpandWildCardPathH 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 is not encoding-neutral can lead to mismatches that result in compilation or runtime errors. For more information, see [Conventions for Function Prototypes](#).

Requirements

Minimum supported client	Windows XP [desktop apps only]
Minimum supported server	Windows Server 2003 [desktop apps only]
Target Platform	Windows
Header	pdh.h
Library	Pdh.lib
DLL	Pdh.dll

See also

[PdhBindInputDataSourceA function](#)

[PdhEnumObjectItemsHA function](#)

[PdhEnumObjectsHA function](#)

[PdhExpandWildCardPathW function](#)

Feedback

Was this page helpful?

 Yes

 No

[Get help at Microsoft Q&A](#)

PdhExpandWildCardPathW function (pdh.h)

Article02/09/2023

Examines the specified computer or log file and returns those counter paths that match the given counter path which contains wildcard characters.

To use handles to data sources, use the [PdhExpandWildCardPathH](#) function.

Syntax

C++

```
PDH_FUNCTION PdhExpandWildCardPathW(
    [in]      LPCWSTR szDataSource,
    [in]      LPCWSTR szWildCardPath,
    [out]     PZZWSTR mszExpandedPathList,
    [in, out] LPDWORD pcchPathListLength,
    [in]      DWORD   dwFlags
);
```

Parameters

[in] *szDataSource*

Null-terminated string that contains the name of a log file. The function uses the performance objects and counters defined in the log file to expand the path specified in the *szWildCardPath* parameter.

If **NULL**, the function searches the computer specified in *szWildCardPath*.

[in] *szWildCardPath*

Null-terminated string that specifies the counter path to expand. The maximum length of a counter path is PDH_MAX_COUNTER_PATH.

If the *szDataSource* parameter is **NULL**, the function searches the computer specified in the path for matches. If the path does not specify a computer, the function searches the local computer.

[out] *mszExpandedPathList*

Caller-allocated buffer that receives a list of **null**-terminated counter paths that match the wildcard specification in the *szWildCardPath*. The list is terminated by two **NULL** characters. Set to **NULL** if *pcchPathListLength* is zero.

[in, out] *pcchPathListLength*

Size of the *mszExpandedPathList* buffer, in **TCHARs**. If zero on input and the object exists, the function returns **PDH_MORE_DATA** and sets this parameter to the required buffer size. If the buffer is larger than the required size, the function sets this parameter to the actual size of the buffer that was used. If the specified size on input is greater than zero but less than the required size, you should not rely on the returned size to reallocate the buffer.

Note You must add one to the required size on Windows XP.

[in] *dwFlags*

Flags that indicate which wildcard characters not to expand. You can specify one or more flags.

Value	Meaning
PDH_NOEXPANDCOUNTERS	Do not expand the counter name if the path contains a wildcard character for counter name.
PDH_NOEXPANDINSTANCES	Do not expand the instance name if the path contains a wildcard character for parent instance, instance name, or instance index.
PDH_REFRESHCOUNTERS	Refresh the counter list.

Return value

If the function succeeds, it returns **ERROR_SUCCESS**.

If the function fails, the return value is a [system error code](#) or a [PDH error code](#).

Return code	Description
PDH_MORE_DATA	The <i>mszExpandedPathList</i> buffer is not large enough to contain the list of paths. This return value is expected if <i>pcchPathListLength</i> is zero on input. If the specified size on input is greater than zero but less than the required

	size, you should not rely on the returned size to reallocate the buffer.
PDH_INVALID_ARGUMENT	A parameter is not valid. For example, on some releases you could receive this error if the specified size on input is greater than zero but less than the required size.
PDH_INVALID_PATH	The specified object does not contain an instance.
PDH_MEMORY_ALLOCATION_FAILURE	Unable to allocate memory to support this function.
PDH_CSTATUS_NO_OBJECT	Unable to find the specified object on the computer or in the log file.

Remarks

You should call this function twice, the first time to get the required buffer size (set *mszExpandedPathList* to **NULL** and *pcchPathListLength* to 0), and the second time to get the data.

PdhExpandWildCardPath differs from [PdhExpandCounterPath](#) in the following ways:

1. Lets you control which wildcard characters are expanded.
2. The contents of a log file can be used as the source of counter names.

The general counter path format is as follows:

\computer\object(parent/instance#index)\counter

The parent, instance, index, and counter components of the counter path may contain either a valid name or a wildcard character. The computer, parent, instance, and index components are not necessary for all counters.

The following is a list of the possible formats:

- \\computer\object(parent/instance#index)\counter
- \\computer\object(parent/instance)\counter
- \\computer\object(instance#index)\counter
- \\computer\object(instance)\counter
- \\computer\object\counter
- \object(parent/instance#index)\counter
- \object(parent/instance)\counter
- \object(instance#index)\counter
- \object(instance)\counter
- \object\counter

Use an asterisk (*) as the wildcard character, for example, \object(*)\counter.

If a wildcard character is specified in the parent name, all instances of the specified object that match the specified instance and counter fields will be returned. For example, \object(*\instance)\counter.

If a wildcard character is specified in the instance name, all instances of the specified object and parent object will be returned if all instance names corresponding to the specified index match the wildcard character. For example, \object(parent/*)\counter. If the object does not contain an instance, an error occurs.

If a wildcard character is specified in the counter name, all counters of the specified object are returned.

Partial counter path string matches (for example, "pro*") are supported.

Prior to Windows Vista: Partial wildcard matches are not supported.

 **Note**

The pdh.h header defines PdhExpandWildCardPath 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 is not encoding-neutral can lead to mismatches that result in compilation or runtime errors. For more information, see [Conventions for Function Prototypes](#).

Requirements

Minimum supported client	Windows XP [desktop apps only]
Minimum supported server	Windows Server 2003 [desktop apps only]
Target Platform	Windows
Header	pdh.h
Library	Pdh.lib
DLL	Pdh.dll

See also

[PdhBindInputDataSourceA function](#)

[PdhEnumObjectItemsHA function](#)

[PdhEnumObjectsHA function](#)

[PdhExpandWildCardPathHW function](#)

Feedback

Was this page helpful?

 Yes

 No

[Get help at Microsoft Q&A](#)

PdhFormatFromRawValue function (pdh.h)

Article 10/13/2021

Computes a displayable value for the given raw counter values.

Syntax

C++

```
PDH_FUNCTION PdhFormatFromRawValue(
    [in]  DWORD          dwCounterType,
    [in]  DWORD          dwFormat,
    [in]  LONGLONG       *pTimeBase,
    [in]  PPDH_RAW_COUNTER pRawValue1,
    [in]  PPDH_RAW_COUNTER pRawValue2,
    [out] PPDH_FMT_COUNTERVALUE pFmtValue
);
```

Parameters

[in] dwCounterType

Type of counter. Typically, you call [PdhGetCounterInfo](#) to retrieve the counter type at the time you call [PdhGetRawCounterValue](#) to retrieve the raw counter value.

For a list of counter types, see the Counter Types section of the [Windows Server 2003 Deployment Kit](#). (The constant values are defined in Winperf.h.)

Note that you cannot specify base types, for example, PERF_LARGE_RAW_BASE.

[in] dwFormat

Determines the data type of the calculated value. Specify one of the following values.

[+] Expand table

Value	Meaning
PDH_FMT_DOUBLE	Return the calculated value as a double-precision floating point real.
	Return the calculated value as a 64-bit integer.

PDH_FMT_LARGE

PDH_FMT_LONG

Return the calculated value as a long integer.

You can use the bitwise inclusive OR operator (\mid) to combine the data type with one of the following scaling factors.

[\[+\] Expand table](#)

Value	Meaning
PDH_FMT_NOSCALE	Do not apply the counter's scaling factor in the calculation.
PDH_FMT_NOCAP100	Counter values greater than 100 (for example, counter values measuring the processor load on multiprocessor computers) will not be reset to 100. The default behavior is that counter values are capped at a value of 100.
PDH_FMT_1000	Multiply the final value by 1,000.

[in] pTimeBase

Pointer to the time base, if necessary for the format conversion. If time base information is not necessary for the format conversion, the value of this parameter is ignored. To retrieve the time base of the counter, call [PdhGetCounterTimeBase](#).

[in] pRawValue1

Raw counter value used to compute the displayable counter value. For details, see [PDH_RAW_COUNTER](#).

[in] pRawValue2

Raw counter value used to compute the displayable counter value. For details, see [PDH_RAW_COUNTER](#). Some counters, for example, rate counters, require two raw values to calculate a displayable value. If the counter type does not require a second value, set this parameter to **NULL**. This value must be the older of the two raw values.

[out] pFmtValue

A [PDH_FMT_COUNTERVALUE](#) structure that receives the calculated counter value.

Return value

If the function succeeds, it returns ERROR_SUCCESS.

If the function fails, the return value is a [system error code](#) or a [PDH error code](#).

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	pdh.h
Library	Pdh.lib
DLL	Pdh.dll

See also

[PDH_FMT_COUNTERVALUE](#)

[PDH_RAW_COUNTER](#)

[PdhGetCounterInfo](#)

[PdhGetCounterTimeBase](#)

[PdhGetRawCounterValue](#)

[PdhReadRawLogRecord](#)

Feedback

Was this page helpful?

 Yes

 No

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

PdhGetCounterInfoA function (pdh.h)

Article02/09/2023

Retrieves information about a counter, such as data size, counter type, path, and user-supplied data values.

Syntax

C++

```
PDH_FUNCTION PdhGetCounterInfoA(
    [in]      PDH_HCOUNTER      hCounter,
    [in]      BOOLEAN          bRetrieveExplainText,
    [in, out] LPDWORD          pdwBufferSize,
    [out]     PPDH_COUNTER_INFO_A lpBuffer
);
```

Parameters

[in] hCounter

Handle of the counter from which you want to retrieve information. The [PdhAddCounter](#) function returns this handle.

[in] bRetrieveExplainText

Determines whether explain text is retrieved. If you set this parameter to **TRUE**, the explain text for the counter is retrieved. If you set this parameter to **FALSE**, the field in the returned buffer is **NULL**.

[in, out] pdwBufferSize

Size of the *lpBuffer* buffer, in bytes. If zero on input, the function returns **PDH_MORE_DATA** and sets this parameter to the required buffer size. If the buffer is larger than the required size, the function sets this parameter to the actual size of the buffer that was used. If the specified size on input is greater than zero but less than the required size, you should not rely on the returned size to reallocate the buffer.

[out] lpBuffer

Caller-allocated buffer that receives a [PDH_COUNTER_INFO](#) structure. The structure is variable-length, because the string data is appended to the end of the fixed-format

portion of the structure. This is done so that all data is returned in a single buffer allocated by the caller. Set to **NULL** if *pdwBufferSize* is zero.

Return value

If the function succeeds, it returns **ERROR_SUCCESS**.

If the function fails, the return value is a [system error code](#) or a [PDH error code](#). The following are possible values.

[+] [Expand table](#)

Return code	Description
PDH_INVALID_ARGUMENT	A parameter is not valid or is incorrectly formatted. For example, on some releases you could receive this error if the specified size on input is greater than zero but less than the required size.
PDH_INVALID_HANDLE	The counter handle is not valid.
PDH_MORE_DATA	The <i>lpBuffer</i> buffer is too small to hold the counter information. This return value is expected if <i>pdwBufferSize</i> is zero on input. If the specified size on input is greater than zero but less than the required size, you should not rely on the returned size to reallocate the buffer.

Remarks

You should call this function twice, the first time to get the required buffer size (set *lpBuffer* to **NULL** and *pdwBufferSize* to 0), and the second time to get the data.

Note

The pdh.h header defines PdhGetCounterInfo 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 is 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	pdh.h
Library	Pdh.lib
DLL	Pdh.dll

See also

[PDH_COUNTER_INFO](#)

Feedback

Was this page helpful?

 Yes

 No

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

PdhGetCounterInfoW function (pdh.h)

Article 02/09/2023

Retrieves information about a counter, such as data size, counter type, path, and user-supplied data values.

Syntax

C++

```
PDH_FUNCTION PdhGetCounterInfoW(
    [in]      PDH_HCOUNTER      hCounter,
    [in]      BOOLEAN          bRetrieveExplainText,
    [in, out] LPDWORD          pdwBufferSize,
    [out]     PPDH_COUNTER_INFO_W lpBuffer
);
```

Parameters

[in] hCounter

Handle of the counter from which you want to retrieve information. The [PdhAddCounter](#) function returns this handle.

[in] bRetrieveExplainText

Determines whether explain text is retrieved. If you set this parameter to **TRUE**, the explain text for the counter is retrieved. If you set this parameter to **FALSE**, the field in the returned buffer is **NULL**.

[in, out] pdwBufferSize

Size of the *lpBuffer* buffer, in bytes. If zero on input, the function returns **PDH_MORE_DATA** and sets this parameter to the required buffer size. If the buffer is larger than the required size, the function sets this parameter to the actual size of the buffer that was used. If the specified size on input is greater than zero but less than the required size, you should not rely on the returned size to reallocate the buffer.

[out] lpBuffer

Caller-allocated buffer that receives a [PDH_COUNTER_INFO](#) structure. The structure is variable-length, because the string data is appended to the end of the fixed-format

portion of the structure. This is done so that all data is returned in a single buffer allocated by the caller. Set to **NULL** if *pdwBufferSize* is zero.

Return value

If the function succeeds, it returns **ERROR_SUCCESS**.

If the function fails, the return value is a [system error code](#) or a [PDH error code](#). The following are possible values.

Return code	Description
PDH_INVALID_ARGUMENT	A parameter is not valid or is incorrectly formatted. For example, on some releases you could receive this error if the specified size on input is greater than zero but less than the required size.
PDH_INVALID_HANDLE	The counter handle is not valid.
PDH_MORE_DATA	The <i>lpBuffer</i> buffer is too small to hold the counter information. This return value is expected if <i>pdwBufferSize</i> is zero on input. If the specified size on input is greater than zero but less than the required size, you should not rely on the returned size to reallocate the buffer.

Remarks

You should call this function twice, the first time to get the required buffer size (set *lpBuffer* to **NULL** and *pdwBufferSize* to 0), and the second time to get the data.

Note

The pdh.h header defines PdhGetCounterInfo 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 is not encoding-neutral can lead to mismatches that result in compilation or runtime errors. For more information, see [Conventions for Function Prototypes](#).

Requirements

Minimum supported client	Windows XP [desktop apps only]
Minimum supported server	Windows Server 2003 [desktop apps only]
Target Platform	Windows
Header	pdh.h
Library	Pdh.lib
DLL	Pdh.dll

See also

[PDH_COUNTER_INFO](#)

Feedback

Was this page helpful?

 Yes

 No

[Get help at Microsoft Q&A](#)

PdhGetCounterTimeBase function (pdh.h)

Article 10/13/2021

Returns the time base of the specified counter.

Syntax

C++

```
PDH_FUNCTION PdhGetCounterTimeBase(
    [in] PDH_HCOUNTER hCounter,
    [out] LONGLONG     *pTimeBase
);
```

Parameters

[in] hCounter

Handle to the counter. The [PdhAddCounter](#) function returns this handle.

[out] pTimeBase

Time base that specifies the number of performance values a counter samples per second.

Return value

If the function succeeds, it returns `ERROR_SUCCESS`.

If the function fails, the return value is a [system error code](#) or a [PDH error code](#). The following are possible values.

[+] Expand table

Return code	Description
<code>PDH_INVALID_ARGUMENT</code>	The specified counter does not use a time base.
<code>PDH_INVALID_HANDLE</code>	The counter handle is not valid.

Remarks

If you use the [PdhFormatFromRawValue](#) function to calculate a displayable value instead of calling the [PdhCalculateCounterFromRawValue](#) function, you must call the [PdhGetCounterTimeBase](#) function to retrieve the time base.

Each counter that returns time-based performance data has a time base defined for it. The time base of a counter is the number of times a counter samples data per second.

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	pdh.h
Library	Pdh.lib
DLL	Pdh.dll

See also

[PdhFormatFromRawValue](#)

Feedback

Was this page helpful?

 Yes

 No

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

PdhGetDataSourceTimeRangeA function (pdh.h)

Article 02/22/2024

Determines the time range, number of entries and, if applicable, the size of the buffer containing the performance data from the specified input source.

To use handles to data sources, use the [PdhGetDataSourceTimeRangeH](#) function.

Syntax

C++

```
PDH_FUNCTION PdhGetDataSourceTimeRangeA(
    [in]    LPCSTR          szDataSource,
    [out]   LPDWORD         pdwNumEntries,
    [out]   PPDH_TIME_INFO pInfo,
    [in]    LPDWORD         pdwBufferSize
);
```

Parameters

[in] `szDataSource`

Null-terminated string that specifies the name of a log file from which the time range information is retrieved.

[out] `pdwNumEntries`

Number of structures in the `pInfo` buffer. This function collects information for only one time range, so the value is typically 1, or zero if an error occurred.

[out] `pInfo`

A [PDH_TIME_INFO](#) structure that receives the time range.

[in] `pdwBufferSize`

Size of the [PDH_TIME_INFO](#) structure, in bytes.

Return value

If the function succeeds, it returns ERROR_SUCCESS.

If the function fails, the return value is a [system error code](#) or a [PDH error code](#). The following are possible values.

 Expand table

Return code	Description
PDH_INVALID_ARGUMENT	A parameter is not valid or is incorrectly formatted.
PDH_INVALID_HANDLE	The counter handle is not valid.
PDH_DATA_SOURCE_IS_REAL_TIME	The current data source is a real-time data source.

