OMRON

FinsGateway

UDM API

Programmer's Manual

Contents

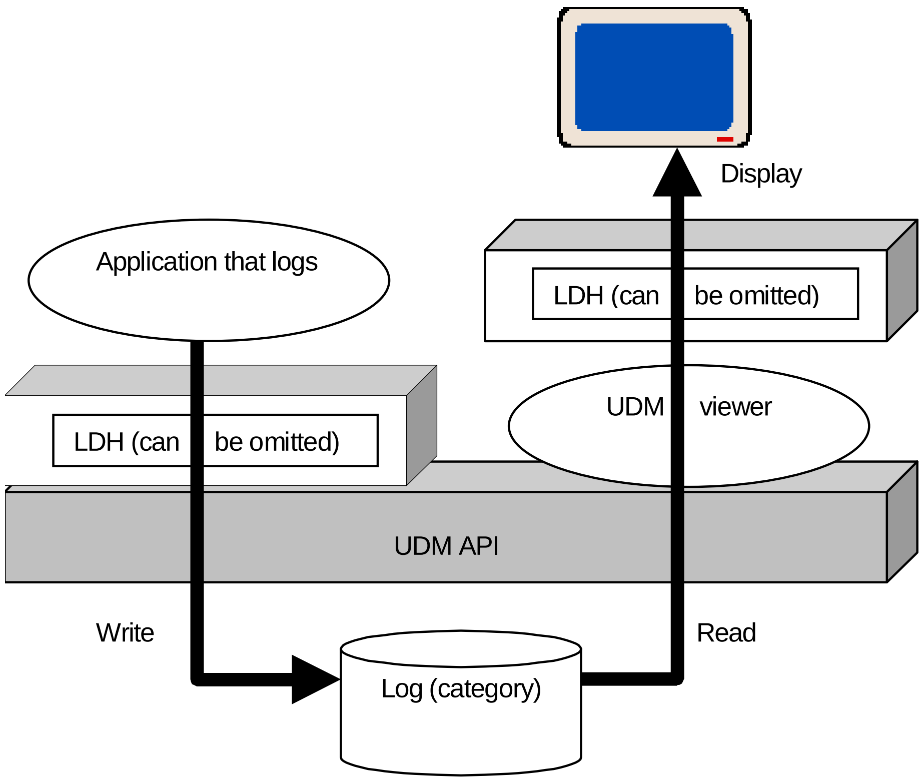
Revision History

Revision code	Date	Revised content
1.00	August 1998	Original production

1 Introduction

This manual describes the API for using the Universal Data Monitor (UDM), which provides general-purpose data logging/viewing functions. The UDM is supplied as a component of FinsGateway.

1.1 Overview



The UDM provides general-purpose data logging (through the data logging API), and a common viewer for viewing logged data. The UDM API is a set of library interfaces that constitute this mechanism.

The main features are as follows:

- High-speed logging
- Expandable log report method with the log data handler (LDH)
- Expandable display styles with the LDH

1.2 Operating Environment

The following files are required to develop applications:

DLL	FgwUdm32.dll
Import library	FgwUdm32.lib
Include file	FgwUdm.h

1.3 UDM API

Structure

Data that can be accessed from multiple applications, such as log data and non-volatile data available for the LDH, is accessed exclusively from the UDM API. The UDM API is divided into the following function groups:

- Log report interface
  Enables log reporting from applications.

- Log read interface
  Enables reading logged data. The UDM viewer uses this interface. An application can also use this interface directly to provide a different display style.
- Log management interface
  Enables log creation, deletion, status management, etc. It can be used from the UDM viewer, UdmAdministrator.exe as an MMI.
- LDH interface
  Enables expansion of the log report method and display style. To use a log report or viewer display style other than the defaults, use this interface to create an LDH.

Log Contents

In the UDM, log data is managed by dividing it into several categories. Applications specify where to log each record using the category name.

Logged records are stored in chronological order. If the number of logs stored exceeds the maximum for a certain category, records are overwritten beginning at the oldest.

In a record, common data (log time, etc.) and log-specific data can be recorded. The log-specific data is arbitrary data for each record. The following common data items are logged:

- Log time (system time)
- Name of the application that reported the log
- Event ID (indicating the meaning of the record)
- Importance of the record
- LDH designation (mainly used for displaying the record)

2 Programming Points

2.1 Before Using the UDM API

Specifying the Version to Execute

Before calling any function that provides UDM services, you must first specify the version of the UDM API to execute. The version to execute is the version number that applications request for the UDM API. This is required so that applications can operate safely without being re-compiled when future versions of the UDM API are used.

To specify the version, use the Udm_requestVersion function. If this function is not executed, all of the functions that the UDM API provides will cause errors (e.g., Execution version not set).

2.2 Writing Logs from Applications

Opening a Log Category

First, use the UdmReport_openCategory function to open a destination log category. Then specify the log category using the handle obtained by opening the category. To close the handle of an unnecessary category, use the UdmReport_closeCategory function.

Loading LDH

When necessary, use the UdmReport_loadLDH function to load the LDH used to specify the log report method. Then specify the log report method using the handle of the LDH obtained by loading the LDH. To unload an unnecessary LDH, use the UdmReport_unloadLDH function.

Writing Log Data

Specify the log category and an LDH, and then use the UdmReport_writeRecord(Ex) function to write the necessary data as a record. You can also write logs by a default operation, without specifying the LDH.

2.3 Creating an LDH

The LDH allows you to specify the log report method and the display style of the viewer. The LDH must be created as a DLL, and incorporate a function that can be called from the UDM or the viewer. Configure the registry information to specify the path of the DLL used as the LDH.

To incorporate the log report method (operation during logging) into an LDH, incorporate the following functions that are called from the UDM:

- Log report function (LDH_Report or LDH_ReportEx)
  If you use the UdmReport_writeRecord function when writing records, you must incorporate the LDH_Report function. If you use the UdmReport_writeRecordEx function, the LDH_ReportEx function is also required.
- The LDH_GetFilterDataSize function
  This is to notify the size of the non-volatile data that can be used to define the log report operation when writing a record. This function must always be incorporated.

