SCSI Pass Through JNI Helper

Technical Description

Table of contents

1. Overview

The SCSI Pass Through JNI (SPT JNI) software package provides the programmatic interface for Java applications on the Windows and Linux operating system platforms for handling SCSI adapters and devices. For Windows operating system platform the native SCSI Pass Through (SPT) programmatic interface is used. For Linux operating system platform the native SCSI Generic (SG) programmatic interface is used.

The package allows Java application to enumerate SCSI host bus adapters and SCSI devices, and to execute SCSI commands for the SCSI devices. Programmatic interface of the package is simple and introduces no limitation on the CDB and data structure. The maximum size of data is limited by the maximum transfer length parameter of the SCSI host bus adapter and depends on the adapter model and type. For Windows 8 and later operating system platforms the large CDB (up to 256 bytes) and bidirectional data transfer are supported. The maximum sense data length is 255 bytes. The package gives the Java application all the advantages of handling the SCSI devices at relatively low system level. The package does not depend on the type of host bus adapter and handles all types of adapters (SCSI, Fibre Channel, iSCSI, etc) in the same

unified way. The package is relatively compact and requires minimum of disk space and memory.

The Java related source code of the package was developed using JDK version 1.8.

2. Supported operating system platforms

The following Windows operating system platforms are supported:

- Windows XP 32-bit and 64-bit
- Windows Server 2003 32-bit and 64-bit
- Windows Vista 32-bit and 64-bit
- Windows Server 2008 32-bit and 64-bit
- Windows 7 32-bit and 64-bit
- Windows 8 32-bit and 64-bit
- Windows Server 2012 64-bit
- Windows 8.1 32-bit and 64-bit
- Linux 32-bit and 64-bit

NOTE: 64-bit operating system platforms are supported only for AMD64 (x64) processor architecture.

3. System requirements

Hardware requirements:

- 800 MHZ or faster Inter Pentium TM or equivalent CPU.
- 256 MB RAM. The optimal size of RAM depends on the installed Windows operating system.
- 10GB hard disk. The optimal size of hard disk depends on the installed Windows operating system.
- 15" or greater SVGA display.
- Standard keyboard and mouse.
- One or more host bus adapters.

Software requirements:

- Supported Windows or Linux operating system with the necessary service pack installed.
- Java run time environment (JRE) version 1.8 or higher is to be installed on the host computer.
- For Windows operating system platforms before Windows Vista the java process or the process that runs the Java VM should be started with local adminstrator user privileges. For Windows Vista and later operating system platforms the java process or the process that runs the Java VM should be started with highest user privileges (Run as Admin) in order to operate the host bus adapter and device objects.

NOTE: It is recommended to install all system patches from Windows Update and Java technology sites.

4. Package features

- Host bus adapter (HBA) enumeration. The application gets the array with identifiers of installed HBAs. Then the application selects the HBA by the identifier and gets the HBA capabilities and enumerates the devices that are connected to HBA.
- SCSI device enumeration. The package offers two methods of device enumeration. The first method enumerates all the devices for selected HBA and returns the array of device information units. The second method enumerates all the devices of selected type (disk, tape drive, CD/DVD, and media changer) and returns the array of device information units. The device information unit includes the device parameters (device object name, HBA number, bus number, target ID, and LUN) and identification data (vendor ID, product ID, and revision level).
- Rescanning the SCSI bus for selected HBA. Application can rescan SCSI bus(es) for selected HBA. During the rescan operation the SCSI bus is reset (not supported for some newest HBA and bus types) and the operating system updates the internal list of connected devices. Then the application can enumerate the new set of devices.
- SCSI command execution. The application can execute SCSI commands using legacy and extended command requests. For legacy command request the CDB size is limited by 16 bytes and bidirectional transfer is not supported. For Windows 8 and later operating system platforms the application can execute extended command requests that support large CDBs and bidirectional data transfer.
- Sense data support. The package supports sense data of up to 255 bytes size. The fixed format sense data and descriptor format sense data are parsed transparently and the most important fields (sense key, additional sense code, and additional sense code qualifier) are returned to the application. The raw sense data are also available to the application.
- Large CDB support. For Windows 8 and later operating system platforms the extended command requests with CDB of up to 256 bytes size are supported. The HBA reports the support of extended command requests in its capabilities and the application should analyze them before making extended command request with large CDB.
- Bidirectional data support. For Windows 8 and later operating system platforms the extended command requests with both input and output data are supported. The HBA reports the support of extended command requests in its capabilities and the application should analyze them before making extended command request with bidirectional data transfer.
- 32-bit and 64-bit operating system support. The package supports both x86 and x64 Windows and Linux operating system platforms and JRE versions. The package capacity should match the JRE capacity.
- Sample Java application that demostrates the use of classes and methods for all operations.

