C++ CIM Client
OpenPegasus

Denise Eckstein
Hewlett-Packard
Module Content

C++ Client Overview
  - Concept Overview
  - Client Example
  - Client API
A **CIM Operation** describes a management action (i.e., a monitor or control request) on a CIM modeled resource.

A **CIM Client** sends CIM Operation requests and receives CIM Operation responses.

A **CIM Server** receives CIM Operation requests and sends CIM Operation responses.

A CIM Client can be used to monitor and control local and/or remote resources.
CIM Client Example

Management Node

CIM Client

osinfo

CIM Server

Operating System Provider

HP-UX

Managed Node

Output:

```
# osinfo
OperatingSystem Information
Host: bodie.cup.hp.com
Name: HP-UX
Version: B.11.00
UserLicense: Unlimited user license
OSCapability: 32 bit
LastBootTime: May 12, 2003 8:54:59 (-0700)
LocalDateTime: Jun 14, 2003 10:46:5 (-0700)
SystemUpTime: 2857866 seconds = 33 days, 1 hr, 51
```
Client Application Logic

CIM Client Application Logic contains the application logic required to perform the desired management action (e.g. "gather and display information regarding the running operating system").
CIM Object Abstraction

The CIM Object Abstraction layer contains the implementation of the CIM object and operation semantics.
CIM Object Abstraction

**EnumerateInstances**

```xml
<namedInstance>*EnumerateInstances (  
    [IN] <className> ClassName,  
    [IN,OPTIONAL] boolean LocalOnly = true,  
    [IN,OPTIONAL] boolean DeepInheritance = true,  
    [IN,OPTIONAL] boolean IncludeQualifiers = false,  
    [IN,OPTIONAL] boolean IncludeClassOrigin = false,  
    [IN,OPTIONAL,NULL] string PropertyList [] = NULL  
)
```

**CIM Operations**

**CIM Objects**

**CIM Instance**

A CIM Instance represents an instance of a Class, including values for the Properties.

```
Instance = Class Name +  
zero or more Properties +  
zero or more Qualifiers
```
Common Information Model

CIM Schema and CIM Operations play a critical role in creating an interoperable solution.

Key Fact:
Communication between the CIM Server and CIM Clients and between the CIM Server and CIM Providers is based on CIM Schema.
CIM Schema

CIM_OperatingSystem

CreationClassName: string [key]
Name: string [key]
OSType: uint16
OtherTypeDescription: string
Version: string
LastBootUpTime: datetime
LocalDateTime: datetime
CurrentTimeZone: sint16
NumberOfLicensedUsers: uint32
NumberOfUsers: uint32
NumberOfProcesses: uint32
MaxNumberOfProcesses: uint32
TotalSwapSpaceSize: uint64
TotalVirtualMemorySize: uint64
FreeVirtualMemory: uint64
FreePhysicalMemory: uint64
TotalVisibleMemorySize: uint64
SizeStoredInPagingFiles: uint64
FreeSpaceInPagingFiles: uint64
MaxProcessMemorySize: uint64
Distributed: boolean
MaxProcessesPerUser: uint32

Reboot(): uint32
Shutdown(): uint32
CIM Schema Extensions

PG_OperatingSystem "extends" the semantics of CIM_OperatingSystem

// ===================================================================
// PG_OperatingSystem
// ===================================================================

[Version ("2.2.0"), Description ( "An extension of CIM_OperatingSystem which adds information "
"not contained in the superclass.")]
class PG_OperatingSystem : CIM_OperatingSystem
{
    [Description ( "The capability of this operating system. "
    "One capability is '32 bits' or '64 bits'. ")]
    string OperatingSystemCapability;
    [Description ( "The elapsed time, in seconds, since the OS was booted."
    "A convenience property, versus having to calculate"
    "the time delta from LastBootUpTime to LocalDateTime.")]
    uint64 SystemUpTime;
};
CIM Schema

**CIM_OperatingSystem**
- CreationClassName: string [key]
- Name: string [key]
- OSType: uint16
- OtherTypeDescription: string
- Version: string
- LastBootUpTime: datetime
- LocalDateTime: datetime
- CurrentTimeZone: sint16
- NumberOfLicensedUsers: uint32
- NumberOfUsers: uint32
- NumberOfProcesses: uint32
- MaxNumberOfProcesses: uint32
- TotalSwapSpaceSize: uint64
- TotalVirtualMemorySize: uint64
- FreeVirtualMemory: uint64
- FreePhysicalMemory: uint64
- TotalVisibleMemorySize: uint64
- SizeStoredInPagingFiles: uint64
- FreeSpaceInPagingFiles: uint64
- MaxProcessMemorySize: uint64
- Distributed: boolean
- MaxProcessesPerUser: uint32