Remarks

Note

The pdh.h header defines PdhGetDataSourceTimeRange 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 is 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	pdh.h
Library	Pdh.lib
DLL	Pdh.dll

See also

[PdhGetDataSourceTimeRangeH](#)

[PdhSetQueryTimeRange](#)

Feedback

Was this page helpful?



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

PdhGetDataSourceTimeRangeH function (pdh.h)

Article02/22/2024

Determines the time range, number of entries and, if applicable, the size of the buffer containing the performance data from the specified input source.

This function is identical to the [PdhGetDataSourceTimeRange](#) function, except that it supports the use of handles to data sources.

Syntax

C++

```
PDH_FUNCTION PdhGetDataSourceTimeRangeH(
    [in] PDH_HLOG      hDataSource,
    [out] LPDWORD      pdwNumEntries,
    [out] PPDH_TIME_INFO pInfo,
    [in] LPDWORD      pdwBufferSize
);
```

Parameters

[in] hDataSource

Handle to a data source returned by the [PdhBindInputDataSource](#) function.

[out] pdwNumEntries

Number of structures in the *pInfo* buffer. This function collects information for only one time range, so the value is typically 1, or zero if an error occurred.

[out] pInfo

A [PDH_TIME_INFO](#) structure that receives the time range. The information spans all bound log files.

[in] pdwBufferSize

Size of the [PDH_TIME_INFO](#) structure, in bytes.

Return value

If the function succeeds, it returns ERROR_SUCCESS.

If the function fails, the return value is a [system error code](#) or a [PDH error code](#). The following are possible values.

 [Expand table](#)

Return code	Description
PDH_INVALID_ARGUMENT	A parameter is not valid or is incorrectly formatted.
PDH_INVALID_HANDLE	The counter handle is not valid.
PDH_DATA_SOURCE_IS_REAL_TIME	The current data source is a real-time data source.

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	pdh.h
Library	Pdh.lib
DLL	Pdh.dll

See also

[PdhBindInputDataSource](#)

Feedback

Was this page helpful?

 Yes

 No

PdhGetDataSourceTimeRangeW function (pdh.h)

Article 02/09/2023

Determines the time range, number of entries and, if applicable, the size of the buffer containing the performance data from the specified input source.

To use handles to data sources, use the [PdhGetDataSourceTimeRangeH](#) function.

Syntax

C++

```
PDH_FUNCTION PdhGetDataSourceTimeRangeW(
    [in]    LPCWSTR      szDataSource,
    [out]   LPDWORD      pdwNumEntries,
    [out]   PPDH_TIME_INFO pInfo,
    [in]    LPDWORD      pdwBufferSize
);
```

Parameters

[in] `szDataSource`

Null-terminated string that specifies the name of a log file from which the time range information is retrieved.

[out] `pdwNumEntries`

Number of structures in the `pInfo` buffer. This function collects information for only one time range, so the value is typically 1, or zero if an error occurred.

[out] `pInfo`

A [PDH_TIME_INFO](#) structure that receives the time range.

[in] `pdwBufferSize`

Size of the [PDH_TIME_INFO](#) structure, in bytes.

Return value

If the function succeeds, it returns ERROR_SUCCESS.

If the function fails, the return value is a [system error code](#) or a [PDH error code](#). The following are possible values.

Return code	Description
PDH_INVALID_ARGUMENT	A parameter is not valid or is incorrectly formatted.
PDH_INVALID_HANDLE	The counter handle is not valid.
PDH_DATA_SOURCE_IS_REAL_TIME	The current data source is a real-time data source.

Remarks

Note

The pdh.h header defines PdhGetDataSourceTimeRange 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

Minimum supported client	Windows XP [desktop apps only]
Minimum supported server	Windows Server 2003 [desktop apps only]
Target Platform	Windows
Header	pdh.h
Library	Pdh.lib
DLL	Pdh.dll

See also

[PdhGetDataSourceTimeRangeH](#)

Feedback

Was this page helpful?

 Yes

 No

Get help at Microsoft Q&A

PdhGetDefaultPerfCounterA function (pdh.h)

Article02/09/2023

Retrieves the name of the default counter for the specified object. This name can be used to set the initial counter selection in the Browse Counter dialog box.

To use handles to data sources, use the [PdhGetDefaultPerfCounterH](#) function.

Syntax

C++

```
PDH_FUNCTION PdhGetDefaultPerfCounterA(
    [in]      LPCSTR szDataSource,
    [in]      LPCSTR szMachineName,
    [in]      LPCSTR szObjectName,
    [out]     LPSTR  szDefaultCounterName,
    [in, out] LPDWORD pcchBufferSize
);
```

Parameters

[in] `szDataSource`

Should be **NULL**.

If you specify a log file, `szDefaultCounterName` will be a **null** string.

[in] `szMachineName`

Null-terminated string that specifies the name of the computer used to verify the object name. If **NULL**, the local computer is used to verify the object name.

[in] `szObjectName`

Null-terminated string that specifies the name of the object whose default counter name you want to retrieve.

[out] `szDefaultCounterName`

Caller-allocated buffer that receives the null-terminated default counter name. Set to **NULL** if *pcchBufferSize* is zero.

[in, out] *pcchBufferSize*

Size of the *szDefaultCounterName* buffer, in TCHARs. If zero on input, the function returns PDH_MORE_DATA and sets this parameter to the required buffer size. If the buffer is larger than the required size, the function sets this parameter to the actual size of the buffer that was used. If the specified size on input is greater than zero but less than the required size, you should not rely on the returned size to reallocate the buffer.

Return value

If the function succeeds, it returns ERROR_SUCCESS.

If the function fails, the return value is a [system error code](#) or a [PDH error code](#). The following are possible values.

[+] Expand table

Return code	Description
PDH_MORE_DATA	The <i>szDefaultCounterName</i> buffer is too small to contain the counter name. This return value is expected if <i>pcchBufferSize</i> is zero on input. If the specified size on input is greater than zero but less than the required size, you should not rely on the returned size to reallocate the buffer.
PDH_INVALID_ARGUMENT	A required parameter is not valid. For example, on some releases you could receive this error if the specified size on input is greater than zero but less than the required size.
PDH_MEMORY_ALLOCATION_FAILURE	Unable to allocate memory in order to complete the function.
PDH_CSTATUS_NO_MACHINE	The specified computer is offline or unavailable.
PDH_CSTATUS_NO_COUNTERNAME	The default counter name cannot be read or found.
PDH_CSTATUS_NO_OBJECT	The specified object could not be found.
PDH_CSTATUS_NO_COUNTER	The object did not specify a default counter.

Remarks

You should call this function twice, the first time to get the required buffer size (set *szDefaultCounterName* to **NULL** and *pcchBufferSize* to 0), and the second time to get the data.

Note

The pdh.h header defines PdhGetDefaultPerfCounter 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 is 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	pdh.h
Library	Pdh.lib
DLL	Pdh.dll

See also

[PdhGetDefaultPerfCounterH](#)

[PdhGetDefaultPerfObject](#)

Feedback

Was this page helpful?

 Yes

 No

PdhGetDefaultPerfCounterHA function (pdh.h)

Article 02/09/2023

Retrieves the name of the default counter for the specified object. This name can be used to set the initial counter selection in the Browse Counter dialog box.

This function is identical to [PdhGetDefaultPerfCounter](#), except that it supports the use of handles to data sources.

Syntax

C++

```
PDH_FUNCTION PdhGetDefaultPerfCounterHA(
    [in]      PDH_HLOG hDataSource,
    [in]      LPCSTR   szMachineName,
    [in]      LPCSTR   szObjectName,
    [out]     LPSTR    szDefaultCounterName,
    [in, out] LPDWORD  pcchBufferSize
);
```

Parameters

[in] `hDataSource`

Should be **NULL**.

If you specify a log file handle, `szDefaultCounterName` will be a **null** string.

[in] `szMachineName`

Null-terminated string that specifies the name of the computer used to verify the object name. If **NULL**, the local computer is used to verify the name.

[in] `szObjectName`

Null-terminated string that specifies the name of the object whose default counter name you want to retrieve.

[out] `szDefaultCounterName`

Caller-allocated buffer that receives the null-terminated default counter name. Set to **NULL** if *pcchBufferSize* is zero.

[in, out] *pcchBufferSize*

Size of the *szDefaultCounterName* buffer, in TCHARs. If zero on input, the function returns PDH_MORE_DATA and sets this parameter to the required buffer size. If the buffer is larger than the required size, the function sets this parameter to the actual size of the buffer that was used. If the specified size on input is greater than zero but less than the required size, you should not rely on the returned size to reallocate the buffer.

Return value

If the function succeeds, it returns ERROR_SUCCESS.

If the function fails, the return value is a [system error code](#) or a [PDH error code](#). The following are possible values.

[+] Expand table

Return code	Description
PDH_MORE_DATA	The <i>szDefaultCounterName</i> buffer is too small to contain the counter name. This return value is expected if <i>pcchBufferSize</i> is zero on input. If the specified size on input is greater than zero but less than the required size, you should not rely on the returned size to reallocate the buffer.
PDH_INVALID_ARGUMENT	A required parameter is not valid. For example, on some releases you could receive this error if the specified size on input is greater than zero but less than the required size.
PDH_MEMORY_ALLOCATION_FAILURE	Unable to allocate memory in order to complete the function.
PDH_CSTATUS_NO_MACHINE	The specified computer is offline or unavailable.
PDH_CSTATUS_NO_COUNTERNAME	The default counter name cannot be read or found.
PDH_CSTATUS_NO_OBJECT	The specified object could not be found.
PDH_CSTATUS_NO_COUNTER	The object did not specify a default counter.

Remarks

You should call this function twice, the first time to get the required buffer size (set *szDefaultCounterName* to **NULL** and *pcchBufferSize* to 0), and the second time to get the data.

ⓘ Note

The pdh.h header defines PdhGetDefaultPerfCounterH 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 is 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	pdh.h
Library	Pdh.lib
DLL	Pdh.dll

See also

[PdhBindInputDataSource](#)

[PdhBrowseCountersH](#)

[PdhGetDefaultPerfObjectH](#)

Feedback

Was this page helpful?

 Yes

 No

Provide product feedback  | Get help at Microsoft Q&A

PdhGetDefaultPerfCounterHW function (pdh.h)

Article 02/09/2023

Retrieves the name of the default counter for the specified object. This name can be used to set the initial counter selection in the Browse Counter dialog box.

This function is identical to [PdhGetDefaultPerfCounter](#), except that it supports the use of handles to data sources.

Syntax

C++

```
PDH_FUNCTION PdhGetDefaultPerfCounterHW(
    [in]      PDH_HLOG hDataSource,
    [in]      LPCWSTR szMachineName,
    [in]      LPCWSTR szObjectName,
    [out]     LPWSTR  szDefaultCounterName,
    [in, out] LPDWORD pcchBufferSize
);
```

Parameters

[in] hDataSource

Should be **NULL**.

If you specify a log file handle, *szDefaultCounterName* will be a **null** string.

[in] szMachineName

Null-terminated string that specifies the name of the computer used to verify the object name. If **NULL**, the local computer is used to verify the name.

[in] szObjectName

Null-terminated string that specifies the name of the object whose default counter name you want to retrieve.

[out] szDefaultCounterName

Caller-allocated buffer that receives the null-terminated default counter name. Set to **NULL** if *pcchBufferSize* is zero.

[in, out] *pcchBufferSize*

Size of the *szDefaultCounterName* buffer, in TCHARs. If zero on input, the function returns PDH_MORE_DATA and sets this parameter to the required buffer size. If the buffer is larger than the required size, the function sets this parameter to the actual size of the buffer that was used. If the specified size on input is greater than zero but less than the required size, you should not rely on the returned size to reallocate the buffer.

Return value

If the function succeeds, it returns ERROR_SUCCESS.

If the function fails, the return value is a [system error code](#) or a [PDH error code](#). The following are possible values.

Return code	Description
PDH_MORE_DATA	The <i>szDefaultCounterName</i> buffer is too small to contain the counter name. This return value is expected if <i>pcchBufferSize</i> is zero on input. If the specified size on input is greater than zero but less than the required size, you should not rely on the returned size to reallocate the buffer.
PDH_INVALID_ARGUMENT	A required parameter is not valid. For example, on some releases you could receive this error if the specified size on input is greater than zero but less than the required size.
PDH_MEMORY_ALLOCATION_FAILURE	Unable to allocate memory in order to complete the function.
PDH_CSTATUS_NO_MACHINE	The specified computer is offline or unavailable.
PDH_CSTATUS_NO_COUNTERNAME	The default counter name cannot be read or found.
PDH_CSTATUS_NO_OBJECT	The specified object could not be found.
PDH_CSTATUS_NO_COUNTER	The object did not specify a default counter.

Remarks

You should call this function twice, the first time to get the required buffer size (set *szDefaultCounterName* to **NULL** and *pcchBufferSize* to 0), and the second time to get the data.

ⓘ Note

The pdh.h header defines PdhGetDefaultPerfCounterH 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 is not encoding-neutral can lead to mismatches that result in compilation or runtime errors. For more information, see **Conventions for Function Prototypes**.

Requirements

Minimum supported client	Windows XP [desktop apps only]
Minimum supported server	Windows Server 2003 [desktop apps only]
Target Platform	Windows
Header	pdh.h
Library	Pdh.lib
DLL	Pdh.dll

See also

[PdhBindInputDataSource](#)

[PdhBrowseCountersH](#)

[PdhGetDefaultPerfObjectH](#)

Feedback

Was this page helpful?

 Yes

 No

Get help at Microsoft Q&A

PdhGetDefaultPerfCounterW function (pdh.h)

Article02/09/2023

Retrieves the name of the default counter for the specified object. This name can be used to set the initial counter selection in the Browse Counter dialog box.

To use handles to data sources, use the [PdhGetDefaultPerfCounterH](#) function.

Syntax

C++

```
PDH_FUNCTION PdhGetDefaultPerfCounterW(
    [in]      LPCWSTR szDataSource,
    [in]      LPCWSTR szMachineName,
    [in]      LPCWSTR szObjectName,
    [out]     LPWSTR  szDefaultCounterName,
    [in, out] LPDWORD pcchBufferSize
);
```

Parameters

[in] `szDataSource`

Should be **NULL**.

If you specify a log file, `szDefaultCounterName` will be a **null** string.

[in] `szMachineName`

Null-terminated string that specifies the name of the computer used to verify the object name. If **NULL**, the local computer is used to verify the object name.

[in] `szObjectName`

Null-terminated string that specifies the name of the object whose default counter name you want to retrieve.

[out] `szDefaultCounterName`

Caller-allocated buffer that receives the null-terminated default counter name. Set to **NULL** if *pcchBufferSize* is zero.

[in, out] *pcchBufferSize*

Size of the *szDefaultCounterName* buffer, in TCHARs. If zero on input, the function returns PDH_MORE_DATA and sets this parameter to the required buffer size. If the buffer is larger than the required size, the function sets this parameter to the actual size of the buffer that was used. If the specified size on input is greater than zero but less than the required size, you should not rely on the returned size to reallocate the buffer.

Return value

If the function succeeds, it returns ERROR_SUCCESS.

If the function fails, the return value is a [system error code](#) or a [PDH error code](#). The following are possible values.

Return code	Description
PDH_MORE_DATA	The <i>szDefaultCounterName</i> buffer is too small to contain the counter name. This return value is expected if <i>pcchBufferSize</i> is zero on input. If the specified size on input is greater than zero but less than the required size, you should not rely on the returned size to reallocate the buffer.
PDH_INVALID_ARGUMENT	A required parameter is not valid. For example, on some releases you could receive this error if the specified size on input is greater than zero but less than the required size.
PDH_MEMORY_ALLOCATION_FAILURE	Unable to allocate memory in order to complete the function.
PDH_CSTATUS_NO_MACHINE	The specified computer is offline or unavailable.
PDH_CSTATUS_NO_COUNTERNAME	The default counter name cannot be read or found.
PDH_CSTATUS_NO_OBJECT	The specified object could not be found.
PDH_CSTATUS_NO_COUNTER	The object did not specify a default counter.

Remarks

You should call this function twice, the first time to get the required buffer size (set *szDefaultCounterName* to **NULL** and *pcchBufferSize* to 0), and the second time to get the data.

ⓘ Note

The pdh.h header defines PdhGetDefaultPerfCounter 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 is not encoding-neutral can lead to mismatches that result in compilation or runtime errors. For more information, see **Conventions for Function Prototypes**.

Requirements

Minimum supported client	Windows XP [desktop apps only]
Minimum supported server	Windows Server 2003 [desktop apps only]
Target Platform	Windows
Header	pdh.h
Library	Pdh.lib
DLL	Pdh.dll

See also

[PdhGetDefaultPerfCounterH](#)

[PdhGetDefaultPerfObject](#)

Feedback

Was this page helpful?

Yes

No

[Get help at Microsoft Q&A](#)

PdhGetDefaultPerfObjectA function (pdh.h)

Article 02/09/2023

Retrieves the name of the default object. This name can be used to set the initial object selection in the Browse Counter dialog box.

To use handles to data sources, use the [PdhGetDefaultPerfObjectH](#) function.

Syntax

C++

```
PDH_FUNCTION PdhGetDefaultPerfObjectA(
    [in]      LPCSTR szDataSource,
    [in]      LPCSTR szMachineName,
    [out]     LPSTR  szDefaultObjectName,
    [in, out] LPDWORD pcchBufferSize
);
```

Parameters

[in] `szDataSource`

Should be **NULL**.

If you specify a log file, the `szDefaultObjectName` parameter will be a **null** string.

[in] `szMachineName`

Null-terminated string that specifies the name of the computer used to verify the object name. If **NULL**, the local computer is used to verify the name.

[out] `szDefaultObjectName`

Caller-allocated buffer that receives the **null**-terminated default object name. Set to **NULL** if the `pcchBufferSize` parameter is zero.