5. Package classes

5.1. ScsiPassThrough class

The scsi.ScsiPassThrough class is the main class in the SPT JNI package. It provides all necessary methods to work with the SPT interface.

The class declaration is located in the scsi\ScsiPassThrough.java source file.

The class implements the following methods.

Method	Description
enumAdapters	Returns the list of identifiers of installed SCSI adapters.
enumDevices	Returns the list of SCSI devices for selected SCSI adapter.

enumDevicesType	Returns the list of SCSI devices for selected SCSI device type.
executeCommand	Executes SCSI command for selected SCSI device using legacy SCSI request block.
executeCommandEx	Executes SCSI command for selected SCSI device using storage request block.
getAdapterCaps	Returns the parameters of selected SCSI adapter.
getAdapterCount	Returns the total number of SCSI adapters.
rescanBus	Rescans the SCSI bus(es) for selected SCSI adapter.

Other definitions of class.

Definition	Description
public static final byte SCSIOP_XXX	SCSI command code values as defined by the SCSI standard. These definitions include most but not all commands. Refer to the SCSI standard for more information.

enumAdapters method

```
public int [] enumAdapters ()
```

The enumAdapters method returns the array of zero-based SCSI adapter numbers. These values should be passed to the other methods that require adapter number. Method does not have arguments.

enumDevices method

```
public SptDeviceInfo [] enumDevices (int nAdapter)
```

The enumDevices method enumerates the SCSI devices that are connected to the SCSI adapter and returns the array of instances of SptDeviceInfo class. Each instance of SptDeviceInfo class represents single SCSI device. If no SCSI devices are connected to the adapter, the array is empty. If wrong adapter number is passed to the method or the internal error occurred, the returned value is null. Method accepts the following arguments.

Argument	Description
int nAdapter	Zero-based SCSI adapter number. If -1 value is passed the returned device list includes all devices connected to all adapters.

enumDevicesType method

```
public SptDeviceInfo [] enumDevicesType (int nType)
```

The enumDevicesType method enumerates the SCSI devices of the specified type and returns the array of instances of SptDeviceInfo class. Each instance of SptDeviceInfo class represents single SCSI device. If no SCSI devices of the specified type were detected, the array is empty. If wrong or unsupported device type value is passed to the method or the internal error occurred, the returned value is null. Method accepts the following arguments.

Argument	Description
int nType	SCSI device type as specified by the SCSI standard. If -1 value is passed the returned device list includes devices of all types.

The following device types are supported by the method.

Type	Description
0	Direct access block device (e.g., magnetic disk).
1	Sequential-access device (e.g., magnetic tape).
5	CD/DVD device.
7	Optical memory device (e.g., some optical disks).
8	Media changer device (e.g., jukeboxes).

executeCommand method

```
public SptStatus executeCommand (String sDeviceName, int nAdapterNumber, int nBusNumber, int nTargetNumber, int nLogicalUnitNumber, byte[] chCdb, boolean bDataIn, byte[] chData, int nRequestTimeout)
```

The executeCommand method passes the SCSI command to the SCSI device using legacy SCSI request block and returns the command status in the instance of SptStatus class. If wrong device parameters (adapter number, bus number, target SCSI ID, LUN) are passed to the method or the internal error occurred, the returned value is null. Method accepts the following arguments.