**Reboot(): uint32**
**Shutdown(): uint32**

**PG_OperatingSystem**
- OperatingSystemCapability: string
- SystemUpTime: uint64;

**osinfo**

```
# osinfo
OperatingSystem Information
Host: bodie.cup.hp.com
Name: HP-UX
Version: B.11.00
UserLicense: Unlimited user
OSCapability: 32 bit
LastBootTime: May 12, 2003 8:54:59 (-0700)
LocalDateTime: Jun 14, 2003 10:46:5 (-0700)
SystemUpTime: 2857866 seconds = 33 days, 1 hr, 51
# ```
## CIM Operation Overview

The DMTF has defined a set of CIM Operations.

<table>
<thead>
<tr>
<th>Functional Group</th>
<th>CIM Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic read</td>
<td>GetClass, EnumerateClasses, EnumerateClassNames, GetInstance, EnumerateInstances, EnumerateInstanceNames, GetProperty</td>
</tr>
<tr>
<td>Basic Write</td>
<td>SetProperty</td>
</tr>
<tr>
<td>Schema Manipulation</td>
<td>CreateClass, ModifyClass, DeleteClass</td>
</tr>
<tr>
<td>Instance Manipulation</td>
<td>CreateInstance, ModifyInstance, DeleteInstance</td>
</tr>
<tr>
<td>Association Traversal</td>
<td>Associators, AssociatorNames, References, ReferenceNames</td>
</tr>
<tr>
<td>Query</td>
<td>ExecQuery</td>
</tr>
<tr>
<td>Qualifier Declaration</td>
<td>GetQualifier, SetQualifier, DeleteQualifier, EnumerateQualifier</td>
</tr>
</tbody>
</table>
CIM Client

A **CIM Client** issues CIM Operation requests and receives and processes CIM Operation responses.

What is the value of CIM_OperatingSystem. NumberOfProcesses?

A **CIM Server** receives and processes CIM Operation requests and issues CIM Operation responses.
CIM Provider

A **CIM Server** receives CIM Operation requests, coordinates the processing of requests and responses among the Providers and sends CIM Operation responses back to the CIM Client.

A **CIM Provider** translates CIM formatted requests into resource-specific operations and translates resource-specific responses into CIM formatted responses.

A **Managed Resource** is a manageable entity (e.g., memory, process, system, application, network) plus the resource-specific instrumentation capable of monitoring and controlling the resource.

What is the value of CIM_OperatingSystem. NumberOfProcesses?

Get the value of psd_activeprocs using pstat_getdynamic.
Key Fact: The WBEM Standard allows the design and implementation of the CIM Client Application to be Provider neutral.
Module Content

C++ Client Overview
  ▪ Concept Overview
  ▪ **Client Example**
  ▪ Client API
Client Application Logic

1. Connect to Target CIM Server
2. Issue CIM Operation Request
3. Process CIM Operation Response
4. Disconnect from Target CIM Server

Diagram:
- Management Node
- System Administrator
- CIM Client
- CIM Server
- Managed Node
- Managed Node
- System
- System 1
Connection Points

<table>
<thead>
<tr>
<th>ID</th>
<th>Requestor</th>
<th>Responder</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>&quot;connect&quot; CIM Client</td>
<td>CIM Server</td>
</tr>
<tr>
<td>2</td>
<td>&quot;connectLocal&quot; CIM Client</td>
<td>CIM Server</td>
</tr>
</tbody>
</table>
Connect to Target CIM Server

- connect
  
  ```java
  client.connect ( hostName,.portNumber, sslcontext, userName, password );
  ```

- connectLocal
  
  ```java
  client.connectLocal();
  ```

- Managed Node

- CIM Server

- Connect to Target CIM Server

- CIM Client
"connectLocal" Connection

<table>
<thead>
<tr>
<th>Requester</th>
<th>Responder</th>
<th>Encoding</th>
<th>Protocol</th>
<th>CIM Server Configuration Mechanisms</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIM Client</td>
<td>CIM Server</td>
<td>CIM-XML</td>
<td>Proprietary</td>
<td>Varies by platform. On HP-UX this option is not configurable. Always enabled.</td>
</tr>
</tbody>
</table>

The `connectLocal()` Client API creates a connection to the server for local clients. The connection is automatically authenticated for the current user.

**Note:** The `connectLocal` interface is NOT STANDARD and only supported on certain platforms C++ CIM Clients.
#include <Pegasus/Client/CIMClient.h>

PEGASUS_USING_PEGASUS;
PEGASUS_USING_STD;

int main(int argc, char** argv)
{
    const CIMNamespaceName NAMESPACE = CIMNamespaceName("root/cimv2");
    const CIMName CLASSNAME = CIMName("CIM_OperatingSystem");

    try
    {
        Boolean deepInheritance = true;
        Boolean localOnly = true;
        Boolean includeQualifiers = false;
        Boolean includeClassOrigin = false;
        Array<CIMInstance> cimInstances;
        CIMClient client;

        // The connectLocal client API creates a connection to the server for
        // local clients. The connection is automatically authenticated
        // for the current user.
        client.connectLocal();

        // Enumerate instances.
        cimInstances = client.enumerateInstances(
            NAMESPACE,
            CLASSNAME,
            deepInheritance, 
            localOnly, 
            includeQualifiers, 
            includeClassOrigin);

        cout << "Total Number of Instances: " << cimInstances.size() << endl;

        client.disconnect();
    }
    catch(Exception & e)
    {
        cerr << "Error: " << e.getMessage() << endl;
        exit(1);
    }
    return 0;
}
connectLocal

CIMClient client;
client.connectLocal();
client.disconnect();
connectLocal

Connection fails if CIM Server is not running.

CIMClient client;
client.connectLocal();
client.disconnect();

Error: Cannot connect to local CIM server. Connection failed.
"connect" Connection Point

<table>
<thead>
<tr>
<th>ID</th>
<th>Requestor</th>
<th>Responder</th>
<th>Protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>&quot;connect&quot; CIM Client</td>
<td>CIM Server</td>
<td>HTTP</td>
</tr>
<tr>
<td>1</td>
<td>&quot;connect&quot; CIM Client</td>
<td>CIM Server</td>
<td>HTTPS</td>
</tr>
</tbody>
</table>
The DMTF recommends the use of the following well-known IP ports for use in compliant CIM Servers. This is a recommendation only and not a requirement for compliance with this specification. These port addresses have been acquired from IANA by the DMTF and are registered with IANA so are for the exclusive use for DMTF functions, in particular CIM Servers.

CIM-XML (http) 5988/tcp
CIM-XML (https) 5989/tcp
"connect" Connection

<table>
<thead>
<tr>
<th>Requester</th>
<th>Responder</th>
<th>Encoding</th>
<th>Protocol</th>
<th>Port</th>
<th>CIM Server Configuration Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIM Client</td>
<td>CIM Server</td>
<td>CIM-XML</td>
<td>HTTPS over TCP/IP</td>
<td>5989</td>
<td>enableHttpsConnection Default = TRUE</td>
</tr>
<tr>
<td>CIM Client</td>
<td>CIM Server</td>
<td>CIM-XML</td>
<td>HTTP over TCP/IP</td>
<td>5988</td>
<td>enableHttpConnection Default = FALSE</td>
</tr>
</tbody>
</table>

The `connect()` Client API, can be used to create an HTTP connection with the server defined by the URL in address. User name, Password and SSL information can be passed using the `connect` Client API.

**Note:** This interface implements the DMTF CIM-XML Standard.
"connect" Connection

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Port</th>
<th>CIM Server Configuration Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTP over TCP/IP</td>
<td>5989</td>
<td>enableHttpsConnection: Default = Platform Specific</td>
</tr>
<tr>
<td>HTTPS over TCP/IP</td>
<td>5988</td>
<td>enableHttpConnection: Default = Platform Specific</td>
</tr>
</tbody>
</table>

**Key Fact:** On some platforms, port 5988 is not enabled by default. `cimconfig` can be used to enable this port.