To incorporate a specific viewer display style into an LDH, incorporate the following functions called from the UDM viewer. Basically, these functions must be always incorporated. The UDM viewer uses the default operations if not all of the functions are incorporated in the LDH.

- The LDH_getSummary function creates messages to be displayed in the log summary. By default, no detailed data is displayed.
- The LDH_showEssenceDialog function calls the dialog to display the detailed log data. By default, the detailed data, composed of the common data and a binary data stream, is displayed in the dialog.

- The LDH_showFilterDialog function calls the dialog to configure the non-volatile data available for defining the log write operation. By default, the dialog is not displayed.
- The LDH_getCSV function creates the data to be stored when the log is stored in CSV format. By default, the detailed data, composed of the common data and a binary data stream, is stored in CSV format.

To specify the path of the DLL used as the LDH, specify the LDH DLL path in the entry under the following registry sub-key:

HKEY_LOCAL_MACHINE¥SOFTWARE¥OMRON¥FinsGateway¥UdmLDH

You can use REG_EXPAND_SZ or REG_SZ as a type code for the entry.

From programs, specify the LDH using the entry name set in the registry.

When loading an LDH for a log report, specify the entry name used in the registry.

Use the entry name in the log record as well, to specify the LDH to use for display in the viewer.

3 Errors

3.1 Function Call Errors

All errors that occur in the UDM API are notified by an error returned from the function. If a UDM API library function call fails, the function returns one of the following values:

- Null (0)
- UDM_RETURN_CODE_FAILED (-1)
- FALSE (0)

Applications can identify the cause of a failure by calling the GetLastError function. Error codes are managed on a thread basis.

An error code is a 32-bit value. (Bit 31 is the most significant bit.) For any error code the UDM API defines, bit 29 is always set to 1. If bit 29 is not set, the error code is from the Win32 API. The following table is a list of the error codes from the UDM API. The GetLastError function obtains the error code by calculating the OR of the error code in the following table and 0x20000000 for each bit.

List of Error Codes for Functions

Code	Defined Name (UDM_ERROR_*)	Explanation
1	INVALID_CATEGORY_NAME	The category name is invalid.
2	INVALID_MAX_RECORDS	The maximum number of records is invalid.
3	INVALID_MAX_RECORD_SIZE	The number of records is invalid.
4	INVALID_REPORT_STATUS	The report ON/OFF setting is invalid.
5	EXCEPTION_BREAKOUT	An exception occurred during processing.
6	CREATE_MUTEX_FAIL	Mutex creation failed.
7	OWN_MUTEX_FAIL	Mutex ownership acquisition failed.
8	CATEGORY_ALREADY_EXISTS	The specified category already exists.
9	CREATE_FOLDER_FAIL	Folder creation failed.
10	CREATE_FILE_FAIL	File creation (open) failed.
11	CREATE_SHARED_MEMORY	Shared memory creation failed.
12	MEMORY_ALLOC_FAIL	Memory allocation failed.
13	WRONG_CHECK_SUM	The file checksum is invalid. The log file is corrupted.
14	CATEGORY_NOT_EXISTS	The specified category does not exist.
15	CATEGORY_IN_USE	The specified category is in use.
16	DELETE_FILE_FAIL	File deletion failed.
17	FILE_PATH_GET_FAIL	File path acquisition failed.
18	FIND_FILE_FAIL	File find failed.
19	INVALID_CATEGORY_INDEX	The index to specify the category is invalid.
20	CATEGORY_NAME_BUFFER_SHORT	The buffer to store category names is too small.
21	NOW_WRITE_ENABLE	The specified category is now write-enabled.
22	REPORT_STATUS_UNCHANGEABLE	The report ON/OFF setting cannot be changed.
23	CREATE_EVENT_FAIL	Event object creation failed.
24	NOW_WRITE_DISABLE	The specified category is now write-disabled.
25	OPEN_SUBKEY_FAIL	Registry sub-key open failed.
26	QUERY_VALUE_FAIL	Registry value acquisition failed.
27	LOAD_LIBRARY_FAIL	DLL load failed.
28	NO_FUNCTIONS	Functions required for the specified DLL do not exist.
29	ENUM_VALUE_FAIL	Registry enumeration failed.
30	ATTACH_NON_VOLATILE_DATA_FAIL	Non-volatile data attach failed.
31	NON_VOLATILE_DATA_NOT_SET	Non-volatile data is not set.
32	EXE_VERSION_ALREADY_SET	The version to execute has already been set.
33	REQUEST_VERSION_NOT_EXISTS	The specified version to execute does not exist.
34	EXE_VERSION_NOT_SET	The version to execute is not set.
35	WRITE_AREA_SHORT	Write area is too small. Part of the data is not written.
36	READ_AREA_SHORT	Read area is too small. Part of the data is not read.
37	NO_LDH_WRITE	The LDH does not contain the log write function.

4 UDM API Reference

The library consists of the functions shown below:

Version Management

Udm_requestVersion	Specifies the version to execute.
Udm_getVersion	Obtains the DLL release version.

Log Write/Read

UdmReport_openCategory	Opens a category.
UdmReport_closeCategory	Closes a category.
UdmReport_writeRecord	Writes a log record.
UdmReport_writeRecordEx	Writes a log record (vector mode)
UdmReport_loadLDH	Loads an LDH.
UdmReport_unloadLDH	Unloads an LDH.
UdmView_enumRecord	Reads a log record.

Category Management

UdmAdm_createCategory	Creates a category.
UdmAdm_deleteCategory	Deletes a category.
UdmAdm_enumCategory	Enumerates the categories.
UdmAdm_clearRecords	Clears a log record.
UdmAdm_changeCategorySize	Changes the size of a category.
UdmAdm_getCategoryUsedCount	Obtains how many times the category has been used.
UdmAdm_changeReportOnOff	Switches the log ON/OFF status of a category.
UdmView_getCategoryInformation	Obtains category information.
UdmView_getCategoryInformationByName	Obtains category information (by name)

LDH Incorporated Functions

LDH_Report	Runs a log report.
LDH_ReportEx	Runs a log report (vector mode)
LDH_GetFilterDataSize	Obtains the size of non-volatile data.
LDH_getSummary	Obtains the log summary.
LDH_showEssenceDialog	Displays the dialog for detailed data.
LDH_showFilterDialog	Displays the dialog for configuring non-volatile data.
LDH_getCSV	Obtains CSV data.

