August 15, 2006

Mr. Robert Snively  
Chair, INCITS Technical Committee T11  
iVivity, Inc.  
5555 Oakbrook Parkway  
Atlanta, GA 30093

Re: Notice of Patent Issuance

Dear Mr. Robert Snively,

iVivity, Inc. has recently been granted U.S. Patent No. 6,765,871 titled “Application Interface Access to Hardware Services for Storage Management Applications.” A copy is attached for your information. The patent generally relates to subject matter of the T11 Fabric Application Interface Standard.

If the proposed standards are adopted and any claims of the patent are necessary for practicing these standards, iVivity will offer the patent on reasonable, non-discriminatory terms, with reciprocity, to the extent necessary to implement the standards.

Please contact the Office of General Counsel at iVivity, Inc. 678-990-1550 concerning any licensing matters.

Sincerely,

[Signature]
Brian Anderson  
General Counsel & Secretary
A method and device for using a set of APIs are provided. Some of the functions which used to be performed by software are now accelerated through hardware.

6 Claims, 7 Drawing Sheets
Fig. 1 (Prior Art)
Fig. 2

control path 1

software

storage data management silicon 3

VAAPI 4
Fig. 3 (Prior Art)

- Management application
- Virtualization repository
- Storage Virtualization Engine
- Data path
- Control path
- Control flow
- Data
Fig. 4

Management application

control path 22

data path

virtulization repository 24

VAAPI 12

Hardware Acceleration interface 18

Accelerated data path 16

22

24
Fig. 6

1. Identify virtual disk
2. Identify I/O execution plan
3. Check for exceptions
   - Yes: Control path software
   - No: Check for hardware
     - Yes: Execute I/O plan using hardware acceleration
     - No: Analyze I/O plan
Fig. 7

iDisX Storage Virtualization Engine
I/O Processing

Vendor Proprietary Virtualization Information Repository

Repository Populator & Synchronizer

CIM/WBEM-APIs
(RPS-APIs)

iVivity's CIM-Based Virtualization Information Repository

RI-APIs
(Repository Interface)
(CIM/WBEM-API-Based)

Control Path (CP) in Software

AP-APIs

I/O-APIs

Mapping/Conversion

Accelerated Path (AP) in Hardware

Complex/Faulted I/O Plans From AP-H/W

Yes

execute the I/O

plan using

hardware acceleration

Out Q

In Q

CP Analyzed/Generated I/O Plans

RI-APIs
(Alternative)
(Based on Vendor APIs)
APPLICATION PROGRAM INTERFACE-ACCESS TO HARDWARE SERVICES FOR STORAGE MANAGEMENT APPLICATIONS

This application claims an invention which was disclosed in Provisional Application No. 60/380,160, filed May 6, 2002, entitled "APPLICATION PROGRAM INTERFACE-ACCESS TO HARDWARE SERVICES FOR STORAGE MANAGEMENT APPLICATIONS". The benefit under 35 U.S.C. §119(e) of the United States provisional application is fully claimed, and the aforementioned application is hereby incorporated herein by reference.

BACKGROUND OF THE INVENTION

1. Field of the Invention
The present invention generally relates to an application program interface (API), more specifically, the present invention relates to an API having access to hardware services for storage management applications. Yet more specifically, the present invention relates to a Virtualization Application Programming Interface (VAAPI).

2. Description of the Related Art
Application program interface (API), also known as application programming interface (API), is known in the art. API can be considered as a set of specific methods prescribed by a computer operating system or by an application program, which a programmer who is writing an application program can make requests of the operating system or another application.

The explosive growth of storage networks is being driven by the collaboration of business computing and the need for business continuity. The storage data management silicon model makes the assumption that the next logical step in managing storage networks is to move some of the storage management functionality into the storage network with the implementation located in switches, routers, appliances, NAS and SAN attached arrays. This model envisions storage virtualization application implemented onto storage network nodes using specialized storage data management silicon to ensure that the node does not become a severe performance bottleneck to the network traffic flowing through it.