```
# cinconfig -l -c
shutdownTimeout=10
enableRemotePrivilegedUserAccess=true
enableHttpsConnection=true
enableNamespaceAuthorization=false
enableHttpConnection=false
# cinconfig -s enableHttpConnection=true -p
Planned value for the property 'enableHttpConnection' is set to "true" in CIMServer.
# cinserver -s
Shutdown timeout expired, CIM Server process killed.
# cinserver
# cinconfig -l -c
shutdownTimeout=10
enableRemotePrivilegedUserAccess=true
enableHttpsConnection=true
enableNamespaceAuthorization=false
enableHttpConnection=true
# cinconfig -s enableHttpConnection=true -p
```

**Default Setting**

```shell
cimconfig -s enableHttpConnection=true
cimconfig -s enableHttpConnection=false
```

```shell
cimconfig -s enableHttpConnection=true -p
```
```
#include <Pegasus/Client/CIMClient.h>
PEGASUS_USING_PEGASUS;
PEGASUS_USING_STD;

int main(int argc, char** argv)
{
    const CIMNameName NAMESPACE = CIMNameName("/root/cimv2");
    const CIMName CLASSNAME = CIMName("_CIM_OperatingSystem");
    try
    {
        String hostName = "localhost";
        Uint32 portNumber = 5988;
        String userName = "guest";
        String password = "guest";
        client.connect(hostName, portNumber, userName, password);
        // The connect client API, can be used to create an HTTP
        // connection with the server defined by the URL in address.
        // user name and password information can be passed
        // using the connect Client API.
        client.connect(hostName, portNumber, userName, password);
        // Enumerate Instances
        CIMInstances = client.enumerateInstances(
            NAMESPACE,
            CLASSNAME,
            deepInheritance = true,
            locaOnly = true,
            includeQualifiers = false,
            includeClassOrigin = false);
        cout << "Total Number of Instances: " << CIMInstances.size() << endl;
        client.disconnect();
    }
    catch(Exception& e)
    {
        cerr << "Error: " << e.getMessage() << endl;
        exit(1);
    }
    return 0;
}
```
connect

String  hostName = "localhost";
Uint32  portNumber = 5988;
String  userName = "guest";
String  password = "guest";

```
# pwd
/opt/wbem/sample/ClassClients/ConnectExample
# ls
ConnectExample.cpp  Makefile
# make
   aCC +DD64 -AA -mt -c -o ConnectExample.o -I/opt/wbem/include -DPEGASUS_PLAT FORM_HPUX_IA64_ACC -DHPUX_IA64_NATIVE_COMPILER -DPEGASUS_TEMP_HARDCODED_IND_DELIVER Y -DINDICATION_DIR="/var/opt/wbem" ConnectExample.cpp
   aCC +DD64 -AA -mt -L/opt/wbem/lib -oConnectExample ConnectExample.o -lpegco mmon -lpegclient -lpthread -lrt
# ./ConnectExample
Total Number of Instances: 1
#  
```
Client Application Logic

1. Connect to Target CIM Server
2. Issue CIM Operation Request
3. Process CIM Operation Response
4. Disconnect from Target CIM Server

Management Node

CIM Client

CIM Server

Managed Node

System Administrator

CIM Schema
Disconnect to Target Server

- disconnect

