C++ CIM Client OpenPegasus

Denise Eckstein

Hewlett-Packard

Module Content

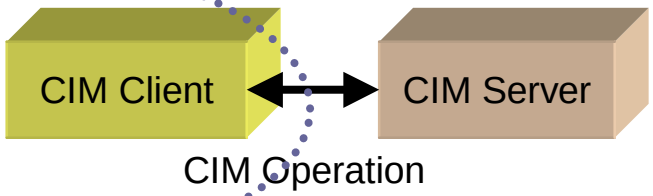
C++ Client Overview

- **Concept Overview**
- Client Example
- Client API

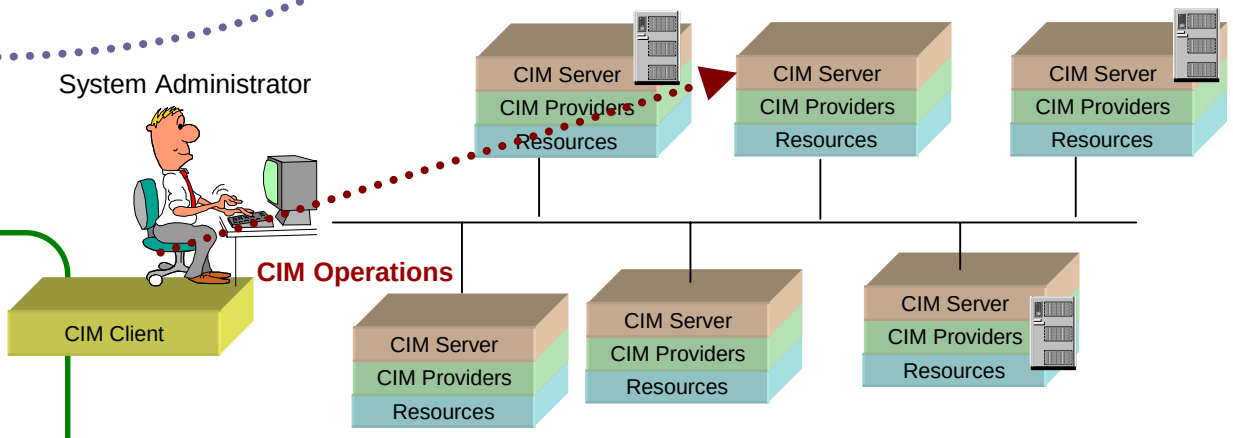
CIM Operations

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.

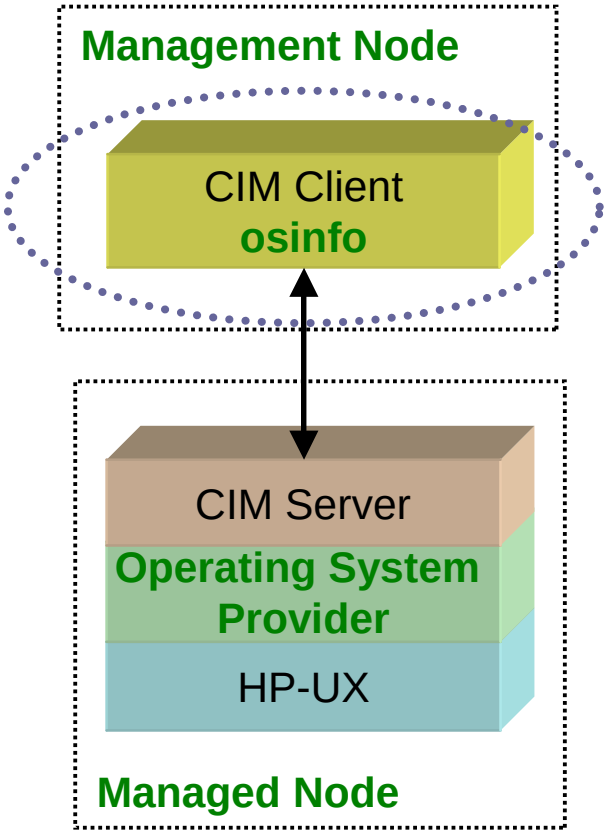


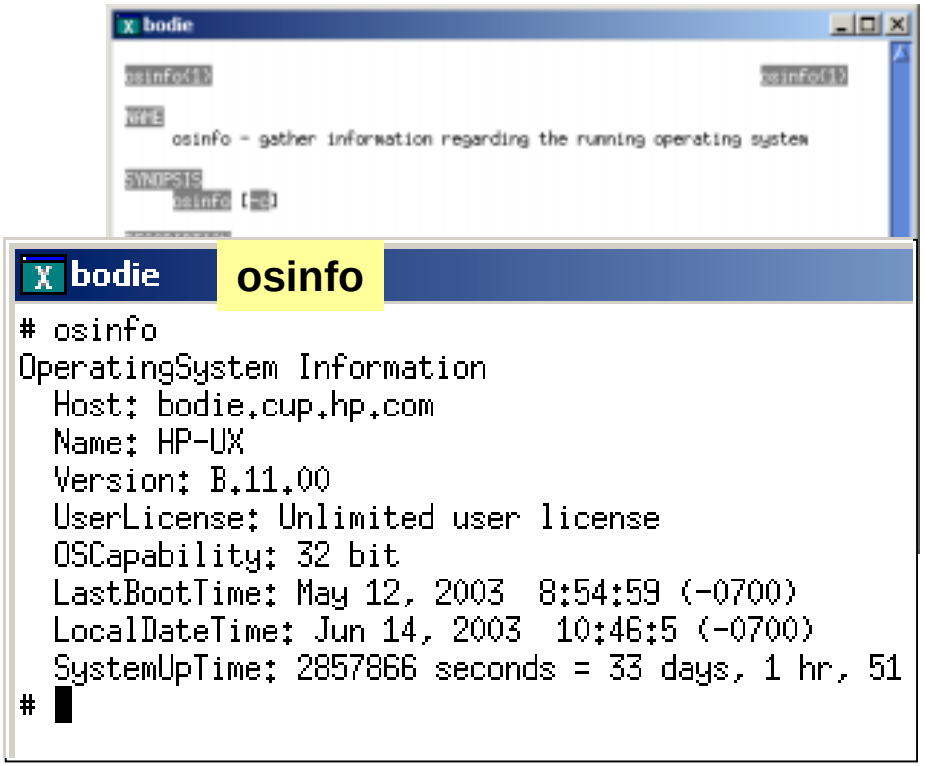
A **CIM Server** receives CIM Operation requests and sends CIM Operation responses.



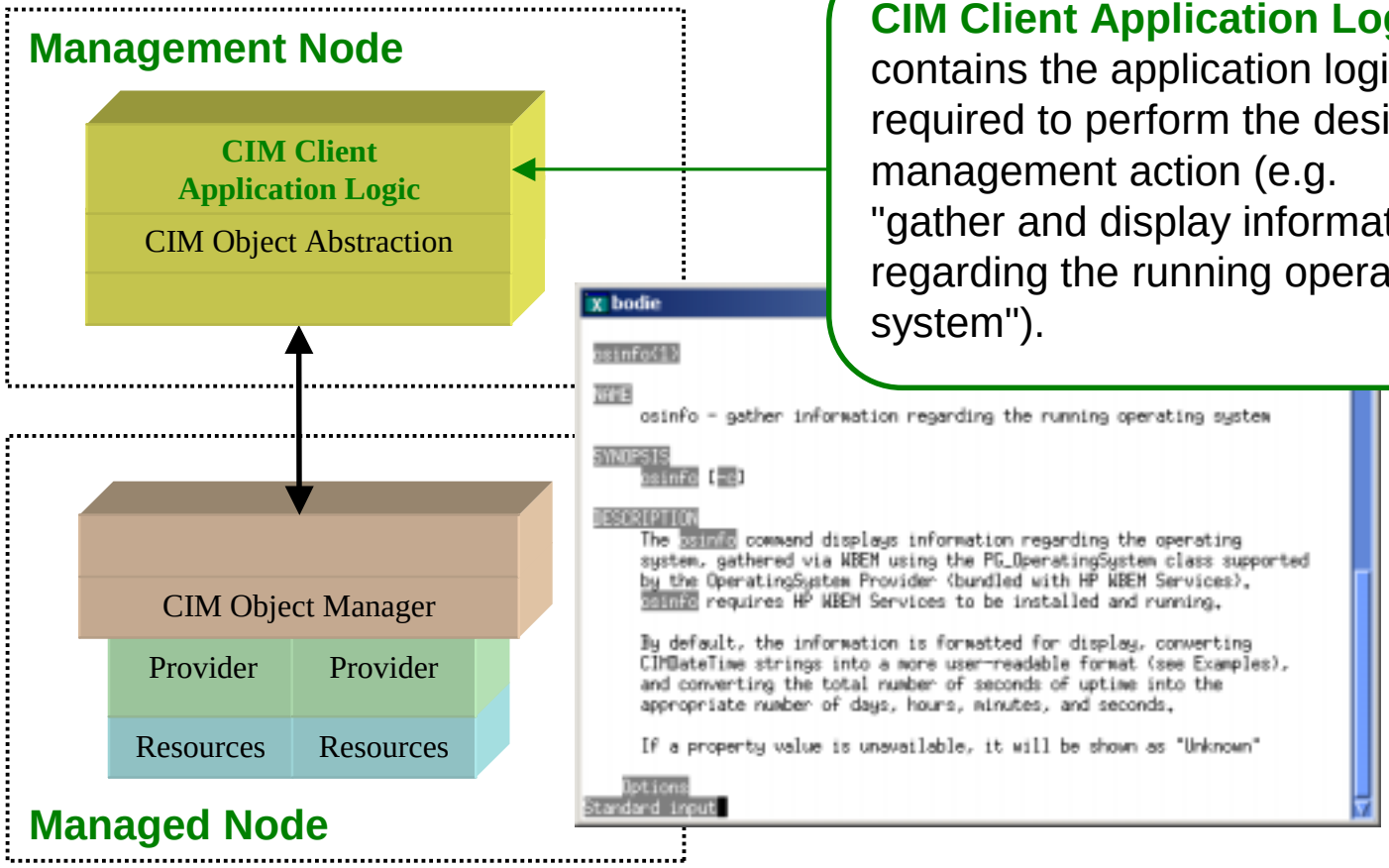
A CIM Client can be used to monitor and control local and/or remote resources.

