Chobik SCSI Programming Interface Reference Guide

Revision 4

Contents

1. Introduction

Chobik SCSI Programming Interface product provides suitable interface to applications written in C/C++ language for handling host bus adapters and storage devices on Windows operating system platforms.

The programming interface is based on Windows native SCSI Pass Through interface. It allows user-mode applications to enumerate host bus adapters and storage devices, and execute SCSI commands. The programming interface supports storage devices of all types. Application is responsible for proper handling of the storage device. Multiple independent applications can handle multiple storage devices in parallel.

The programming interface is implemented as kernel mode driver and accompanying user mode API DLL module. The kernel mode driver communicates with host bus adapters and storage devices using device I/O control and other requests. The API module accepts the requests from controlling application and passes them to the kernel mode driver.

The product supports 32-bit and 64-bit Windows operating system platforms.

2. Supported platforms

The following operating system platforms are supported.

- Windows 10
- Windows 8.1
- Windows Server 2012
- Windows 8
- Windows 7
- Windows Server 2008
- Windows Vista
- Windows Server 2003
- Windows XP

Both 32-bit and 64-bit configurations are supported. 64-bit configuration is supported only for AMD64 architecture.

3. Product features

3.1. Host bus adapter enumeration

Application can enumerate host bus adapters connected to the host. The returned list of adapters includes IDE/SATA, SCSI, FC, SAS, iSCSI, virtual, and other adapter types.

3.2. Host bus adapter information

Application can get detailed information for the specified host bus adapter. The information includes such items as maximum data transfer length, display name string, etc.

3.3. Rescanning SCSI bus(es) for host bus adapter

Application can rescan SCSI bus(es) for specified host bus adapter. Rescan operation updates internal list of devices connected to host bus adapter. After SCSI bus is rescanned the application can get updated list of devices.

3.4. SCSI device enumeration

Application can enumerate devices connected to the specified host bus adapter. Returned device information includes such items as adapter number, bus number, target SCSI ID, logical unit number, device type, vendor ID string, product ID string, etc.

3.5. Synchronous SCSI command execution

Application can execute SCSI commands for the specified SCSI device in synchronous mode. SCSI commands can transfer no data, transfer data from device, and transfer data to device.

3.6. Sample client application with source code

Product includes source code of sample console client application. The source code includes functions for all major features. It also includes separate functions for typical SCSI commands that transfer no data, transfer data from device, and transfer data to device.

4. Module structure

The solution includes the following modules:

1. ChobScsiApi.dll is user mode API module. The API module accepts the requests from controlling application and passes them to the kernel mode driver. The API module is copied to System32 directory during installation. For 64-bit platform the 32-bit version of API module is also copied to SysWOW64 directory.
2. chobscsi.sys is kernel mode driver. The driver communicates with HBAs and SCSI devices using device I/O control and other requests. The driver module is copied to System32/drivers directory during installation and is registered as kernel mode driver service.
3. Include file ChobScsiApi.h. The header file contains definitions of functions and structures.
4. Static libraries ChobScsiApi.lib. The library files (32-bit and 64-bit) allow to statically link to the functions exported from ChobScsiApi.dll module.

5. API functions

5.1. ChobScsiApiEnumAdapters function

The function enumerates available host bus adapters (HBAs) and returns adapter identifiers.

Syntax

```
int ChobScsiApiEnumAdapters (
    unsigned long * pulIdentifiers,
    int nMaxCount);
```

Parameters

pulIdentifiers

Pointer to the array of unsigned long integer values. Adapter identifiers will be stored into the array. If NULL pointer is passed, no adapter identifiers are returned. If NULL pointer is passed, the nMaxCount argument must also have zero value.

nMaxCount

Maximum number of elements in the array pointed to by pulIdentifiers argument. If argument has zero value, the NULL pointer must also be passed as pulIdentifiers argument.

Return value

In case of success the function returns the number of adapter identifiers stored in the array pointed to by pulIdentifiers argument. If no buffer is specified for adapter identifiers, the function returns total number of available adapters in the system.

In case of failure the function returns -1 value. To get extended error information, call GetLastError.

5.2. ChobScsiApiEnumDevices function

The function enumerates available devices for specified adapter and returns device information.