```java
client.disconnect();
```
CIMClient client;

client.connectLocal();

client.disconnect();
Client Application Logic

- Connect to Target Server
- Issue CIM Operation Request
- Process CIM Operation Response
- Disconnect from Target Server

System Administrator

CIM Client

CIM Server

Managed Node

Management Node
Issue CIM Operation Request

OpenPegasus C++ Client API

```c++
Array<CIMInstance> enumerateInstances (  
    const CIMNamespaceName& nameSpace,  
    const CIMName& className,  
    Boolean deepInheritance = true,  
    Boolean localOnly = true,  
    Boolean includeQualifiers = false,  
    Boolean includeClassOrigin = false,  
    const CIMPropertyList& propertyList = CIMPropertyList()  
);
```

CIM Operation

```c++
<namedInstance>* EnumerateInstances (  
    [IN] <className> ClassName,  
    [IN,OPTIONAL] boolean LocalOnly = true,  
    [IN,OPTIONAL] boolean DeepInheritance = true,  
    [IN,OPTIONAL] boolean IncludeQualifiers = false,  
    [IN,OPTIONAL] boolean IncludeClassOrigin = false,  
    [IN,OPTIONAL,NULL] string PropertyList [] = NULL  
)  
```
const CIMNamespaceName NAMESPACE = CIMNamespaceName("root/cimv2");
const CIMName CLASSNAME = CIMName("CIM_OperatingSystem");

Boolean deepInheritance = true;
Boolean localOnly = true;
Boolean includeQualifiers = false;
Boolean includeClassOrigin = false;

Array<CIMInstance> cimInstances;
CIMClient client;

cimInstances = client.enumerateInstances(
    NAMESPACE, CLASSNAME, deepInheritance, localOnly,
    includeQualifiers, includeClassOrigin );
Client Application Logic

1. Connect to Target Server
2. Issue CIM Operation Request
3. Process CIM Operation Response
4. Disconnect from Target Server
cimInstances = client.enumerateInstances(
    NAMESPACE,
    CLASSNAME,
    deepInheritance,
    localOnly,
    includeQualifiers,
    includeClassOrigin);

Uint32 index = cimInstances[0].findProperty("NumberOfProcesses");
Uint32 numberOfProcesses;
cimInstances[0].getProperty(index).getValue().get(numberOfProcesses);
cout << "Total Number of Processes: " << numberOfProcesses << endl;
Module Content

C++ Client Overview
- Concept Overview
- Client Example
- Client API

<table>
<thead>
<tr>
<th>Functional Group</th>
<th>Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic read</td>
<td>GetClass, EnumerateClasses, EnumerateClassNames, GetInstance, EnumerateInstances, EnumerateInstanceNames, GetProperty</td>
</tr>
<tr>
<td>Basic Write</td>
<td>SetProperty</td>
</tr>
<tr>
<td>Schema Manipulation</td>
<td>CreateClass, ModifyClass, DeleteClass</td>
</tr>
<tr>
<td>Instance Manipulation</td>
<td>CreateInstance, ModifyInstance, DeleteInstance</td>
</tr>
<tr>
<td>Association Traversal</td>
<td>Associators, AssociatorNames, References, ReferenceNames</td>
</tr>
<tr>
<td>Query</td>
<td>ExecQuery</td>
</tr>
<tr>
<td>Qualifier Declaration</td>
<td>GetQualifier, SetQualifier, DeleteQualifier, EnumerateQualifier</td>
</tr>
</tbody>
</table>
EnumerateInstanceNames

<table>
<thead>
<tr>
<th>CIM_Process</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSCreationClassName: string [Key]</td>
</tr>
<tr>
<td>CSName: string [Key]</td>
</tr>
<tr>
<td>OSCreationClassName: string [Key]</td>
</tr>
<tr>
<td>OSName: string [Key]</td>
</tr>
<tr>
<td>ClassName: string [Key]</td>
</tr>
<tr>
<td>Handle: string [Key]</td>
</tr>
<tr>
<td>Priority: uint32</td>
</tr>
<tr>
<td>ExecutionState: uint16</td>
</tr>
<tr>
<td>OtherExecutionDescription: string</td>
</tr>
<tr>
<td>CreationDate: datetime</td>
</tr>
<tr>
<td>TerminationDate: datetime</td>
</tr>
<tr>
<td>KernelModeTime: uint64</td>
</tr>
<tr>
<td>UserModeTime: uint64</td>
</tr>
<tr>
<td>WorkingSetSize: uint64</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PG_UnixProcess</th>
</tr>
</thead>
<tbody>
<tr>
<td>ParentProcessID: string</td>
</tr>
<tr>
<td>RealUserID: uint64</td>
</tr>
<tr>
<td>ProcessGroupID: uint64</td>
</tr>
<tr>
<td>ProcessSessionID: uint64</td>
</tr>
<tr>
<td>ProcessTTY: string</td>
</tr>
<tr>
<td>ModulePath: string</td>
</tr>
<tr>
<td>Parameters: string</td>
</tr>
<tr>
<td>ProcessNiceValue: uint32</td>
</tr>
<tr>
<td>ProcessWaitingForEvent: string</td>
</tr>
</tbody>
</table>

What processes are running on "goreme"?
Process Info Client Example

Let's start simple and just display the number of instances.

Return the InstanceName of each process running on goreme.
The **EnumerateInstanceNames** operation is used to retrieve the names of the instances of a class in a namespace.

### CIM Operation

**EnumerateInstanceNames**