CIM Client Example



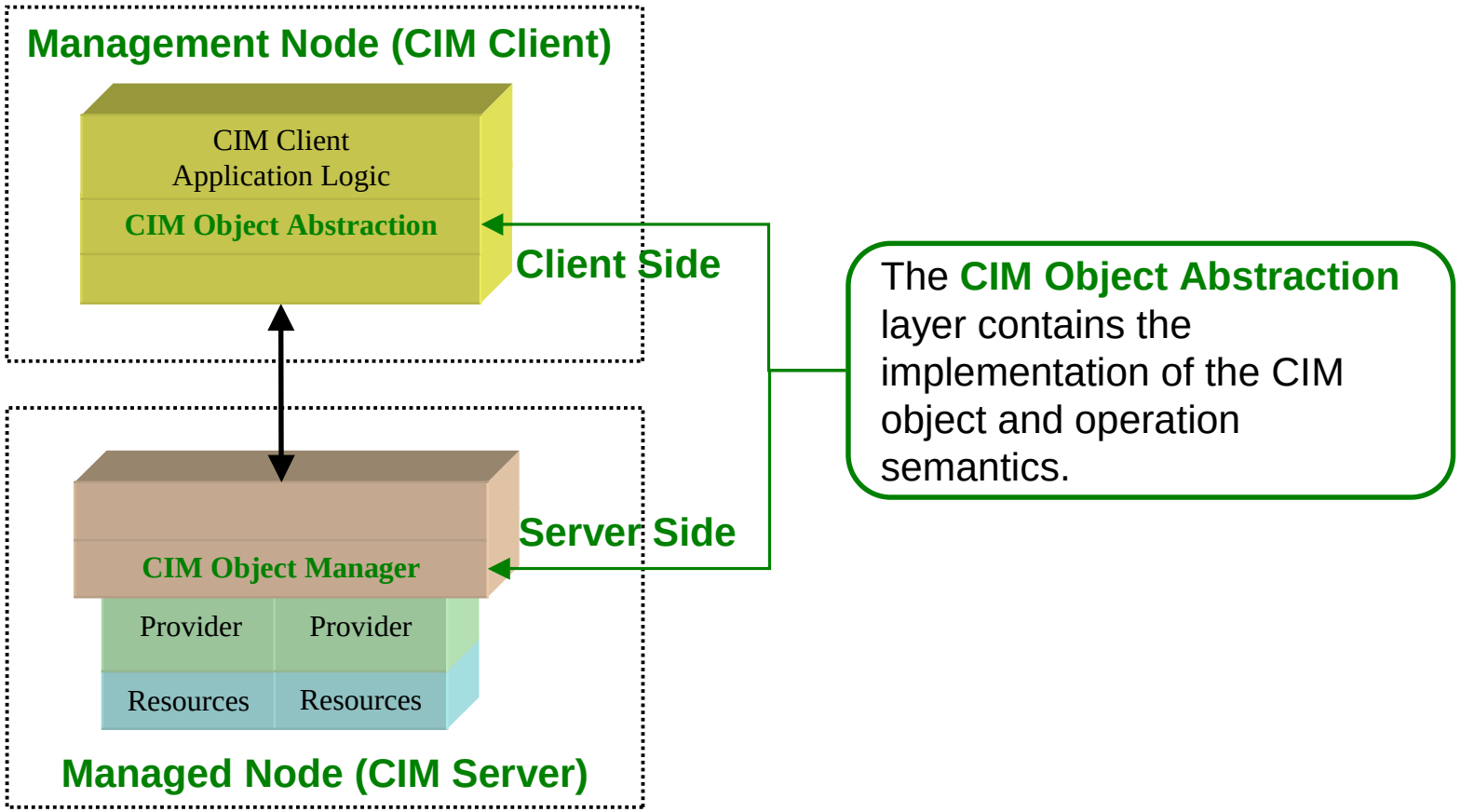


Client Application Logic

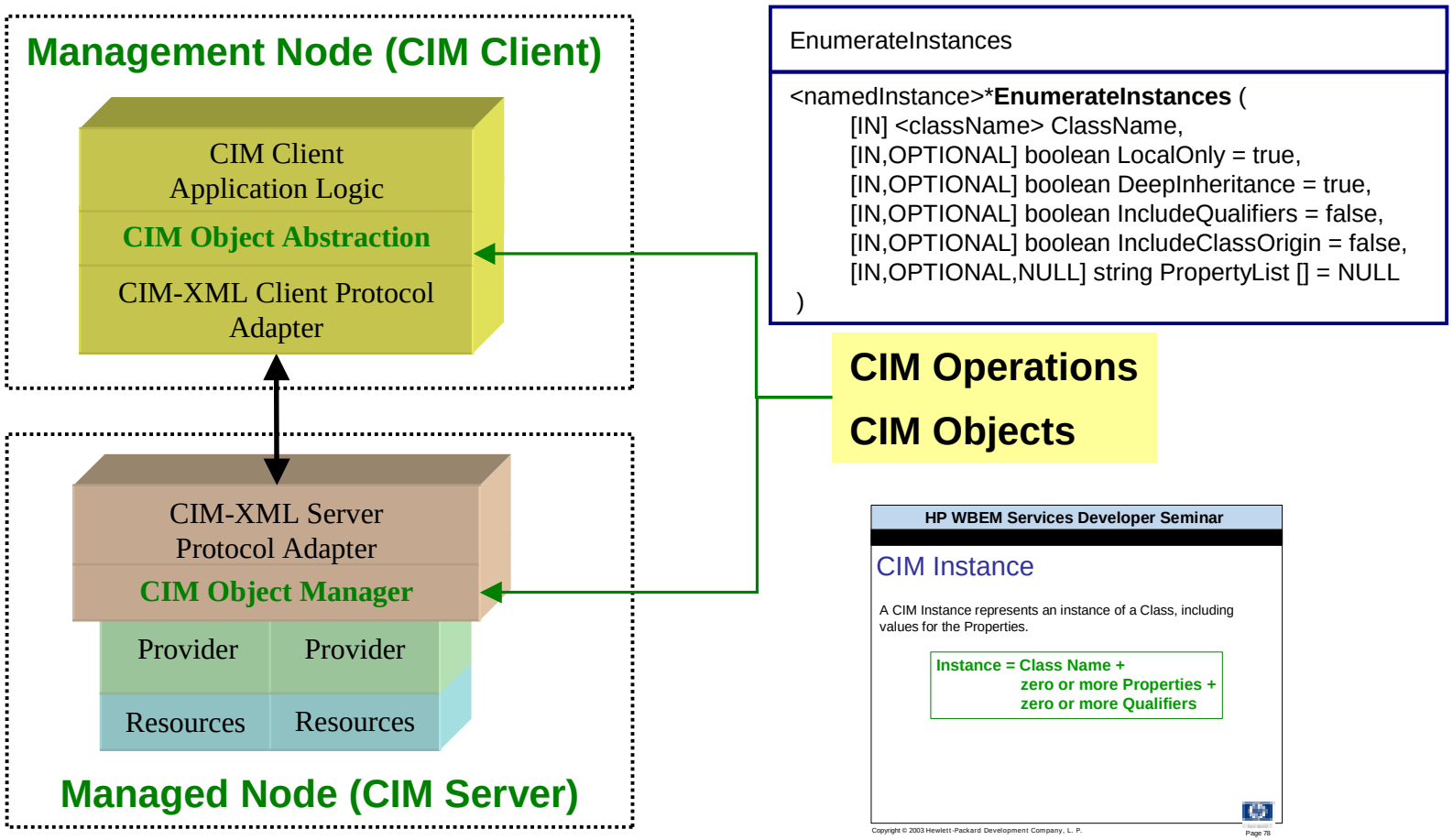


CIM Client Application Logic contains the application logic required to perform the desired management action (e.g. "gather and display information regarding the running operating system").

CIM Object Abstraction

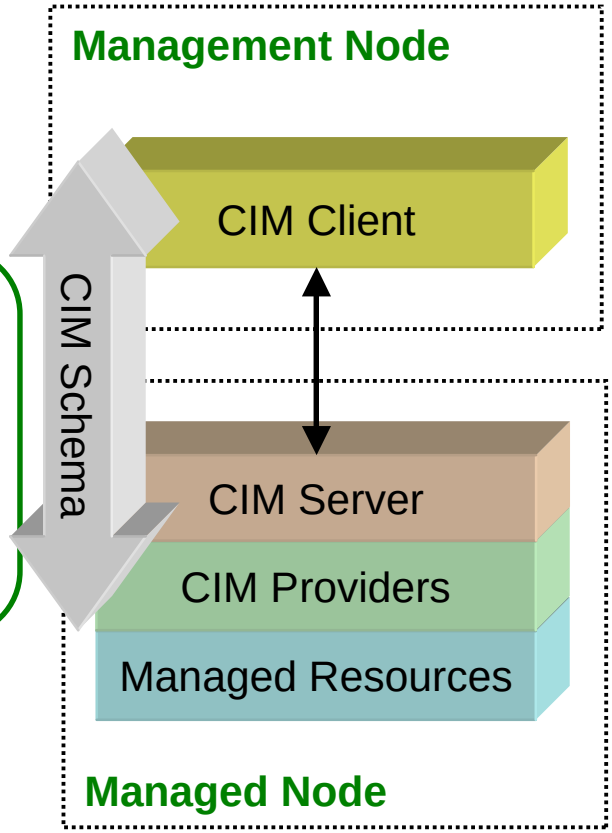


CIM Object Abstraction

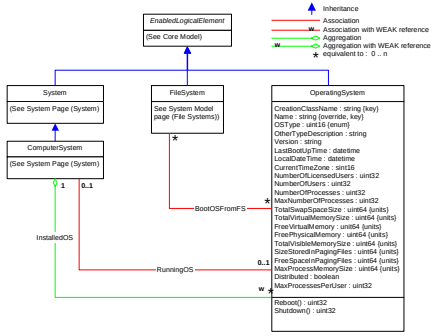


Common Information Model

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

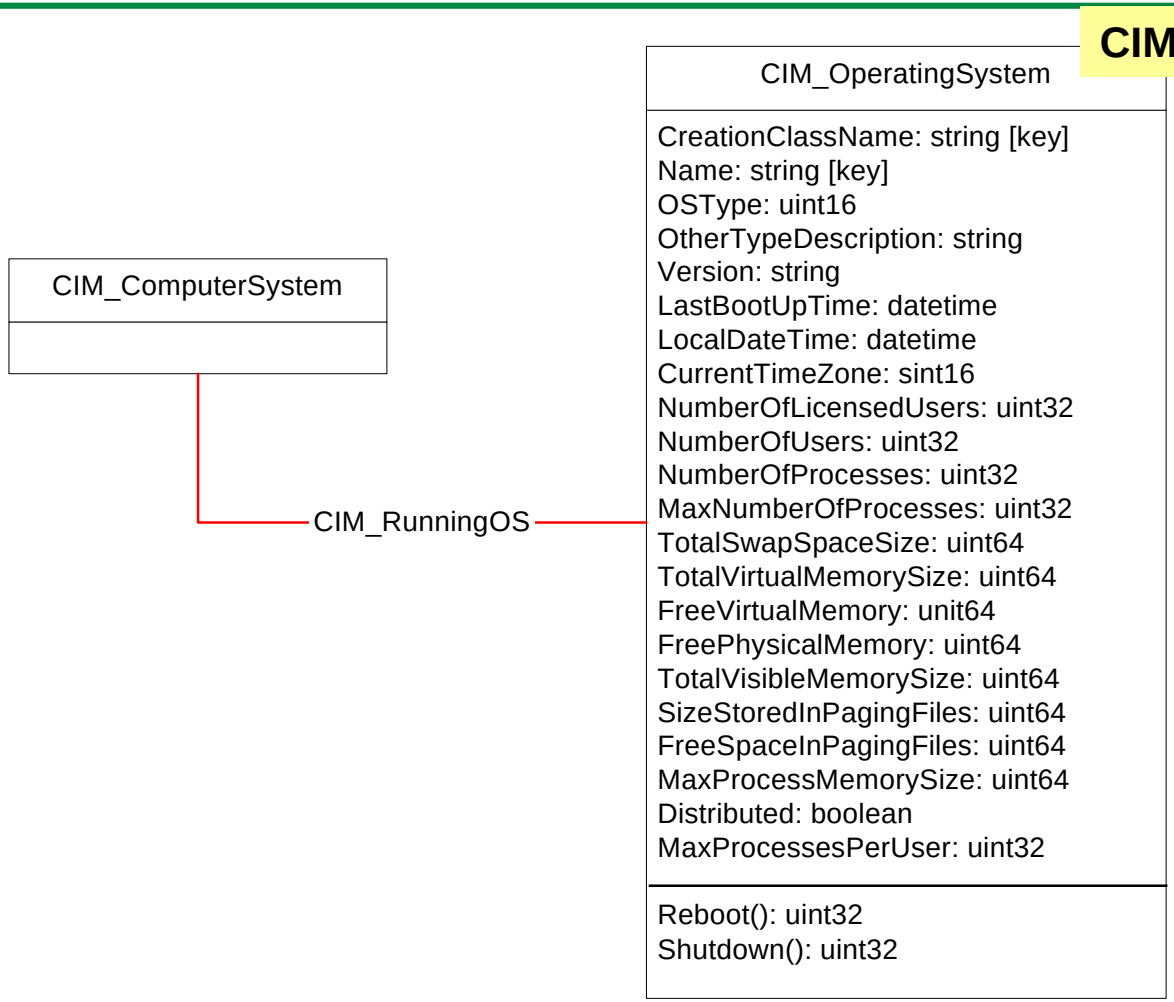


Key Fact:
Communication between the CIM Server and CIM Clients and between the CIM Server and CIM Providers is based on CIM Schema.