Note that PDH always returns Processor for the default object name.

[in, out] `pcchBufferSize`

Size of the *szDefaultObjectName* buffer, in TCHARs. If zero on input, the function returns PDH_MORE_DATA and sets this parameter to the required buffer size. If the buffer is larger than the required size, the function sets this parameter to the actual size of the buffer that was used. If the specified size on input is greater than zero but less than the required size, you should not rely on the returned size to reallocate the buffer.

Return value

If the function succeeds, it returns ERROR_SUCCESS.

If the function fails, the return value is a [system error code](#) or a [PDH error code](#). The following are possible values.

[+] Expand table

Return code	Description
PDH_MORE_DATA	The <i>szDefaultObjectName</i> buffer is too small to contain the object name. This return value is expected if <i>pcchBufferSize</i> is zero on input. If the specified size on input is greater than zero but less than the required size, you should not rely on the returned size to reallocate the buffer.
PDH_INVALID_ARGUMENT	A required parameter is not valid. For example, on some releases you could receive this error if the specified size on input is greater than zero but less than the required size.
PDH_MEMORY_ALLOCATION_FAILURE	Unable to allocate memory in order to complete the function.
PDH_CSTATUS_NO_MACHINE	The specified computer is offline or unavailable.

Remarks

You should call this function twice, the first time to get the required buffer size (set *szDefaultObjectName* to **NULL** and *pcchBufferSize* to 0), and the second time to get the data.

! Note

The pdh.h header defines PdhGetDefaultPerfObject 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 is 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	pdh.h
Library	Pdh.lib
DLL	Pdh.dll

See also

[PdhGetDefaultPerfCounter](#)

[PdhGetDefaultPerfObjectH](#)

Feedback

Was this page helpful?

 Yes

 No

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

PdhGetDefaultPerfObjectHA function (pdh.h)

Article 02/22/2024

Retrieves the name of the default object. This name can be used to set the initial object selection in the Browse Counter dialog box.

This function is identical to the [PdhGetDefaultPerfObject](#) function, except that it supports the use of handles to data sources.

Syntax

C++

```
PDH_FUNCTION PdhGetDefaultPerfObjectHA(  
    [in]      PDH_HLOG hDataSource,  
    [in]      LPCSTR   szMachineName,  
    [out]     LPSTR    szDefaultObjectName,  
    [in, out] LPDWORD  pcchBufferSize  
) ;
```

Parameters

[in] `hDataSource`

Should be **NULL**.

If you specify a log file handle, `szDefaultObjectName` will be a **null** string.

[in] `szMachineName`

Null-terminated string that specifies the name of the computer used to verify the object name. If **NULL**, the local computer is used to verify the name.

[out] `szDefaultObjectName`

Caller-allocated buffer that receives the **null**-terminated default object name. Set to **NULL** if `pcchBufferSize` is zero.

Note that PDH always returns Processor for the default object name.

[in, out] `pcchBufferSize`

Size of the *szDefaultObjectName* buffer, in TCHARs. If zero on input, the function returns PDH_MORE_DATA and sets this parameter to the required buffer size. If the buffer is larger than the required size, the function sets this parameter to the actual size of the buffer that was used. If the specified size on input is greater than zero but less than the required size, you should not rely on the returned size to reallocate the buffer.

Return value

If the function succeeds, it returns ERROR_SUCCESS.

If the function fails, the return value is a [system error code](#) or a [PDH error code](#). The following are possible values.

[+] Expand table

Return code	Description
PDH_MORE_DATA	The <i>szDefaultObjectName</i> buffer is too small to contain the object name. This return value is expected if <i>pcchBufferSize</i> is zero on input. If the specified size on input is greater than zero but less than the required size, you should not rely on the returned size to reallocate the buffer.
PDH_INVALID_ARGUMENT	A required parameter is not valid. For example, on some releases you could receive this error if the specified size on input is greater than zero but less than the required size.
PDH_MEMORY_ALLOCATION_FAILURE	Unable to allocate memory in order to complete the function.
PDH_CSTATUS_NO_MACHINE	The specified computer is offline or unavailable.
PDH_CSTATUS_NO_COUNTERNAME	The default object name cannot be read or found.

Remarks

You should call this function twice, the first time to get the required buffer size (set *szDefaultObjectName* to **NULL** and *pcchBufferSize* to 0), and the second time to get the data.

(!) Note

The pdh.h header defines PdhGetDefaultPerfObjectH 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 is 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	pdh.h
Library	Pdh.lib
DLL	Pdh.dll

See also

[PdhBindInputDataSource](#)

[PdhBrowseCountersH](#)

[PdhGetDefaultPerfCounterH](#)

Feedback

Was this page helpful?

 Yes

 No

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

PdhGetDefaultPerfObjectHW function (pdh.h)

Article 02/09/2023

Retrieves the name of the default object. This name can be used to set the initial object selection in the Browse Counter dialog box.

This function is identical to the [PdhGetDefaultPerfObject](#) function, except that it supports the use of handles to data sources.

Syntax

C++

```
PDH_FUNCTION PdhGetDefaultPerfObjectHW(
    [in]      PDH_HLOG hDataSource,
    [in]      LPCWSTR  szMachineName,
    [out]     LPWSTR   szDefaultObjectName,
    [in, out] LPDWORD  pcchBufferSize
);
```

Parameters

[in] `hDataSource`

Should be **NULL**.

If you specify a log file handle, `szDefaultObjectName` will be a **null** string.

[in] `szMachineName`

Null-terminated string that specifies the name of the computer used to verify the object name. If **NULL**, the local computer is used to verify the name.

[out] `szDefaultObjectName`

Caller-allocated buffer that receives the **null**-terminated default object name. Set to **NULL** if `pcchBufferSize` is zero.

Note that PDH always returns Processor for the default object name.

[in, out] `pcchBufferSize`

Size of the *szDefaultObjectName* buffer, in TCHARs. If zero on input, the function returns PDH_MORE_DATA and sets this parameter to the required buffer size. If the buffer is larger than the required size, the function sets this parameter to the actual size of the buffer that was used. If the specified size on input is greater than zero but less than the required size, you should not rely on the returned size to reallocate the buffer.

Return value

If the function succeeds, it returns ERROR_SUCCESS.

If the function fails, the return value is a [system error code](#) or a [PDH error code](#). The following are possible values.

Return code	Description
PDH_MORE_DATA	The <i>szDefaultObjectName</i> buffer is too small to contain the object name. This return value is expected if <i>pcchBufferSize</i> is zero on input. If the specified size on input is greater than zero but less than the required size, you should not rely on the returned size to reallocate the buffer.
PDH_INVALID_ARGUMENT	A required parameter is not valid. For example, on some releases you could receive this error if the specified size on input is greater than zero but less than the required size.
PDH_MEMORY_ALLOCATION_FAILURE	Unable to allocate memory in order to complete the function.
PDH_CSTATUS_NO_MACHINE	The specified computer is offline or unavailable.
PDH_CSTATUS_NO_COUNTERNAME	The default object name cannot be read or found.

Remarks

You should call this function twice, the first time to get the required buffer size (set *szDefaultObjectName* to **NULL** and *pcchBufferSize* to 0), and the second time to get the data.

Note

The pdh.h header defines PdhGetDefaultPerfObjectH 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 is not encoding-neutral can lead to mismatches that result in compilation or runtime errors. For more information, see [Conventions for Function Prototypes](#).

Requirements

Minimum supported client	Windows XP [desktop apps only]
Minimum supported server	Windows Server 2003 [desktop apps only]
Target Platform	Windows
Header	pdh.h
Library	Pdh.lib
DLL	Pdh.dll

See also

[PdhBindInputDataSource](#)

[PdhBrowseCountersH](#)

[PdhGetDefaultPerfCounterH](#)

Feedback

Was this page helpful?

 Yes

 No

[Get help at Microsoft Q&A](#)

PdhGetDefaultPerfObjectW function (pdh.h)

Article 02/09/2023

Retrieves the name of the default object. This name can be used to set the initial object selection in the Browse Counter dialog box.

To use handles to data sources, use the [PdhGetDefaultPerfObjectH](#) function.

Syntax

C++

```
PDH_FUNCTION PdhGetDefaultPerfObjectW(
    [in]      LPCWSTR szDataSource,
    [in]      LPCWSTR szMachineName,
    [out]     LPWSTR  szDefaultObjectName,
    [in, out] LPDWORD pcchBufferSize
);
```

Parameters

[in] `szDataSource`

Should be **NULL**.

If you specify a log file, the `szDefaultObjectName` parameter will be a **null** string.

[in] `szMachineName`

Null-terminated string that specifies the name of the computer used to verify the object name. If **NULL**, the local computer is used to verify the name.

[out] `szDefaultObjectName`

Caller-allocated buffer that receives the **null**-terminated default object name. Set to **NULL** if the `pcchBufferSize` parameter is zero.

Note that PDH always returns Processor for the default object name.

[in, out] `pcchBufferSize`

Size of the *szDefaultObjectName* buffer, in TCHARs. If zero on input, the function returns PDH_MORE_DATA and sets this parameter to the required buffer size. If the buffer is larger than the required size, the function sets this parameter to the actual size of the buffer that was used. If the specified size on input is greater than zero but less than the required size, you should not rely on the returned size to reallocate the buffer.

Return value

If the function succeeds, it returns ERROR_SUCCESS.

If the function fails, the return value is a [system error code](#) or a [PDH error code](#). The following are possible values.

Return code	Description
PDH_MORE_DATA	The <i>szDefaultObjectName</i> buffer is too small to contain the object name. This return value is expected if <i>pcchBufferSize</i> is zero on input. If the specified size on input is greater than zero but less than the required size, you should not rely on the returned size to reallocate the buffer.
PDH_INVALID_ARGUMENT	A required parameter is not valid. For example, on some releases you could receive this error if the specified size on input is greater than zero but less than the required size.
PDH_MEMORY_ALLOCATION_FAILURE	Unable to allocate memory in order to complete the function.
PDH_CSTATUS_NO_MACHINE	The specified computer is offline or unavailable.

Remarks

You should call this function twice, the first time to get the required buffer size (set *szDefaultObjectName* to **NULL** and *pcchBufferSize* to 0), and the second time to get the data.

Note

The pdh.h header defines PdhGetDefaultPerfObject 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 is not encoding-neutral can lead to mismatches that result in compilation or runtime errors. For more information, see [Conventions for Function Prototypes](#).

Requirements

Minimum supported client	Windows XP [desktop apps only]
Minimum supported server	Windows Server 2003 [desktop apps only]
Target Platform	Windows
Header	pdh.h
Library	Pdh.lib
DLL	Pdh.dll

See also

[PdhGetDefaultPerfCounter](#)

[PdhGetDefaultPerfObjectH](#)

Feedback

Was this page helpful?

 Yes

 No

[Get help at Microsoft Q&A](#)

PdhGetDllVersion function (pdh.h)

Article02/22/2024

Returns the version of the currently installed Pdh.dll file.

Note This function is obsolete and no longer supported.

Syntax

C++

```
PDH_FUNCTION PdhGetDllVersion(  
    [out] LPDWORD lpdwVersion  
);
```

Parameters

[out] lpdwVersion

Pointer to a variable that receives the version of Pdh.dll. This parameter can be one of the following values.

[+] Expand table

Value	Meaning
PDH_CVERSION_WIN50	The file version is a legacy operating system.
PDH_VERSION	The file version is Windows XP.

Return value

If the function succeeds, it returns ERROR_SUCCESS.

If the function fails, the return value is a [system error code](#) or a [PDH error code](#). The following are possible values.

Remarks

This function is used to help in determining the functionality that the currently installed version of Pdh.dll supports.

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	pdh.h
Library	Pdh.lib
DLL	Pdh.dll

Feedback

Was this page helpful?

 Yes

 No

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

PdhGetFormattedCounterArrayA function (pdh.h)

Article 02/09/2023

Returns an array of formatted counter values. Use this function when you want to format the counter values of a counter that contains a wildcard character for the instance name.

Syntax

C++

```
PDH_FUNCTION PdhGetFormattedCounterArrayA(
    [in]      PDH_HCOUNTER           hCounter,
    [in]      DWORD                 dwFormat,
    [in, out] LPDWORD              lpdwBufferSize,
    [out]     LPDWORD              lpdwItemCount,
    [out]     PPDH_FMT_COUNTERVALUE_ITEM_A ItemBuffer
);
```

Parameters

[in] hCounter

Handle to the counter whose current value you want to format. The [PdhAddCounter](#) function returns this handle.

[in] dwFormat

Determines the data type of the formatted value. Specify one of the following values.

[+] Expand table

Value	Meaning
PDH_FMT_DOUBLE	Return data as a double-precision floating point real.
PDH_FMT_LARGE	Return data as a 64-bit integer.
PDH_FMT_LONG	Return data as a long integer.

You can use the bitwise inclusive OR operator (|) to combine the data type with one of the following scaling factors.

[Expand table

Value	Meaning
PDH_FMT_NOSCALE	Do not apply the counter's default scaling factor.
PDH_FMT_NOCAP100	Counter values greater than 100 (for example, counter values measuring the processor load on multiprocessor computers) will not be reset to 100. The default behavior is that counter values are capped at a value of 100.
PDH_FMT_1000	Multiply the actual value by 1,000.

[in, out] lpdwBufferSize

Size of the *ItemBuffer* buffer, in bytes. If zero on input, the function returns PDH_MORE_DATA and sets this parameter to the required buffer size. If the buffer is larger than the required size, the function sets this parameter to the actual size of the buffer that was used. If the specified size on input is greater than zero but less than the required size, you should not rely on the returned size to reallocate the buffer.

[out] lpdwItemCount

Number of counter values in the *ItemBuffer* buffer.

[out] ItemBuffer

Caller-allocated buffer that receives an array of [PDH_FMT_COUNTERVALUE_ITEM](#) structures; the structures contain the counter values. Set to NULL if *lpdwBufferSize* is zero.

Return value

If the function succeeds, it returns ERROR_SUCCESS.

If the function fails, the return value is a [system error code](#) or a [PDH error code](#). The following are possible values.

[Expand table

Return code	Description
-------------	-------------

PDH_MORE_DATA	The <i>ItemBuffer</i> buffer is not large enough to contain the object name. This return value is expected if <i>lpdwBufferSize</i> is zero on input. If the specified size on input is greater than zero but less than the required size, you should not rely on the returned size to reallocate the buffer.
PDH_INVALID_ARGUMENT	A parameter is not valid or is incorrectly formatted. For example, on some releases you could receive this error if the specified size on input is greater than zero but less than the required size.
PDH_INVALID_HANDLE	The counter handle is not valid.

Remarks

You should call this function twice, the first time to get the required buffer size (set *ItemBuffer* to **NULL** and *lpdwBufferSize* to 0), and the second time to get the data.

The data for the counter is locked for the duration of the call to **PdhGetFormattedCounterArray** to prevent any changes during the processing of the call.

Examples

The following example shows how to use this function.

C++

```
#include <windows.h>
#include <stdio.h>
#include <pdh.h>
#include <pdhmsg.h>

#pragma comment(lib, "pdh.lib")

CONST PWSTR COUNTER_PATH = L"\Processor(*)\% Processor Time";
CONST ULONG SAMPLE_INTERVAL_MS = 1000;

void main()
{
    PDH_HQUERY hQuery = NULL;
    PDH_STATUS status = ERROR_SUCCESS;
    PDH_HCOUNTER hCounter = NULL;
    DWORD dwBufferSize = 0;           // Size of the pItems buffer
    DWORD dwItemCount = 0;           // Number of items in the pItems buffer
    PDH_FMT_COUNTERVALUE_ITEM *pItems = NULL; // Array of
```

PDH_FMT_COUNTERVALUE_ITEM structures

```
if (status = PdhOpenQuery(NULL, 0, &hQuery))
{
    wprintf(L"PdhOpenQuery failed with 0x%x.\n", status);
    goto cleanup;
}

// Specify a counter object with a wildcard for the instance.
if (status = PdhAddCounter(hQuery, COUNTER_PATH, 0, &hCounter))
{
    wprintf(L"PdhAddCounter failed with 0x%x.\n", status);
    goto cleanup;
}

// Some counters need two sample in order to format a value, so
// make this call to get the first value before entering the loop.
if (status = PdhCollectQueryData(hQuery))
{
    wprintf(L"PdhCollectQueryData failed with 0x%x.\n", status);
    goto cleanup;
}

for (int i = 0; i < 10; i++)
{
    Sleep(SAMPLE_INTERVAL_MS);

    if (status = PdhCollectQueryData(hQuery))
    {
        wprintf(L"PdhCollectQueryData failed with 0x%x.\n", status);
        goto cleanup;
    }

    // Get the required size of the pItems buffer.
    status = PdhGetFormattedCounterArray(hCounter, PDH_FMT_DOUBLE,
&dwBufferSize, &dwItemCount, pItems);
    if (PDH_MORE_DATA == status)
    {
        pItems = (PDH_FMT_COUNTERVALUE_ITEM *) malloc(dwBufferSize);
        if (pItems)
        {
            status = PdhGetFormattedCounterArray(hCounter,
PDH_FMT_DOUBLE, &dwBufferSize, &dwItemCount, pItems);
            if (ERROR_SUCCESS == status)
            {
                // Loop through the array and print the instance name
and counter value.
                for (DWORD i = 0; i < dwItemCount; i++)
                {
                    wprintf(L"counter: %s, value %.20g\n",
pItems[i].szName, pItems[i].FmtValue.doubleValue);
                }
            }
            else
            {

```

```

        wprintf(L"Second PdhGetFormattedCounterArray call failed
with 0x%x.\n", status);
        goto cleanup;
    }

    free(pItems);
    pItems = NULL;
    dwBufferSize = dwItemCount = 0;
}
else
{
    wprintf(L"malloc for PdhGetFormattedCounterArray
failed.\n");
    goto cleanup;
}
else
{
    wprintf(L"PdhGetFormattedCounterArray failed with 0x%x.\n",
status);
    goto cleanup;
}
}

cleanup:

if (pItems)
    free(pItems);

if (hQuery)
    PdhCloseQuery(hQuery); // Closes all counter handles and the query
handle
}

```

ⓘ Note

The pdh.h header defines PdhGetFormattedCounterArray 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	pdh.h
Library	Pdh.lib
DLL	Pdh.dll

See also

[PDH_FMT_COUNTERVALUE_ITEM](#)

[PdhAddCounter](#)

[PdhGetFormattedCounterValue](#)

[PdhGetRawCounterArray](#)

[PdhGetRawCounterValue](#)

Feedback

Was this page helpful?