Argument	Description
String sDeviceName	Device object name. The device object name string should be taken from device information returned during device enumeration. If device name string is valid the other device parameters (adapter number, bus number, target SCSI ID, and logical unit number) are ignored and command request is passed to target device object. If device name string is empty the device parameters are used and command request is passed to SCSI port device object.
int nAdapter	Zero-based SCSI adapter number.
int nBusNumber	Zero-based SCSI bus number. The value of this argument should be taken from the device information data.
int nTargetNumber	Target SCSI ID. The value of this argument should be taken from the device information data.
int nLogicalUnitNumber	Logical unit number. The value of this argument should be taken from the device information data.
byte[] chCdb	Binary data of command descriptor block (CDB). Refer to the SCSI standard or to the SCSI device reference for the structure of CDB. The CDB length is limited by 16 bytes.
boolean bDataIn	Flag of inbound (from the device to the initiator) data. If this flag is false, the data are outbound (from the initiator to the device).
byte[] chData	Buffer for input (from the device) or output (to the device) data. If command does not pass data, this argument should have null value. For inbound data the initial size of the buffer will be used as allocation length parameter. Actually the device may return less data than requested.
int nRequestTimeout	Command timeout in seconds.

executeCommandEx method

```
public SptStatusEx executeCommandEx (String sDeviceName, int nAdapterNumber, int nBusNumber, int nTargetNumber, int nLogicalUnitNumber, byte[] chCdb, byte[] chInputData, byte[] chOutputData, int nRequestTimeout)
```

The executeCommandEx method passes the SCSI command to the SCSI device using storage request block and returns the command status in the instance of SptStatusEx class. If wrong device parameters (adapter number, bus number, target SCSI ID, LUN) are passed to the method or the internal error occurred, the returned value is null. The storage request block can be passed to the device only when adapter capabilities data have the value 1 in the nSrbType field. Method accepts the following arguments.

Argument	Description
String sDeviceName	Device object name. The device object name string should be taken from device information returned during device enumeration. If device name string is valid the other device parameters (adapter number, bus number, target SCSI ID, and logical unit number) are ignored and command request is passed to target device object. If device name string is empty the device parameters are used and command request

	is passed to SCSI port device object.
int nAdapterNumber	Zero-based SCSI adapter number.
int nBusNumber	Zero-based SCSI bus number. The value of this argument should be taken from the device information data.
int nTargetNumber	Target SCSI ID. The value of this argument should be taken from the device information data.
int nLogicalUnitNumber	Logical unit number. The value of this argument should be taken from the device information data.
byte[] chCdb	Binary data of command descriptor block (CDB). Refer to the SCSI standard or to the SCSI device reference for the structure of CDB. The CDB length is limited by 256 bytes.
byte[] chInputData	Buffer for input (from the device) data. If command does not pass input data, this argument should have null value. Actual number of received data is returned in the status data.
byte[] chOutputData	Buffer for output (to the device) data. If command does not pass output data, this argument should have null value. Actual number of sent data is returned in the status data.
int nRequestTimeout	Command timeout in seconds.

getAdapterCaps method

```
public SptAdapterCaps getAdapterCaps (int nAdapter)
```

The getAdapterStatus method returns the instance of the SptAdapterCaps class that contains the parameters of the SCSI adapter. If wrong adapter number is passed to the method or the internal error occurred, the returned value is null. Method accepts the following arguments.

Argument	Description
int nAdapter	Zero-based SCSI adapter number.

getAdapterCount method

```
public int getAdapterCount ()
```

The getAdapterCount method returns total number of SCSI adapters that are installed on the system. In case of internal error the -1 value is returned. Method does not have arguments.

rescanBus method

```
public boolean rescanBus (int nAdapter)
```

The rescanBus method rescans SCSI bus(es) of the SCSI adapter. If operation was performed successfully the returned value is true. If wrong adapter number was passed to the method or the internal error occurred, the returned value is false. Method accepts the following arguments.

Argument	Description
int nAdapter	Zero-based SCSI adapter number.

5.2. SptAdapterCaps class

The scsi.ScsiAdapterCaps class is the container class that includes data fields for SCSI adapter parameters. The instance of this class is returned by the getAdapterCaps method of scsi.ScsiPassThrough class.