CIM Schema

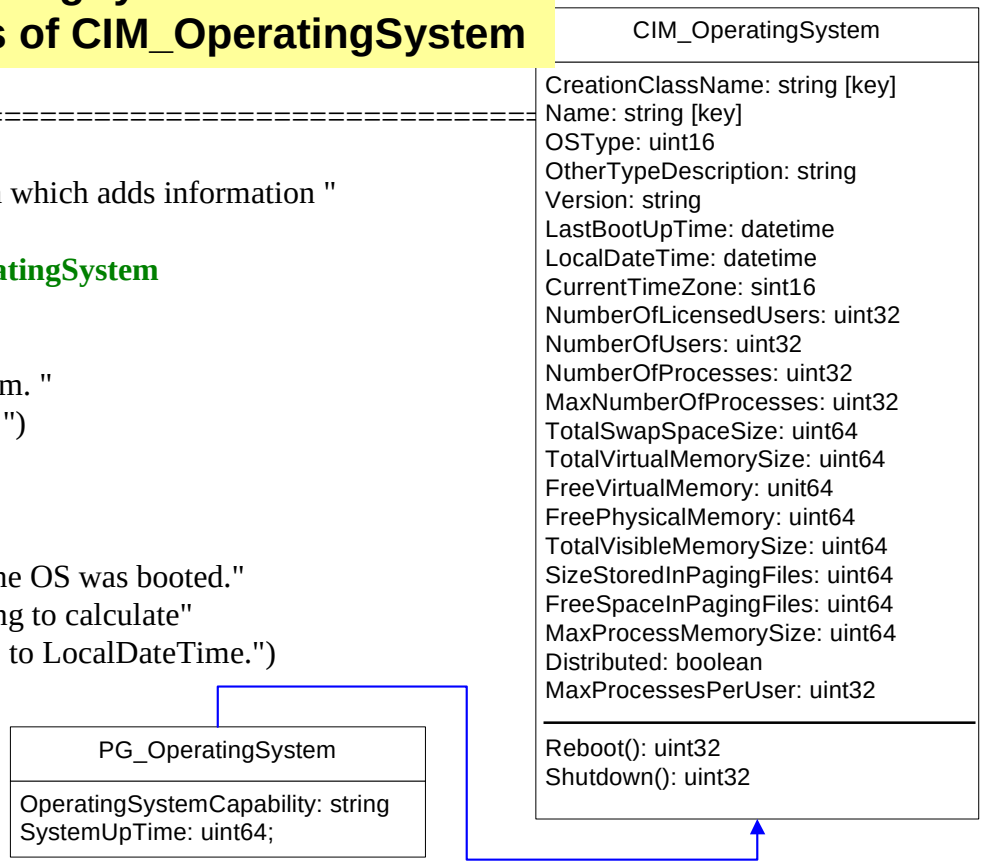
CIM_OperatingSystem



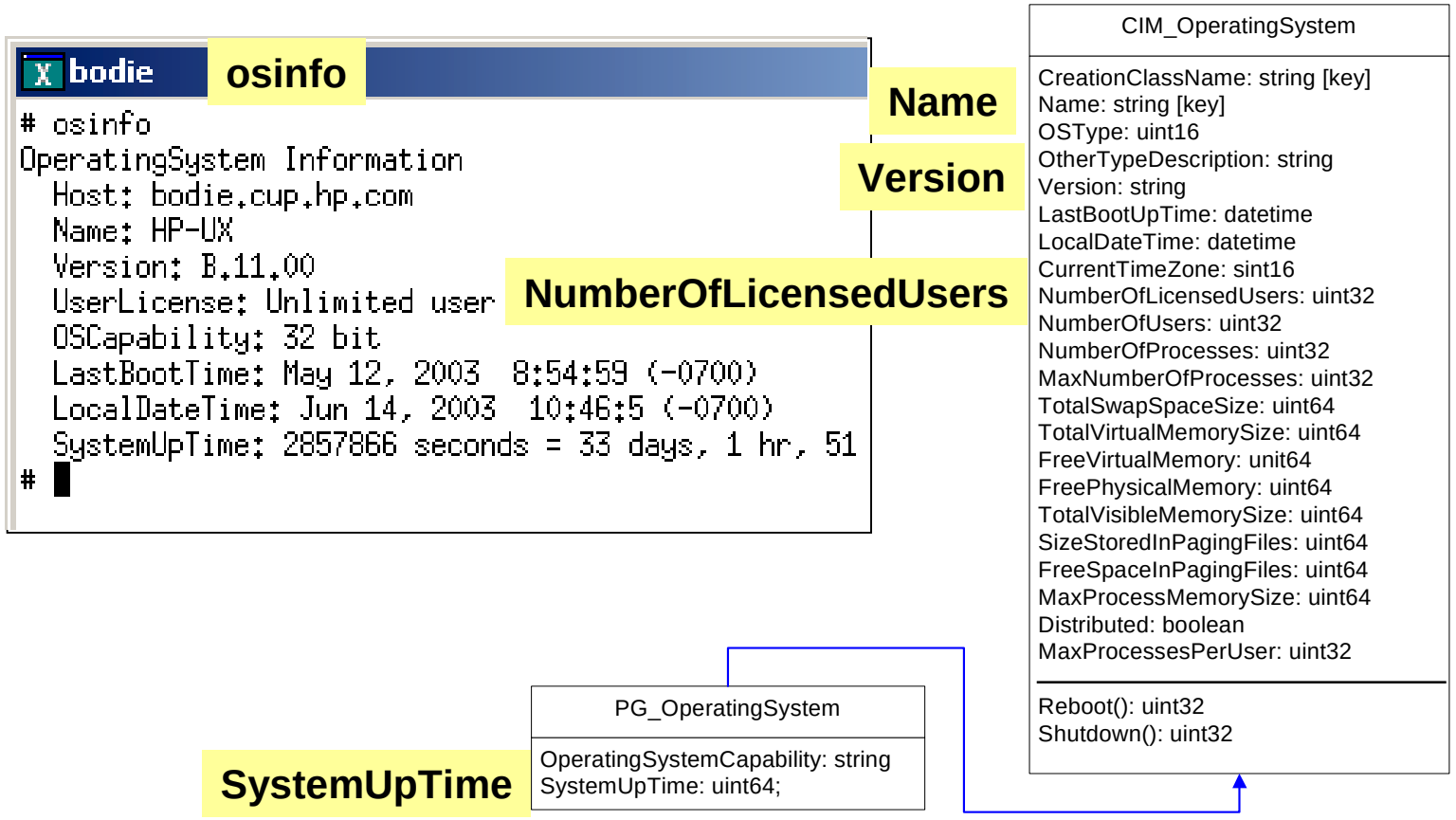
CIM Schema Extensions

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 ;
};
```



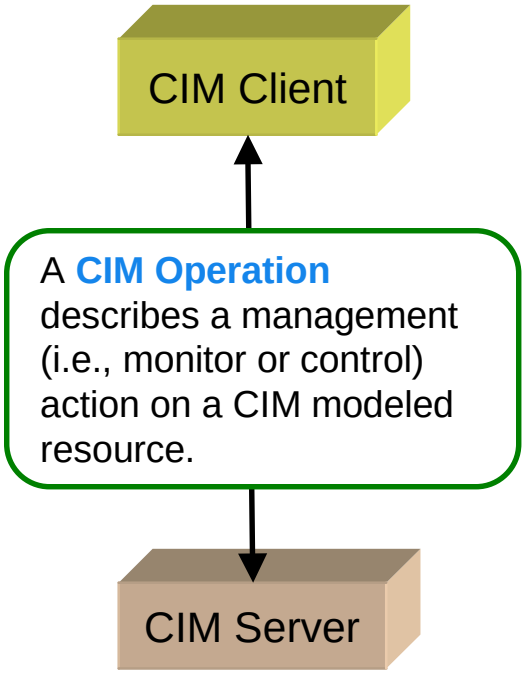
CIM Schema



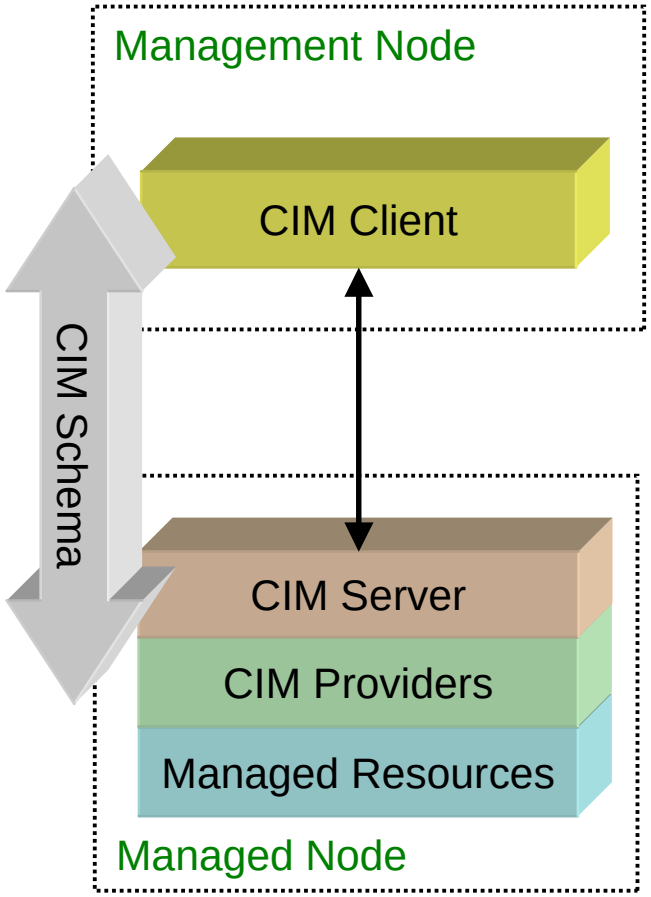
CIM Operation Overview

The DMTF has defined a set of CIM Operations.

Functional Group	CIM Operations
Basic read	GetClass, EnumerateClasses, EnumerateClassNames, GetInstance, EnumerateInstances, EnumerateInstanceNames, GetProperty
Basic Write	SetProperty
Schema Manipulation	CreateClass, ModifyClass, DeleteClass
Instance Manipulation	CreateInstance, ModifyInstance, DeleteInstance
Association Traversal	Associators, AssociatorNames, References, ReferenceNames
Query	ExecQuery
Qualifier Declaration	GetQualifier, SetQualifier, DeleteQualifier, EnumerateQualifier

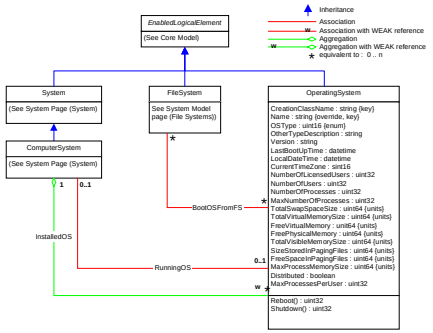


CIM Client



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

What is the value of CIM_OperatingSystem.NumberOfProcesses?



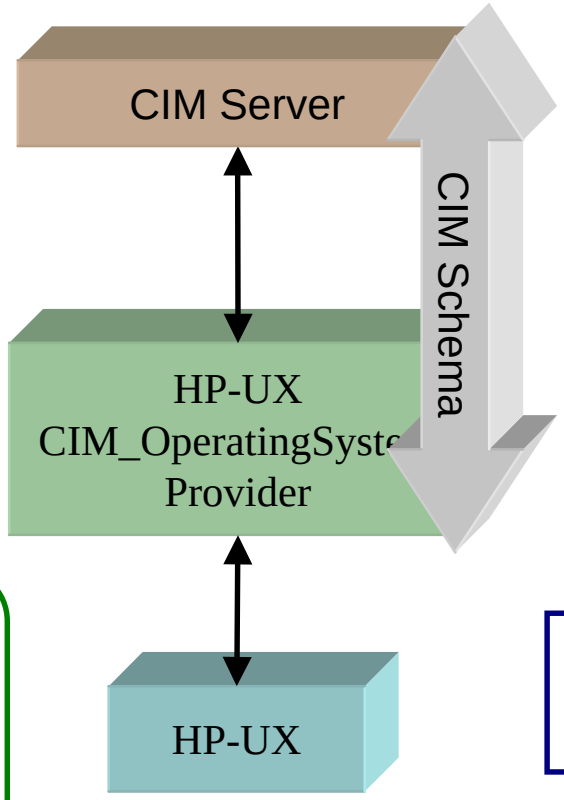
A **CIM Server** receives and processes CIM Operation requests and issues CIM Operation responses.

CIM Provider

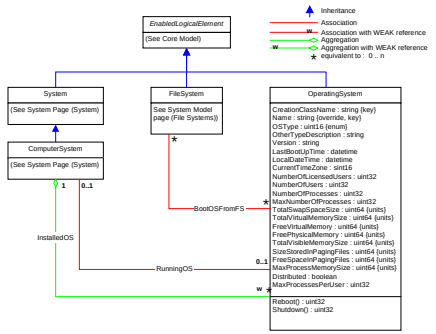
A **CIM Server** receives CIM Operation requests, coordinates the processing of requests and responses among the Providers and sends CIM Operation responses back to the CIM Client.

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.

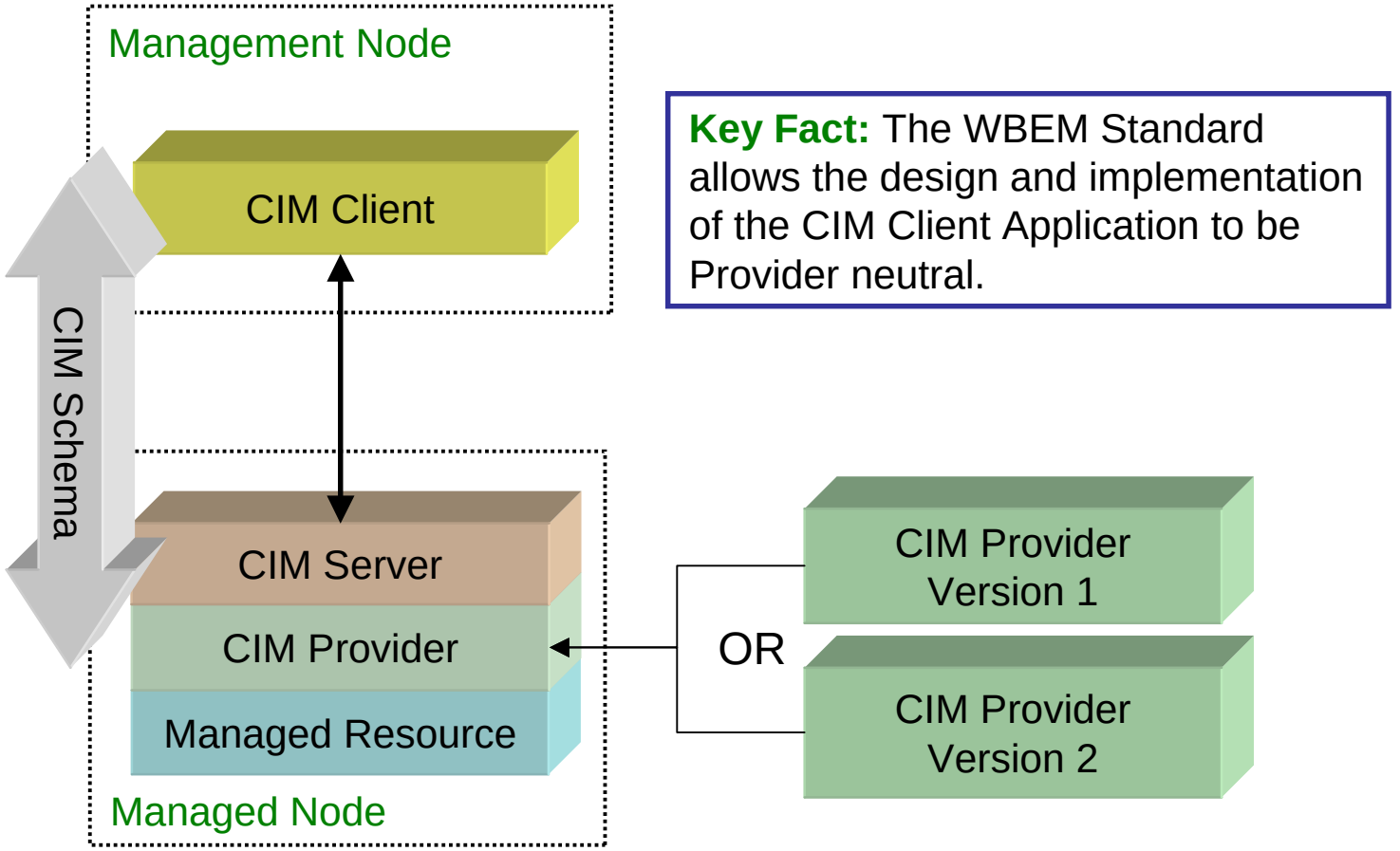


What is the value of CIM_OperatingSystem. NumberOfProcesses?



Get the value of psd_activeprocs using pstat_getdynamic.

CIM Client



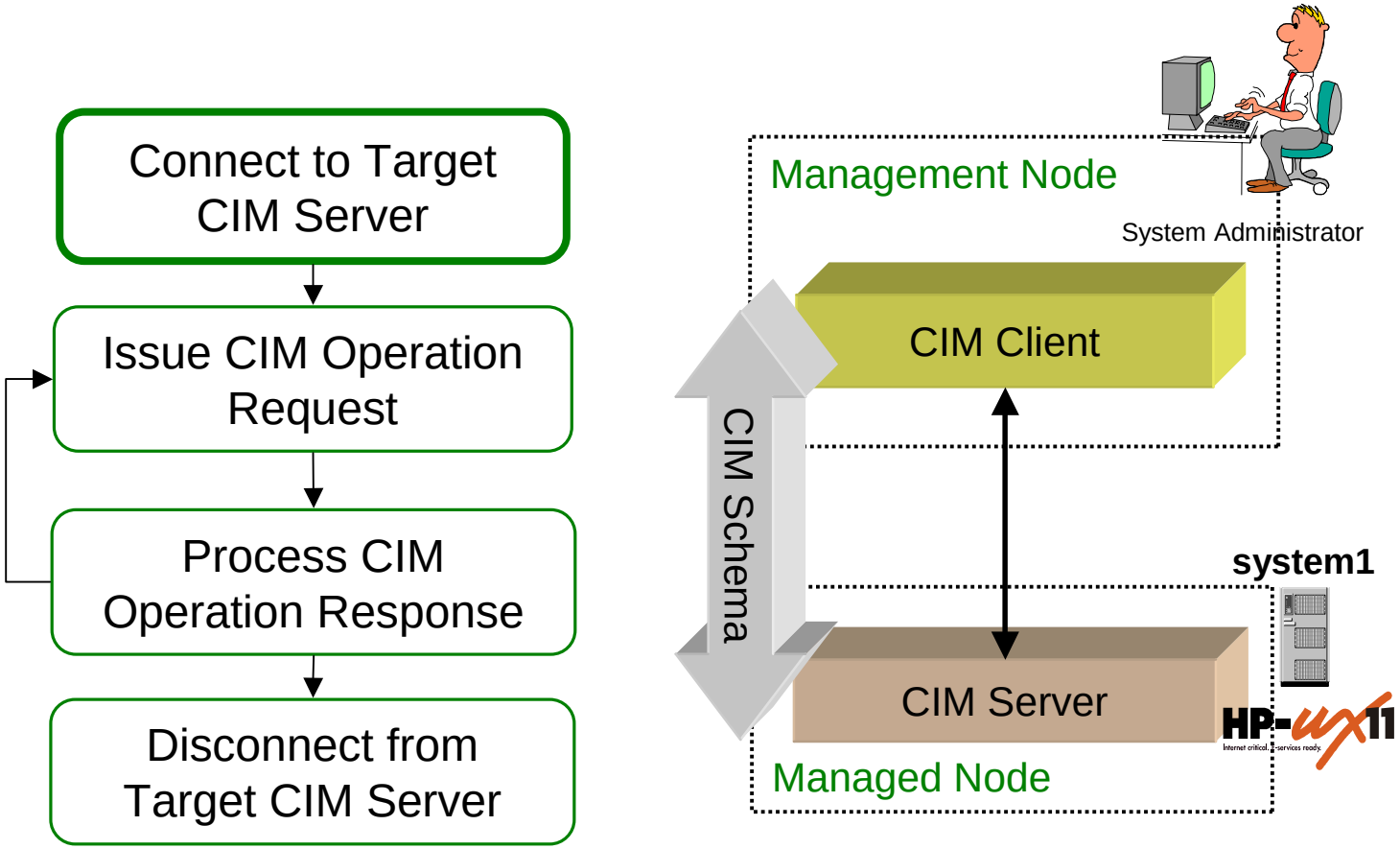
Key Fact: The WBEM Standard allows the design and implementation of the CIM Client Application to be Provider neutral.

Module Content

C++ Client Overview

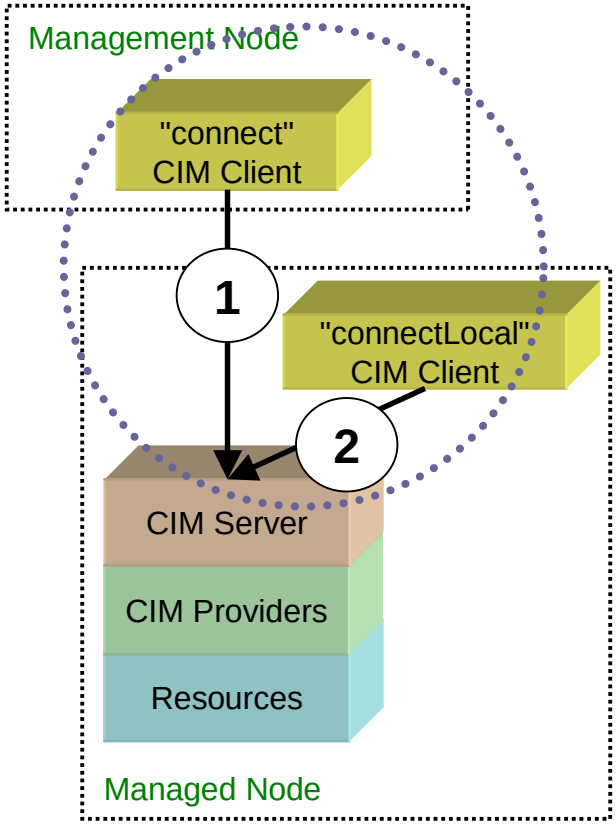
- Concept Overview
- **Client Example**
- Client API

Client Application Logic

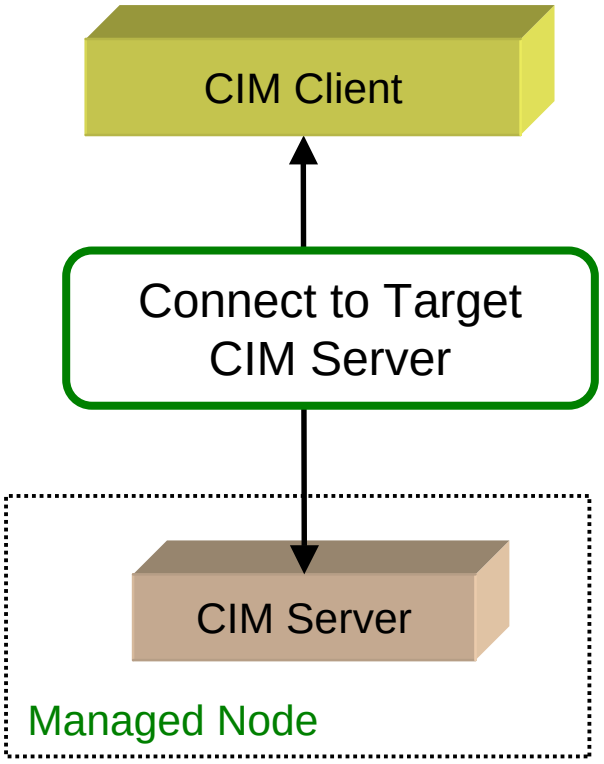


Connection Points

ID	Requestor	Responder
1	"connect" CIM Client	CIM Server
2	"connectLocal" CIM Client	CIM Server



Connect to Target CIM Server



- connect

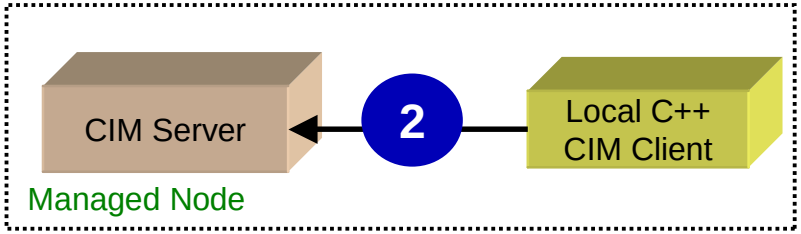
```
client.connect ( hostName,
                 portNumber, sslcontext,
                 userName, password );
```

  - host name
  - port number
  - user identification
  - SSL information
- connectLocal