LDH Helper Functions

UdmView_getLDHPath	Obtains the LDH DLL path.
UdmView_enumLDHPath	Enumerates the LDH DLL paths.
UdmLDH_setData	Configures the non-volatile data for an LDH.
UdmLDH_getData	Obtains the non-volatile data for an LDH.

Other

Udm_getLastErrorMessage	Obtains an error message.

4.1 Udm_requestVersion Function

Function

Defines the UDM API version to execute.

```
BOOL Udm_requestVersion(
        BYTE byMajor,    //Major version
        BYTE byMinor     //Minor version
        )
```

Comments

The Udm_requestVersion function requests that the UDM operate in the specified version. If the version to execute is not specified using this function, all of the UDM API functions will fail. The UDM_STARTUP() macro includes the Udm_requestVersion function specifying the release version.

Parameters	Description
byMajor	Specify the major version number. In FgwUdm.h, the major version at release is defined as UDM_CURRENT_MAJOR_VERSION.
byMinor	Specify the minor version number. In FgwUdm.h, the minor version at release is defined as UDM_CURRENT_MINOR_VERSION.

Return Value

The return value is TRUE if the function completes normally. Otherwise, the return value is FALSE. To obtain the error data, use the GetLastError function.

If the version to execute has been specified already, it is not changed and the function completes normally. In this case, the GetLastError function will return the UDM_ERROR_EXE_VERSION_ALREADY_SET error code.

If a version not in the release history is specified, an error will occur.

See Also

Udm*

4.2 Udm_getVersion Function

Function

Obtains the release version of FgwUdm32.dll.

UDM_VERSION Udm_getVersion (VOID)

Comments

The Udm_getVersion function obtains the release version of the FgwUdm32.dll file.

This version is irrelevant to the execution version requested by Udm_requestVersion.

Parameters	Description
None	---

Return Value

The return value is the UDM_VERSION structure. This function never fails.

```
typedef struct _UDM_VERSION {
        BYTE    ByMajor; //Major version.
        BYTE    ByMinor; //Minor version.
        BYTE    ByRevision; //Revision.
        BYTE    ByReserved; //Reserved.
} UDM_VERSION;
```

See Also

Udm_requestVersion

4.3 UdmReport_openCategory Function

Function

Opens the specified category.

```
HANDLE WINAPI UdmReport_openCategory (  //Category handle
        LPCTSTR szCategory,             //Category name (null terminated)
        LPCTSTR szApplication           //Application name (null terminated)
        )
```

Comments

The UdmReport_openCategory function opens the specified category. You can write a log record by specifying the handle of an open category.

Parameters	Description
szCategory	Specify the category name using a string ending with null. The maximum number of characters of szCategory is UDM_CATEGORY_NAME_MAX_LENGTH (=128).
szApplication	Specify an application name using a string ending with null. The maximum number of characters of szApplication is UDM_APP_NAME_LENGTH (=64). The name specified here is recorded in the log as the application name.

Return Value

The return value is the category handle if the function completes normally. Otherwise, the return value is null. To obtain the error data, use the GetLastError function.

If a category that does not exist is specified, an error will occur.

See Also

UdmReport_writeRecord, UdmReport_writeRecordEx, UdmView_enumRecord

4.4 UdmReport_closeCategory Function

Function

Closes the specified category.

```
BOOL WINAPI UdmReport_closeCategory (      //TRUE|FALSE
            HANDLE hCategory)              //Category handle
```

Comments

The UdmReport_closeCategory function closes the specified category.

Parameters	Description
hCategory	Specify the handle of the category to be closed.

Return Value

The return value is TRUE if the function completes normally. Otherwise, the return value is FALSE. To obtain the error data, use the GetLastError function.

See Also

UdmReport_openCategory

4.5 UdmReport_writeRecord Function

Function

Writes a log record into the specified category.

```
BOOL WINAPI UdmReport_writeRecord(          //TRUE|FALSE
        HANDLE hCategory,                   //Category handle
        HANDLE hLDH,                        //LDH handle
        PUDM_COMMON_DATA    pCommonData,    //Common data
        LPVOID  pvEssenceData,
                            //Pointer to Essential Data Buffer
        int     lDataSize                   //Size of Essential Data
        )
```

Comments

The UdmReport_writeRecord function writes a log record into the specified category. In the default log report, the data specified by the UdmReport_writeRecord function is stored as a record together with the system time when the data was written. For a Win32 API event object, the function executes PulseEvent to notify applications that a log record was written. For the name of the target event object, the suffix UDM_EVENT_NAME_SUFFIX (="_UDM_event") is added to the end of the category name.

Parameters	Description
hCategory	Specify the category handle.
hLDH	Specify the LDH handle. If null is specified, the default log operation is executed without using an LDH.
pCommonData	This is the common log data handled by the UDM. Each member is as follows: lImportance: Importance of the record lEventID: EventID that defines the meaning of the record. LogHandlerDll: Specify the LDH used for displaying the record in the viewer. lLDHtype: Specify either of following parameters: eUdm_useDefaultLDH / eUdm_useRegistryName specifyDll.RegistryName: The entry name specified in the registry to define the LDH. If you specify eUdm_useDefaultLDH in lLDHtype, then RegistryName is not required.
pvEssenceData	Specify the pointer to the buffer to store the log-specific write data.
lDataSize	Specify the number of bytes of log-specific write data.

```
typedef struct _ UDM_COMMON_DATA {
        int     lImportance;        //Importance of the record
        int     lEventID;
                //EventID, specify the importance of the record
        UDM_LDH_DLL    logHandlerDll;    //Log handler dll
  } UDM_COMMON_DATA;
typedef struct _UDM_LDH_DLL {       //FLogDataHandler
        int     lLDHtype;
        union {
        //Name of the registry that has the LogDataHandler (DLL) path
        char    registryName[UDM_LOG_HANDLER_NAME_MAX_LENGTH];
        CLSID   clsID;              //CLSID
        } specifyDll;
} UDM_LDH_DLL, *PUDM_LDH_DLL;
```

Return Value

The return value is TRUE if the function completes normally. Otherwise, the return value is FALSE. To obtain the error data, use the GetLastError function.