```
<instanceName>* EnumerateInstanceNames (    [IN] <className> ClassName)
```

### OpenPegasus C++ Client API

```cpp
Array<CIMObjectPath> enumerateInstanceNames (    const CIMNamespaceName& nameSpace,    const CIMName& className
);
```
EnumerateInstanceNames

```
<?xml version="1.0" encoding="utf-8" ?>
<CIM CIMVERSION="2.0" DTDVERSION="2.0">
  <MESSAGE ID="10101" PROTOCOLVERSION="1.0">
    <SIMPLEREQ>
      <IMETHODCALL NAME="EnumerateInstanceNames">
        <LOCALNAMESPACEPATH>
          <NAMESPACE NAME="root"/>
          <NAMESPACE NAME="cimv2"/>
        </LOCALNAMESPACEPATH>
        <IPARAMVALUE NAME="ClassName">
          <CLASSNAME NAME="CIM_Process"/>
        </IPARAMVALUE>
      </IMETHODCALL>
    </SIMPLEREQ>
  </MESSAGE>
</CIM>
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Input</td>
</tr>
<tr>
<td>Name</td>
<td>Namespace</td>
</tr>
<tr>
<td></td>
<td>root/cimv2</td>
</tr>
<tr>
<td></td>
<td>Input</td>
</tr>
<tr>
<td>Name</td>
<td>ClassName</td>
</tr>
<tr>
<td></td>
<td>CIM_Process</td>
</tr>
</tbody>
</table>

What processes are running on goreme?
NAMESPACE = CIMNameSpaceName ("root/cimv2");
CLASSNAME = CIMName ("CIM_Process");

Array<CIMObjectPath> cimInstanceNames;

cimInstanceNames = client.enumerateInstanceNames (NAMESPACE, CLASSNAME);
cout << "Total Number of Processes: " << cimInstanceNames.size() << endl;

There are 145 processes running on goreme.
Process Info Client Example

Now let's display the Instance Names.

Return the InstanceName of each process running on goreme.

Managed Node

Management Node

CIM Client

CIM Schema

CIM Server

CIM Providers

Managed Resources

Process InstanceName

Process InstanceName

Process InstanceName

Process InstanceName
```cpp
#include <Pegasus/Client/CIMClient.h>

PEGASUS_USING_PEGASUS;
PEGASUS_USING_STD;

int main(int argc, char** argv)
{
    const CIMNamespaceName NAMESPACE = CIMNamespaceName("root/cimv2");
    const CIMName CLASSNAME = CIMName("CIM_Process");

    try
    {
        Array<CIMObjectPath> cimInstanceNames;
        CIMClient client;

        client.connectLocal();
        // enumerateInstanceNames
        // cimInstanceNames = client.enumerateInstanceNames(
        //     NAMESPACE, CLASSNAME);

        for (Uint32 i = 0; i < cimInstanceNames.size(); i++)
        {
            cout << cimInstanceNames[i].toString() << endl;
        }
        client.disconnect();
    }
    catch(Exception& e)
    {
        cerr << "Error: " << e.getMessage() << endl;
        exit(1);
    }

    return 0;
}
```
EnumerateInstanceNames

cimInstanceNames = client.enumerateInstanceNames(NAMESPACE, CLASSNAME);
for (Uint32 i = 0; i < cimInstanceNames.size(); i++)
{
    cout << cimInstanceNames[i].toString() << endl;
}
EnumerateInstanceNames

EnumerateInstanceNames (Input Values)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>Namespace</td>
<td>root/cimv2</td>
</tr>
<tr>
<td>Input</td>
<td>ClassName</td>
<td>CIM_Process</td>
</tr>
</tbody>
</table>

EnumerateInstanceNames (Return Values)

<table>
<thead>
<tr>
<th>CSCreationClassName</th>
<th>CSName</th>
<th>OSCreationClassName</th>
<th>OSName</th>
<th>CreationClassName</th>
<th>Handle</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIM_UnitaryComputerSystem</td>
<td>goreme..</td>
<td>CIM_OperatingSystem</td>
<td>HP-UX</td>
<td>PG_UnixProcess</td>
<td>0</td>
</tr>
<tr>
<td>CIM_UnitaryComputerSystem</td>
<td>goreme..</td>
<td>CIM_OperatingSystem</td>
<td>HP-UX</td>
<td>PG_UnixProcess</td>
<td>1</td>
</tr>
<tr>
<td>CIM_UnitaryComputerSystem</td>
<td>goreme..</td>
<td>CIM_OperatingSystem</td>
<td>HP-UX</td>
<td>PG_UnixProcess</td>
<td>713</td>
</tr>
<tr>
<td>CIM_UnitaryComputerSystem</td>
<td>goreme..</td>
<td>CIM_OperatingSystem</td>
<td>HP-UX</td>
<td>PG_UnixProcess</td>
<td>9</td>
</tr>
<tr>
<td>CIM_UnitaryComputerSystem</td>
<td>goreme..</td>
<td>CIM_OperatingSystem</td>
<td>HP-UX</td>
<td>PG_UnixProcess</td>
<td>10</td>
</tr>
</tbody>
</table>