```
client.connectLocal();
```

"connectLocal" Connection

Requester	Responder	Encoding	Protocol	CIM Server Configuration Mechanisms
CIM Client	CIM Server	CIM-XML	Proprietary	Varies by platform. On HP-UX this option is not configurable. Always enabled.



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.

ConnectLocalExample.cpp - Notepad

File Edit Format Help

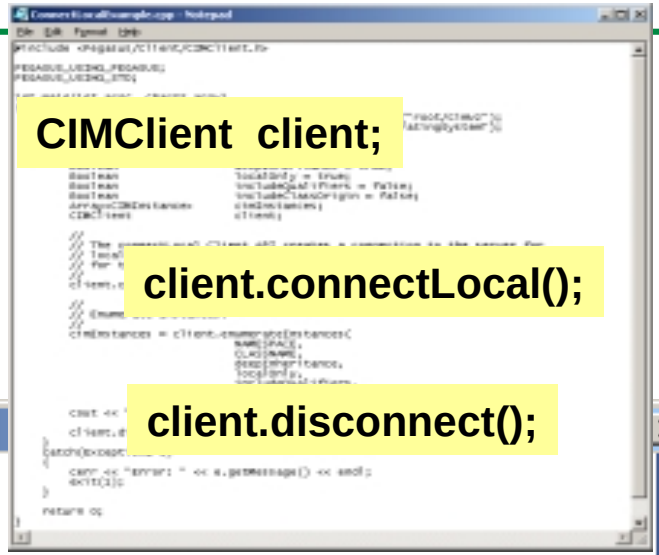
```
#include <Pegasus/Client/CIMClient.h>

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;
}
```

CIMClient client;

client.connectLocal();

client.disconnect();

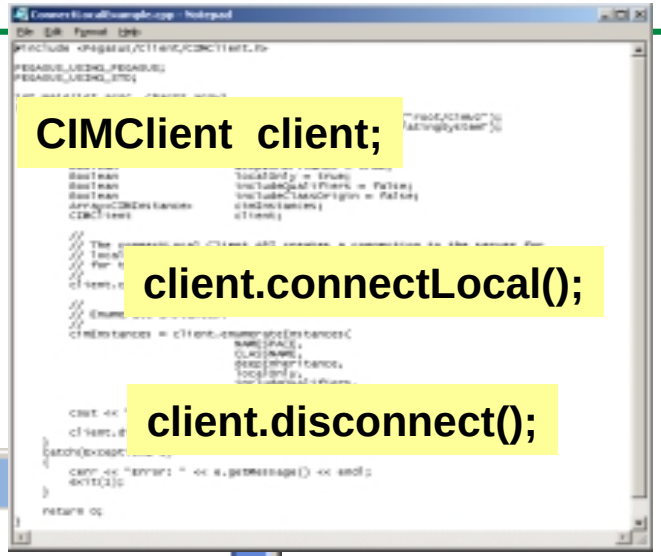
connectLocal




```
hpterm (goreme.cup.hp.com via TELNET)
# pwd
/opt/wbem/sample/ClassClients/ConnectLocalExample
# ls
ConnectLocalExample.cpp  Makefile
# make
        aCC +DD64 -AA -mt -c -o ConnectLocalExample.o -I/opt/wbem/include -DPEGASUS_PLATFORM_HPUX_IA64_ACC -
DHPUX_IA64_NATIVE_COMPILER -DPEGASUS_TEMP_HARDCODED_IND_DELIVERY -DINDICATION_DIR=\"/var/opt/wbem\"  Connect
LocalExample.cpp
        aCC +DD64 -AA -mt -L/opt/wbem/lib -oConnectLocalExample ConnectLocalExample.o -lpegcommon -lpegclien
t -lpthread -lrt
# ls
ConnectLocalExample      ConnectLocalExample.cpp  ConnectLocalExample.o    Makefile
# ./ConnectLocalExample
Total Number of Instances: 1
#
```

connectLocal

Connection fails if CIM Server is not running.

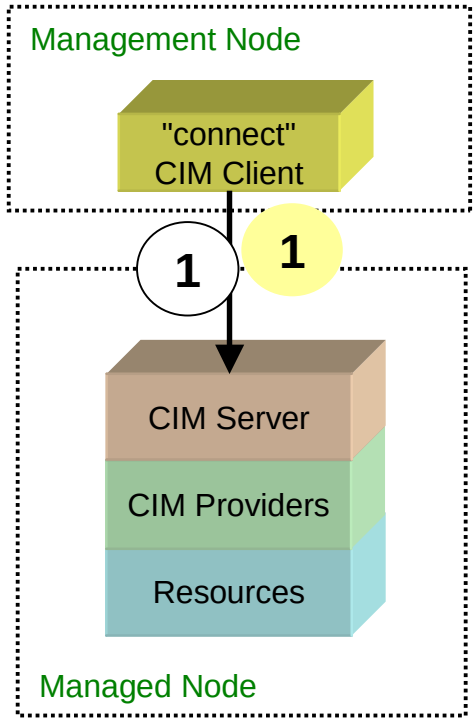