To implement storage virtualization in the network, the storage virtualization application is effectively split into two function components: the control path and the data path, as shown in FIG. 1. The control path is responsible for all of the control functions of virtualization; including setting up the configuration, changing the configuration, network and availability management, fault tolerance, and error recovery. The data path component is responsible for moving the I/O through the virtualization application.

The performance characteristics of the storage virtualization engine in this paradigm depends on the amount of the data path that is implemented in hardware. A silicon-assisted solution can significantly reduce latencies over software solutions and increase IOP performance many times.

Therefore, it is desirable to have specialized APIs residing in the datapath. Further, it is desirable to have a storage network I/O handling framework and a set of APIs for better performance.

SUMMARY OF THE INVENTION
A storage network I/O handling system including a set of APIs are provided for enabling the separation of Control path (configuration and complex exception handling) and data path (storage I/O execution and relatively simpler exception handling) related computing.

A storage network I/O handling system including a set of APIs is provided, in which the data path processing is kept relatively simple in comparison to control path processing, and the system is being accelerated with specialized hardware (HW) for achieving higher performance.

A storage network I/O handling system including a set of specialized APIs is provided for defining abstracted interfaces to the configuration information repository from the Storage Management applications in the control path.

A storage network I/O handling system including a set of APIs is provided for defining a set of APIs for device configuration, configuration loading, exception reporting, and access to HW accelerated I/O processing pipeline such as a storage management processor.

A storage network I/O handling system including a set of APIs is provided for optimizing storage network environments with emphasis on performance and ease of development.

A storage network I/O handling system including a set of APIs is provided for facilitating implementations with 10x or greater performance scalability characteristics as compared to known processor implementations.

A storage network I/O handling system including a set of APIs is provided with the system further having an extensible and partition-able framework that allows easy integration with a vendor's unique content and APIs.

A storage network I/O handling system including a set of APIs is provided for leveraging the industry standardization efforts as much as possible. For example, CIM and WBEM are heavily leveraged in the repository component of the present application.

A storage network I/O handling system including a set of APIs is provided for easy adaptation for implementations other than only CIM/WBEM, including SNMP and proprietary interfaces.

A storage network I/O handling system including a set of APIs is provided for a wide adoptability, or support to other vendor storage systems.

Accordingly, a storage network I/O handling system including a set of APIs is provided.

Accordingly, a method is provided. The method includes: providing a virtual disk for an I/O request; providing an I/O execution plan based upon the I/O request; providing an I/O plan executor in hardware; and using the I/O plan executor to execute the I/O plan, thereby at least some storage related function are performed by the I/O plan executor in hardware.

Accordingly, a storage virtualization engine coupled to a control path and a data path is provided. The engine comprising: a software sub-engine having the control path and data path; and a virtualization repository; a hardware sub-engine having an accelerated data path; an VAAPI coupling the software sub-engine with the hardware sub-engine; a management application coupled to the software sub-engine, wherein command therefrom are processed by the control path, thereby some function are performed by hardware through the VAAPI and data are accelerated through the accelerated data path.

Accordingly, a storage management system having a control path and a data path is provided. The system comprising: a storage virtualization engine, the engine includes: a software sub-engine having the control path and data path; and a virtualization repository; a hardware sub-engine having an accelerated data path; an VAAPI coupling the software sub-engine with the hardware sub-engine; a management application coupled to the software sub-engine,
The present invention provides a Virtualization Acceleration Application Programming Interface (VAAPI) which is interposed between a hardware layer and a software layer. For detailed description of VAAPI, please refer to infra. The present invention intends to create or modify existing storage virtualization applications to take advantage of the fast path acceleration provided by storage data management silicon, which is included in a commonly assigned application, entitled STORAGE MANAGEMENT PROCESSOR, provisional application No. 60/427,593 filed on Nov. 19, 2002. Further, VAAPI is a strategy to bring concurrency within the storage virtualization industry for the use of a common platform. By providing hardware-assisted data movement and related functionality through VAAPI, virtualization application vendors can boost their performance while positioning their technology on an open platform.