Syntax

```
int ChobScsiApiEnumDevices (

    unsigned long ulAdapter,

    struct CHOBSCSI_API_DEVICE_INFO * pInfo,

    int nMaxCount);
```

Parameters

ulAdapter

Adapter identifier.

pInfo

Pointer to the array of device information structures. Device information will be stored into the array. If NULL pointer is passed, no device information is returned. If NULL pointer is passed, the nMaxCount argument must also have zero value.

nMaxCount

Maximum number of elements in the array pointed to by pInfo argument. If argument has zero value, the NULL pointer must also be passed as pInfo argument.

Return value

In case of success the function returns the number of device information structures stored in the array pointed to by pInfo argument. If no buffer is specified for device information, the function returns total number of available devices for specified adapter.

In case of failure the function returns -1 value. To get extended error information, call GetLastError.

5.3. ChobScsiApiExecuteCommand function

The function synchronously executes SCSI command for the specified device.

Syntax

```
int ChobScsiApiExecuteCommand (
    struct CHOBSCSI_API_COMMAND_DESCRIPTOR * pCommand);
```

Parameters

pCommand

Pointer to SCSI command descriptor. On input the command descriptor contains command parameters like device address, CDB, buffers and transfer length, timeout, etc. On output the command descriptor contains various command status values like adapter status, SCSI status, sense data, data transfer counters, etc.

Return value

In case of success the function returns non-zero value.

In case of failure the function returns zero value. To get extended error information, call GetLastError.

5.4. ChobScsiApiGetAdapterInfo function

The function returns adapter information for the specified adapter.

Syntax

```
int ChobScsiApiGetAdapterInfo (
    unsigned long ulAdapter,
    struct CHOBSCSI_API_ADAPTER_INFO * pInfo);
```

Parameters

ulAdapter

Adapter identifier.

pInfo

Pointer to the adapter information data structure. On successful return the information will be stored in the data structure.

Return value

In case of success the function returns non-zero value.

In case of failure the function returns zero value. To get extended error information, call GetLastError.

5.5. ChobScsiApiRescanBus function

The function rescans bus(es) for the specified adapter.

Syntax

```
int ChobScsiApiRescanBus (
    unsigned long ulAdapter);
```

Parameters

ulAdapter

Adapter identifier.

Return value

In case of success the function returns non-zero value.

In case of failure the function returns zero value. To get extended error information, call GetLastError.

6. API structures

6.1. CHOBSCSI_API_ADAPTER_INFO structure

The structure contains adapter information.

Syntax

```
struct CHOBSCSI_API_ADAPTER_INFO {
    unsigned long ulMaximumTransferLength;
    unsigned long ulNumberOfBuses;
    unsigned long ulInitiatorIdentifiers [CHOBSCSI_API_MAX_BUSES];
    wchar_t wszDiplayName [CHOBSCSI_API_MAX_NAME_LENGTH];
    int nRequestType;
    int nAddressType;
};
```

Members

ulMaximumTransferLength

Maximum amount of data in bytes that can be transferred by single command.

ulNumberOfBuses

The number of storage buses reported by adapter driver.

ulInitiatorIdentifiers

Initiator SCSI IDs for first 8 buses.

wszDisplayName

Display name for user interface.

nRequestType

Type of supported SCSI request block. The following values are defined for this field:

- 0 - Legacy SCSI Request Block
- 1 - Storage Request Block

nAddressType

Type of supported SCSI device address. The following values are defined for this field:

- 0 - 8-bit bus, target, and LUN addressing

6.2. CHOBSCSI_API_COMMAND_DESCRIPTOR structure

The structure contains command parameters and status values.

Syntax

```
struct CHOBSCSI_API_COMMAND_DESCRIPTOR {
    unsigned long ulAdapter;
    unsigned long ulBus;
    unsigned long ulTarget;
    unsigned long ulLun;
    char szDeviceObjectName [CHOBSCSI_API_MAX_NAME_LENGTH];
    int nAdapterStatus;
    int nScsiStatus;
    int nSenseLength;
    unsigned char ucSenseBuffer [CHOBSCSI_API_MAX_SENSE_LENGTH];
    int nInputTransferLength;
    void * pvInputBuffer;
    int nOutputTransferLength;
    void * pvOutputData;
    int nTimeout;
    int nCdbLength;
    unsigned char ucCdb [CHOBSCSI_API_MAX_CDB_LENGTH];
};
```

Members

ulAdapter

On input specifies adapter identifier. If device object name is not specified this member will be used for addressing the device during command execution. If valid device object name is specified this member is ignored.

ulBus

On input specifies bus number. If device object name is not specified this member will be used for addressing the device during command execution. If valid device object name is specified this member is ignored.