```
hpterm (goreme.cup.hp.com via TELNET)
# ./ConnectLocalExample
Total Number of Instances: 1
# cimserver -s
Shutdown timeout expired.  CIM Server process killed.
# ./ConnectLocalExample
Error: Cannot connect to local CIM server. Connection failed.
# cimserver
# ./ConnectLocalExample
Total Number of Instances: 1
#
```

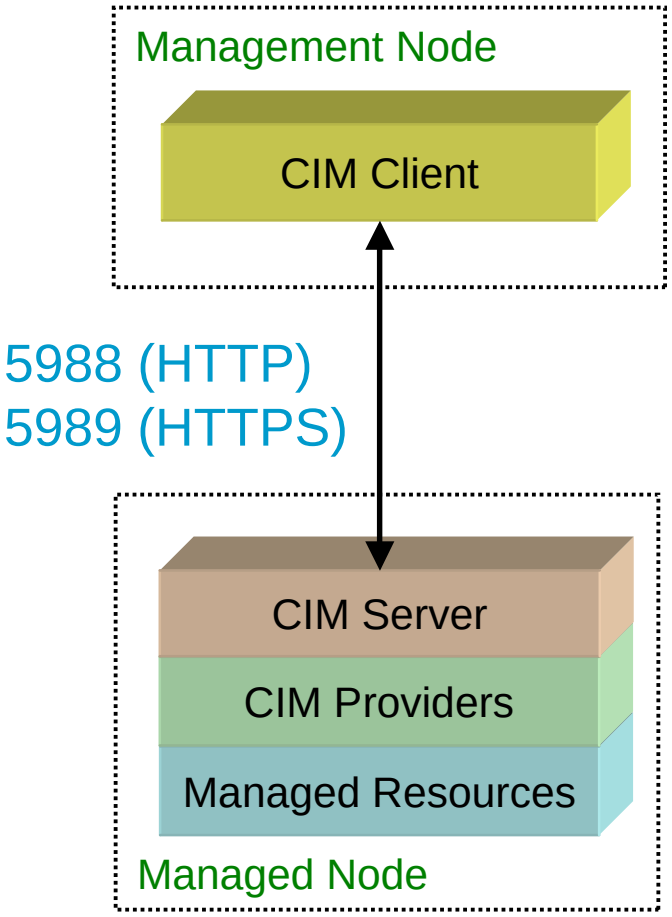
Error: Cannot connect to local CIM server. Connection failed.

"connect" Connection Point

ID	Requestor	Responder	
1	"connect" CIM Client	CIM Server	HTTP
1	"connect" CIM Client	CIM Server	HTTPS



Port Numbers

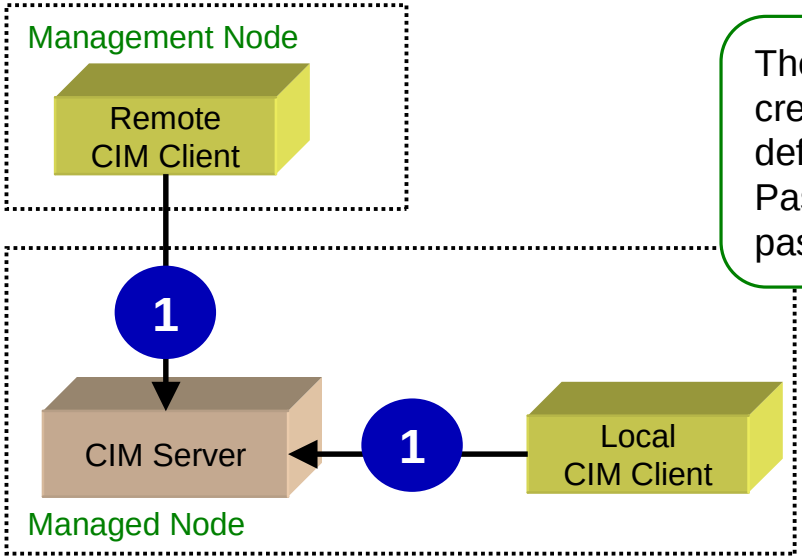


The DMTF recommends the use of the following well-known IP ports for use in compliant CIM Servers. This is a recommendation only and not a requirement for compliance with this specification. These port addresses have been acquired from IANA by the DMTF and are registered with IANA so are for the exclusive use for DMTF functions, in particular CIM Servers.

```
CIM-XML(http)        5988/tcp
CIM-XML(https)       5989/tcp
```

"connect" Connection

Requester	Responder	Encoding	Protocol	Port	CIM Server Configuration Parameter
CIM Client	CIM Server	CIM-XML	HTTPS over TCP/IP	5989	enableHttpsConnection Default = TRUE
CIM Client	CIM Server	CIM-XML	HTTP over TCP/IP	5988	enableHttpConnection Default = FALSE

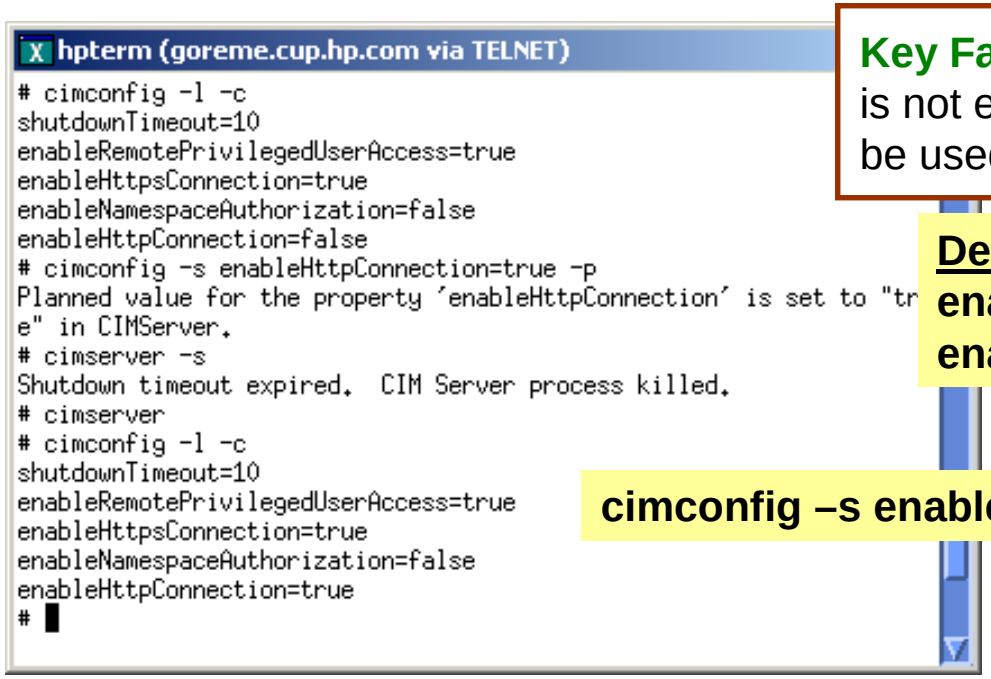


The **connect()** Client API, can be used to create an HTTP connection with the server defined by the URL in address. User name, Password and SSL information can be passed using the connect Client API.

Note: This interface implements the DMTF CIM-XML Standard.

"connect" Connection

Protocol	Port	CIM Server Configuration Parameter
HTTP over TCP/IP	5989	enableHttpsConnection: Default =Platform Specific
HTTPS over TCP/IP	5988	enableHttpConnection: Default = Platform Specific



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

Default Setting
enableHttpsConnection=true
enableHttpConnection=false

cimconfig –s enableHttpConnection=true -p

ConnectExample.cpp - Notepad

File Edit Format Help

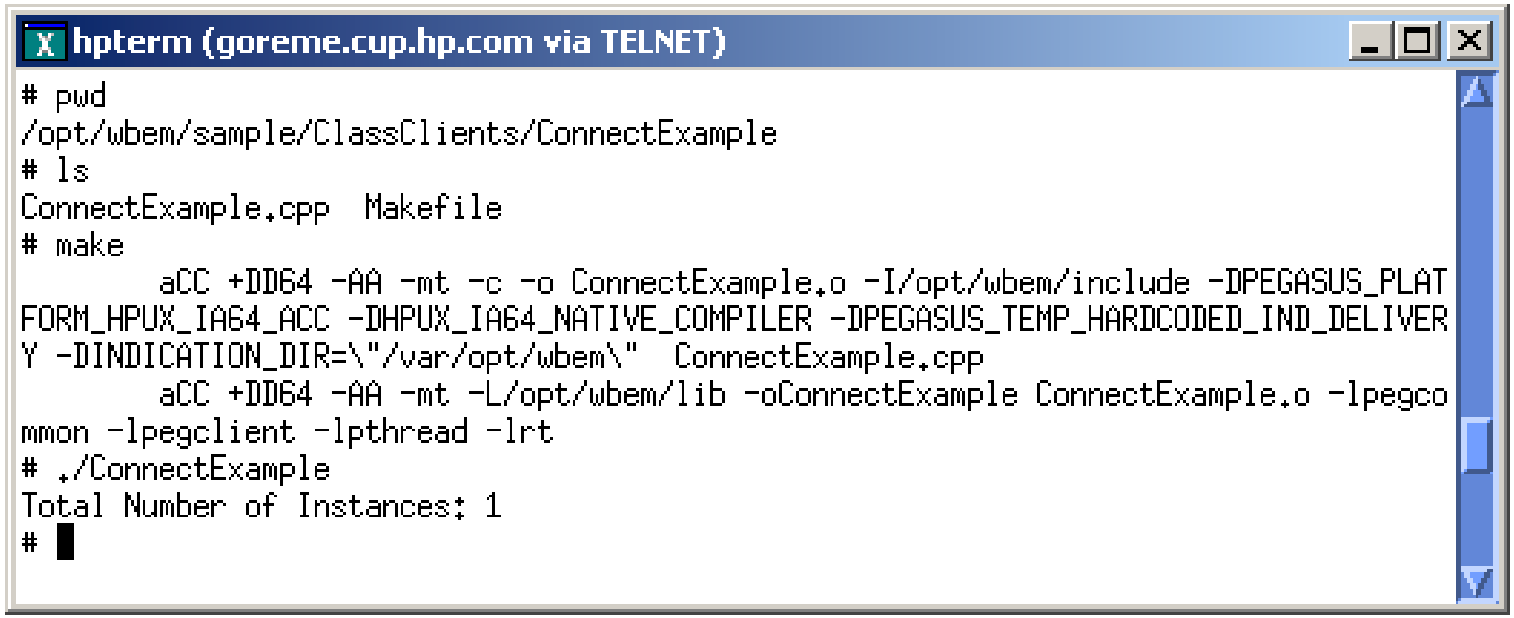
```
#include <Pegasus/Client/CIMClient.h>