...
### EnumeratelnInstances Example

**CIM_Process**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSCreationClassName</td>
<td>string [Key]</td>
<td></td>
</tr>
<tr>
<td>CSName</td>
<td>string [Key]</td>
<td></td>
</tr>
<tr>
<td>OSCreationClassName</td>
<td>string [Key]</td>
<td></td>
</tr>
<tr>
<td>OSName</td>
<td>string [Key]</td>
<td></td>
</tr>
<tr>
<td>CreationClassName</td>
<td>string [key]</td>
<td></td>
</tr>
<tr>
<td>Handle</td>
<td>string [key]</td>
<td></td>
</tr>
<tr>
<td>Priority</td>
<td>uint32</td>
<td></td>
</tr>
<tr>
<td>ExecutionState</td>
<td>uint16</td>
<td></td>
</tr>
<tr>
<td>OtherExecutionDescription</td>
<td>string</td>
<td></td>
</tr>
<tr>
<td>CreationDate</td>
<td>datetime</td>
<td></td>
</tr>
<tr>
<td>TerminationDate</td>
<td>datetime</td>
<td></td>
</tr>
<tr>
<td>KernelModeTime</td>
<td>uint64</td>
<td></td>
</tr>
<tr>
<td>UserModeTime</td>
<td>uint64</td>
<td></td>
</tr>
<tr>
<td>WorkingSetSize</td>
<td>uint64</td>
<td></td>
</tr>
<tr>
<td>OtherExecutionDescription</td>
<td>string</td>
<td></td>
</tr>
</tbody>
</table>

**PG_UnixProcess**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ParentProcessID</td>
<td>string</td>
<td></td>
</tr>
<tr>
<td>RealUserID</td>
<td>uint64</td>
<td></td>
</tr>
<tr>
<td>ProcessGroupID</td>
<td>uint64</td>
<td></td>
</tr>
<tr>
<td>ProcessSessionID</td>
<td>uint64</td>
<td></td>
</tr>
<tr>
<td>ProcessTTY</td>
<td>string</td>
<td></td>
</tr>
<tr>
<td>ModulePath</td>
<td>string</td>
<td></td>
</tr>
<tr>
<td>Parameters</td>
<td>string</td>
<td></td>
</tr>
<tr>
<td>ProcessNiceValue</td>
<td>uint32</td>
<td></td>
</tr>
<tr>
<td>ProcessWaitingForEvent</td>
<td>string</td>
<td></td>
</tr>
</tbody>
</table>

Can you list processes with their names?
EnumerateInstances

The `EnumerateInstances` operation is used to retrieve the instances of a class in a namespace.

```cpp
Array<CIMInstance> enumerateInstances ( const CIMNamespaceName& nameSpace,
const CIMName& className,
Boolean deepInheritance = true,
Boolean localOnly = true,
Boolean includeQualifiers = false,
Boolean includeClassOrigin = false,
const CIMPropertyList& propertyList = CIMPropertyList()
);
```
const CIMNamespaceName NAMESPACE = CIMNamespaceName("root/cimv2");
const CIMName CLASSNAME = CIMName("CIM_Process");
const CIMName NAMEPROPERTYNAME = CIMName("Name");
const CIMName HANDLEPROPERTYNAME = CIMName("Handle");

Array<CIMName> propertyNames(1, NAMEPROPERTYNAME);
CIMPropertyList propertyList(propertyNames);

cimInstances = client.enumerateInstances(
  NAMESPACE, CLASSNAME,
  deepInheritance, localOnly,
  includeQualifiers, includeClassOrigin,
  propertyList);

string processName;
string PID;
Uint32 index;
for (Uint32 i = 0; i < cimInstances.size(); i++)
    index = cimInstances[i].findProperty(HANDLEPROPERTYNAME);
EnumerateInstances

```cpp
String processName;
String PID;
Uint32 index;
for (Uint32 i = 0; i < cimInstances.size(); i++)
{
    index = cimInstances[i].findProperty(HANDLEPROPERTYNAME);
    cimInstances[i].getProperty(index).getValue().get(PID);
    index = cimInstances[i].findProperty(NAMEPROPERTYNAME);
    cimInstances[i].getProperty(index).getValue().get(processName);
    cout << "The process name for PID " << PID << " is " << processName << endl;
}
```
DeepInheritance & LocalOnly

```c++
Array<CIMInstance> enumerateInstances ( const CIMNamespaceName& nameSpace,
const CIMName& className,
Boolean deepInheritance = true,
Boolean localOnly = true,
Boolean includeQualifiers = false,
Boolean includeClassOrigin = false,
const CIMPropertyList& propertyList = CIMPropertyList() ) ;
```
### Deep Inheritance and Local Only Flags