Referring to FIG. 2, VAAPI 4 is a storage network I/O handling framework and a set of APIs for the following purposes. The purposes includes: enabling separation of a control path 1 (configuration and complex exception handling) and data path 2 (storage I/O execution and relatively simpler exception handling) related computing. The data path 2 processing is kept relatively simple in comparison to control path 1 processing and data path 2 is being accelerated with specialized HW for achieving higher performance. VAAPI 4 further defines abstracted interfaces to the configuration information repository from the Storage Management applications in the control path 1; and defines a set of APIs for device configuration, configuration loading, exception reporting and access to HW accelerated I/O processing pipeline in a storage management processor 3 (silicon).

VAAPI 4 resides in the datapath 2 and is a mechanism for implementing the steady state portion of I/O in hardware for maximum performance. A storage virtualization map (not shown) is created in the control portion 1 of the storage virtualization and is then pushed to the silicon 3 via VAAPI 4 without requiring I/O interruption. The steady state component of the data path 2 that is implemented in the storage management silicon 3 is referred to as the Accelerated Path (AP).

A typical prior art enterprise vendor solution is shown in FIG. 3.

The present invention provides the VAAPI which may operate in new virtualization environments that use Common Information Model/Web Based Enterprise Management (CIM/WBEM) interfaces look like the one shown in FIG. 4. Compared with FIG. 3, the interface of the present invention includes a VAAPI layer 12 interposed between a hardware subsystem 14 which includes an accelerated data path 16 and a hardware acceleration interface 18. Hardware subsystem 14 is adapted to receive data flow 20, which terminates at terminating points 22, 24. Terminating points 22, 24 may be such devices as hard disks, virtual disks, or tapes. Hardware acceleration interface 18 is interposed between accelerated data path 16 and VAAPI layer 12. In the present invention, such as in the CIM-based approach, necessary strategic foundations are provided while offering a common basis for adapting to a variety of other environments such as those using Simple Network Management Protocol (SNMP) or proprietary protocols.

Further, the present invention contemplates a system that has a management application component 30 and a Virtualization Engine 40. The management application 30 generates and handles the control path information. For example, it may use CIM/WBEM-based interfaces to exchange control information with the Virtualization Engine 40, which is implemented in the hardware.

As can be seen, the present invention provides VAAPI layer 12 and hardware subsystem 14 over prior art systems such as the one shown in FIG. 3.

The control path 22 may populate a virtualization repository 24 such as the CIM-based repository using standard CIM/WBEM formats. A Mapping Table (not shown) is implemented in the hardware and provides the mapping from the virtual storage to the physical storage. The CIM-based repository 24 provides the static information for the storage mapping in the hardware.

FIG. 5 illustrates the VAAPI support for a virtualization application using SNMP or proprietary protocols. As can be seen, a CIM based repository 50 is required. Repository 50 is implemented in hardware and is coupled to VAAPI 12, hardware acceleration interface 18 and accelerated data path 16 respectively.

In FIG. 6, there are two repositories shown, one for the software environment and one for the hardware environment. The software repository 24 supports existing vendor's current protocols and related data structures. The hardware repository 50 supports CIM/WBEM and is provided by the hardware acceleration vendor. The two repositories 24, 50 need to populate each other and maintain a certain level of synchronization. This functionality is, in part, accomplished by the VAAPI interface 12.

Along with normal data and address flows 20, VAAPI 12 also supports delegation of high-usage control functions from the software virtualization engine 40 to the hardware.
API group is preferably based on top of CIM/WBEM APIs by the control path software for interfacing with the storage with the repository related software provided. RI-APIs (Repository Interface) and AP (Accelerated Path) are APIs used to access a CIM implementation. These APIs are defined in standards documents. RI-APIs are APIs used in the control path software to propagate the virtualization information to the acceleration hardware with the storage virtualization information that it gets with the RI-APIs. I/O-APIs are APIs used in the control path software for sharing the control and data related to an I/O plan with the acceleration hardware. UA-APIs are APIs that provide utility functions, (e.g. Free buffers, etc.)