The class declaration is located in the scsi\SptAdapterCaps.java source file.

The class includes the following fields.

Field	Description
public int nMaximumTransferLength	Maximum amount of data in bytes that can be transferred by single command.
public int nInitiatorId	Initiator SCSI ID for bus 0.

public String sDisplayName	Display name for user interface.
public int nSrbType	Type of supported SCSI request block. The following values are defined for this field: • 0 Legacy SCSI Request Block • 1 Storage Request Block
public int nAddressType	Type of supported SCSI device address. The following values are defined for this field: • 0 8-bit bus, target, and LUN addressing

If adapter supports legacy SCSI request block the CDB length is limited by 16 bytes and bidirectional data transfer is not supported. If adapter supports storage request block the CDB length is limited by 256 bytes and bidirectional data transfer is supported. Refer to the description of executeCommand(Ex) methods of ScsiPassThrough class for more information.

5.3. SptDeviceInfo class

The scsi.SptDeviceInfo class is the container class that includes data fields for SCSI device parameters. The array of instances of this class is returned by the enumDevices and enumDevicesType methods of scsi.ScsiPassThrough class.

The class declaration is located in the scsi\SptDeviceInfo.java source file.

The class includes the following fields.

Field	Description
public int nAdapter	Zero-based SCSI adapter number.
public int nBus	Zero-based SCSI bus number.
public int nTargetId	Target SCSI ID.
public int nLun	Logical unit number (LUN).
public int nType	SCSI device type as specified by the SCSI standard.
public int nDevice	Zero-based device number. If named device object is not available the device number is -1.
public String sName	Device name string. See the comment below about the device name structure. If named device object is not available the device name string is empty.
public String sVendorId	Vendor ID string.
public String sProductId	Product ID string.
public String sRevisionLevel	Device firmware revision level 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.

5.4. SptStatus class

The scsi.SptStatus class is the container for SCSI command status data. The instance of this class is returned by the executeCommand method of the scsi.ScsiPassThrough class.

The class declaration is located in the scsi\SptStatus.java source file.

The class includes the following fields.

Field	Description
public byte byteAdapterStatus	Adapter status. If this field has SPT_ADAPTER_STATUS_GOOD value, the SCSI status data field is valid.

public byte byteScsiStatus	SCSI status. If this field has SCSI_STATUS_CHECK_CONDITION value, the sense data fields are valid.
public byte byteSenseKey	Sense key field of sense data. This field is valid only when SCSI status has SCSI_STATUS_CHECK_CONDITION value.
public byte byteAdditionalSenseCode	ASC field of sense data. This field is valid only when SCSI status has SCSI_STATUS_CHECK_CONDITION value.
public byte byteAdditionalSenseCodeQualifier	ASCQ field of sense data. This field is valid only when SCSI status has SCSI_STATUS_CHECK_CONDITION value.
public byte byteSenseData	Raw sense data. This field is valid only when SCSI status has SCSI_STATUS_CHECK_CONDITION value.
public int nDataCount	The size of received data for data IN commands. This field is always valid.
public int nExecutionTime	Total time of command execution in milliseconds. This field is always valid.

Other definitions of class.

Definition	Description
public static final byte SPT_ADAPTER_STATUS_XXX	Adapter status values.
public static final byte SCSI_STATUS_XXX	SCSI status values. These values are defined by the SCSI standard.
public static final byte SCSI_SENSE_XXX	Sense key values. These values are defined by the SCSI standard.

5.5. SptStatusEx class

The scsi.SptStatusEx class is the container for SCSI command status data. The instance of this class is returned by the executeCommandEx method of the scsi.ScsiPassThrough class.

The class declaration is located in the scsi\SptStatusEx.java source file.

The class includes the following fields.