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

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

        //
        // 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 Instanc
        //
        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;
}
```

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

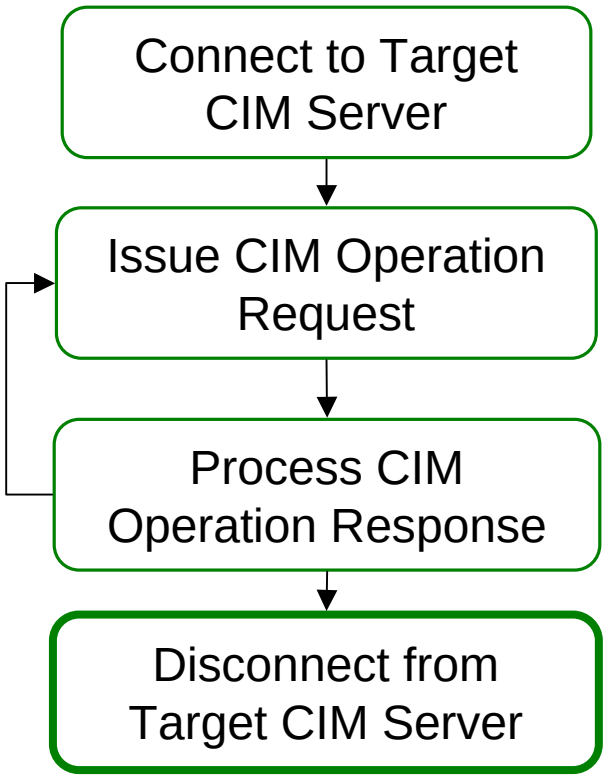
client.connect(hostName, portNumber, userName, password);

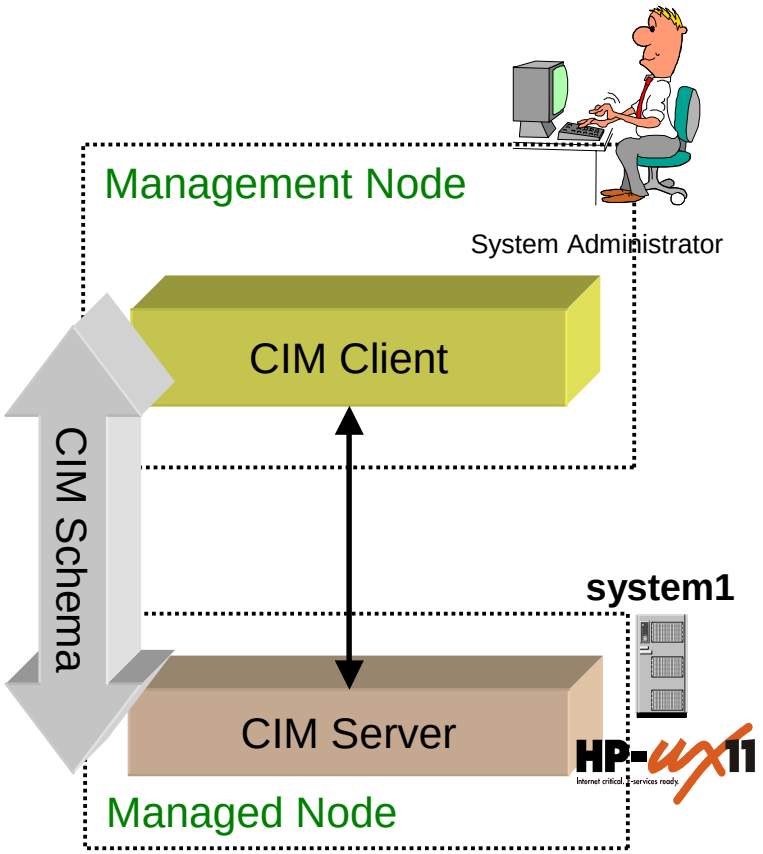
connect

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

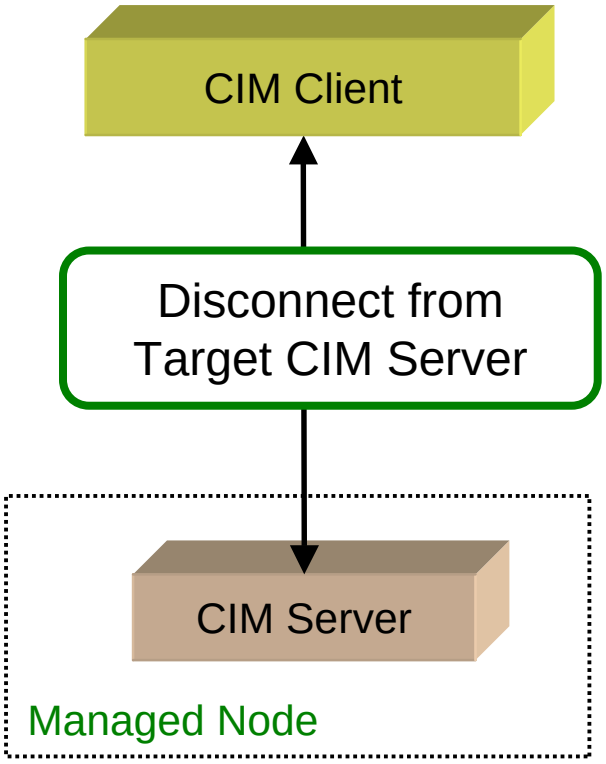


Client Application Logic





Disconnect to Target Server



- disconnect

```
client.disconnect();
```

ConnectExample.cpp - Notepad

File Edit Format Help

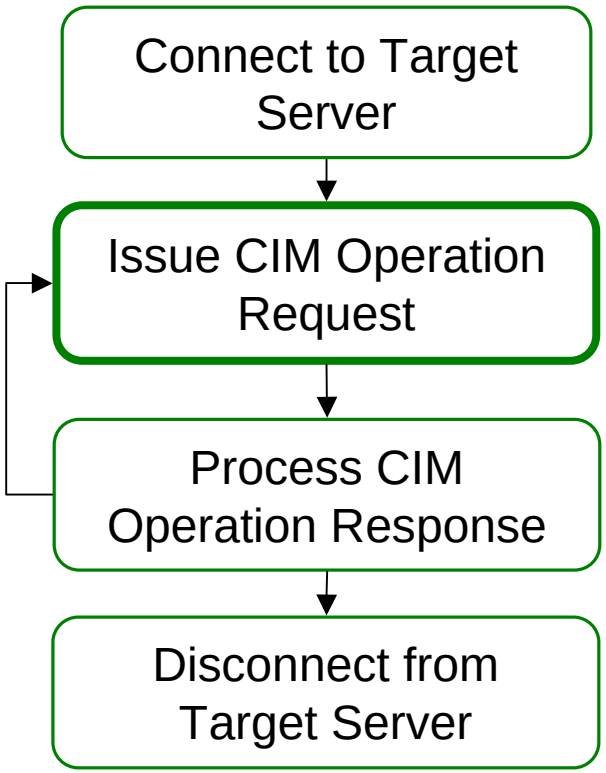
```
#include <Pegasus/Client/CIMClient.h>

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

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

        //
        // 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,
                                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;
}
```

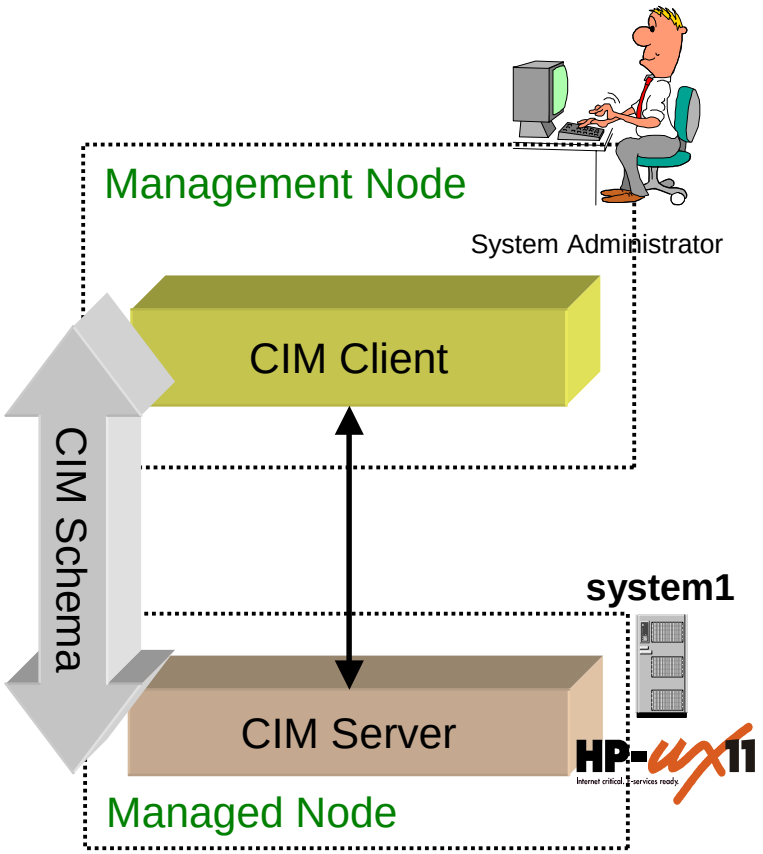
CIMClient client;

client.connectLocal();

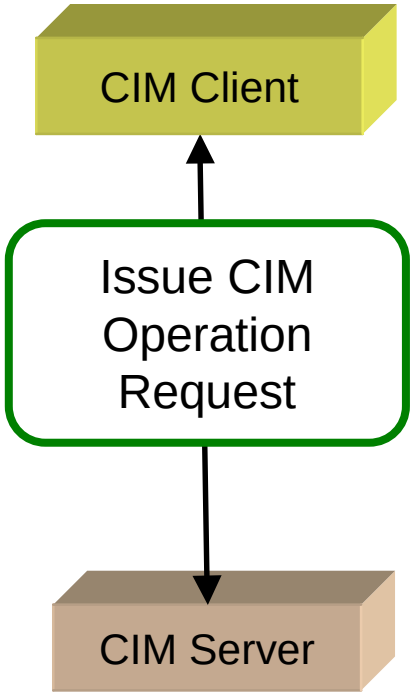
client.disconnect();

Client Application Logic





Issue CIM Operation Request



CIM Operation

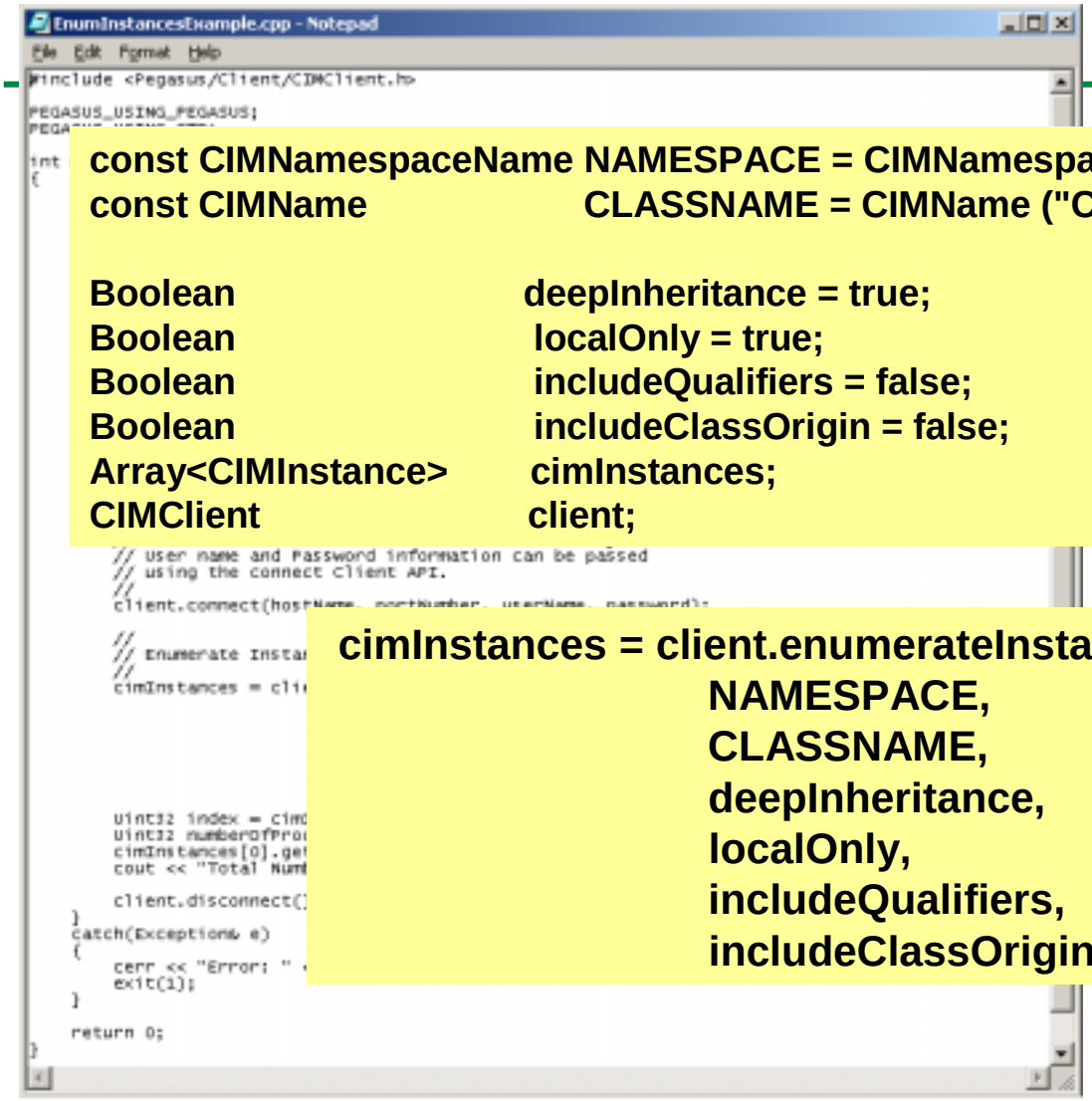
EnumerateInstances

```
<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
 )
```

OpenPegasus C++ Client API

enumerateInstances

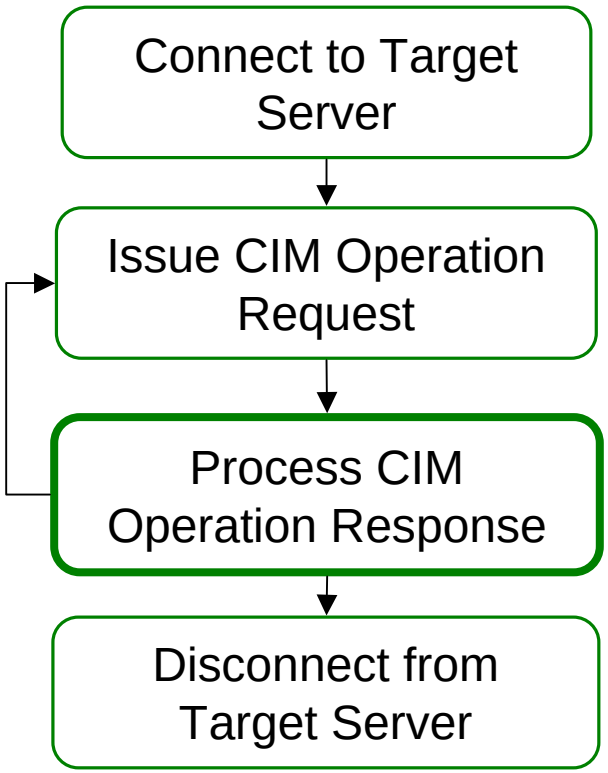
```
 Array<CIMInstance> enumerateInstances (
        const CIMNamespaceName& nameSpace,
        const CIMName& className,
        Boolean deepInheritance = true,
        Boolean localOnly = true,
        Boolean includeQualifiers = false,
        Boolean includeClassOrigin = false,
        const CIMPropertyList&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