If the number of bytes of the specified data exceeds the maximum record size for a category, the data is written until the maximum record size is reached, and the function completes normally. In this case, the GetLastError function will obtain the UDM_ERROR_WRITE_AREA_SHORT error code.

See Also

UdmReport_openCategory, UdmReport_loadLDH

4.6 UdmReport_writeRecordEx Function

Function

Writes a log report into the specified category.

```
BOOL WINAPI UdmReport_writeRecordEx(      //TRUE|FALSE
        HANDLE hCategory,                 //Category handle
        HANDLE hLDH,              //LDH handle
        PUDM_COMMON_DATA    pCommonData,    //Common data
        PUDM_ESSENCE_DATA   pEssenceDataTable,
                            //Pointer to essential data table
        int       lNumberOfEssenceData
                            //Number of essential data
        )
```

Comments

The UdmReport_writeRecordEx function writes a log report into the specified category. Unlike the UdmReport_writeRecord function, this function can write multiple memory block data using the vector mode. In the default log report, the data specified by the UdmReport_writeRecord function is stored as a record together with the system time when the data was written. For a Win32 API event object, the function executes PulseEvent to notify applications that a log record was written. For the name of the target event object, the suffix UDM_EVENT_NAME_SUFFIX (="_UDM_event") is added to the end of the category name.

Parameters	Description
hCategory	Specify the category handle.
hLDH	Specify the LDH handle. If null is specified, the default log operation is executed without using an LDH.
pCommonData	This is the common log data handled by the UDM. For details, refer to the same item described in the UdmReport_writeRecord function.
pEssenceDataTable	Specify the pointer to the vector table to specify the data for the log-specific write data. The vector table is a UDM_ESSENCE_DATA structure array. Each member for the structure is as follows: lDataSize: The number of bytes of log-specific write data. pvDataBuffer: The pointer to the buffer to store the log-specific write data. lNumberOfEssenceData: The number of UDM_ESSENCE_DATA structures in the vector table.

Return Value

The return value is TRUE if the function completes normally. Otherwise, the return value is FALSE. To obtain the error data, use the GetLastError function.

If the number of bytes of the specified data exceeds the maximum record size for a category, the data is written until the maximum record size is reached, and the function completes normally. In this case, the GetLastError function will obtain the UDM_ERROR_WRITE_AREA_SHORT error code.

See Also

UdmReport_openCategory, UdmReport_loadLDH

4.7 UdmReport_loadLDH Function

Function

Loads the specified LDH.

```
HANDLE WINAPI UdmReport_loadLDH(          //LDH handle, null for failure
          LPCTSTR szDataHandler
          //Name of LDH (=Name of the registry
          //that has the LDH (DLL) path)
          )
```

Comments

This function loads the LDH used by the log report. By using the handle of the loaded LDH, the log report operation can be specified.

Parameters	Description
szDataHandler	Specify the entry name in the registry to define the LDH using a string ending with null. The maximum number of characters of szDataHandler is UDM_LOG_HANDLER_NAME_MAX_LENGTH (=16).

Return Value

The return value is the LDH handle if the function completes normally. Otherwise, the return value is null. To obtain the error data, use the GetLastError function.

See Also

UdmReport_writeRecord, UdmReport_writeRecordEx

4.8 UdmReport_unloadLDH Function

Function

Unloads the specified LDH.

```
BOOL WINAPI UdmReport_unloadLDH(        //TRUE|FALSE
        HANDLE hLDH                     //LDH handle
        )
```

Comments

Unloads the specified LDH.

Parameters	Description
hLDH	Specify the LDH handle.

Return Value

The return value is TRUE if the function completes normally. Otherwise, the return value is FALSE. To obtain the error data, use the GetLastError function.

See Also

UdmReport_loadLDH

4.9 UdmView_enumRecord Function

Function

Enumerates the log records stored in the specified category.

```
int WINAPI UdmView_enumRecord(
        HANDLE hCategory,           //Category handle
        int    relativeIndex,       //Relative index for the last record:
        //UDM_LAST_RECORD_INDEX specifies the current last record
        //Older records than the last one are specified by relative index
        //ex. (UDM_LAST_RECORD_INDEX + 1) specifies the next record to last one
        PUDM_RECORD_STATE     pGetRecordState,
        //Pointer to record state buffer
        LPFILETIME      pLoggedTime,      //Pointer to logged time buffer
        LPTSTR szApplication,       //Pointer to application name buffer
        int    lApplicationSize,    //Size of application name buffer
        PUDM_COMMON_DATA      pCommonData,     //Pointer to common data buffer
        LPVOID pvEssenceData,       //Pointer to essential data buffer
        PINT   lpDataSize           //Pointer to essential data size buffer
        //IN: Buffer size; OUT: Get data size
        )
```

Comments

The UdmView_enumRecord function enumerates the log records stored in the specified category.

Parameters	Description
hCategory	Specify the category handle.
relativeIndex	This is the relative index for the last record. If UDM_LAST_RECORD_INDEX is first specified in relativeIndex, the newest record will be read. Then, by incrementing the relative index one by one each time a record is read, the record just prior to the record read last, can be read.
pGetRecordState	This is information about the obtained record. Each member for the structure is as follows: lGetRecordAbsoluteIndex: The absolute index in the category dwGetRecordOverWriteCount: The number of overwrites
pLoggedTime	This is the pointer to the buffer to store the log time (system time).
szApplication	This is the pointer to the buffer to store the name of application.
pCommonData	This is the pointer to the buffer to store the common log data. For details, see the same item described in the UdmReport_writeRecord function.
pvEssenceData	This is the pointer to the buffer to store the log-specific read data.
lpDataSize	This is the pointer to the buffer to store the number of bytes of log-specific read data. Before calling the function, specify the number of bytes of the buffer to store read data. After calling the function, the number of bytes of read data is stored.