Repository Population And Synchronization (RPS-APIs)
The repository used by the hardware (AP) environment is an implementation of standard CIM model with standard CIM/WBEM APIs that are supported over an HTTPS/XML protocol. These APIs are not described in this document since they are described elsewhere in standards documents.

Repository Interface (RI-APIs) and Accelerated Path (AP-APIs)
The AP-APIs and the corresponding RI-APIs are further classified into the following groups based on their information content. Normally, for any AP-APIs, there will be a complimentary API in the RI-API.

The following are subcategories associated with VAAPI. These configurations are Virtual Disk Configuration, Storage Services Configuration, I/O Plan Exception Handling Configuration, CP-AP Shared I/O plans, AP Pass-through I/O plans, Physical Devices Discovery and Management, CP-AP Transaction Management, Event Handling, Performance and Statistics, and Utility Functions.

Virtual Disk Configuration
This group of APIs deals with configuration related to individual virtual disk and basic virtualization (i.e., disk concatenation and striping). In the VAAPI framework, I/Os that require involvement of multiple virtual disks are categorized as Storage Services related I/Os. For example, mirroring, snapshot, on-line migration etc. are termed as storage services and configuration requirements for these services are handled through a group of APIs termed as Storage Services Configuration that is described later.

The following are examples of VAAPIs of the present invention. The prefixes used to mark this group of APIs are RI (Repository Interface) and AP (Accelerated Path).

<table>
<thead>
<tr>
<th>RI_API Function</th>
<th>AP_API Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gets the list of all virtual disks from the repository.</td>
<td>Gets the information for a Virtual</td>
</tr>
<tr>
<td>Gets the information for a virtual disk from the repository.</td>
<td>Disk from the repository.</td>
</tr>
<tr>
<td>Gets the full map of a virtual disk from the repository.</td>
<td>Sets the full map of a virtual disk in AP hardware, if a map already exists then it is replaced with the new one.</td>
</tr>
<tr>
<td>Gets the information for a client from the repository.</td>
<td>Gets the ACL setup for a virtual disk.</td>
</tr>
<tr>
<td>Gets the ACL setup for a client from the repository.</td>
<td>Gets the ACL setup for a Client for a virtual disk.</td>
</tr>
<tr>
<td>Gets the ACL setup for a virtual disk in the AP hardware.</td>
<td>Gets the ACL setup for a Client for a virtual disk in AP hardware.</td>
</tr>
<tr>
<td>Gets the ACL setup for a Client for a virtual disk in AP hardware.</td>
<td>Gets Class of Service for a virtual disk from the repository.</td>
</tr>
<tr>
<td>API</td>
<td>Description</td>
</tr>
<tr>
<td>-----</td>
<td>-------------</td>
</tr>
<tr>
<td>AP_SetCoSVD_vaVendor</td>
<td>Sets Class of Service for a virtual disk in AP hardware.</td>
</tr>
<tr>
<td>RL_GetCoSVDClient_vaVendor</td>
<td>Gets Class of Service for a Client for a virtual disk from the repository.</td>
</tr>
<tr>
<td>AP_SetCoSVDClient_vaVendor</td>
<td>Sets Class of Service for a Client for a virtual disk in AP hardware.</td>
</tr>
<tr>
<td>AP_SetStatStatusVD_vaVendor</td>
<td>Sets the status of a virtual disk. The state applies to all Clients on a virtual disk. (enable, disable, quiesce, etc.).</td>
</tr>
<tr>
<td>AP_SetStatStatusVDClient_vaVendor</td>
<td>Sets the status of a virtual disk for a Client in AP hardware.</td>
</tr>
<tr>
<td>RL_GetStatCollectionDirectiveVD_vaVendor</td>
<td>Gets the statistics collection directive for a virtual disk from the repository.</td>
</tr>
<tr>
<td>AP_GetVDStorageExtent_vaVendor</td>
<td>Gets the map of a specific storage extent within an allocation for a virtual disk from the repository.</td>
</tr>
<tr>
<td>AP_SetVDStorageExtent_vaVendor</td>
<td>Sets the map of a specific storage extent within an allocation for a virtual disk in the acceleration path. This API could be used to replace part of the map of a VD in the accelerated path at the storage extent granularity.</td>
</tr>
</tbody>
</table>