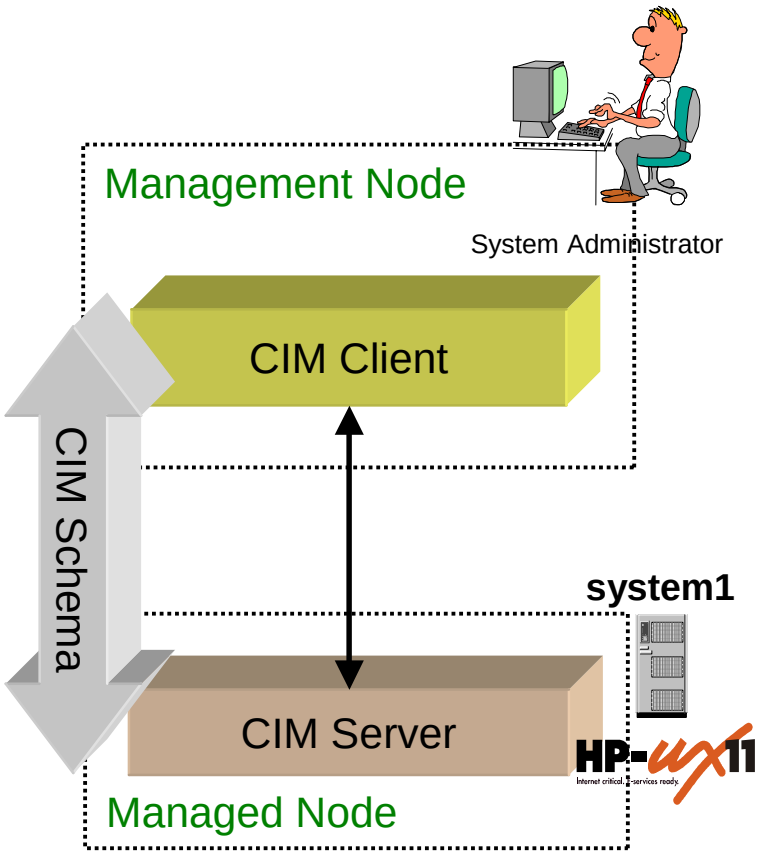


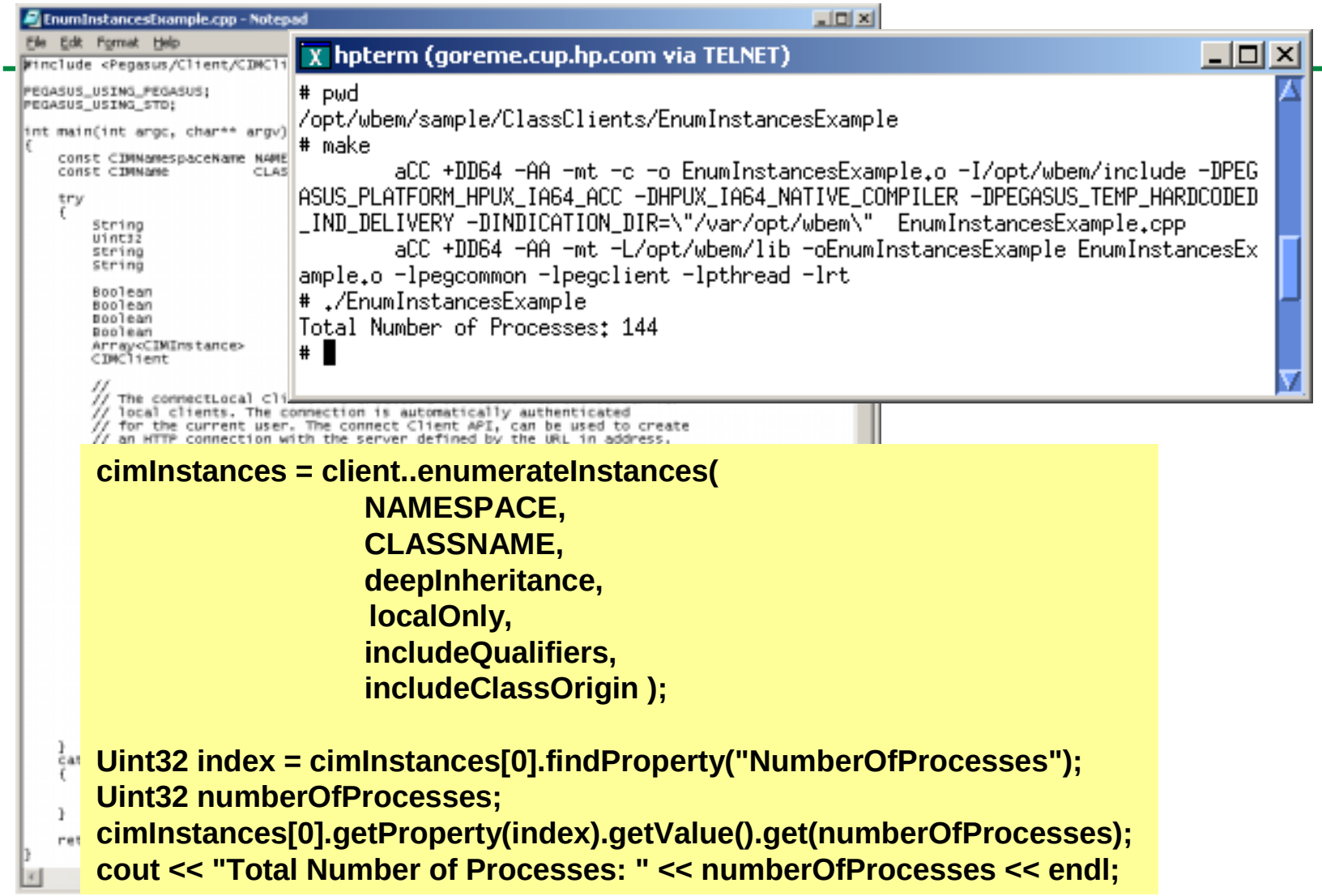

```
const CIMNamespaceName NAMESPACE = CIMNamespaceName ("root/cimv2");
const CIMName          CLASSNAME = CIMName ("CIM_OperatingSystem");

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








```
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;
```

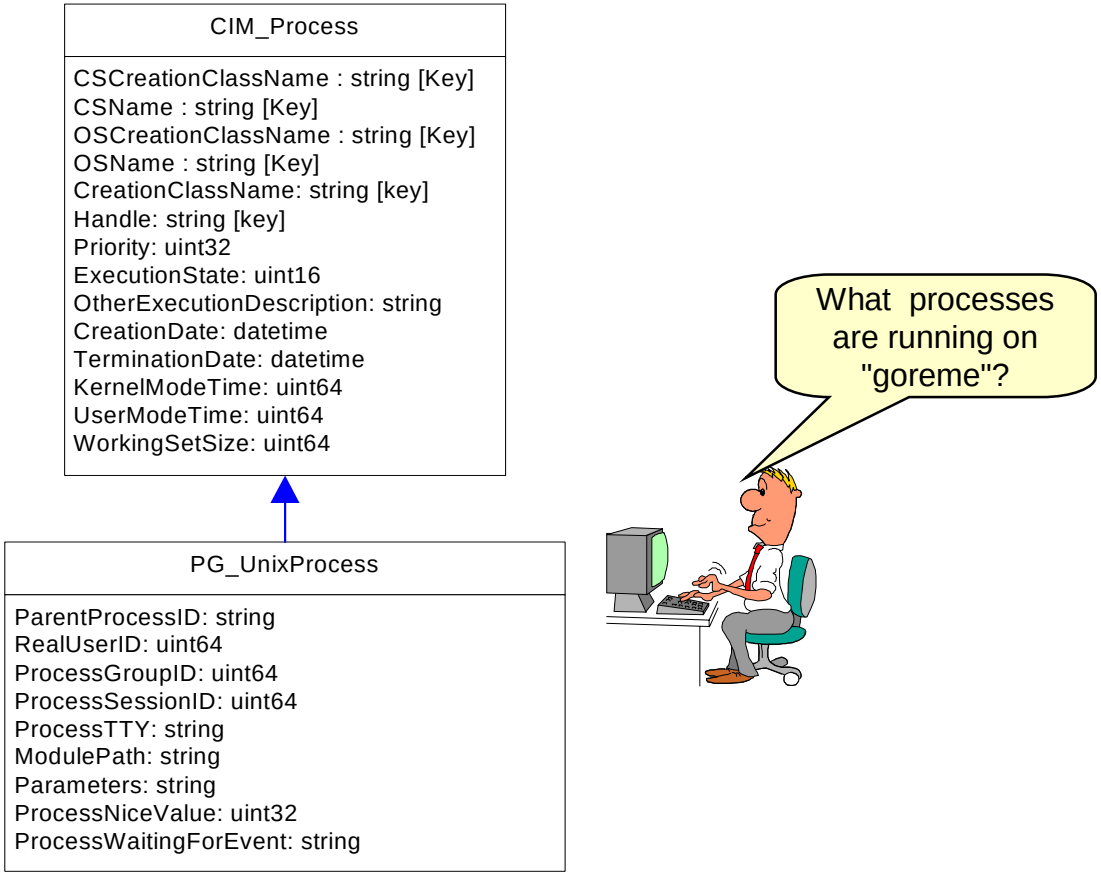
Module Content

C++ Client Overview

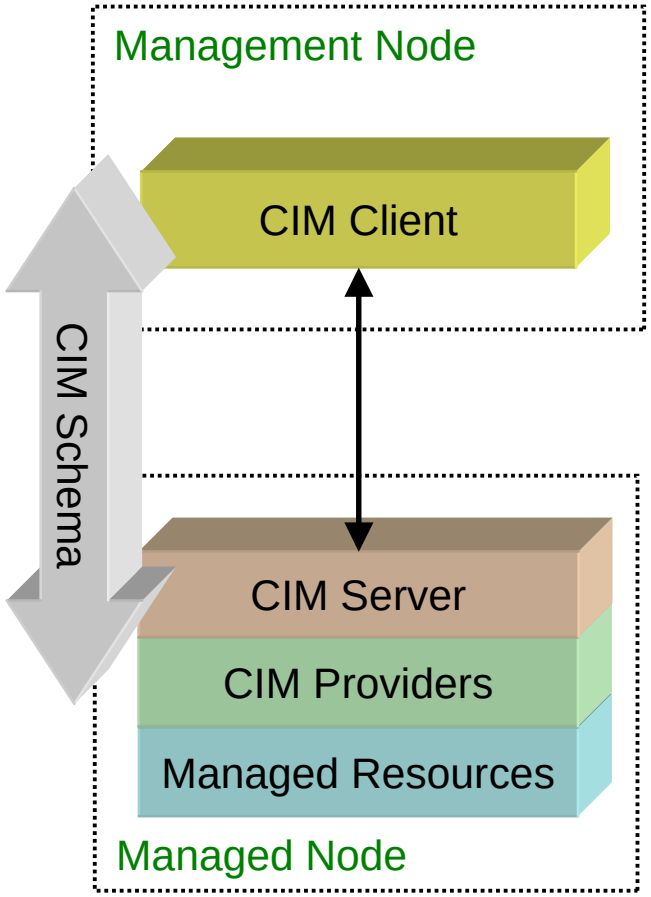
- Concept Overview
- Client Example
- **Client API**

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

EnumerateInstanceNames



Process Info Client Example



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

Return the InstanceName of each process running on goreme.

Process InstanceName

EnumerateInstanceNames

The **EnumerateInstanceNames** operation is used to retrieve the names of the instances of a class in a namespace.

CIM Operation

CIM Operation

EnumerateInstanceNames

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

OpenPegasus C++ Client API

enumerateInstanceNames

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

EnumerateInstanceNames

Encoded Using CIM-XML

EnumerateInstanceNames		
Parameter		Value
Type	Name	
Input	Namespace	root/cimv2
Input	ClassName	CIM_Process