Return Value

The return value is UDM_NO_RECORD if the specified record does not exist. In this case, the record is not read.

It is UDM_OLDEST_RECORD if the read record is the oldest in the category.

It is UDM_RETURN_CODE_FAILED if the function does not complete normally.

Otherwise, it is UDM_RETURN_CODE_SUCCESS.

If the number of bytes of the specified buffer is less than the size of the record, the data is read until the number of bytes of the buffer is reached, and the function completes normally. In this case, the GetLastError function will obtain the UDM_ERROR_READ_AREA_SHORT error code.

See Also

UdmReport_openCategory

4.10 UdmAdm_createCategory Function

Function

Creates the specifed category.

```
BOOL WINAPI UdmAdm_createCategory(        //TRUE|FALSE
        LPCTSTR             szCategory,           //Category name (null terminated)
        PUDM_CATEGORY             pCategory             //Pointer to category buffer
        )
```

Comments

The UdmAdm_createCategory function creates the specified category.

Parameters	Description
szCategory	Specify the category name using a string ending with null. The maximum number of characters of szCategory is UDM_CATEGORY_NAME_MAX_LENGTH (=128).
pCategory	This is the pointer to the buffer to specify the data of the category to create. Each member is as follows: lMaxRecordCount: The maximum record count lMaxEssenceDataSize: The maximum size of detailed data per record lReportEnable: The log ON/OFF status at creation UDM_REPORT_DISABLE: Report OFF UDM_REPORT_ENABLE: Report ON UDM_REPORT_UNCHANGEABLE: Always Report ON szVersion: Version (a string ending with null) szComment: Comment (a string ending with null)

```
typedef struct _UDM_CATEGORY {
        int lMaxRecordCount;           //Maximum record count
        int lMaxEssenceDataSize;    //Maximum size of detailed data for each record
        int lReportEnable;  //Report ON/OFF
        char szVersion[UDM_CATEGORY_VERSION_LENGTH]; //Version
        char szComment[UDM_CATEGORY_COMMENT_LENGTH];//Comment
} UDM_CATEGORY, *PUDM_CATEGORY;
```

Return Value

The return value is TRUE if the function completes normally. Otherwise, the return value is FALSE. To obtain the error data, use the GetLastError function.

See Also

UdmAdm_deleteCategory

4.11 UdmAdm_deleteCategory Function

Function

Deletes the specified category.

```
BOOL WINAPI UdmAdm_deleteCategory(        //TRUE|FALSE
        LPCTSTR         szCategory        //Category name (null terminated)
        )
```

Comments

The UdmAdm_deleteCategory function deletes the specified category. If the category is in use, it cannot be deleted.

Parameters	Description
szCategory	Specify the category name using a string ending with null. The maximum number of characters of szCategory is UDM_CATEGORY_NAME_MAX_LENGTH (=128).

Return Value

The return value is TRUE if the function completes normally. Otherwise, the return value is FALSE. To obtain the error data, use the GetLastError function.

See Also

UdmAdm_createCategory

4.12 UdmAdm_enumCategory Function

Function

Enumerates the specified category information.

```
int WINAPI UdmAdm_enumCategory(
        int index, //Index: First specify UDM_FIRST_CATEGORY_INDEX
        //Then specify any other value
        //UDM_FIRST_CATEGORY_INDEX.
        LPTSTR  szCategory,             //Pointer to category name buffer
        int     lBufferSize,            //Size of category name buffer
        PUDM_CATEGORY       pCategory       //Pointer to category buffer
        )
```

Comments

The UdmAdm_enumCategory function enumerates the specified category information.

Parameters	Description
index	Specify the index for the obtained category. To call UdmAdm_enumCategory, first specify UDM_FIRST_CATEGORY_INDEX. Then, specify any other value.
szCategory	Specify the pointer to the buffer to store the category name.
lBufferSize	Specify the size of the buffer to store the category name.
Pcategory	Specify the pointer to the buffer to store the category information.

Return Value

The return value is UDM_RETURN_CODE_SUCCESS if the function completes normally. The return value is UDM_NO_CATEGORY if the category to be obtained does not exist. In this case, category information is not obtained. Otherwise, the return value is FALSE. To obtain the error data, use the GetLastError function.

See Also

UdmAdm_createCategory, UdmAdm_deleteCategory

4.13 UdmAdm_clearRecords Function

Function

Clears the records from the specified category.

```
BOOL WINAPI UdmAdm_clearRecords(
        LPCTSTR        szCategory        //Category name (null terminated)
        )
```

Comments

The UdmAdm_clearRecords function clears all the records in the specified category. The records can be cleared only when the category is not in use and the log report is set to OFF.

Parameters	Description
szCategory	Specify the category name using a string ending with null. The maximum number of characters of szCategory is UDM_CATEGORY_NAME_MAX_LENGTH (=128).

Return Value

The return value is TRUE if the function completes normally. Otherwise, the return value is FALSE. To obtain the error data, use the GetLastError function.

See Also

UdmAdm_createCategory, UdmAdm_getCategoryUsedCount, UdmAdm_changeReportOnOff

4.14 UdmAdm_changeCategorySize Function

Function

Changes the size of the specified category.

```
BOOL WINAPI UdmAdm_changeCategorySize(
        LPCTSTR         szCategory,         //Category name (null terminated)
        int         lMaxRecordCount,        //Maximum record count
        int         lMaxEssenceDataSize     //Maximum size of detailed data for each
record
        )
```

Comments

The UdmAdm_changeCategorySize function changes the size of the specified category (i.e., the maximum record count, and maximum size of detailed data for each record). If the category is in use, the size cannot be changed.