 Yes

 No

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

PdhGetFormattedCounterArrayW function (pdh.h)

Article 02/09/2023

Returns an array of formatted counter values. Use this function when you want to format the counter values of a counter that contains a wildcard character for the instance name.

Syntax

C++

```
PDH_FUNCTION PdhGetFormattedCounterArrayW(
    [in]      PDH_HCOUNTER           hCounter,
    [in]      DWORD                 dwFormat,
    [in, out] LPDWORD              lpdwBufferSize,
    [out]     LPDWORD              lpdwItemCount,
    [out]     PPDH_FMT_COUNTERVALUE_ITEM_W ItemBuffer
);
```

Parameters

[in] hCounter

Handle to the counter whose current value you want to format. The [PdhAddCounter](#) function returns this handle.

[in] dwFormat

Determines the data type of the formatted value. Specify one of the following values.

Value	Meaning
PDH_FMT_DOUBLE	Return data as a double-precision floating point real.
PDH_FMT_LARGE	Return data as a 64-bit integer.
PDH_FMT_LONG	Return data as a long integer.

You can use the bitwise inclusive OR operator (`|`) to combine the data type with one of the following scaling factors.

Value	Meaning
PDH_FMT_NOSCALE	Do not apply the counter's default scaling factor.
PDH_FMT_NOCAP100	Counter values greater than 100 (for example, counter values measuring the processor load on multiprocessor computers) will not be reset to 100. The default behavior is that counter values are capped at a value of 100.
PDH_FMT_1000	Multiply the actual value by 1,000.

[in, out] *lpdwBufferSize*

Size of the *ItemBuffer* buffer, in bytes. If zero on input, the function returns PDH_MORE_DATA and sets this parameter to the required buffer size. If the buffer is larger than the required size, the function sets this parameter to the actual size of the buffer that was used. If the specified size on input is greater than zero but less than the required size, you should not rely on the returned size to reallocate the buffer.

[out] *lpdwItemCount*

Number of counter values in the *ItemBuffer* buffer.

[out] *ItemBuffer*

Caller-allocated buffer that receives an array of [PDH_FMT_COUNTERVALUE_ITEM](#) structures; the structures contain the counter values. Set to **NULL** if *lpdwBufferSize* is zero.

Return value

If the function succeeds, it returns **ERROR_SUCCESS**.

If the function fails, the return value is a [system error code](#) or a [PDH error code](#). The following are possible values.

Return code	Description
PDH_MORE_DATA	The <i>ItemBuffer</i> buffer is not large enough to contain the object name. This return value is expected if <i>lpdwBufferSize</i> is zero on input. If the specified size on input is greater than zero but less than the required size, you should not rely on the returned size to reallocate the buffer.
PDH_INVALID_ARGUMENT	A parameter is not valid or is incorrectly formatted. For example, on some releases you could receive this error if

	the specified size on input is greater than zero but less than the required size.
PDH_INVALID_HANDLE	The counter handle is not valid.

Remarks

You should call this function twice, the first time to get the required buffer size (set *ItemBuffer* to **NULL** and *lpdwBufferSize* to 0), and the second time to get the data.

The data for the counter is locked for the duration of the call to **PdhGetFormattedCounterArray** to prevent any changes during the processing of the call.

Examples

The following example shows how to use this function.

C++

```
#include <windows.h>
#include <stdio.h>
#include <pdh.h>
#include <pdhmsg.h>

#pragma comment(lib, "pdh.lib")

CONST PWSTR COUNTER_PATH = L"\Processor(*)\% Processor Time";
CONST ULONG SAMPLE_INTERVAL_MS = 1000;

void main()
{
    PDH_HQUERY hQuery = NULL;
    PDH_STATUS status = ERROR_SUCCESS;
    PDH_HCOUNTER hCounter = NULL;
    DWORD dwBufferSize = 0;           // Size of the pItems buffer
    DWORD dwItemCount = 0;           // Number of items in the pItems buffer
    PDH_FMT_COUNTERVALUE_ITEM *pItems = NULL; // Array of
    PDH_FMT_COUNTERVALUE_ITEM structures

    if (status = PdhOpenQuery(NULL, 0, &hQuery))
    {
        wprintf(L"PdhOpenQuery failed with 0x%x.\n", status);
        goto cleanup;
    }

    // Specify a counter object with a wildcard for the instance.
    if (status = PdhAddCounter(hQuery, COUNTER_PATH, 0, &hCounter))

```

```

{
    wprintf(L"PdhAddCounter failed with 0x%x.\n", status);
    goto cleanup;
}

// Some counters need two sample in order to format a value, so
// make this call to get the first value before entering the loop.
if (status = PdhCollectQueryData(hQuery))
{
    wprintf(L"PdhCollectQueryData failed with 0x%x.\n", status);
    goto cleanup;
}

for (int i = 0; i < 10; i++)
{
    Sleep(SAMPLE_INTERVAL_MS);

    if (status = PdhCollectQueryData(hQuery))
    {
        wprintf(L"PdhCollectQueryData failed with 0x%x.\n", status);
        goto cleanup;
    }

    // Get the required size of the pItems buffer.
    status = PdhGetFormattedCounterArray(hCounter, PDH_FMT_DOUBLE,
&dwBufferSize, &dwItemCount, pItems);
    if (PDH_MORE_DATA == status)
    {
        pItems = (PDH_FMT_COUNTERVALUE_ITEM *) malloc(dwBufferSize);
        if (pItems)
        {
            status = PdhGetFormattedCounterArray(hCounter,
PDH_FMT_DOUBLE, &dwBufferSize, &dwItemCount, pItems);
            if (ERROR_SUCCESS == status)
            {
                // Loop through the array and print the instance name
                // and counter value.
                for (DWORD i = 0; i < dwItemCount; i++)
                {
                    wprintf(L"counter: %s, value %.20g\n",
pItems[i].szName, pItems[i].FmtValue.doubleValue);
                }
            }
            else
            {
                wprintf(L"Second PdhGetFormattedCounterArray call failed
with 0x%x.\n", status);
                goto cleanup;
            }
        }
        free(pItems);
        pItems = NULL;
        dwBufferSize = dwItemCount = 0;
    }
    else

```

```

    {
        wprintf(L"malloc for PdhGetFormattedCounterArray
failed.\n");
        goto cleanup;
    }
}
else
{
    wprintf(L"PdhGetFormattedCounterArray failed with 0x%x.\n",
status);
    goto cleanup;
}
}

cleanup:

if (pItems)
    free(pItems);

if (hQuery)
    PdhCloseQuery(hQuery); // Closes all counter handles and the query
handle
}

```

ⓘ Note

The pdh.h header defines PdhGetFormattedCounterArray 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 is not encoding-neutral can lead to mismatches that result in compilation or runtime errors. For more information, see [Conventions for Function Prototypes](#).

Requirements

Minimum supported client	Windows XP [desktop apps only]
Minimum supported server	Windows Server 2003 [desktop apps only]
Target Platform	Windows
Header	pdh.h
Library	Pdh.lib

DLL

Pdh.dll

See also

[PDH_FMT_COUNTERVALUE_ITEM](#)

[PdhAddCounter](#)

[PdhGetFormattedCounterValue](#)

[PdhGetRawCounterArray](#)

[PdhGetRawCounterValue](#)

Feedback

Was this page helpful?

 Yes

 No

[Get help at Microsoft Q&A](#)

PdhGetFormattedCounterValue function (pdh.h)

Article 10/13/2021

Computes a displayable value for the specified counter.

Syntax

C++

```
PDH_FUNCTION PdhGetFormattedCounterValue(
    [in] PDH_HCOUNTER           hCounter,
    [in] DWORD                  dwFormat,
    [out] LPDWORD                lpdwType,
    [out] PPDH_FMT_COUNTERVALUE pValue
);
```

Parameters

[in] hCounter

Handle of the counter for which you want to compute a displayable value. The [PdhAddCounter](#) function returns this handle.

[in] dwFormat

Determines the data type of the formatted value. Specify one of the following values.

[] [Expand table](#)

Value	Meaning
PDH_FMT_DOUBLE	Return data as a double-precision floating point real.
PDH_FMT_LARGE	Return data as a 64-bit integer.
PDH_FMT_LONG	Return data as a long integer.

You can use the bitwise inclusive OR operator (`|`) to combine the data type with one of the following scaling factors.

Value	Meaning
PDH_FMT_NOSCALE	Do not apply the counter's default scaling factor.
PDH_FMT_NOCAP100	Counter values greater than 100 (for example, counter values measuring the processor load on multiprocessor computers) will not be reset to 100. The default behavior is that counter values are capped at a value of 100.
PDH_FMT_1000	Multiply the actual value by 1,000.

[out] lpdwType

Receives the counter type. For a list of counter types, see the Counter Types section of the [Windows Server 2003 Deployment Kit](#). This parameter is optional.

[out] pValue

A [PDH_FMT_COUNTERVALUE](#) structure that receives the counter value.

Return value

If the function succeeds, it returns `ERROR_SUCCESS`.

If the function fails, the return value is a [system error code](#) or a [PDH error code](#). The following are possible values.

Return code	Description
PDH_INVALID_ARGUMENT	A parameter is not valid or is incorrectly formatted.
PDH_INVALID_DATA	The specified counter does not contain valid data or a successful status code.
PDH_INVALID_HANDLE	The counter handle is not valid.

Remarks

The data for the counter is locked (protected) for the duration of the call to `PdhGetFormattedCounterValue` to prevent any changes during the processing of the

call. Reading the data (calling this function successfully) clears the data-changed flag for the counter.

Some counters, such as rate counters, require two counter values in order to compute a displayable value. In this case you must call [PdhCollectQueryData](#) twice before calling [PdhGetFormattedCounterValue](#). For more information, see [Collecting Performance Data](#).

If the specified counter instance does not exist, the method will return **PDH_INVALID_DATA** and set the **CStatus** member of the [PDH_FMT_COUNTERVALUE](#) structure to **PDH_CSTATUS_NO_INSTANCE**.

Prior to Windows Server 2003: The format call may fail for counters that require only a single value when the instance is not found. Try calling the query and format calls again. If the format call fails the second time, the instance is not found. As an alternative, you can call the [PdhEnumObjects](#) function with the refresh option set to **TRUE** to refresh the counter instances before querying and formatting the counter data.

Examples

For an example, see [Browsing Performance Counters](#) or [Reading Performance Data from a Log File](#).

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	pdh.h
Library	Pdh.lib
DLL	Pdh.dll

See also

[PdhCollectQueryData](#)

[PdhGetRawCounterValue](#)

[PdhSetCounterScaleFactor](#)

Feedback

Was this page helpful?

 Yes

 No

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

PdhGetLogFileSize function (pdh.h)

Article02/22/2024

Returns the size of the specified log file.

Syntax

C++

```
PDH_FUNCTION PdhGetLogFileSize(
    [in] PDH_HLOG hLog,
    [out] LONGLONG *l1Size
);
```

Parameters

[in] hLog

Handle to the log file. The [PdhOpenLog](#) or [PdhBindInputDataSource](#) function returns this handle.

[out] l1Size

Size of the log file, in bytes.

Return value

If the function succeeds, it returns ERROR_SUCCESS.

If the function fails, the return value is a [system error code](#) or a [PDH error code](#). The following are possible values.

[+] Expand table

Return code	Description
PDH_LOG_FILE_OPEN_ERROR	An error occurred when trying to open the log file.
PDH_INVALID_HANDLE	The handle is not valid.

Remarks

If the log file handle points to multiple bound log files, the size is the sum of all the log files. If the log file is a SQL log file, the *lFileSize* parameter is the number of records in the log file.

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	pdh.h
Library	Pdh.lib
DLL	Pdh.dll

See also

[PdhOpenLog](#)

Feedback

Was this page helpful?

 Yes

 No

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

PdhGetRawCounterArrayA function (pdh.h)

Article 02/09/2023

Returns an array of raw values from the specified counter. Use this function when you want to retrieve the raw counter values of a counter that contains a wildcard character for the instance name.

Syntax

C++

```
PDH_FUNCTION PdhGetRawCounterArrayA(
    [in]      PDH_HCOUNTER           hCounter,
    [in, out] LPDWORD               lpdwBufferSize,
    [out]     LPDWORD               lpdwItemCount,
    [out]     PPDH_RAW_COUNTER_ITEM_A ItemBuffer
);
```

Parameters

[in] hCounter

Handle of the counter for whose current raw instance values you want to retrieve. The [PdhAddCounter](#) function returns this handle.

[in, out] lpdwBufferSize

Size of the *ItemBuffer* buffer, in bytes. If zero on input, the function returns PDH_MORE_DATA and sets this parameter to the required buffer size. If the buffer is larger than the required size, the function sets this parameter to the actual size of the buffer that was used. If the specified size on input is greater than zero but less than the required size, you should not rely on the returned size to reallocate the buffer.

[out] lpdwItemCount

Number of raw counter values in the *ItemBuffer* buffer.

[out] ItemBuffer

Caller-allocated buffer that receives the array of [PDH_RAW_COUNTER_ITEM](#) structures; the structures contain the raw instance counter values. Set to **NULL** if *lpdwBufferSize* is zero.

Return value

If the function succeeds, it returns [ERROR_SUCCESS](#).

If the function fails, the return value is a [system error code](#) or a [PDH error code](#). The following are possible values.

[] [Expand table](#)

Return code	Description
PDH_MORE_DATA	The <i>ItemBuffer</i> buffer is not large enough to contain the object name. This return value is expected if <i>lpdwBufferSize</i> is zero on input. If the specified size on input is greater than zero but less than the required size, you should not rely on the returned size to reallocate the buffer.
PDH_INVALID_ARGUMENT	A parameter is not valid or is incorrectly formatted. For example, on some releases you could receive this error if the specified size on input is greater than zero but less than the required size.
PDH_INVALID_HANDLE	The counter handle is not valid.

Remarks

You should call this function twice, the first time to get the required buffer size (set *ItemBuffer* to **NULL** and *lpdwBufferSize* to 0), and the second time to get the data.

The data for the counter is locked for the duration of the call to [PdhGetRawCounterArray](#) to prevent any changes during processing of the call.

Note

The pdh.h header defines PdhGetRawCounterArray 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 is 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	pdh.h
Library	Pdh.lib
DLL	Pdh.dll

See also

[PDH_RAW_COUNTER_ITEM](#)

[PdhCalculateCounterFromRawValue](#)

[PdhGetFormattedCounterArray](#)

[PdhGetFormattedCounterValue](#)

Feedback

Was this page helpful?

 Yes

 No

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

PdhGetRawCounterArrayW function (pdh.h)

Article 02/09/2023

Returns an array of raw values from the specified counter. Use this function when you want to retrieve the raw counter values of a counter that contains a wildcard character for the instance name.

Syntax

C++

```
PDH_FUNCTION PdhGetRawCounterArrayW(
    [in]      PDH_HCOUNTER          hCounter,
    [in, out] LPDWORD              lpdwBufferSize,
    [out]     LPDWORD              lpdwItemCount,
    [out]     PPDH_RAW_COUNTER_ITEM_W ItemBuffer
);
```

Parameters

[in] hCounter

Handle of the counter for whose current raw instance values you want to retrieve. The [PdhAddCounter](#) function returns this handle.

[in, out] lpdwBufferSize

Size of the *ItemBuffer* buffer, in bytes. If zero on input, the function returns PDH_MORE_DATA and sets this parameter to the required buffer size. If the buffer is larger than the required size, the function sets this parameter to the actual size of the buffer that was used. If the specified size on input is greater than zero but less than the required size, you should not rely on the returned size to reallocate the buffer.

[out] lpdwItemCount

Number of raw counter values in the *ItemBuffer* buffer.

[out] ItemBuffer

Caller-allocated buffer that receives the array of [PDH_RAW_COUNTER_ITEM](#) structures; the structures contain the raw instance counter values. Set to **NULL** if *lpdwBufferSize* is zero.

Return value

If the function succeeds, it returns [ERROR_SUCCESS](#).

If the function fails, the return value is a [system error code](#) or a [PDH error code](#). The following are possible values.

Return code	Description
PDH_MORE_DATA	The <i>ItemBuffer</i> buffer is not large enough to contain the object name. This return value is expected if <i>lpdwBufferSize</i> is zero on input. If the specified size on input is greater than zero but less than the required size, you should not rely on the returned size to reallocate the buffer.
PDH_INVALID_ARGUMENT	A parameter is not valid or is incorrectly formatted. For example, on some releases you could receive this error if the specified size on input is greater than zero but less than the required size.
PDH_INVALID_HANDLE	The counter handle is not valid.

Remarks

You should call this function twice, the first time to get the required buffer size (set *ItemBuffer* to **NULL** and *lpdwBufferSize* to 0), and the second time to get the data.