**Storage Services Configuration**

This group of APIs deals with configuration related to various storage services applications like mirroring, snapshot, online migration, dynamic multi-path etc. This configuration group may involve more than one virtual disks. For example, establishing a mirror virtual disk for another virtual disk is done through an API in this group.

The prefixes used by this group of APIs are SSRI (Storage Services Repository Interface) and SSAP (Storage Services Accelerated Path).

**I/O Plan Exception Handling Configuration**

The APIs in this group provide configuration related to handling of exceptions in an I/O plan in the accelerated path. The APIs are prefixed with PERI (Plan Exception Repository Interface) and PEAP (Plan Exception Accelerated Path).
PerI_GetIOPlanParam vaVendor

This API will get the value of a given parameter in an I/O plan component. For example, the time-out value for an I/O to a mirror virtual disk. The list of parameters will be defined during the course of the implementation as needs are identified.

PERI_SetIOPlanParam vaVendor

This API will set up the value of a given parameter in an I/O plan within the accelerated path.

PERI_IOPlanContinuationMask vaVendor

The API sets a mask in order to determine if the I/O plan execution for an I/O should continue in case of failure of an I/O plan component.

PERI_IOPlanSuccessMask vaVendor

The API sets a mask in order to determine if the I/O from a client on a virtual disk is to be reported as a success or failure. For example, in one storage management environment, it may be set so that I/O to all mirrors in a plan must succeed in order to report success to an I/O client. But, if the virtual disk exposed to the client is based on a RAID-5 device, then a determination could be made to succeed the client I/O even if all the mirrors in the I/O plan fail.

PERI_IOPlanLogMask vaVendor

This API sets up a mask in order to determine which I/O components of an I/O plan need to be logged in case of failure. Also provided in this mask is information regarding whether the original data needs to be logged or not. For example, in case of a failure of a replication component - in one I/O plan, it may be decided.

PERI_VDDeactivateMask vaVendor

The API sets up a mask in order to determine if failure of an I/O component results in making a virtual disk unavailable to the clients. The client access is resumed only when the status of the virtual disk is modified from the control path software.

CP-AP Shared I/O Plans

The I/O APIs provide the facility for dealing with I/Os that are generated in the acceleration path and then handled through the control path in case of I/O exception. These APIs are prefixed with IO.

<table>
<thead>
<tr>
<th>API Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IO_GetPlan vaVendor</td>
<td>Gets the first I/O plan that was sent from the accelerated path to the control path software.</td>
</tr>
<tr>
<td>IO_GetPlanVD vaVendor</td>
<td>Gets the first I/O plan for a virtual disk that was sent from the accelerated path to the control path software.</td>
</tr>
<tr>
<td>IO_GetPlanVDAllInap vaVendor</td>
<td>Gets a list of all the outstanding I/O plans for a virtual disk in the accelerated path. These I/O plans have not yet encountered any exception. Based on a parameter, the owner of these plans is either kept unchanged or changed to the control path software as part of this list generation.</td>
</tr>
<tr>
<td>IO_ChgPlanVDAllInap vaVendor</td>
<td>Change the owner of an I/O Plan from the accelerated path to the control path.</td>
</tr>
<tr>
<td>IO_ResubmitPlan vaVendor</td>
<td>Control path software puts back an I/O plan after doing necessary handling of the exception(s) in the I/O plan.</td>
</tr>
<tr>
<td>IO_AbortPlan vaVendor</td>
<td>Aborts an I/O plan.</td>
</tr>
<tr>
<td>IO_SubmitPlan vaVendor</td>
<td>For data movement from one virtual disk to another virtual disk, the control path software may generate an I/O plan itself and submit it to the accelerated path with this API.</td>
</tr>
<tr>
<td>IO_AddDivertRange vaVendor</td>
<td>For a given virtual disk, add a block range to the acceleration path so that I/Os involving the block range are diverted to the control path software.</td>
</tr>
</tbody>
</table>
IO_RemoveDivertRange_vaVendor: For a given virtual disk, remove a previously specified block range from the acceleration path.