Parameters	Description
szCategory	Specify the category name using a string ending with null. The maximum number of characters of szCategory is UDM_CATEGORY_NAME_MAX_LENGTH (=128).
lMaxRecordCount	Specify the maximum record count.
lMaxEssenceDataSize	Specify the maximum size of detailed data for each record.

Return Value

The return value is TRUE if the function completes normally. Otherwise, the return value is FALSE. To obtain the error data, use the GetLastError function.

See Also

UdmAdm_createCategory, UdmReport_openCategory, UdmAdm_getCategoryUsedCount

4.15 UdmAdm_changeReportOnOff Function

Function

Switches the log report of the specified category ON/OFF.

```
BOOL WINAPI UdmAdm_changeReportOnOff(
        LPCTSTR         szCategory,         //Category name (null terminated)
        BOOL    bReportEnable               //Report ON/OFF
        )
```

Comments

The UdmAdm_changeReportOnOff function switches the log report of the specified category ON/OFF.

Parameters	Description
szCategory	Specify the category name using a string ending with null. The maximum number of characters of szCategory is UDM_CATEGORY_NAME_MAX_LENGTH (=128).
bReportEnable	Set the log report to ON or OFF.

Return Value

The return value is TRUE if the function completes normally. Otherwise, the return value is FALSE. To obtain the error data, use the GetLastError function.

See Also

UdmAdm_createCategory

4.16 UdmView_getCategoryInformation Function

Function

Obtains the specified category information.

```
BOOL WINAPI UdmView_getCategoryInformation(          //TRUE|FALSE
            HANDLE hCategory,                        //Category handle
            PUDM_CATEGORY          pCategory         //Pointer to category buffer
            )
```

Comments

The UdmView_getCategoryInformation function obtains category information.

Parameters	Description
hCategory	Specify the category handle.
pCategory	Specify the pointer to the buffer to store the category information.

Return Value

The return value is TRUE if the function completes normally. Otherwise, the return value is FALSE. To obtain the error data, use the GetLastError function.

See Also

UdmAdm_createCategory

4.17 UdmView_getCategoryInformationByName Function

Function

Obtains the specified category information by name.

```
BOOL WINAPI UdmView_getCategoryInformationByName(
        LPCTSTR         szCategory,           //Pointer to category name buffer
        PUDM_CATEGORY          pCategory      //Pointer to category buffer
        )
```

Comments

The UdmView_getCategoryInformationByName function obtains the specified category information using the category name.

Parameters	Description
szCategory	Specify the category name using a string ending with null. The maximum number of characters of szCategory is UDM_CATEGORY_NAME_MAX_LENGTH (=128).
pCategory	Specify the pointer to the buffer to store the category information.

Return Value

The return value is TRUE if the function completes normally. Otherwise, the return value is FALSE. To obtain the error data, use the GetLastError function.

See Also

UdmAdm_createCategory

4.18 LDH_Report Function

Function

Runs a log report.

```
BOOL WINAPI LDH_Report(
        PLDH_REPORT_IF pReportSingleIf
        )
```

Comments

The LDH_Report function is to be implemented into an LDH. If the UdmReport_writeRecord function is called by an application, the function runs a log report using the specified LDH.

Parameters	Description
pReportSingleIf	This is the interface to perform a log report with an LDH. Each member is as follows: dataWriteIf: The interface to write the log. This contains the data specified by UdmReport_writeRecord, and the function and the handle of the category to write the log record. eventSendIf: The interface to execute PulseEvent for Win32 API event objects. This contains the function and the handle of the event object to set the event. filterData: The non-volatile data that can be used to define the log report operation. To set this data, use the UdmLDH_setData function.

```
typedef struct _LDH_SINGLE_DATUM_WRITE_IF
{
        LPCTSTR             szApplication;      //Application name
        PUDM_COMMON_DATA    pCommonData;    //Pointer to common data
        int         lEssenceSize;       //Size of essential data
        PVOID       pEssenceData;
        //Pointer to Essential Data Buffer.
        LDH_WRITE_SINGLE_DATA           fWriteData;         //Data-write function
        HANDLE hWritePort
        //Write port: write data to the port, refer to a category
} LDH_SINGLE_DATUM_WRITE_IF, *PLDH_SINGLE_DATUM_WRITE_IF;
typedef struct _LDH_EVENT_SEND_IF
{
        LDH_SEND_EVENT          fSendEvent;         //Event-send function
        HANDLE hSendPort;
        //Send port: send event to the port, refer to a category.
} LDH_EVENT_SEND_IF, *PLDH_EVENT_SEND_IF;
typedef struct _LDH_FILTER_DATA
{
        int         lDataSize;          //Non-volatile data size
        PVOID       pvDataBuffer;       //Pointer to non-volatile data
} LDH_FILTER_DATA, *PLDH_FILTER_DATA;
typedef struct _LDH_REPORT_IF
{
        LDH_SINGLE_DATUM_WRITE_IF    dataWriteIf;        //Interface to write data
        LDH_EVENT_SEND_IF       eventSendIf;        //Interface to send event
        LDH_FILTER_DATA         filterData;         //Filter Data for LDH
} LDH_REPORT_IF, *PLDH_REPORT_IF;
```

Return Value

The return value is the same as that of the UdmReport_writeRecord function.

See Also

UdmReport_writeRecord, UdmLDH_setData, LDH_GetFilterDataSize

4.19 LDH_ReportEx Function

Function

Runs a log report.

```
BOOL WINAPI LDH_ReportEx(
        PLDH_REPORT_EX_IF pReportMultiIf
        )
```

Comments

The LDH_ReportEx function is to be implemented into an LDH. If the UdmReport_writeRecordEx function is called by an application, the function performs a log report using the specified LDH.