Field	Description
public byte byteAdapterStatus	Adapter status. If this field has SPT_ADAPTER_STATUS_GOOD value, the SCSI status data field is valid.
public byte byteScsiStatus	SCSI status. If this field has SCSI_STATUS_CHECK_CONDITION value, the sense data fields are valid.
public byte byteSenseKey	Sense key field of sense data. This field is valid only when SCSI status has SCSI_STATUS_CHECK_CONDITION value.
public byte byteAdditionalSenseCode	ASC field of sense data. This field is valid only when SCSI status has SCSI_STATUS_CHECK_CONDITION value.
public byte	ASCQ field of sense data. This field is valid only when

byteAdditionalSenseCodeQualifier	SCSI status has SCSI_STATUS_CHECK_CONDITION value.
public byte byteSenseData	Raw sense data. This field is valid only when SCSI status has SCSI_STATUS_CHECK_CONDITION value.
public int nInputDataCount	The size of received IN data in bytes. This field is always valid.
public int nOutputDataCount	The size of sent OUT data in bytes. This field is always valid.
public int nExecutionTime	Total time of command execution in milliseconds. This field is always valid.

6. Code samples

The sample code that illustrates the use of classes is included into the package help system for the following operations.

- Host bus adapter enumeration.
- Device enumeration.
- Rescanning SCSI bus.
- SCSI command with no data transfer (Test Unit Ready (00h) command).
- SCSI command with input data transfer (Inquiry (12h) command for Unit Serial Number (80h) page).
- SCSI command with output data transfer (Write Buffer (3Bh) for writing the data to echo buffer).

Refer to the package help system for Java source code fragments.

The package also includes simple sample Java application that demostrates the use of classes and methods for all operations. The Test.java source file is located in the archive with source code in the Packages subfolder.

7. Distribution

The package is distributed in binary and source code form.

The binary form includes executable (DLL, CLASS, JAR) modules as well as compiled MSI-files for 32-bit and 64-bit configurations. There are separate packages for Windows and some Linux operating system platforms.

The source code form includes the entire solution that can be built in MS Visual Studio environment or on Linux operating system platform.

The following tools are necessary in order to successfully build the package on Windows.

- MS Visual Studio 2010
- Windows Driver Kit version 7.1
- Java Development Kit version 1.8

The following tools are necessary in order to successfully build the package on Linux.

- G++ compiler
- Java Development Kit version 1.8

The Doc subfolder in the project tree contains the build instruction document that provides detailed description of all build steps.

8. Modules

The software package includes the following modules:

scsi\ScsiPassThrough.java

Java source file that contains declaration of the scsi.ScsiPassThrough class.

scsi\SptAdapterCaps.java

Java source file that contains declaration of the scsi.SptAdapterCaps class.

scsi\SptDeviceInfo.java

Java source file that contains declaration of the scsi.SptDeviceInfo class.

scsi\SptStatus.java

Java source file that contains declaration of the scsi.SptStatus class.

scsi\SptStatusEx.java

Java source file that contains declaration of the scsi.SptStatusEx class.

spt.jar

Compiled Java source code that is included into the JAR-file. The JAR-file is made up using the JDK version 1.7.0_75.

SptJni.dll

JNI interface DLL library. In order to make this module lodable for any application, it should be located in the folder that is included into the system environment variable "PATH". The alternative way to load the module is to specify the path to folder using **java.library.path** definition. This module is available for Windows operating system platform in 32-bit and 64-bit forms.

libSptJni.so

JNI interface shared object library. In order to make this module lodable for any application, it should be located in the folder that is included into the system environment variable "PATH". The alternative way to load the module is to specify the path to folder using **java.library.path** definition. This module is available for Linux operating system platform in 32-bit and 64-bit forms.

SptJni.chm

Hyper text help file.

All modules are fully redistributable.

9. Running the package

9.1. Windows

On Windows operating system platform the Java process should be run with elevated user privileges (Run as Admin).

9.2. Linux

On Linux operating system platform the necessary privilege level depends on the type of device being accessed.

For CD/DVD/BD drive devices the normal user privilege level is usually sufficient.

For disk devices the highest super user (or root) privilege level is necessary.

For other devices the necessary privilege level may depend on Linux distribution and device type.

If storage device management application needs to access devices of all types, the Java process needs to run at the highest super user (or root) privilege level.