IO_PlanStatusDecode_vaVendor: Decodes the processing status of the I/O plan components and provides the next I/O component on which exception occurred.

AP Pass-through I/O Plans

These APIs are used to create I/O plans from the control path and send it to the devices in a pass-through mode through the acceleration path. These APIs are prefixed with IOP.

IOP_CreateIOPlan_vaVendor: This creates a new I/O plan, which can further be filled with I/O commands.

IOP_AddIO_vaVendor: An I/O is added to the I/O plan.

IOP_ChangeIO_vaVendor: The information of an I/O is changed.

IOP_GetErrorCode_vaVendor: Returns the error code for a given I/O in the I/O plan.

IOP_RelinitIOPlan_vaVendor: Re-initializes the I/O plan.

IOP_DestroyIOPlan_vaVendor: This releases the I/O plan resources.

IOP_AllocPayIdSGLBuf_vaVendor: If user wants to send down the payload in the form of SGL, he should build the SGL on the 256-byte memory area provided by this API.

IOP_FreePayIdSGLBuf_vaVendor: Free the above-allocated SGL buffer.

Devices Discovery and Management

The following APIs are related to devices discovery and management.

ISCSI Management APIs

<table>
<thead>
<tr>
<th>ISCSIAPI_Get_Global_Params</th>
<th>Gets the global ISCSI parameters from the repository.</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISCSIAPI_Get_Target_List</td>
<td>Gets the Target List from the repository.</td>
</tr>
<tr>
<td>ISCSIAPI_Get_Target_Info</td>
<td>Gets the information for a Target from the repository.</td>
</tr>
<tr>
<td>ISCSIAPI_Get_Initiator_List_VD</td>
<td>Gets the Initiator List for a VD from the repository.</td>
</tr>
<tr>
<td>ISCSIAPI_Get_Initiator_List_Target</td>
<td>Gets the Initiator List for a Target from the repository.</td>
</tr>
<tr>
<td>UA_FreeBufPointer_vaVendor</td>
<td>Free the allocated buffer.</td>
</tr>
</tbody>
</table>

CP-AP Transaction Management

These APIs are used to provide a transaction management facility for updating the shared data structures between the control path and the acceleration path in a way that preserves the integrity of the modified data with respect to its use by multiple processors.

These APIs are prefixed with TXCP for the control path part and TXAP for the acceleration path.

Event Handling

In case of any exception while processing an I/O from a client according to an I/O plan along with the data is made available to the control path software. The APIs in this group provide the facilities to decode information from the I/O plans. Also, this API group provides APIs for determining the recipients of the exception information and APIs for sending the exception information.

The APIs in this group are prefixed with EHRI (Event Handling Repository Interface) and EHAP (Event Handling Accelerated Path).

EHAP_Register_EventHandler_vaVendor: This API registers a function that is called for a particular type of event.

EHAP_UnRegister_EventHandler_vaVendor: This API un-registers the event handler.

EHRI_EventReportingSetup_vaVendor: This API sets up the infrastructure for the control path software for reporting events.