Parameters	Description
pReportMultiIf	This is the interface to run a log report. Each member is as follows: dataWriteIf: The interface to write the log. This contains the data specified by UdmReport_writeRecord, and the function and handle of the category to write the log record. eventSendIf: The interface to execute PulseEvent for Win32 API event objects. This contains the function and handle of the event object to set the event. filterData: The non-volatile data that can be used to define the log report operation. To set this data, use the UdmLDH_setData function.

```
typedef struct _LDH_MULTI_DATA_WRITE_IF
{
        LPCTSTR             szApplication;      //Application name
        PUDM_COMMON_DATA    pCommonData;    //Pointer to common data
        int         lNumberOfEssenceData;    //Number of essential data
        PUDM_ESSENCE_DATA   pEssenceDataTable;
        //Pointer to essential data table
        LDH_WRITE_MULTI_DATA fWriteData;         //Data-write function
        HANDLE hWritePort;
        //Write port: write data to the port, refer to a category
} LDH_MULTI_DATA_WRITE_IF, *PLDH_MULTI_DATA_WRITE_IF;
typedef struct _LDH_EVENT_SEND_IF
{
        LDH_SEND_EVENT          fSendEvent;         //Event-send function
        HANDLE hSendPort;
        //Send port: send event to the port, refer to a category
} LDH_EVENT_SEND_IF, *PLDH_EVENT_SEND_IF;
typedef struct _LDH_FILTER_DATA
{
        int         lDataSize;          //Non-volatile data size
        PVOID       pvDataBuffer;       //Pointer to non-volatile data
} LDH_FILTER_DATA, *PLDH_FILTER_DATA;
typedef struct _LDH_REPORT_EX_IF
{
        LDH_MULTI_DATA_WRITE_IF         dataWriteIf;        //Interface to write data
```

```
		LDH_EVENT_SEND_IF	eventSendIf;		//Interface to send event
		LDH_FILTER_DATA		filterData;			//Filter data for
LDH
} LDH_REPORT_EX _IF, *PLDH_REPORT_EX _IF;
```

Return Value

The return value is the same as that of the UdmReport_writeRecordEx function.

See Also

UdmReport_writeRecordEx, UdmLDH_setData, LDH_GetFilterDataSize

4.20 LDH_GetFilterDataSize Function

Function

Obtains the size of non-volatile data that can be used in the log report.

```
int WINAPI LDH_GetFilterDataSize(
          )
```

Comments

The LDH_GetFilterDataSize function obtains the number of bytes of non-volatile data used by the LDH_Report and LDH_ReportEx functions. If the UdmReport_loadLDH function is called by an application, this function is called by the UDM.

Parameters	Description
None	---

Return Value

The return value is the number of bytes of non-volatile data. It is 0 if non-volatile data is not used by the LDH_Report function.

See Also

UdmReport_loadLDH, UdmReport_writeRecord, UdmReport_writeRecordEx, UdmLDH_setData, UdmLDH_getData

4.21 LDH_getSummary Function

Function

Obtains the message displayed by the viewer.

```
int WINAPI LDH_getSummary(
        int     nEventID,
        LPVOID  pData,
        int     lDataSize,
        char    *szSummary,
        int     nSummarySize
        )
```

Comments

The LDH_getSummary function obtains the message displayed as a summary by the viewer.
If the viewer displays a log list, the viewer calls this function.

Parameters	Description
nEventID	Specify the event ID.
pData	Specify the pointer to the buffer to store the detailed data.
lDataSize	Specify the number of bytes of detailed data.
szSummary	Specify the leading pointer to a string buffer to store the message. Message strings must be terminated with null.
nSummarySize	Specify the number of bytes of the string buffer to store the message.

Return Value

The return value is the number of bytes for the message string.

See Also

UdmReport_writeRecord, UdmReport_writeRecordEx, UdmView_enumRecord

4.22 LDH_showEssenceDialog Function

Function

Shows the detailed data dialog.

```
BOOL WINAPI LDH_showEssenceDialog(
        HWND    hParentWnd,
        const FILETIME    FileTime,
        LPCTSTR           szApplication,
        const PUDM_COMMON_DATA        pCommonData,
        LPVOID  pData,
        int         lDataSize
        )
```

Comments

The LDH_showEssenceDialog function shows the detailed data dialog. When the viewer shows the detailed data, it calls this function.

Parameters	Description
hParentWnd	Specify the handle of the parent window.
FileTime	This is the log time (system time).
szApplication	Specify the pointer to the buffer to store the name of the application that reported the log record.
pCommonData	Specify the pointer to the buffer to store the common log data.
pData	Specify the leading pointer to the buffer to store the detailed data.
lDataSize	Specify the number of bytes of detailed data.

Return Value

The return value is TRUE if the dialog is displayed properly. The return value is FALSE if the dialog is not displayed.

See Also

UdmReport_writeRecord, UdmReport_writeRecordEx, UdmView_enumRecord

4.23 LDH_showFilterDialog Function

Function

Displays the dialog to perform the LDH-specific settings, such as the log report method.

```
BOOL WINAPI LDH_showFilterDialog(
        HWND hParentWnd
        )
```

Comments

The LDH_showFilterDialog function displays the dialog to perform the LDH-specific settings. The viewer calls this function.

Parameters	Description
hParentWnd	Specify the handle of the parent window.

Return Value

The return value is TRUE if the dialog is displayed properly. The return value is FALSE if the dialog is not displayed.

See Also

UdmLDH_setData

4.24 LDH_getCSV Function

Function

Obtains the CSV data logged by the viewer.

```
int WINAPI LDH_getCSV(
        const FILETIME    FileTime,
        LPCTSTR           szApplication,
        const PUDM_COMMON_DATA        pCommonData,
        LPVOID  pData,
        int     lDataSize,
        char    *szCSV,
        int     nCSVSize
        )
```

Comments

The LDH_getCSV function obtains the CSV format data that the viewer uses for saving and displaying. When the viewer saves a log record, it calls this function.

Parameters	Description
FileTime	This is the log time (system time).
szApplication	Specify the pointer to the buffer to store the name of the application that reported the log record.
pCommonData	Specify the pointer to the buffer to store the common log data.
pData	Specify the leading pointer to the buffer to store the detailed data.
lDataSize	Specify the number of bytes of detailed data.
szCSV	Specify the pointer to the string buffer to store the CSV data. The CSV data must be terminated with null.
nCSVSize	Specify the number of bytes of the string buffer to store the CSV data.