The data for the counter is locked for the duration of the call to [PdhGetRawCounterArray](#) to prevent any changes during processing of the call.

Note

The pdh.h header defines PdhGetRawCounterArray 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 is not encoding-neutral can lead to mismatches that result in compilation or runtime errors. For more information, see [Conventions for Function Prototypes](#).

Requirements

Minimum supported client	Windows XP [desktop apps only]
Minimum supported server	Windows Server 2003 [desktop apps only]
Target Platform	Windows
Header	pdh.h
Library	Pdh.lib
DLL	Pdh.dll

See also

[PDH_RAW_COUNTER_ITEM](#)

[PdhCalculateCounterFromRawValue](#)

[PdhGetFormattedCounterArray](#)

[PdhGetFormattedCounterValue](#)

Feedback

Was this page helpful?



[Get help at Microsoft Q&A](#)

PdhGetRawCounterValue function (pdh.h)

Article 02/22/2024

Returns the current raw value of the counter.

Syntax

C++

```
PDH_FUNCTION PdhGetRawCounterValue(
    [in] PDH_HCOUNTER hCounter,
    [out] LPDWORD     lpdwType,
    [out] PPDH_RAW_COUNTER pValue
);
```

Parameters

[in] hCounter

Handle of the counter from which to retrieve the current raw value. The [PdhAddCounter](#) function returns this handle.

[out] lpdwType

Receives the counter type. For a list of counter types, see the Counter Types section of the [Windows Server 2003 Deployment Kit](#). This parameter is optional.

[out] pValue

A [PDH_RAW_COUNTER](#) structure that receives the counter value.

Return value

If the function succeeds, it returns [ERROR_SUCCESS](#).

If the function fails, the return value is a [system error code](#) or a [PDH error code](#). The following are possible values.

[+] Expand table

Return code	Description
PDH_INVALID_ARGUMENT	A parameter is not valid or is incorrectly formatted.
PDH_INVALID_HANDLE	The counter handle is not valid.

Remarks

The data for the counter is locked (protected) for the duration of the call to [PdhGetRawCounterValue](#) to prevent any changes during processing of the call.

If the specified counter instance does not exist, this function will return [ERROR_SUCCESS](#) and the [CStatus](#) member of the [PDH_RAW_COUNTER](#) structure will contain [PDH_CSTATUS_NO_INSTANCE](#).

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	pdh.h
Library	Pdh.lib
DLL	Pdh.dll

See also

[PdhCalculateCounterFromRawValue](#)

[PdhCollectQueryData](#)

[PdhGetFormattedCounterValue](#)

Feedback

Was this page helpful?

 Yes

 No

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

PdhIsRealTimeQuery function (pdh.h)

Article10/13/2021

Determines if the specified query is a real-time query.

Syntax

C++

```
BOOL PdhIsRealTimeQuery(
    [in] PDH_HQUERY hQuery
);
```

Parameters

[in] hQuery

Handle to the query. The [PdhOpenQuery](#) function returns this handle.

Return value

If the query is a real-time query, the return value is **TRUE**.

If the query is not a real-time query, the return value is **FALSE**.

Remarks

The term *real-time* as used in the description of this function does not imply the standard meaning of the term *real-time*. Instead, it describes the collection of performance data from a source providing current information (for example, the registry or a WMI provider) rather than from a log file.

Requirements

[] [Expand table](#)

Requirement	Value
Minimum supported client	Windows XP [desktop apps only]

Requirement	Value
Minimum supported server	Windows Server 2003 [desktop apps only]
Target Platform	Windows
Header	pdh.h
Library	Pdh.lib
DLL	Pdh.dll

See also

[PdhOpenQuery](#)

[PdhSelectDataSource](#)

Feedback

Was this page helpful?

 Yes

 No

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

PdhLookupPerfIndexByNameA function (pdh.h)

Article 02/09/2023

Returns the counter index corresponding to the specified counter name.

Syntax

C++

```
PDH_FUNCTION PdhLookupPerfIndexByNameA(
    [in]  LPCSTR  szMachineName,
    [in]  LPCSTR  szNameBuffer,
    [out] LPDWORD pdwIndex
);
```

Parameters

[in] szMachineName

Null-terminated string that specifies the name of the computer where the specified counter is located. The computer name can be specified by the DNS name or the IP address. If **NULL**, the function uses the local computer.

[in] szNameBuffer

Null-terminated string that contains the counter name.

[out] pdwIndex

Index of the counter.

Return value

If the function succeeds, it returns **ERROR_SUCCESS**.

If the function fails, the return value is a [system error code](#) or a [PDH error code](#). The following is a possible value.

[+] Expand table

Return code	Description
PDH_INVALID_ARGUMENT	A parameter is not valid or is incorrectly formatted.

Remarks

Note

The pdh.h header defines PdhLookupPerfIndexByName 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 is 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	pdh.h
Library	Pdh.lib
DLL	Pdh.dll

See also

[PdhLookupPerfNameByIndex](#)

Feedback

Was this page helpful?

 Yes

 No

Provide product feedback  | Get help at Microsoft Q&A

PdhLookupPerfIndexByNameW function (pdh.h)

Article 02/09/2023

Returns the counter index corresponding to the specified counter name.

Syntax

C++

```
PDH_FUNCTION PdhLookupPerfIndexByNameW(
    [in]  LPCWSTR szMachineName,
    [in]  LPCWSTR szNameBuffer,
    [out] LPDWORD pdwIndex
);
```

Parameters

[in] szMachineName

Null-terminated string that specifies the name of the computer where the specified counter is located. The computer name can be specified by the DNS name or the IP address. If **NULL**, the function uses the local computer.

[in] szNameBuffer

Null-terminated string that contains the counter name.

[out] pdwIndex

Index of the counter.

Return value

If the function succeeds, it returns **ERROR_SUCCESS**.

If the function fails, the return value is a [system error code](#) or a [PDH error code](#). The following is a possible value.

Return code	Description

Remarks

ⓘ Note

The pdh.h header defines PdhLookupPerfIndexByName 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 is not encoding-neutral can lead to mismatches that result in compilation or runtime errors. For more information, see [Conventions for Function Prototypes](#).

Requirements

Minimum supported client	Windows XP [desktop apps only]
Minimum supported server	Windows Server 2003 [desktop apps only]
Target Platform	Windows
Header	pdh.h
Library	Pdh.lib
DLL	Pdh.dll

See also

[PdhLookupPerfNameByIndex](#)

Feedback

Was this page helpful?

 Yes

 No

[Get help at Microsoft Q&A](#)

PdhLookupPerfNameByIndexA function (pdh.h)

Article 02/09/2023

Returns the performance object name or counter name corresponding to the specified index.

Syntax

C++

```
PDH_FUNCTION PdhLookupPerfNameByIndexA(
    [in]      LPCSTR  szMachineName,
    [in]      DWORD   dwNameIndex,
    [out]     LPSTR   szNameBuffer,
    [in, out] LPDWORD pcchNameBufferSize
);
```

Parameters

[in] szMachineName

Null-terminated string that specifies the name of the computer where the specified performance object or counter is located. The computer name can be specified by the DNS name or the IP address. If **NULL**, the function uses the local computer.

[in] dwNameIndex

Index of the performance object or counter.

[out] szNameBuffer

Caller-allocated buffer that receives the null-terminated name of the performance object or counter. Set to **NULL** if *pcchNameBufferSize* is zero.

[in, out] pcchNameBufferSize

Size of the *szNameBuffer* buffer, in TCHARs. If zero on input, the function returns PDH_MORE_DATA and sets this parameter to the required buffer size. If the buffer is larger than the required size, the function sets this parameter to the actual size of the

buffer that was used. If the specified size on input is greater than zero but less than the required size, you should not rely on the returned size to reallocate the buffer.

Return value

If the function succeeds, it returns ERROR_SUCCESS.

If the function fails, the return value is a [system error code](#) or a [PDH error code](#). The following are possible values.

[+] Expand table

Return code	Description
PDH_MORE_DATA	The <i>szNameBuffer</i> buffer is not large enough to contain the counter name. This return value is expected if <i>pcchNameBufferSize</i> is zero on input. If the specified size on input is greater than zero but less than the required size, you should not rely on the returned size to reallocate the buffer.
PDH_INVALID_ARGUMENT	A parameter is not valid or is incorrectly formatted. For example, on some releases you could receive this error if the specified size on input is greater than zero but less than the required size.

Remarks

You should call this function twice, the first time to get the required buffer size (set *szNameBuffer* to **NULL** and *pcchNameBufferSize* to 0), and the second time to get the data.

Windows XP: You must specify a buffer and buffer size. The function sets *pcchNameBufferSize* to either the required size or the size of the buffer that was used. If the buffer is too small, the function returns PDH_INSUFFICIENT_BUFFER instead of PDH_MORE_DATA. The maximum string size in bytes is PDH_MAX_COUNTER_NAME * sizeof(TCHAR).

The index value that you specify must match one of the index values associated with the objects or counters that were loaded on the computer. The index/name value pairs are stored in the **Counters** registry value in the following registry location.

HKEY_LOCAL_MACHINE
 \SOFTWARE

```
\Microsoft
  \Windows NT
    \CurrentVersion
      \Perflib
        Last Counter = highest counter index
        Last Help = highest help index
        \009
          Counters = 2 System 4 Memory...
          Help = 3 The System Object Type...
        \supported Language, other than English
          Counters = ...
          Help = ...
```

ⓘ Note

The pdh.h header defines PdhLookupPerfNameByIndex 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 is 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	pdh.h
Library	Pdh.lib
DLL	Pdh.dll

See also

[PdhLookupPerfIndexByName](#)

Feedback

Was this page helpful?

 Yes

 No

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

PdhLookupPerfNameByIndexW function (pdh.h)

Article 02/09/2023

Returns the performance object name or counter name corresponding to the specified index.

Syntax

C++

```
PDH_FUNCTION PdhLookupPerfNameByIndexW(
    [in]      LPCWSTR szMachineName,
    [in]      DWORD   dwNameIndex,
    [out]     LPWSTR  szNameBuffer,
    [in, out] LPDWORD pcchNameBufferSize
);
```

Parameters

[in] szMachineName

Null-terminated string that specifies the name of the computer where the specified performance object or counter is located. The computer name can be specified by the DNS name or the IP address. If **NULL**, the function uses the local computer.

[in] dwNameIndex

Index of the performance object or counter.

[out] szNameBuffer

Caller-allocated buffer that receives the null-terminated name of the performance object or counter. Set to **NULL** if *pcchNameBufferSize* is zero.

[in, out] pcchNameBufferSize

Size of the *szNameBuffer* buffer, in TCHARs. If zero on input, the function returns PDH_MORE_DATA and sets this parameter to the required buffer size. If the buffer is larger than the required size, the function sets this parameter to the actual size of the

buffer that was used. If the specified size on input is greater than zero but less than the required size, you should not rely on the returned size to reallocate the buffer.

Return value

If the function succeeds, it returns ERROR_SUCCESS.

If the function fails, the return value is a [system error code](#) or a [PDH error code](#). The following are possible values.

Return code	Description
PDH_MORE_DATA	The <i>szNameBuffer</i> buffer is not large enough to contain the counter name. This return value is expected if <i>pcchNameBufferSize</i> is zero on input. If the specified size on input is greater than zero but less than the required size, you should not rely on the returned size to reallocate the buffer.
PDH_INVALID_ARGUMENT	A parameter is not valid or is incorrectly formatted. For example, on some releases you could receive this error if the specified size on input is greater than zero but less than the required size.

Remarks

You should call this function twice, the first time to get the required buffer size (set *szNameBuffer* to **NULL** and *pcchNameBufferSize* to 0), and the second time to get the data.

Windows XP: You must specify a buffer and buffer size. The function sets *pcchNameBufferSize* to either the required size or the size of the buffer that was used. If the buffer is too small, the function returns PDH_INSUFFICIENT_BUFFER instead of PDH_MORE_DATA. The maximum string size in bytes is PDH_MAX_COUNTER_NAME * sizeof(TCHAR).

The index value that you specify must match one of the index values associated with the objects or counters that were loaded on the computer. The index/name value pairs are stored in the **Counters** registry value in the following registry location.

```
HKEY_LOCAL_MACHINE  
  \SOFTWARE  
    \Microsoft  
      \Windows NT
```

```
\CurrentVersion
  \Perflib
    Last Counter = highest counter index
    Last Help = highest help index
    \009
      Counters = 2 System 4 Memory...
      Help = 3 The System Object Type...
      \supported Language, other than English
        Counters = ...
        Help = ...
```

ⓘ Note

The pdh.h header defines PdhLookupPerfNameByIndex 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

Minimum supported client	Windows XP [desktop apps only]
Minimum supported server	Windows Server 2003 [desktop apps only]
Target Platform	Windows
Header	pdh.h
Library	Pdh.lib
DLL	Pdh.dll

See also

[PdhLookupPerfIndexByName](#)

Feedback



Was this page helpful?  

Get help at Microsoft Q&A

PdhMakeCounterPathA function (pdh.h)

Article 02/09/2023

Creates a full counter path using the members specified in the [PDH_COUNTER_PATH_ELEMENTS](#) structure.

Syntax

C++

```
PDH_FUNCTION PdhMakeCounterPathA(
    [in]      PPDH_COUNTER_PATH_ELEMENTS_A pCounterPathElements,
    [out]     LPSTR                      szFullPathBuffer,
    [in, out] LPDWORD                   pcchBufferSize,
    [in]      DWORD                     dwFlags
);
```

Parameters

[in] `pCounterPathElements`

A [PDH_COUNTER_PATH_ELEMENTS](#) structure that contains the members used to make up the path. Only the `szObjectName` and `szCounterName` members are required, the others are optional.

If the instance name member is `NULL`, the path will not contain an instance reference and the `szParentInstance` and `dwInstanceId` members will be ignored.

[out] `szFullPathBuffer`

Caller-allocated buffer that receives a null-terminated counter path. The maximum length of a counter path is `PDH_MAX_COUNTER_PATH`. Set to `NULL` if `pcchBufferSize` is zero.

[in, out] `pcchBufferSize`

Size of the `szFullPathBuffer` buffer, in TCHARs. If zero on input, the function returns `PDH_MORE_DATA` and sets this parameter to the required buffer size. If the buffer is larger than the required size, the function sets this parameter to the actual size of the buffer that was used. If the specified size on input is greater than zero but less than the required size, you should not rely on the returned size to reallocate the buffer.

[in] dwFlags

Format of the input and output counter values. You can specify one of the following values.

[+] Expand table

Value	Meaning
PDH_PATH_WBEM_RESULT	Converts a PDH path to the WMI class and property name format.
PDH_PATH_WBEM_INPUT	Converts the WMI class and property name to a PDH path.
0	Returns the path in the PDH format, for example, \\computer\object(parent-instance#index)\counter.

Return value

If the function succeeds, it returns ERROR_SUCCESS.

If the function fails, the return value is a [system error code](#) or a [PDH error code](#). The following are possible values.

[+] Expand table

Return code	Description
PDH_MORE_DATA	The <i>szFullPathBuffer</i> buffer is too small to contain the counter name. This return value is expected if <i>pcchBufferSize</i> is zero on input. If the specified size on input is greater than zero but less than the required size, you should not rely on the returned size to reallocate the buffer.
PDH_INVALID_ARGUMENT	A parameter is not valid or is incorrectly formatted. For example, on some releases you could receive this error if the specified size on input is greater than zero but less than the required size.

Remarks

You should call this function twice, the first time to get the required buffer size (set *szFullPathBuffer* to **NULL** and *pcchBufferSize* to 0), and the second time to get the data.

Note

The pdh.h header defines PdhMakeCounterPath 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 is 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	pdh.h
Library	Pdh.lib
DLL	Pdh.dll

See also

[PDH_COUNTER_PATH_ELEMENTS](#)

[PdhParseCounterPath](#)

Feedback

Was this page helpful?

 Yes

 No

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

PdhMakeCounterPathW function (pdh.h)

Article 02/09/2023

Creates a full counter path using the members specified in the [PDH_COUNTER_PATH_ELEMENTS](#) structure.

Syntax

C++

```
PDH_FUNCTION PdhMakeCounterPathW(
    [in]      PPDH_COUNTER_PATH_ELEMENTS_W pCounterPathElements,
    [out]     LPWSTR                      szFullPathBuffer,
    [in, out] LPDWORD                     pcchBufferSize,
    [in]      DWORD                       dwFlags
);
```

Parameters

[in] pCounterPathElements

A [PDH_COUNTER_PATH_ELEMENTS](#) structure that contains the members used to make up the path. Only the **szObjectName** and **szCounterName** members are required, the others are optional.

If the instance name member is **NULL**, the path will not contain an instance reference and the **szParentInstance** and **dwInstanceId** members will be ignored.

[out] szFullPathBuffer

Caller-allocated buffer that receives a null-terminated counter path. The maximum length of a counter path is **PDH_MAX_COUNTER_PATH**. Set to **NULL** if **pcchBufferSize** is zero.

[in, out] pcchBufferSize

Size of the **szFullPathBuffer** buffer, in **TCHARs**. If zero on input, the function returns **PDH_MORE_DATA** and sets this parameter to the required buffer size. If the buffer is larger than the required size, the function sets this parameter to the actual size of the

buffer that was used. If the specified size on input is greater than zero but less than the required size, you should not rely on the returned size to reallocate the buffer.

[in] dwFlags

Format of the input and output counter values. You can specify one of the following values.

Value	Meaning
PDH_PATH_WBEM_RESULT	Converts a PDH path to the WMI class and property name format.
PDH_PATH_WBEM_INPUT	Converts the WMI class and property name to a PDH path.
0	Returns the path in the PDH format, for example, \\computer\\object(parent/instance#index)\\counter.