EHRI_SendEvent_vaVendor: This API sends the event to whoever has registered for receiving the event.
Performance and Statistics

This API group provides access to various performance related counters and values in the accelerated path of the Storage Virtualization Engine. The API group is prefixed with PSRI (PerformanceStatisticsRepositoryInterface) and PSAP (PerformanceStatisticsAcceleratedPath).

<table>
<thead>
<tr>
<th>PSRI/PSAP</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSRI_UpdateVDStats_vavendor</td>
<td>Updates all the statistics in the repository for a given virtual disk</td>
</tr>
<tr>
<td>PSAP_CopyVDStats_vavendor</td>
<td>Gets all the statistics for a given virtual disk from the accelerated path</td>
</tr>
<tr>
<td>PSAP_ResetVDStats_vavendor</td>
<td>Resets all statistics for a virtual disk in the accelerated path</td>
</tr>
<tr>
<td>PSAP_GetMapSizeVD_vavendor</td>
<td>Gets the map size for a virtual disk</td>
</tr>
<tr>
<td>PSAP_GetMemReqVD_vavendor</td>
<td>Gets the full memory requirement for the virtual disk in the SVE</td>
</tr>
</tbody>
</table>

Utility APIs

These APIs will provide utility functions and are prefixed with UA. Two examples of the API in this category are:

<table>
<thead>
<tr>
<th>UA</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UA_FreeBuffPtrArray_vavendor</td>
<td>This will free all buffers related to an API that requires a parameter of pointer to an array of pointers</td>
</tr>
<tr>
<td>UA_FreeBuffPointer_vavendor</td>
<td>This will free the buffer pointed by the pointer</td>
</tr>
</tbody>
</table>

Briefly, the following changes need to be implemented in an existing virtualization environment to utilize VAAPI with hardware acceleration. The primary driver will support API calls, including the verbs and formats, as specified in VAAPI. The following identifies several of the important areas of impact.

If the Information Repository of the existing application is not CIM-based, the vendor will either need to convert the existing SNMP or proprietary formats into the CIM object model so that the current VAAPI implementation can get required information from the CIM or the vendor needs to implement the repository interface components of VAAPI on top of the proprietary repository.

The hardware acceleration component may not be able to handle certain error conditions. These error conditions need to be forwarded to the virtualization engine (software-based) to process and report them. The vendor needs to provide entry points into the existing code to allow this access.

The data path and control path of the existing software-based virtualization engine will also need to support the hardware-based accelerated data path through VAAPI. This will require changes to the control path and data path components of the virtualization engine.

One embodiment of the invention is implemented as a program product for use with a computer system such as, for example, the storage network environment as shown in FIGS. 4 and 5 and described below. The program(s) of the program product defines functions of the embodiments (including the methods described below with reference to FIGS. 6 and 7 and can be contained on a variety of signal-bearing media. Illustrative signal-bearing media include, but are not limited to: (i) information permanently stored on non-writable storage media (e.g., read-only memory devices within a computer such as CD-ROM disks readable by a CD-ROM drive); (ii) alterable information stored on writable storage media (e.g., floppy disks within a diskette drive or hard-disk drive); or (iii) information conveyed to a computer by a communications medium, such as through a computer or telephone network, including wire-
2. The storage virtualization engine of claim 1, wherein the VA-API and the hardware sub-engine are embedded in a storage management processor.

3. The storage virtualization engine of claim 1, further comprising a virtualization repository in a hardware portion of the storage virtualization engine.

4. A storage management system having a control path and a data path, the system comprising:
   a software sub-engine having the control path and data path; and
   a virtualization repository;
   a hardware sub-engine having an accelerated data path; an VA-API coupling the software sub-engine with the hardware sub-engine;
   a management application coupled to the software sub-engine, wherein command therefrom are processed by the control path, thereby some functions are performed by hardware through VA-API and data are accelerated through the accelerated data path.

5. The system of claim 4, wherein the VA-API and hardware sub-engine are embedded in a storage management processor.

6. The system of claim 4, further comprising a virtualization repository in a hardware portion of the virtualization engine.