Return Value

The return value is the number of bytes of the CSV data string.

See Also

UdmReport_writeRecord, UdmReport_writeRecordEx, UdmView_enumRecord

4.25 UdmView_getLDHPath Function

Function

Obtains the LDH DLL path from the registry.

```
BOOL WINAPI UdmView_getLDHPath(          //TRUE|FALSE
        LPCTSTR         szDataHandler,   //Name of LDH (=Name of registry entry
        //          that defines LDH (DLL) path)
        LPTSTR  szLDHPath,          //Not expanded LDH (DLL) path
        int         lBufferSize         //Buffer size of LDH path
        )
```

Comments

The UdmView_getLDHPath function obtains the LDH DLL path from the registry.

Parameters	Description
szDataHandler	Specify the entry name in the registry that defines the LDH using a string ending with null. The maximum number of characters of szDataHandler is UDM_LOG_HANDLER_NAME_MAX_LENGTH (=16).
szLDHPath	The leading pointer to the buffer to store the LDH DLL path.
lBufferSize	The number of bytes of the buffer to store the LDH DLL path.

Return Value

The return value is TRUE if the function completes normally. Otherwise, the return value is FALSE. To obtain the error data, use the GetLastError function.

See Also

LDH_getSummary, LDH_showEssenceDialog, LDH_showFilterDialog, LDH_getCSV

4.26 UdmView_enumLDHPath Function

Function

Enumerates the LDH DLL paths from the registry entries.

```
BOOL WINAPI UdmView_enumLDHPath(          //TRUE|FALSE
          DWORD   index,       //Index: First specify 0
          //Then, increment index 1, by 1
          LPTSTR  szDataHandler,      //Name of LDH (=Name of registry entry
          //      that defines LDH (DLL) path)
          int     lNameBufferSize,   //BufferSize of LDH name
          LPTSTR  szLDHPath,          //Not expanded LDH (DLL) path
          int     lPathBufferSize     //Buffer size of LDH path
          )
```

Comments

The UdmView_enumLDHPath function enumerates the LDH DLL paths from the registry entries.

Parameters	Description
index	Specify the LDH index to be obtained. To call UdmView_enumLDHPath first, specify 0. From then on, increment the index by 1.
szDataHandler	Specify the leading pointer to the buffer to store the entry name in the registry that defines the LDH.
lNameBufferSize	Specify the number of bytes of the buffer to store the entry name in the registry that defines the LDH.
szLDHPath	Specify the leading pointer to the buffer to store the LDH DLL path.
lPathBufferSize	Specify the number of bytes of the buffer to store the LDH DLL path.

Return Value

The return value is TRUE if the LDH to enumerate exists and the function completes normally. Otherwise, the return value is FALSE. To obtain the error data, use the GetLastError function.

See Also

LDH_getSummary, LDH_showEssenceDialog, LDH_showFilterDialog, LDH_getCSV

4.27 UdmLDH_setData Function

Function

Sets the non-volatile data that can be used to define the log report operation.

```
BOOL WINAPI UdmLDH_setData(      //TRUE|FALSE
        LPCTSTR        szDataHandler,      //Name of LDH (=Name of registry entry
        //        that defines LDH (DLL) path)
        PVOID   pvDataBuffer,      //DataBuffer to set.
        int     lDataSize          //DataSize to set.
        )
```

Comments

The UdmLDH_setData function sets the non-volatile data that can be used to define the log report operation.

Parameters	Description
szDataHandler	Specify the leading pointer to the buffer to store the entry name in the registry that defines the LDH.
pvDataBuffer	Specify the leading pointer to the buffer to store the data to be set.
lDataSize	Specify the number of bytes to be set.

Return Value

The return value is TRUE if the function completes normally. Otherwise, the return value is FALSE. To obtain the error data, use the GetLastError function.

See Also

UdmLDH_getData, UdmReport_writeRecord, UdmReport_writeRecordEx, LDH_Report, LDH_ReportEx, LDH_showFilterDialog

4.28 UdmLDH_getData Function

Function

Obtains the non-volatile data that can be used to define the log report operation.

```
int WINAPI UdmLDH_getData(                //Size of data to obtain
        LPCTSTR         szDataHandler,    //Name of LDH (=Name of registry entry
        //      that defines LDH (DLL) path)
        PVOID   pvDataBuffer,     //Data buffer to obtain data
        PINT    lpDataBufferSize  //Data buffer size (IN)/size of data (OUT)
        )
```

Comments

The UdmLDH_getData function obtains the non-volatile data that can be used to define the log report operation.

Parameters	Description
szDataHandler	Specify the leading pointer to the buffer to store the entry name in the registry that defines the LDH.
pvDataBuffer	Specify the leading pointer to the buffer to store the configuration data.
lpDataBufferSize	Specify the pointer to the buffer to store the number of bytes of obtained configuration data. Before calling this function, specify the number of bytes of the buffer to store the read data. After calling this function, the specified number of bytes of read data is stored.

Return Value

The return value is the number of bytes of the data obtained, if the function completes normally. Otherwise, the return value is UDM_RETURN_CODE_FAILED. To obtain the error data, use the GetLastError function.

See Also

UdmLDH_setData, LDH_showFilterDialog

4.29 Udm_getLastErrorMessage Function

Function

Obtains the error message for an error code.

```
int WINAPI Udm_getLastErrorMessage(          //Size of message to obtain
        DWORD  dwErrorCode,      //Error code obtained by GetLastError()
        LPTSTR pMessage,         //Error message
        int    lBufferSize       //Buffer size of message
        )
```

Comments

The Udm_getLastErrorMessage function obtains the error message from the specified error code.

Parameters	Description
dwErrorCode	Specify the error code set by the UDM API.
pMessage	Specify the leading pointer to the buffer to store the error message.
lBufferSize	Specify the number of bytes of the buffer to store the error message.

Return Value

The return value is the number of bytes of the error message obtained, if the function completes normally. Otherwise, the return value is UDM_RETURN_CODE_FAILED. To obtain the error data, use the GetLastError function.

See Also

Udm*