Return value

If the function succeeds, it returns ERROR_SUCCESS.

If the function fails, the return value is a [system error code](#) or a [PDH error code](#). The following are possible values.

Return code	Description
PDH_MORE_DATA	The <i>szFullPathBuffer</i> buffer is too small to contain the counter name. This return value is expected if <i>pcchBufferSize</i> is zero on input. If the specified size on input is greater than zero but less than the required size, you should not rely on the returned size to reallocate the buffer.
PDH_INVALID_ARGUMENT	A parameter is not valid or is incorrectly formatted. For example, on some releases you could receive this error if the specified size on input is greater than zero but less than the required size.

Remarks

You should call this function twice, the first time to get the required buffer size (set *szFullPathBuffer* to **NULL** and *pcchBufferSize* to 0), and the second time to get the data.

Note

The pdh.h header defines PdhMakeCounterPath 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 is not encoding-neutral can lead to mismatches that result in compilation or runtime errors. For more information, see [Conventions for Function Prototypes](#).

Requirements

Minimum supported client	Windows XP [desktop apps only]
Minimum supported server	Windows Server 2003 [desktop apps only]
Target Platform	Windows
Header	pdh.h
Library	Pdh.lib
DLL	Pdh.dll

See also

[PDH_COUNTER_PATH_ELEMENTS](#)

[PdhParseCounterPath](#)

Feedback

Was this page helpful?

 Yes

 No

[Get help at Microsoft Q&A](#)

PdhOpenLogA function (pdh.h)

Article02/09/2023

Opens the specified log file for reading or writing.

Syntax

C++

```
PDH_FUNCTION PdhOpenLogA(
    [in]    LPCSTR      szLogFileName,
    [in]    DWORD       dwAccessFlags,
    [in]    LPDWORD     lpdwLogType,
    [in]    PDH_HQUERY  hQuery,
    [in]    DWORD       dwMaxSize,
    [in]    LPCSTR      szUserCaption,
    [out]   PDH_HLOG    *phLog
);
```

Parameters

[in] szLogFileName

Null-terminated string that specifies the name of the log file to open. The name can contain an absolute or relative path.

If the *lpdwLogType* parameter is **PDH_LOG_TYPE_SQL**, specify the name of the log file in the form, **SQL:DataSourceName!LogFileName**.

[in] dwAccessFlags

Type of access to use to open the log file. Specify one of the following values.

[+] Expand table

Value	Meaning
PDH_LOG_READ_ACCESS	Open the log file for reading.
PDH_LOG_WRITE_ACCESS	Open a new log file for writing.
PDH_LOG_UPDATE_ACCESS	Open an existing log file for writing.

You can use the bitwise inclusive OR operator (|) to combine the access type with one or more of the following creation flags.

[+] Expand table

Value	Meaning
PDH_LOG_CREATE_NEW	Creates a new log file with the specified name.
PDH_LOG_CREATE_ALWAYS	Creates a new log file with the specified name. If the log file already exists, the function removes the existing log file before creating the new file.
PDH_LOG_OPEN_EXISTING	Opens an existing log file with the specified name. If a log file with the specified name does not exist, this is equal to PDH_LOG_CREATE_NEW.
PDH_LOG_OPEN_ALWAYS	Opens an existing log file with the specified name or creates a new log file with the specified name.
PDH_LOG_OPT_CIRCULAR	Creates a circular log file with the specified name. When the file reaches the value of the <i>dwMaxSize</i> parameter, data wraps to the beginning of the log file. You can specify this flag only if the <i>lpdwLogType</i> parameter is PDH_LOG_TYPE_BINARY.
PDH_LOG_USER_STRING	Used with PDH_LOG_TYPE_TSV to write the user caption or log file description indicated by the <i>szUserString</i> parameter of PdhUpdateLog or PdhOpenLog . The user caption or log file description is written as the last column in the first line of the text log.

[in] *lpdwLogType*

Type of log file to open. This parameter can be one of the following values.

[+] Expand table

Value	Meaning
PDH_LOG_TYPE_UNDEFINED	Undefined log file format. If specified, PDH determines the log file type. You cannot specify this value if the <i>dwAccessFlags</i> parameter is PDH_LOG_WRITE_ACCESS.
PDH_LOG_TYPE_CSV	Text file containing column headers in the first line, and individual data records in each subsequent line. The fields of each data record are comma-delimited. The first line also contains information about the format of the file, the PDH version used to create the log file, and

	the names and paths of each of the counters.
PDH_LOG_TYPE_SQL	The data source of the log file is an SQL database.
PDH_LOG_TYPE_TSV	<p>Text file containing column headers in the first line, and individual data records in each subsequent line. The fields of each data record are tab-delimited.</p> <p>The first line also contains information about the format of the file, the PDH version used to create the log file, and the names and paths of each of the counters.</p>
PDH_LOG_TYPE_BINARY	Binary log file format.

[in] hQuery

Specify a query handle if you are writing query data to a log file. The [PdhOpenQuery](#) function returns this handle.

This parameter is ignored and should be **NULL** if you are reading from the log file.

[in] dwMaxSize

Maximum size of the log file, in bytes. Specify the maximum size if you want to limit the file size or if *dwAccessFlags* specifies **PDH_LOG_OPT_CIRCULAR**; otherwise, set to 0.

For circular log files, you must specify a value large enough to hold at least one sample. Sample size depends on data being collected. However, specifying a value of at least one megabyte will cover most samples.

[in] szUserCaption

Null-terminated string that specifies the user-defined caption of the log file. A log file caption generally describes the contents of the log file. When an existing log file is opened, the value of this parameter is ignored.

[out] phLog

Handle to the opened log file.

Return value

If the function succeeds, it returns **ERROR_SUCCESS**.

If the function fails, the return value is a [system error code](#) or a [PDH error code](#).

Remarks

To use this function to write performance data to a log file, you must open a query using [PdhOpenQuery](#) and add the desired counters to it, before calling this function.

Newer operating systems can read log files that were generated on older operating systems; however, log files that were created on Windows Vista and later operating systems cannot be read on earlier operating systems.

The following rules apply to log files

- READ_ACCESS requires OPEN_EXISTING.
- UPDATE_ACCESS cannot be used with file-based logs. It can only be used with database logs.
- WRITE_ACCESS requires one of CREATE_NEW, CREATE_ALWAYS, OPEN_EXISTING, OPEN_ALWAYS.

Examples

For an example, see [Writing Performance Data to a Log File](#).

ⓘ Note

The pdh.h header defines PdhOpenLog 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 is 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	pdh.h
Library	Pdh.lib

Requirement	Value
DLL	Pdh.dll

See also

[PdhGetLogFileSize](#)

[PdhOpenQuery](#)

[PdhUpdateLog](#)

[PdhUpdateLogFileCatalog](#)

Feedback

Was this page helpful?

 Yes

 No

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

PdhOpenLogW function (pdh.h)

Article02/09/2023

Opens the specified log file for reading or writing.

Syntax

C++

```
PDH_FUNCTION PdhOpenLogW(
    [in]    LPCWSTR      szLogFileName,
    [in]    DWORD        dwAccessFlags,
    [in]    LPDWORD      lpdwLogType,
    [in]    PDH_HQUERY   hQuery,
    [in]    DWORD        dwMaxSize,
    [in]    LPCWSTR      szUserCaption,
    [out]   PDH_HLOG    *phLog
);
```

Parameters

[in] szLogFileName

Null-terminated string that specifies the name of the log file to open. The name can contain an absolute or relative path.

If the *lpdwLogType* parameter is **PDH_LOG_TYPE_SQL**, specify the name of the log file in the form, **SQL:DataSourceName!LogFileName**.

[in] dwAccessFlags

Type of access to use to open the log file. Specify one of the following values.

Value	Meaning
PDH_LOG_READ_ACCESS	Open the log file for reading.
PDH_LOG_WRITE_ACCESS	Open a new log file for writing.
PDH_LOG_UPDATE_ACCESS	Open an existing log file for writing.

You can use the bitwise inclusive **OR** operator (**|**) to combine the access type with one or more of the following creation flags.

Value	Meaning
PDH_LOG_CREATE_NEW	Creates a new log file with the specified name.
PDH_LOG_CREATE_ALWAYS	Creates a new log file with the specified name. If the log file already exists, the function removes the existing log file before creating the new file.
PDH_LOG_OPEN_EXISTING	Opens an existing log file with the specified name. If a log file with the specified name does not exist, this is equal to PDH_LOG_CREATE_NEW.
PDH_LOG_OPEN_ALWAYS	Opens an existing log file with the specified name or creates a new log file with the specified name.
PDH_LOG_OPT_CIRCULAR	Creates a circular log file with the specified name. When the file reaches the value of the <i>dwMaxSize</i> parameter, data wraps to the beginning of the log file. You can specify this flag only if the <i>lpdwLogType</i> parameter is PDH_LOG_TYPE_BINARY.
PDH_LOG_USER_STRING	Used with PDH_LOG_TYPE_TSV to write the user caption or log file description indicated by the <i>szUserString</i> parameter of PdhUpdateLog or PdhOpenLog . The user caption or log file description is written as the last column in the first line of the text log.

[in] *lpdwLogType*

Type of log file to open. This parameter can be one of the following values.

Value	Meaning
PDH_LOG_TYPE_UNDEFINED	Undefined log file format. If specified, PDH determines the log file type. You cannot specify this value if the <i>dwAccessFlags</i> parameter is PDH_LOG_WRITE_ACCESS.
PDH_LOG_TYPE_CSV	Text file containing column headers in the first line, and individual data records in each subsequent line. The fields of each data record are comma-delimited. The first line also contains information about the format of the file, the PDH version used to create the log file, and the names and paths of each of the counters.
PDH_LOG_TYPE_SQL	The data source of the log file is an SQL database.
PDH_LOG_TYPE_TSV	Text file containing column headers in the first line, and individual data records in each subsequent line. The fields of each data record are tab-delimited.

The first line also contains information about the format of the file, the PDH version used to create the log file, and the names and paths of each of the counters.

PDH_LOG_TYPE_BINARY	Binary log file format.
---------------------	-------------------------

[in] hQuery

Specify a query handle if you are writing query data to a log file. The [PdhOpenQuery](#) function returns this handle.

This parameter is ignored and should be **NULL** if you are reading from the log file.

[in] dwMaxSize

Maximum size of the log file, in bytes. Specify the maximum size if you want to limit the file size or if *dwAccessFlags* specifies **PDH_LOG_OPT_CIRCULAR**; otherwise, set to 0.

For circular log files, you must specify a value large enough to hold at least one sample. Sample size depends on data being collected. However, specifying a value of at least one megabyte will cover most samples.

[in] szUserCaption

Null-terminated string that specifies the user-defined caption of the log file. A log file caption generally describes the contents of the log file. When an existing log file is opened, the value of this parameter is ignored.

[out] phLog

Handle to the opened log file.

Return value

If the function succeeds, it returns **ERROR_SUCCESS**.

If the function fails, the return value is a [system error code](#) or a [PDH error code](#).

Remarks

To use this function to write performance data to a log file, you must open a query using [PdhOpenQuery](#) and add the desired counters to it, before calling this function.

Newer operating systems can read log files that were generated on older operating systems; however, log files that were created on Windows Vista and later operating

systems cannot be read on earlier operating systems.

The following rules apply to log files

- READ_ACCESS requires OPEN_EXISTING.
- UPDATE_ACCESS cannot be used with file-based logs. It can only be used with database logs.
- WRITE_ACCESS requires one of CREATE_NEW, CREATE_ALWAYS, OPEN_EXISTING, OPEN_ALWAYS.

Examples

For an example, see [Writing Performance Data to a Log File](#).

ⓘ Note

The pdh.h header defines PdhOpenLog 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 is not encoding-neutral can lead to mismatches that result in compilation or runtime errors. For more information, see [Conventions for Function Prototypes](#).

Requirements

Minimum supported client	Windows XP [desktop apps only]
Minimum supported server	Windows Server 2003 [desktop apps only]
Target Platform	Windows
Header	pdh.h
Library	Pdh.lib
DLL	Pdh.dll

See also

[PdhGetLogFileSize](#)

[PdhOpenQuery](#)

[PdhUpdateLog](#)

[PdhUpdateLogFileCatalog](#)

Feedback

Was this page helpful?

 Yes

 No

[Get help at Microsoft Q&A](#)

PdhOpenQueryA function (pdh.h)

Article02/09/2023

Creates a new query that is used to manage the collection of performance data.

To use handles to data sources, use the [PdhOpenQueryH](#) function.

Syntax

C++

```
PDH_FUNCTION PdhOpenQueryA(
    [in]    LPCSTR      szDataSource,
    [in]    DWORD_PTR   dwUserData,
    [out]   PDH_HQUERY *phQuery
);
```

Parameters

[in] szDataSource

Null-terminated string that specifies the name of the log file from which to retrieve performance data. If **NULL**, performance data is collected from a real-time data source.

[in] dwUserData

User-defined value to associate with this query. To retrieve the user data later, call [PdhGetCounterInfo](#) and access the **dwQueryUserData** member of [PDH_COUNTER_INFO](#).

[out] phQuery

Handle to the query. You use this handle in subsequent calls.

Return value

If the function succeeds, it returns **ERROR_SUCCESS**.

If the function fails, the return value is a [system error code](#) or a [PDH error code](#).

Remarks

ⓘ Note

The pdh.h header defines PdhOpenQuery 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 is 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	pdh.h
Library	Pdh.lib
DLL	Pdh.dll

See also

[PdhCloseQuery](#)

[PdhGetCounterInfo](#)

[PdhIsRealTimeQuery](#)

[PdhOpenQueryH](#)

[PdhSetDefaultRealTimeDataSource](#)

Feedback

Was this page helpful?

 Yes

 No

PdhOpenQueryH function (pdh.h)

Article10/13/2021

Creates a new query that is used to manage the collection of performance data.

This function is identical to the [PdhOpenQuery](#) function, except that it supports the use of handles to data sources.

Syntax

C++

```
PDH_FUNCTION PdhOpenQueryH(
    [in] PDH_HLOG    hDataSource,
    [in] DWORD_PTR   dwUserData,
    [out] PDH_HQUERY *phQuery
);
```

Parameters

[in] hDataSource

Handle to a data source returned by the [PdhBindInputDataSource](#) function.

[in] dwUserData

User-defined value to associate with this query. To retrieve the user data later, call the [PdhGetCounterInfo](#) function and access the dwQueryUserData member of [PDH_COUNTER_INFO](#).

[out] phQuery

Handle to the query. You use this handle in subsequent calls.

Return value

If the function succeeds, it returns ERROR_SUCCESS.

If the function fails, the return value is a [system error code](#) or a [PDH error code](#).

Requirements

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

See also

[PdhBindInputDataSource](#)

Feedback

Was this page helpful?

 Yes

 No

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

PdhOpenQueryW function (pdh.h)

Article02/09/2023

Creates a new query that is used to manage the collection of performance data.

To use handles to data sources, use the [PdhOpenQueryH](#) function.

Syntax

C++

```
PDH_FUNCTION PdhOpenQueryW(
    [in]    LPCWSTR      szDataSource,
    [in]    DWORD_PTR    dwUserData,
    [out]   PDH_HQUERY *phQuery
);
```

Parameters

[in] szDataSource

Null-terminated string that specifies the name of the log file from which to retrieve performance data. If **NULL**, performance data is collected from a real-time data source.

[in] dwUserData

User-defined value to associate with this query. To retrieve the user data later, call [PdhGetCounterInfo](#) and access the **dwQueryUserData** member of [PDH_COUNTER_INFO](#).

[out] phQuery

Handle to the query. You use this handle in subsequent calls.

Return value

If the function succeeds, it returns **ERROR_SUCCESS**.

If the function fails, the return value is a [system error code](#) or a [PDH error code](#).

Remarks

Note

The pdh.h header defines PdhOpenQuery 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 is not encoding-neutral can lead to mismatches that result in compilation or runtime errors. For more information, see [Conventions for Function Prototypes](#).

Requirements

Minimum supported client	Windows XP [desktop apps only]
Minimum supported server	Windows Server 2003 [desktop apps only]
Target Platform	Windows
Header	pdh.h
Library	Pdh.lib
DLL	Pdh.dll

See also

[PdhCloseQuery](#)

[PdhGetCounterInfo](#)

[PdhIsRealTimeQuery](#)

[PdhOpenQueryH](#)

[PdhSetDefaultRealTimeDataSource](#)

Feedback

Was this page helpful?

 Yes

 No

[Get help at Microsoft Q&A](#)

PdhParseCounterPathA function (pdh.h)

Article 02/09/2023

Parses the elements of the counter path and stores the results in the [PDH_COUNTER_PATH_ELEMENTS](#) structure.

Syntax

C++

```
PDH_FUNCTION PdhParseCounterPathA(
    [in]      LPCSTR                  szFullPathBuffer,
    [out]     PPDH_COUNTER_PATH_ELEMENTS_A pCounterPathElements,
    [in, out] LPDWORD                 pdwBufferSize,
                           DWORD                   dwFlags
);
```

Parameters

[in] `szFullPathBuffer`

Null-terminated string that contains the counter path to parse. The maximum length of a counter path is [PDH_MAX_COUNTER_PATH](#).

[out] `pCounterPathElements`

Caller-allocated buffer that receives a [PDH_COUNTER_PATH_ELEMENTS](#) structure. The structure contains pointers to the individual string elements of the path referenced by the `szFullPathBuffer` parameter. The function appends the strings to the end of the [PDH_COUNTER_PATH_ELEMENTS](#) structure. The allocated buffer should be large enough for the structure and the strings. Set to `NULL` if `pdwBufferSize` is zero.