<table>
<thead>
<tr>
<th>Deep</th>
<th>Local</th>
<th>Use Model</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>F</td>
<td>&quot;Tell me everything you know&quot;</td>
<td>The elements for all defined Properties are returned. This includes elements for Properties explicitly defined in the designated class, elements for inherited Properties, and elements for any Properties defined in subclasses of the designated class.</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>&quot;I'm not interested in seeing inherited properties.&quot;</td>
<td>Only elements for Properties defined in the designated class or one of the subclasses of the designated class are returned. This does not include elements for inherited Properties, but does include elements for Properties explicitly defined in the designated class or one of the subclasses of the designated class.</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>&quot;I'm only interested in the properties explicitly defined in the class I specified.&quot;</td>
<td>Only elements for Properties defined in the designated class are returned. This does not include elements for inherited Properties or elements for Properties defined in subclasses of the designated class, but does include elements for Properties explicitly defined in the designated class.</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>&quot;I'm not interested in any properties defined in the subclasses.&quot;</td>
<td>Only elements for Properties defined in the designated class or one of the superclasses of the designated class are returned. This includes elements for Properties explicitly defined in the designated class and elements of inherited properties, but does not include elements of properties defined in subclasses of the designated class.</td>
</tr>
</tbody>
</table>
Deep Inheritance and Local Only

Key Fact: This release of the product supports only the values of true for DeepInheritance and false for LocalOnly (i.e., refer to TF in the diagram),
### Deep Inheritance and Local Only Flags

<table>
<thead>
<tr>
<th>Deep</th>
<th>Local</th>
<th>Class</th>
<th>C1 Instances</th>
<th>C2 Instances</th>
<th>C3 Instances</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>C1</td>
<td>K1, K2, P1, C1.P2, C1.P3</td>
<td>K1, K2, P1, C2.P2, C1.P3, P4</td>
<td>K1, K2, P1, C3.P2, C3.P3, P4, P5</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>C1</td>
<td>K1, K2, P1, C1.P2, C1.P3</td>
<td>K1, K2, P1, C2.P2, C1.P3, P4</td>
<td>K1, K2, P1, C3.P2, C3.P3, P4, P5</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>C1</td>
<td>K1, K2, P1, C1.P2, C1.P3</td>
<td>K1, K2, P1, C2.P2, C1.P3</td>
<td>K1, K2, P1, C3.P2, C3.P3</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>C1</td>
<td>K1, K2, P1, C1.P2, C1.P3</td>
<td>K1, K2, P1, C2.P2, C1.P3</td>
<td>K1, K2, P1, C3.P2, C3.P3</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>C2</td>
<td>C2.P2, P4</td>
<td>C3.P2, P4</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>C2</td>
<td>K1, K2, P1, C2.P2, C1.P3, P4</td>
<td>K1, K2, P1, C3.P2, C3.P3, P4</td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>C3</td>
<td>C3.P2, C3.P3, P5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>C3</td>
<td>K1, K2, P1, C3.P2, C3.P3, P4, P5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>C3</td>
<td>C3.P2, C3.P3, P5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>C3</td>
<td>K1, K2, P1, C3.P2, C3.P3, P4, P5</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
GetInstance Example

Tell me about the process 0 that is running on goreme?
GetInstance Example

<table>
<thead>
<tr>
<th>GetInstance</th>
<th>CIM Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;instance&gt;</code> GetInstance (</td>
<td></td>
</tr>
<tr>
<td>[IN] <code>&lt;instanceName&gt;</code> InstanceName,</td>
<td></td>
</tr>
<tr>
<td>[IN,OPTIONAL] boolean LocalOnly = true,</td>
<td></td>
</tr>
<tr>
<td>[IN,OPTIONAL] boolean IncludeQualifiers = false,</td>
<td></td>
</tr>
<tr>
<td>[IN,OPTIONAL] boolean IncludeClassOrigin = false,</td>
<td></td>
</tr>
<tr>
<td>[IN,OPTIONAL,NULL] string PropertyList [] = NULL</td>
<td></td>
</tr>
<tr>
<td>)</td>
<td></td>
</tr>
</tbody>
</table>

getInstance | OpenPegasus C++ Client API |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>CIMInstance getInstance (</td>
<td></td>
</tr>
<tr>
<td>const CIMNamespaceName&amp; nameSpace,</td>
<td></td>
</tr>
<tr>
<td>const CIMObjectPath&amp; instanceName,</td>
<td></td>
</tr>
<tr>
<td>Boolean localOnly = true,</td>
<td></td>
</tr>
<tr>
<td>Boolean includeQualifiers = false,</td>
<td></td>
</tr>
<tr>
<td>Boolean includeClassOrigin = false,</td>
<td></td>
</tr>
<tr>
<td>const CIMPropertyList&amp; propertyList = CIMPropertyList()</td>
<td></td>
</tr>
<tr>
<td>)</td>
<td></td>
</tr>
</tbody>
</table>

The **GetInstance** operation is used to retrieve a single instance of a class in a namespace.
GetInstances Example

Tell me about the process 0 that is running on goreme?