ulTarget

On input specifies target SCSI identifier. If device object name is not specified this member will be used for addressing the device during command execution. If valid device object name is specified this member is ignored.

ulLun

On input specifies logical unit number. If device object name is not specified this member will be used for addressing the device during command execution. If valid device object name is specified this member is ignored.

szDeviceObjectName

On input contains device object name string. If device object name is not available, empty string should be passed. If device object name is available it should be specified.

nAdapterStatus

On output contains adapter status value. CHOBSCSI_API_ADAPTER_STATUS_XXX values are defined in ChobScsiApi.h header file. If this member has CHOBSCSI_API_ADAPTER_STATUS_GOOD value then nScsiStatus member is valid.

nScsiStatus

On output contains SCSI status value. CHOBSCSI_API_SCSI_STATUS_XXX values are defined in ChobScsiApi.h header file. If this member is valid and has CHOBSCSI_API_SCSI_STATUS_CHECK_CONDITION value then sense data are valid.

nSenseLength

On output specifies the size in bytes of sense data in the buffer.

ucSenseBuffer

On output contains SCSI sense data if error occurred.

nInputTransferLength

On input specifies the size in bytes of the buffer for IN data pointed to by pucInputBuffer member. If command does not transfer IN data from the device this member must have zero value. On output this member specifies the size in bytes of IN data that have been transferred from the device.

pvInputBuffer

Pointer to the buffer for IN data. If nInputTransferLength has zero value on input, NULL pointer can be specified.

nOutputTransferLength

On input specifies the size in bytes of the OUT data in the buffer pointed to by pucOutputBuffer member. If command does not transfer OUT data to the device this member must have zero value. On output this member specifies the size in bytes of OUT data that have been transferred to the device.

pvOutputData

Pointer to the buffer with OUT data. If nOutputTransferLength has zero value on input, NULL pointer can be specified.

nTimeout

On input specifies the command timeout in seconds.

nCdbLength

On input specifies the size in bytes of CDB in ucCdb member.

ucCdb

On input contains CDB.

6.3. CHOBSCSI_API_DEVICE_INFO structure

The structure contains device information.

Syntax

```
struct CHOBSCSI_API_DEVICE_INFO {
     unsigned long ulAdapter;
     unsigned long ulBus;
     unsigned long ulTarget;
     unsigned long ulLun;
     int nType;
     char szDeviceObjectName [CHOBSCSI_API_MAX_NAME_LENGTH];
     char szVendor [9];
     char szProduct [17];
     char szRevisionLevel [5];
};
```

Members

ulAdapter

Adapter identifier.

ulBus

Bus number.

ulTarget

Target SCSI identifier.

ulLun

Logical unit number.

nType

SCSI device type as specified by the SCSI standard. The device type value is taken from Standard Inquiry Data.

szDeviceObjectName

Device object name string. For magnetic and optical disk device the device name has the form PhysicalDriveN, where N is the device number. For tape drive device the device name has the form TapeN. For CD/DVD devices the device name has the form CdRomN. For media changer devices the device name has the form ChangerN. If named device object is not available the device name string is empty.

szVendor

Vendor ID string from Standard Inquiry Data.

szProduct

Product ID string from Standard Inquiry Data.

szRevisionLevel

Device firmware revision level string from Standard Inquiry Data.

7. API programming guidelines

API function prototypes and structure definitions are located in the ChobScsiApi.h header file. This file shall be included into C/C++ written application source code. The header file is located in the $(InstallDir)\Include directory.

Static library ChobScsiApi.lib should be linked to controlling application executable file. 32-bit version of the library is located in the $(InstallDir)\Lib directory. 64-bit version of the library is located in the $(InstallDir)\Lib64 directory.

It is recommended to use Microsoft Visual Studio 2010 or later for building applications that use Chobik SCSI Programming Interface.

8. Sample client application

The $(InstallDir)\Sample directory contains source code for sample client application. The source code is the solution for Microsoft Visual Studio 2010. The following functions are implemented for typical operations:

- **EnumAdapters** - adapter enumeration
- **AdapterInformation** - getting adapter information
- **EnumDevices** - device enumeration
- **RescanBus** - SCSI bus rescanning
- **TestUnitReady** - Test Unit Ready (00h) SCSI command
- **Inquiry** - Inquiry (12h) SCSI command
- **WriteBuffer** - Write Buffer (3Bh) SCSI command