[in, out] `pdwBufferSize`

Size of the `pCounterPathElements` buffer, in bytes. If zero on input, the function returns [PDH_MORE_DATA](#) and sets this parameter to the required buffer size. If the buffer is larger than the required size, the function sets this parameter to the actual size of the buffer that was used. If the specified size on input is greater than zero but less than the required size, you should not rely on the returned size to reallocate the buffer.

`dwFlags`

Reserved. Must be zero.

Return value

If the function succeeds, it returns ERROR_SUCCESS.

If the function fails, the return value is a [system error code](#) or a [PDH error code](#). The following are possible values.

[Expand table](#)

Return code	Description
PDH_INVALID_ARGUMENT	A parameter is not valid.
PDH_MORE_DATA	The <i>pCounterPathElements</i> buffer is too small to contain the path elements. This return value is expected if <i>pdwBufferSize</i> is zero on input. If the specified size on input is greater than zero but less than the required size, you should not rely on the returned size to reallocate the buffer.
PDH_INVALID_PATH	The path is not formatted correctly and cannot be parsed. For example, on some releases you could receive this error if the specified size on input is greater than zero but less than the required size.
PDH_MEMORY_ALLOCATION_FAILURE	Unable to allocate memory in order to complete the function.

Remarks

You should call this function twice, the first time to get the required buffer size (set *pCounterPathElements* to **NULL** and *pdwBufferSize* to 0), and the second time to get the data.

Note

The pdh.h header defines PdhParseCounterPath 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 is 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	pdh.h
Library	Pdh.lib
DLL	Pdh.dll

See also

[PDH_COUNTER_PATH_ELEMENTS](#)

[PdhMakeCounterPath](#)

Feedback

Was this page helpful?

 Yes

 No

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

PdhParseCounterPathW function (pdh.h)

Article 02/09/2023

Parses the elements of the counter path and stores the results in the [PDH_COUNTER_PATH_ELEMENTS](#) structure.

Syntax

C++

```
PDH_FUNCTION PdhParseCounterPathW(
    [in]      LPCWSTR             szFullPathBuffer,
    [out]     PPDH_COUNTER_PATH_ELEMENTS_W pCounterPathElements,
    [in, out] LPDWORD            pdwBufferSize,
                           DWORD           dwFlags
);
```

Parameters

[in] `szFullPathBuffer`

Null-terminated string that contains the counter path to parse. The maximum length of a counter path is `PDH_MAX_COUNTER_PATH`.

[out] `pCounterPathElements`

Caller-allocated buffer that receives a [PDH_COUNTER_PATH_ELEMENTS](#) structure. The structure contains pointers to the individual string elements of the path referenced by the `szFullPathBuffer` parameter. The function appends the strings to the end of the [PDH_COUNTER_PATH_ELEMENTS](#) structure. The allocated buffer should be large enough for the structure and the strings. Set to `NULL` if `pdwBufferSize` is zero.

[in, out] `pdwBufferSize`

Size of the `pCounterPathElements` buffer, in bytes. If zero on input, the function returns `PDH_MORE_DATA` and sets this parameter to the required buffer size. If the buffer is larger than the required size, the function sets this parameter to the actual size of the buffer that was used. If the specified size on input is greater than zero but less than the required size, you should not rely on the returned size to reallocate the buffer.

dwFlags

Reserved. Must be zero.

Return value

If the function succeeds, it returns ERROR_SUCCESS.

If the function fails, the return value is a [system error code](#) or a [PDH error code](#). The following are possible values.

Return code	Description
PDH_INVALID_ARGUMENT	A parameter is not valid.
PDH_MORE_DATA	The <i>pCounterPathElements</i> buffer is too small to contain the path elements. This return value is expected if <i>pdwBufferSize</i> is zero on input. If the specified size on input is greater than zero but less than the required size, you should not rely on the returned size to reallocate the buffer.
PDH_INVALID_PATH	The path is not formatted correctly and cannot be parsed. For example, on some releases you could receive this error if the specified size on input is greater than zero but less than the required size.
PDH_MEMORY_ALLOCATION_FAILURE	Unable to allocate memory in order to complete the function.

Remarks

You should call this function twice, the first time to get the required buffer size (set *pCounterPathElements* to **NULL** and *pdwBufferSize* to 0), and the second time to get the data.

ⓘ Note

The pdh.h header defines PdhParseCounterPath 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 is not encoding-neutral can lead to mismatches that result in compilation or runtime errors. For more information, see [Conventions for Function Prototypes](#).

Requirements

Minimum supported client	Windows XP [desktop apps only]
Minimum supported server	Windows Server 2003 [desktop apps only]
Target Platform	Windows
Header	pdh.h
Library	Pdh.lib
DLL	Pdh.dll

See also

[PDH_COUNTER_PATH_ELEMENTS](#)

[PdhMakeCounterPath](#)

Feedback

Was this page helpful?

 Yes

 No

[Get help at Microsoft Q&A](#)

PdhParseInstanceNameA function (pdh.h)

Article 02/09/2023

Parses the elements of an instance string.

Syntax

C++

```
PDH_FUNCTION PdhParseInstanceNameA(
    [in]      LPCSTR  szInstanceString,
    [out]     LPSTR   szInstanceName,
    [in, out] LPDWORD pcchInstanceNameLength,
    [out]     LPSTR   szParentName,
    [in, out] LPDWORD pcchParentNameLength,
    [out]     LPDWORD lpIndex
);
```

Parameters

[in] szInstanceString

Null-terminated string that specifies the instance string to parse into individual components. This string can contain the following formats, and is less than MAX_PATH characters in length:

- instance
- instance#index
- parent/instance
- parent/instance#index

[out] szInstanceName

Caller-allocated buffer that receives the null-terminated instance name. Set to **NULL** if *pcchInstanceNameLength* is zero.

[in, out] pcchInstanceNameLength

Size of the *szInstanceName* buffer, in TCHARs. If zero on input, the function returns PDH_MORE_DATA and sets this parameter to the required buffer size. If the buffer is larger than the required size, the function sets this parameter to the actual size of the

buffer that was used. If the specified size on input is greater than zero but less than the required size, you should not rely on the returned size to reallocate the buffer.

[out] szParentName

Caller-allocated buffer that receives the **null**-terminated name of the parent instance, if one is specified. Set to **NULL** if *pcchParentNameLength* is zero.

[in, out] pcchParentNameLength

Size of the *szParentName* buffer, in **TCHARs**. If zero on input, the function returns **PDH_MORE_DATA** and sets this parameter to the required buffer size. If the buffer is larger than the required size, the function sets this parameter to the actual size of the buffer that was used. If the specified size on input is greater than zero but less than the required size, you should not rely on the returned size to reallocate the buffer.

[out] lpIndex

Index value of the instance. If an index entry is not present in the string, then this value is zero. This parameter can be **NULL**.

Return value

If the function succeeds, it returns **ERROR_SUCCESS**.

If the function fails, the return value is a [system error code](#) or a [PDH error code](#). The following are possible values.

[+] Expand table

Return code	Description
PDH_INVALID_ARGUMENT	A parameter is not valid. For example, on some releases you could receive this error if the specified size on input is greater than zero but less than the required size.
PDH_MORE_DATA	One or both of the string buffers are too small to contain the data. This return value is expected if the corresponding size buffer is zero on input. If the specified size on input is greater than zero but less than the required size, you should not rely on the returned size to reallocate the buffer.
PDH_INVALID_INSTANCE	The instance string is incorrectly formatted, exceeds MAX_PATH characters in length, or cannot be parsed.

Remarks

You should call this function twice, the first time to get the required buffer size (set the buffers to **NULL** and buffer sizes to 0), and the second time to get the data.

ⓘ Note

The pdh.h header defines PdhParseInstanceName 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 is 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	pdh.h
Library	Pdh.lib
DLL	Pdh.dll

See also

[PdhMakeCounterPath](#)

Feedback

Was this page helpful?

 Yes

 No

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

PdhParseInstanceIdNameW function (pdh.h)

Article 02/09/2023

Parses the elements of an instance string.

Syntax

C++

```
PDH_FUNCTION PdhParseInstanceIdNameW(
    [in]      LPCWSTR szInstanceIdNameString,
    [out]     LPWSTR szInstanceIdName,
    [in, out] LPDWORD pcchInstanceIdNameLength,
    [out]     LPWSTR szParentName,
    [in, out] LPDWORD pcchParentNameLength,
    [out]     LPDWORD lpIndex
);
```

Parameters

[in] `szInstanceIdNameString`

Null-terminated string that specifies the instance string to parse into individual components. This string can contain the following formats, and is less than MAX_PATH characters in length:

- instance
- instance#index
- parent/instance
- parent/instance#index

[out] `szInstanceIdName`

Caller-allocated buffer that receives the null-terminated instance name. Set to **NULL** if `pcchInstanceIdNameLength` is zero.

[in, out] `pcchInstanceIdNameLength`

Size of the `szInstanceIdName` buffer, in TCHARs. If zero on input, the function returns PDH_MORE_DATA and sets this parameter to the required buffer size. If the buffer is larger than the required size, the function sets this parameter to the actual size of the

buffer that was used. If the specified size on input is greater than zero but less than the required size, you should not rely on the returned size to reallocate the buffer.

[out] szParentName

Caller-allocated buffer that receives the null-terminated name of the parent instance, if one is specified. Set to **NULL** if *pcchParentNameLength* is zero.

[in, out] pcchParentNameLength

Size of the *szParentName* buffer, in TCHARs. If zero on input, the function returns PDH_MORE_DATA and sets this parameter to the required buffer size. If the buffer is larger than the required size, the function sets this parameter to the actual size of the buffer that was used. If the specified size on input is greater than zero but less than the required size, you should not rely on the returned size to reallocate the buffer.

[out] lpIndex

Index value of the instance. If an index entry is not present in the string, then this value is zero. This parameter can be **NULL**.

Return value

If the function succeeds, it returns ERROR_SUCCESS.

If the function fails, the return value is a [system error code](#) or a [PDH error code](#). The following are possible values.

Return code	Description
PDH_INVALID_ARGUMENT	A parameter is not valid. For example, on some releases you could receive this error if the specified size on input is greater than zero but less than the required size.
PDH_MORE_DATA	One or both of the string buffers are too small to contain the data. This return value is expected if the corresponding size buffer is zero on input. If the specified size on input is greater than zero but less than the required size, you should not rely on the returned size to reallocate the buffer.
PDH_INVALID_INSTANCE	The instance string is incorrectly formatted, exceeds MAX_PATH characters in length, or cannot be parsed.

Remarks

You should call this function twice, the first time to get the required buffer size (set the buffers to **NULL** and buffer sizes to 0), and the second time to get the data.

 **Note**

The pdh.h header defines PdhParseInstanceName 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 is not encoding-neutral can lead to mismatches that result in compilation or runtime errors. For more information, see [Conventions for Function Prototypes](#).

Requirements

Minimum supported client	Windows XP [desktop apps only]
Minimum supported server	Windows Server 2003 [desktop apps only]
Target Platform	Windows
Header	pdh.h
Library	Pdh.lib
DLL	Pdh.dll

See also

[PdhMakeCounterPath](#)

Feedback

Was this page helpful?

 Yes

 No

[Get help at Microsoft Q&A](#)

PdhReadRawLogRecord function (pdh.h)

Article 02/22/2024

Reads the information in the specified binary trace log file.

Syntax

C++

```
PDH_FUNCTION PdhReadRawLogRecord(
    [in] PDH_HLOG           hLog,
    [in] FILETIME          ftRecord,
    [out] PPDH_RAW_LOG_RECORD pRawLogRecord,
    [in] LPDWORD            pdwBufferLength
);
```

Parameters

[in] hLog

Handle to the log file. The [PdhOpenLog](#) or [PdhBindInputDataSource](#) function returns this handle.

[in] ftRecord

Time stamp of the record to be read. If the time stamp does not match a record in the log file, the function returns the record that has a time stamp closest to (but not greater than) the given time stamp.

[out] pRawLogRecord

Caller-allocated buffer that receives a [PDH_RAW_LOG_RECORD](#) structure; the structure contains the log file record information. Set to **NULL** if *pdwBufferLength* is zero.

[in] pdwBufferLength

Size of the *pRawLogRecord* buffer, in **TCHARs**. If zero on input, the function returns **PDH_MORE_DATA** and sets this parameter to the required buffer size. If the buffer is larger than the required size, the function sets this parameter to the actual size of the buffer that was used. If the specified size on input is greater than zero but less than the required size, you should not rely on the returned size to reallocate the buffer.

Return value

If the function succeeds, it returns ERROR_SUCCESS.

If the function fails, the return value is a [system error code](#) or a [PDH error code](#). The following are possible values.

[+] Expand table

Return code	Description
PDH_INVALID_ARGUMENT	A parameter is not valid. For example, on some releases you could receive this error if the specified size on input is greater than zero but less than the required size.
PDH_MORE_DATA	The <i>pRawLogRecord</i> buffer is too small to contain the path elements. This return value is expected if <i>pdwBufferLength</i> is zero on input. If the specified size on input is greater than zero but less than the required size, you should not rely on the returned size to reallocate the buffer.
PDH_MEMORY_ALLOCATION_FAILURE	Unable to allocate memory in order to complete the function.

Remarks

You should call this function twice, the first time to get the required buffer size (set *pRawLogRecord* to **NULL** and *pdwBufferLength* to 0), and the second time to get the data.

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	pdh.h

Requirement	Value
Library	Pdh.lib
DLL	Pdh.dll

See also

[PDH_RAW_LOG_RECORD](#)

[PdhCollectQueryData](#)

[PdhFormatFromRawValue](#)

Feedback

Was this page helpful?

 Yes

 No

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

PdhRemoveCounter function (pdh.h)

Article02/22/2024

Removes a counter from a query.

Syntax

C++

```
PDH_FUNCTION PdhRemoveCounter(
    [in] PDH_HCOUNTER hCounter
);
```

Parameters

[in] hCounter

Handle of the counter to remove from its query. The [PdhAddCounter](#) function returns this handle.

Return value

If the function succeeds, it returns ERROR_SUCCESS.

If the function fails, the return value is a [system error code](#) or a [PDH error code](#).

The following is a possible value.

 Expand table

Return code	Description
PDH_INVALID_HANDLE	The counter handle is not valid.

Remarks

Do not use the counter handle after removing the counter from the query.

The following shows the syntax if calling this function from Visual Basic.

syntax

```
PdhRemoveCounter(  
    ByVal CounterHandle as Long  
)  
as Long
```

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	pdh.h
Library	Pdh.lib
DLL	Pdh.dll

See also

[PdhAddCounter](#)

[PdhOpenQuery](#)

Feedback

Was this page helpful?

 Yes

 No

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

PdhSelectDataSourceA function (pdh.h)

Article 02/09/2023

Displays a dialog window that prompts the user to specify the source of the performance data.

Syntax

C++

```
PDH_FUNCTION PdhSelectDataSourceA(
    [in]      HWND     hWndOwner,
    [in]      DWORD    dwFlags,
    [out]     LPSTR    szDataSource,
    [in, out] LPDWORD  pcchBufferLength
);
```

Parameters

[in] hWndOwner

Owner of the dialog window. This can be **NULL** if there is no owner (the desktop becomes the owner).

[in] dwFlags

Dialog boxes that will be displayed to prompt for the data source. This parameter can be one of the following values.

 Expand table

Value	Meaning
PDH_FLAGS_FILE_BROWSER_ONLY	Display the file browser only. Set this flag when you want to prompt for the name and location of a log file only.
0	Display the data source selection dialog box. The dialog box lets the user select performance data from either a log file or a real-time source. If the user specified that data is to be collected from a log file, a file browser is displayed for the user to specify the name and location of the log file.

[out] szDataSource

Caller-allocated buffer that receives a null-terminated string that contains the name of a log file that the user selected. The log file name is truncated to the size of the buffer if the buffer is too small.

If the user selected a real time source, the buffer is empty.

[in, out] pcchBufferLength

Maximum size of the *szDataSource* buffer, in TCHARs.

Return value

If the function succeeds, it returns ERROR_SUCCESS.

If the function fails, the return value is a [system error code](#) or a [PDH error code](#). The following are possible values.

[] Expand table

Return code	Description
PDH_INVALID_ARGUMENT	The length of the buffer passed in the <i>pcchBufferLength</i> is not equal to the actual length of the <i>szDataSource</i> buffer.
PDH_MEMORY_ALLOCATION_FAILURE	A zero-length buffer was passed in the <i>szDataSource</i> parameter.

Remarks

! Note

The pdh.h header defines PdhSelectDataSource 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 is 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	pdh.h
Library	Pdh.lib
DLL	Pdh.dll

See also

[PdhIsRealTimeQuery](#)

Feedback

Was this page helpful?

 Yes

 No

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

PdhSelectDataSourceW function (pdh.h)

Article 02/09/2023

Displays a dialog window that prompts the user to specify the source of the performance data.

Syntax

C++

```
PDH_FUNCTION PdhSelectDataSourceW(
    [in]      HWND     hWndOwner,
    [in]      DWORD    dwFlags,
    [out]     LPWSTR   szDataSource,
    [in, out] LPDWORD  pcchBufferLength
);
```

Parameters

[in] hWndOwner

Owner of the dialog window. This can be **NULL** if there is no owner (the desktop becomes the owner).

[in] dwFlags

Dialog boxes that will be displayed to prompt for the data source. This parameter can be one of the following values.