<table>
<thead>
<tr>
<th>GetInstance (Input Values)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameter</td>
</tr>
<tr>
<td>Namespace</td>
</tr>
<tr>
<td>InstanceName</td>
</tr>
<tr>
<td>InstanceName</td>
</tr>
<tr>
<td>InstanceName</td>
</tr>
<tr>
<td>InstanceName</td>
</tr>
<tr>
<td>InstanceName</td>
</tr>
<tr>
<td>InstanceName</td>
</tr>
<tr>
<td>LocalOnly</td>
</tr>
</tbody>
</table>

PID = 0
GetInstance

```xml
<?xml version="1.0" encoding="utf-8" ?>
<CIM CIMVERSION="2.0" DTDVERSION="2.0">
  <MESSAGE ID="35002" PROTOCOLVERSION="1.0">
    <SIMPLEREQ>
      <IMETHODCALL NAME="GetInstance">
        <LOCALNAMESPACEPATH>
          <NAMESPACE NAME="root" />
          <NAMESPACE NAME="cimv2" />
        </LOCALNAMESPACEPATH>
        <IPARAMVALUE NAME="InstanceName">
          <INSTANCENAME CLASSNAME="PG_UnixProcess">
            "goreme.cup.hp.com"
          </INSTANCENAME>
        </IPARAMVALUE>
      </IMETHODCALL>
    </SIMPLEREQ>
  </MESSAGE>
</CIM>
```
### GetInstance

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instance Caption</td>
<td>swapper</td>
</tr>
<tr>
<td>Instance Description</td>
<td>swapper</td>
</tr>
<tr>
<td>Instance CSCreationClassName</td>
<td>CIM_UnitaryComputerSystem</td>
</tr>
<tr>
<td>Instance CSName</td>
<td>goreme.cup.hp.com</td>
</tr>
<tr>
<td>Instance OSCreationClassName</td>
<td>CIM_OperatingSystem</td>
</tr>
<tr>
<td>Instance OSName</td>
<td>HP-UX</td>
</tr>
<tr>
<td>Instance CreationClassName</td>
<td>PG_UnixProcess</td>
</tr>
<tr>
<td>Instance Handle</td>
<td>0</td>
</tr>
<tr>
<td>Instance Name</td>
<td>swapper</td>
</tr>
<tr>
<td>Instance Priority</td>
<td>128</td>
</tr>
<tr>
<td>Instance ExecutionState</td>
<td>6</td>
</tr>
<tr>
<td>Instance CreationDate</td>
<td>19691231160000.000000-480</td>
</tr>
<tr>
<td>Instance KernelModeTime</td>
<td>113000</td>
</tr>
<tr>
<td>Instance UserModeTime</td>
<td>0</td>
</tr>
<tr>
<td>Instance WorkingSetSize</td>
<td>0</td>
</tr>
<tr>
<td>Instance ParentProcessID</td>
<td>0</td>
</tr>
<tr>
<td>Instance RealUserID</td>
<td>0</td>
</tr>
<tr>
<td>Instance ProcessGoupID</td>
<td>0</td>
</tr>
<tr>
<td>Instance ProcessSessionID</td>
<td>0</td>
</tr>
<tr>
<td>Instance ProcessTTY</td>
<td>?</td>
</tr>
<tr>
<td>Instance Parameters</td>
<td>swapper</td>
</tr>
<tr>
<td>Instance ProcessNiceValue</td>
<td>20</td>
</tr>
</tbody>
</table>

**Variables:**
- `pst_start` from `pstat_getproc()`
- `pst_status` from `pstat_getproc()`
- `pst_stime` from `pstat_getproc()`
- `pst_utime` from `pstat_getproc()`
- `pst_ppid` from `pstat_getproc()`
- `pst_uid` from `pstat_getproc()`
- `pst_gid` from `pstat_getproc()`
- `pst_sid` from `pstat_getproc()`
- `pst_nice` from `pstat_getproc()`
Array<CIMKeyBinding> keyBindings;
keyBindings.append(CIMKeyBinding(CSCREATIONCLASSPROPERTYNAME, String::EMPTY, CIMKeyBinding::STRING));
keyBindings.append(CIMKeyBinding(CSNAMEPROPERTYNAME, String::EMPTY, String::EMPTY, CIMKeyBinding::STRING));
keyBindings.append(CIMKeyBinding(HANDLEPROPERTYNAME, PID, CIMKeyBinding::STRING));
CIMObjectPath instanceName = CIMObjectPath(String::EMPTY, NAMESPACE, CLASSNAME, keyBindings);
cimInstance = client.getInstance(NAMESPACE, instanceName, localOnly, includeQualifiers, includeClassOrigin);
cout << "Process Information for PID 0:"

    index = cimInstance.findProperty("CSName");
    if (index != PEG_NOT_FOUND)
    {
        cimInstance.getProperty(index).getValue().get(stringValue);
        cout << "The computer system name is " << stringValue << endl;
    }

    index = cimInstance.findProperty("Name");
    if (index != PEG_NOT_FOUND)
    {
        cimInstance.getProperty(index).getValue().get(stringValue);
        cout << "Process name is " << stringValue << endl;
    }

    index = cimInstance.findProperty("Priority");
    if (index != PEG_NOT_FOUND)
    {
        cimInstance.getProperty(index).getValue().get(uint32Value);
        cout << "Process priority is " << uint32Value << endl;
    }