Value	Meaning
PDH_FLAGS_FILE_BROWSER_ONLY	Display the file browser only. Set this flag when you want to prompt for the name and location of a log file only.
0	Display the data source selection dialog box. The dialog box lets the user select performance data from either a log file or a real-time source. If the user specified that data is to be collected from a log file, a file browser is displayed for the user to specify the name and location of the log file.

[out] szDataSource

Caller-allocated buffer that receives a **null**-terminated string that contains the name of a log file that the user selected. The log file name is truncated to the size of the buffer if the buffer is too small.

If the user selected a real time source, the buffer is empty.

[in, out] `pcchBufferLength`

Maximum size of the `szDataSource` buffer, in **TCHARs**.

Return value

If the function succeeds, it returns `ERROR_SUCCESS`.

If the function fails, the return value is a [system error code](#) or a [PDH error code](#). The following are possible values.

Return code	Description
<code>PDH_INVALID_ARGUMENT</code>	The length of the buffer passed in the <code>pcchBufferLength</code> is not equal to the actual length of the <code>szDataSource</code> buffer.
<code>PDH_MEMORY_ALLOCATION_FAILURE</code>	A zero-length buffer was passed in the <code>szDataSource</code> parameter.

Remarks

Note

The pdh.h header defines `PdhSelectDataSource` 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 is not encoding-neutral can lead to mismatches that result in compilation or runtime errors. For more information, see [Conventions for Function Prototypes](#).

Requirements

Minimum supported client	Windows XP [desktop apps only]
--------------------------	--------------------------------

Minimum supported server	Windows Server 2003 [desktop apps only]
Target Platform	Windows
Header	pdh.h
Library	Pdh.lib
DLL	Pdh.dll

See also

[PdhIsRealTimeQuery](#)

Feedback

Was this page helpful?

 Yes

 No

[Get help at Microsoft Q&A](#)

PdhSetCounterScaleFactor function (pdh.h)

Article02/22/2024

Sets the scale factor that is applied to the calculated value of the specified counter when you request the formatted counter value. If the PDH_FMT_NOSCALE flag is set, then this scale factor is ignored.

Syntax

C++

```
PDH_FUNCTION PdhSetCounterScaleFactor(
    [in] PDH_HCOUNTER hCounter,
    [in] LONG          lFactor
);
```

Parameters

[in] hCounter

Handle of the counter to apply the scale factor to. The [PdhAddCounter](#) function returns this handle.

[in] lFactor

Power of ten by which to multiply the calculated value before returning it. The minimum value of this parameter is PDH_MIN_SCALE (-7), where the returned value is the actual value multiplied by 10^{-7} . The maximum value of this parameter is PDH_MAX_SCALE (+7), where the returned value is the actual value multiplied by 10^7 . A value of zero will set the scale to one, so that the actual value is returned.

Return value

If the function succeeds, it returns ERROR_SUCCESS.

If the function fails, the return value is a [system error code](#) or a [PDH error code](#). The following are possible values.

[+] Expand table

Return code	Description
PDH_INVALID_ARGUMENT	The scale value is out of range.
PDH_INVALID_HANDLE	The counter handle is not valid.

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	pdh.h
Library	Pdh.lib
DLL	Pdh.dll

See also

[PdhCalculateCounterFromRawValue](#)

[PdhComputeCounterStatistics](#)

[PdhGetFormattedCounterValue](#)

Feedback

Was this page helpful?

 Yes

 No

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

PdhSetDefaultRealTimeDataSource function (pdh.h)

Article02/22/2024

Specifies the source of the real-time data.

Syntax

C++

```
PDH_FUNCTION PdhSetDefaultRealTimeDataSource(
    [in] DWORD dwDataSourceId
);
```

Parameters

[in] dwDataSourceId

Source of the performance data. This parameter can be one of the following values.

[+] Expand table

Value	Meaning
DATA_SOURCE_REGISTRY	The data source is the registry interface. This is the default.
DATA_SOURCE_WBEM	The data source is a WMI provider.

Return value

If the function succeeds, it returns ERROR_SUCCESS.

If the function fails, the return value is a [system error code](#) or a [PDH error code](#). The following is a possible value.

[+] Expand table

Return code	Description
-------------	-------------

PDH_INVALID_ARGUMENT

The parameter is not valid.

Remarks

The term *real-time* as used in the description of this function does not imply the standard meaning of the term *real-time*. Instead, it describes the collection of performance data from a source providing current information (for example, the registry or a WMI provider) rather than from a log file.

If you want to query real-time data from WMI, you must call **PdhSetDefaultRealTimeDataSource** to set the default real-time data source. You must call this function before calling any other PDH API function.

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	pdh.h
Library	Pdh.lib
DLL	Pdh.dll

See also

[PdhSelectDataSource](#)

Feedback

Was this page helpful?

 Yes

 No

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

PdhSetQueryTimeRange function (pdh.h)

Article02/22/2024

Limits the samples that you can read from a log file to those within the specified time range, inclusively.

Syntax

C++

```
PDH_FUNCTION PdhSetQueryTimeRange(
    [in] PDH_HQUERY      hQuery,
    [in] PPDH_TIME_INFO pInfo
);
```

Parameters

[in] hQuery

Handle to the query. The [PdhOpenQuery](#) function returns this handle.

[in] pInfo

A [PDH_TIME_INFO](#) structure that specifies the time range. Specify the time as local file time. The end time must be greater than the start time. You can specify 0 for the start time and the maximum 64-bit value for the end time if you want to read all records.

Return value

If the function succeeds, it returns ERROR_SUCCESS.

If the function fails, the return value is a [system error code](#) or a [PDH error code](#). The following are possible values.

[] Expand table

Return code	Description
PDH_INVALID_HANDLE	The query handle is not valid.

PDH_INVALID_ARGUMENT

The ending time range value must be greater than the starting time range value.

Remarks

When the end of the specified time range or the end of the log file is reached, the [PdhCollectQueryData](#) function will return PDH_NO_MORE_DATA.

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	pdh.h
Library	Pdh.lib
DLL	Pdh.dll

See also

[PdhCollectQueryData](#)

[PdhGetDataSourceTimeRange](#)

[PdhOpenQuery](#)

Feedback

Was this page helpful?

 Yes

 No

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

PdhUpdateLogA function (pdh.h)

Article 02/22/2024

Collects counter data for the current query and writes the data to the log file.

Syntax

C++

```
PDH_FUNCTION PdhUpdateLogA(
    [in] PDH_HLOG hLog,
    [in] LPCSTR   szUserString
);
```

Parameters

[in] hLog

Handle of a single log file to update. The [PdhOpenLog](#) function returns this handle.

[in] szUserString

Null-terminated string that contains a user-defined comment to add to the data record. The string can not be empty.

Return value

If the function succeeds, it returns ERROR_SUCCESS.

If the function fails, the return value is a [system error code](#) or a [PDH error code](#). The following are possible values.

[] Expand table

Return code	Description
PDH_INVALID_HANDLE	The log file handle is not valid.
PDH_INVALID_ARGUMENT	An empty string was passed in the <i>szUserString</i> parameter.

Remarks

If you are updating a log file from another log file, the comments from the other log file do not migrate.

Examples

For an example, see [Writing Performance Data to a Log File](#).

Note

The pdh.h header defines PdhUpdateLog 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 is 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	pdh.h
Library	Pdh.lib
DLL	Pdh.dll

See also

[PdhGetLogFileSize](#)

[PdhOpenLog](#)

[PdhOpenQuery](#)

Feedback

Was this page helpful?

 Yes

 No

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

PdhUpdateLogFileCatalog function (pdh.h)

Article02/22/2024

Synchronizes the information in the log file catalog with the performance data in the log file.

Note This function is obsolete.

Syntax

C++

```
PDH_FUNCTION PdhUpdateLogFileCatalog(  
    [in] PDH_HLOG hLog  
);
```

Parameters

[in] hLog

Handle to the log file containing the file catalog to update. The [PdhOpenLog](#) function.

Return value

If the function succeeds, it returns ERROR_SUCCESS.

If the function fails, the return value is a [system error code](#) or a [PDH error code](#). The following are possible values.

[+] Expand table

Return code	Description
PDH_NOT_IMPLEMENTED	A handle to a CSV or TSV log file was specified. These log file types do not have catalogs.

PDH_UNKNOWN_LOG_FORMAT	A handle to a log file with an unknown format was specified.
PDH_INVALID_HANDLE	The handle is not valid.

Remarks

The log file catalog serves as an index to the performance data records in the log file, providing for faster searches for individual records in the file.

Catalogs should be updated when the data collection process is complete and the log file has been closed. The catalog can be updated during data collection, but doing this may disrupt the process of logging the performance data because updating the catalogs can be time consuming.

Perfmon, CSV, and TSV log files do not have catalogs. Specifying a handle to these log file types will result in a return value of PDH_NOT_IMPLEMENTED.

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	pdh.h
Library	Pdh.lib
DLL	Pdh.dll

See also

[PdhGetLogFileSize](#)

[PdhUpdateLog](#)

Feedback

Was this page helpful?

 Yes

 No

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

PdhUpdateLogW function (pdh.h)

Article 02/09/2023

Collects counter data for the current query and writes the data to the log file.

Syntax

C++

```
PDH_FUNCTION PdhUpdateLogW(
    [in] PDH_HLOG hLog,
    [in] LPCWSTR szUserString
);
```

Parameters

[in] hLog

Handle of a single log file to update. The [PdhOpenLog](#) function returns this handle.

[in] szUserString

Null-terminated string that contains a user-defined comment to add to the data record. The string can not be empty.

Return value

If the function succeeds, it returns ERROR_SUCCESS.

If the function fails, the return value is a [system error code](#) or a [PDH error code](#). The following are possible values.

Return code	Description
PDH_INVALID_HANDLE	The log file handle is not valid.
PDH_INVALID_ARGUMENT	An empty string was passed in the <i>szUserString</i> parameter.

Remarks

If you are updating a log file from another log file, the comments from the other log file do not migrate.

Examples

For an example, see [Writing Performance Data to a Log File](#).

ⓘ Note

The pdh.h header defines PdhUpdateLog 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 is not encoding-neutral can lead to mismatches that result in compilation or runtime errors. For more information, see [Conventions for Function Prototypes](#).

Requirements

Minimum supported client	Windows XP [desktop apps only]
Minimum supported server	Windows Server 2003 [desktop apps only]
Target Platform	Windows
Header	pdh.h
Library	Pdh.lib
DLL	Pdh.dll

See also

[PdhGetLogFileSize](#)

[PdhOpenLog](#)

[PdhOpenQuery](#)

[PdhUpdateLogFileCatalog](#)

Feedback

Was this page helpful?

 Yes

 No

[Get help at Microsoft Q&A](#)

PdhValidatePathA function (pdh.h)

Article02/22/2024

Validates that the counter is present on the computer specified in the counter path.

Syntax

C++

```
PDH_FUNCTION PdhValidatePathA(  
    [in] LPCSTR szFullPathBuffer  
);
```

Parameters

[in] szFullPathBuffer

Null-terminated string that contains the counter path to validate. The maximum length of a counter path is PDH_MAX_COUNTER_PATH.

Return value

If the function succeeds, it returns ERROR_SUCCESS.

If the function fails, the return value is a [system error code](#) or a [PDH error code](#). The following are possible values.

[+] Expand table

Return code	Description
PDH_CSTATUS_NO_INSTANCE	The specified instance of the performance object was not found.
PDH_CSTATUS_NO_COUNTER	The specified counter was not found in the performance object.
PDH_CSTATUS_NO_OBJECT	The specified performance object was not found on the computer.
PDH_CSTATUS_NO_MACHINE	The specified computer could not be found or connected to.

PDH_CSTATUS_BAD_COUNTERNAME	The counter path string could not be parsed.
PDH_MEMORY_ALLOCATION_FAILURE	The function is unable to allocate a required temporary buffer.

Remarks

Note

The pdh.h header defines PdhValidatePath 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 is 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	pdh.h
Library	Pdh.lib
DLL	Pdh.dll

See also

[PdhMakeCounterPath](#)

Feedback

Was this page helpful?

 Yes

 No

Provide product feedback  | Get help at Microsoft Q&A

PdhValidatePathExA function (pdh.h)

Article02/09/2023

Validates that the specified counter is present on the computer or in the log file.

Syntax

C++

```
PDH_FUNCTION PdhValidatePathExA(
    [in, optional] PDH_HLOG hDataSource,
    [in]           LPCSTR   szFullPathBuffer
);
```

Parameters

[in, optional] hDataSource

Handle to the data source. The [PdhOpenLog](#) and [PdhBindInputDataSource](#) functions return this handle.

To validate that the counter is present on the local computer, specify **NULL** (this is the same as calling [PdhValidatePath](#)).

[in] szFullPathBuffer

Null-terminated string that specifies the counter path to validate. The maximum length of a counter path is **PDH_MAX_COUNTER_PATH**.

Return value

If the function succeeds, it returns **ERROR_SUCCESS**.

If the function fails, the return value is a [system error code](#) or a [PDH error code](#). The following are possible values.

[+] Expand table

Return code	Description
PDH_CSTATUS_NO_INSTANCE	The specified instance of the performance object was

	not found.
PDH_CSTATUS_NO_COUNTER	The specified counter was not found in the performance object.
PDH_CSTATUS_NO_OBJECT	The specified performance object was not found on the computer or in the log file.
PDH_CSTATUS_NO_MACHINE	The specified computer could not be found or connected to.
PDH_CSTATUS_BAD_COUNTERNAME	The counter path string could not be parsed.
PDH_MEMORY_ALLOCATION_FAILURE	The function is unable to allocate a required temporary buffer.

Remarks

 **Note**

The pdh.h header defines PdhValidatePathEx 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 is 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 Vista [desktop apps only]
Minimum supported server	Windows Server 2008 [desktop apps only]
Target Platform	Windows
Header	pdh.h
Library	Pdh.lib
DLL	Pdh.dll

See also

[PdhMakeCounterPath](#)

[PdhValidatePath](#)

Feedback

Was this page helpful?



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

PdhValidatePathExW function (pdh.h)

Article 02/09/2023

Validates that the specified counter is present on the computer or in the log file.

Syntax

C++

```
PDH_FUNCTION PdhValidatePathExW(
    [in, optional] PDH_HLOG hDataSource,
    [in]           LPCWSTR szFullPathBuffer
);
```

Parameters

[in, optional] hDataSource

Handle to the data source. The [PdhOpenLog](#) and [PdhBindInputDataSource](#) functions return this handle.

To validate that the counter is present on the local computer, specify **NULL** (this is the same as calling [PdhValidatePath](#)).

[in] szFullPathBuffer

Null-terminated string that specifies the counter path to validate. The maximum length of a counter path is PDH_MAX_COUNTER_PATH.

Return value

If the function succeeds, it returns **ERROR_SUCCESS**.

If the function fails, the return value is a [system error code](#) or a [PDH error code](#). The following are possible values.

Return code	Description
PDH_CSTATUS_NO_INSTANCE	The specified instance of the performance object was not found.
PDH_CSTATUS_NO_COUNTER	The specified counter was not found in the performance

	object.
PDH_CSTATUS_NO_OBJECT	The specified performance object was not found on the computer or in the log file.
PDH_CSTATUS_NO_MACHINE	The specified computer could not be found or connected to.
PDH_CSTATUS_BAD_COUNTERNAME	The counter path string could not be parsed.
PDH_MEMORY_ALLOCATION_FAILURE	The function is unable to allocate a required temporary buffer.

Remarks

 **Note**

The pdh.h header defines PdhValidatePathEx 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 is not encoding-neutral can lead to mismatches that result in compilation or runtime errors. For more information, see [Conventions for Function Prototypes](#).

Requirements

Minimum supported client	Windows Vista [desktop apps only]
Minimum supported server	Windows Server 2008 [desktop apps only]
Target Platform	Windows
Header	pdh.h
Library	Pdh.lib
DLL	Pdh.dll

See also

[PdhMakeCounterPath](#)

[PdhValidatePath](#)

Feedback

Was this page helpful?

 Yes

 No

[Get help at Microsoft Q&A](#)

PdhValidatePathW function (pdh.h)

Article 02/09/2023

Validates that the counter is present on the computer specified in the counter path.

Syntax

C++

```
PDH_FUNCTION PdhValidatePathW(
    [in] LPCWSTR szFullPathBuffer
);
```

Parameters

[in] szFullPathBuffer

Null-terminated string that contains the counter path to validate. The maximum length of a counter path is PDH_MAX_COUNTER_PATH.

Return value

If the function succeeds, it returns ERROR_SUCCESS.

If the function fails, the return value is a [system error code](#) or a [PDH error code](#). The following are possible values.

Return code	Description
PDH_CSTATUS_NO_INSTANCE	The specified instance of the performance object was not found.
PDH_CSTATUS_NO_COUNTER	The specified counter was not found in the performance object.
PDH_CSTATUS_NO_OBJECT	The specified performance object was not found on the computer.
PDH_CSTATUS_NO_MACHINE	The specified computer could not be found or connected to.
PDH_CSTATUS_BAD_COUNTERNAME	The counter path string could not be parsed.

PDH_MEMORY_ALLOCATION_FAILURE	The function is unable to allocate a required temporary buffer.
-------------------------------	---

Remarks

ⓘ Note

The pdh.h header defines PdhValidatePath 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 is not encoding-neutral can lead to mismatches that result in compilation or runtime errors. For more information, see [Conventions for Function Prototypes](#).

Requirements

Minimum supported client	Windows XP [desktop apps only]
Minimum supported server	Windows Server 2003 [desktop apps only]
Target Platform	Windows
Header	pdh.h
Library	Pdh.lib
DLL	Pdh.dll

See also

[PdhMakeCounterPath](#)

Feedback

Was this page helpful?

 Yes

 No

[Get help at Microsoft Q&A](#)