```
<?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>
```

Operation

Namespace

Class of Interest

EnumInstanceNamesExample.cpp - Notepad

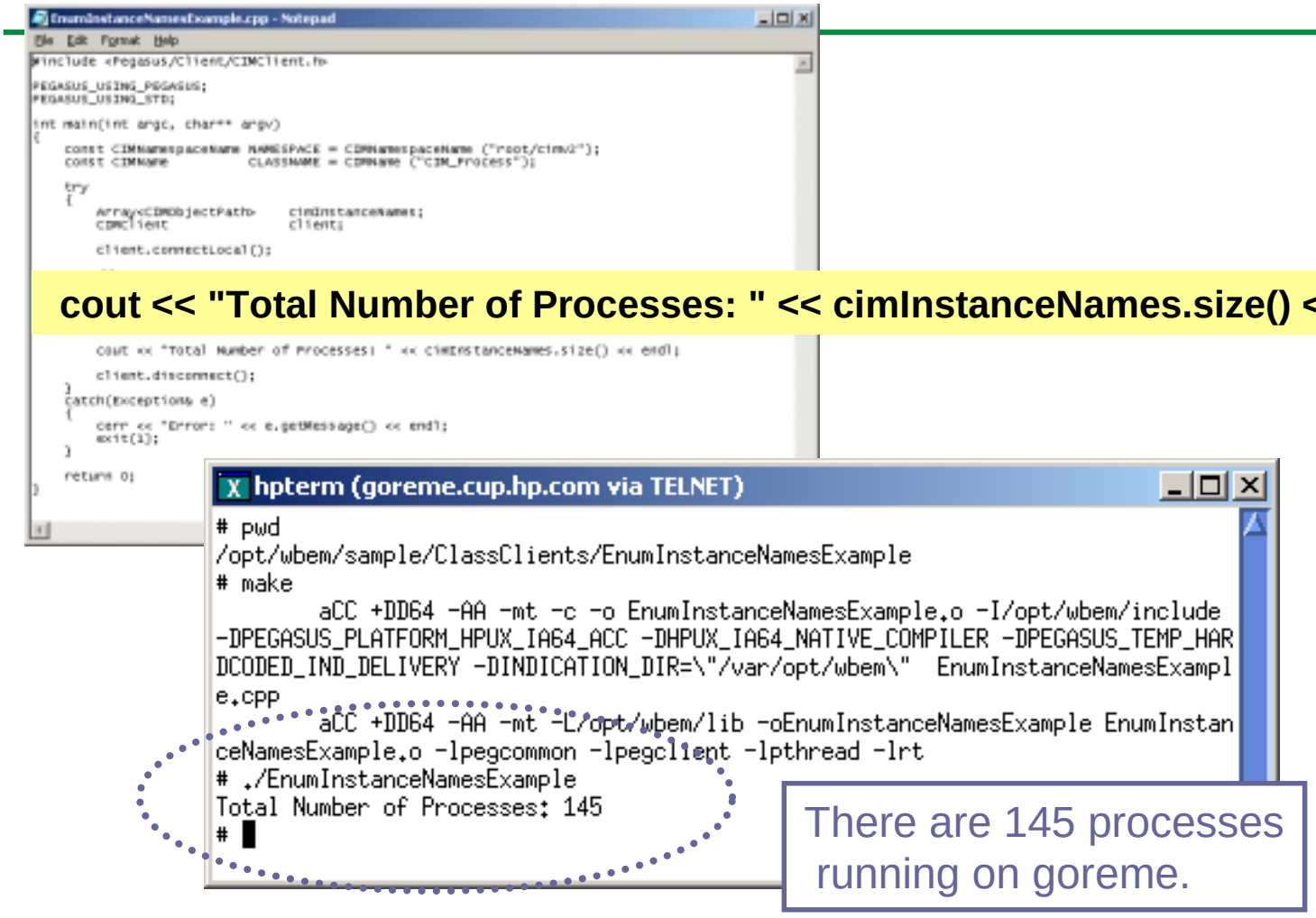
File Edit Format Help

```
#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>    c
        CIMClient               client;

        client.connectLocal();

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

        cout << "Total Number of Processes: " << cimInstanceNames.size() << endl;

        client.disconnect();
    }
    catch(Exception& e)
    {
        cerr << "Error:
        exit(1);
    }

    return 0;
}
```

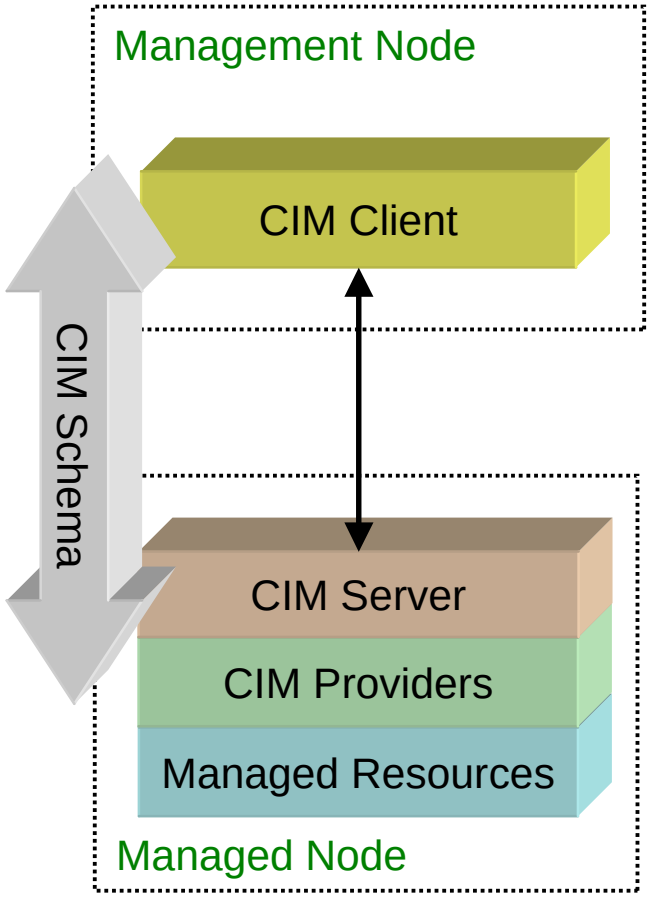
NAMESPACE = CIMNameSpaceName ("root/cimv2");
CLASSNAME = CIMName ("CIM_Process");

Array<CIMObjectPath> cimInstanceNames;

cimInstanceNames = client.enumerateInstanceNames (NAMESPACE, CLASSNAME);



Process Info Client Example



Now let's display the Instance Names.

Return the InstanceName of each process running on goreme.

Process InstanceName

DisplayInstanceNamesExample.cpp - Notepad

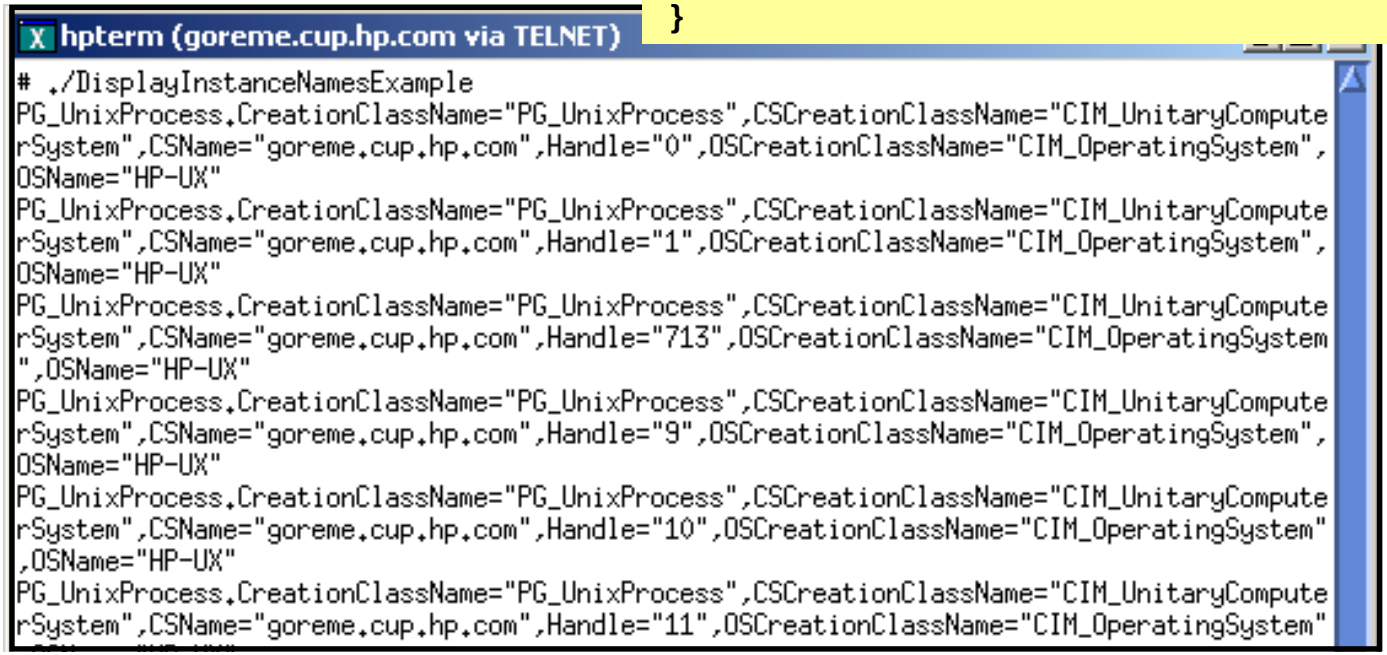
File Edit Format Help

```
#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();

        //
        // enumerateInstanceN
        //
        cimInstanceNames = cl

        for (Uint32 i = 0; i
        {
            cout << cimInstanc
        }

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

    return 0;
}
```

```
cimInstanceNames = client.enumerateInstanceNames(
                                NAMESPACE, CLASSNAME);

for (Uint32 i = 0; i < cimInstanceNames.size(); i++)
{
    cout << cimInstanceNames[i].toString() << endl;
 }
```

EnumerateInstanceNames

```
cimInstanceNames = client.enumerateInstanceNames(
                        NAMESPACE, CLASSNAME);

for (Uint32 i = 0; i < cimInstanceNames.size(); i++)
{
    cout << cimInstanceNames[i].toString() << endl;
 }
```

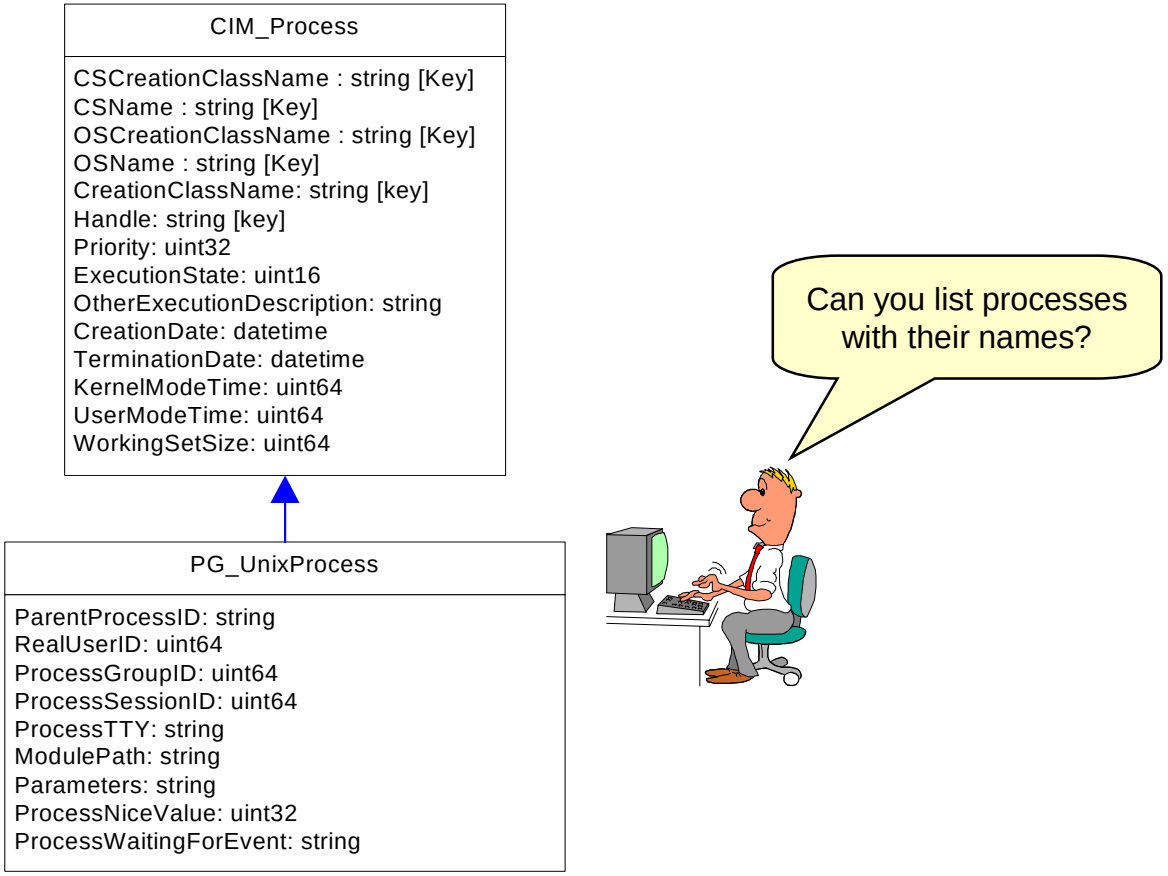


EnumerateInstanceNames

EnumerateInstanceNames (Input Values)		
Parameter		Value
Type	Name	
Input	Namespace	root/cimv2
Input	ClassName	CIM_Process

EnumerateInstanceNames (Return Values)					
<instanceName>*					PID
CSCreationClassName	CSName	OSCreationClassName	OSName	CreationClassName	Handle
CIM_UnitaryComputerSystem	goreme..	CIM_OperatingSystem	HP-UX	PG_UnixProcess	0
CIM_UnitaryComputerSystem	goreme..	CIM_OperatingSystem	HP-UX	PG_UnixProcess	1
CIM_UnitaryComputerSystem	goreme..	CIM_OperatingSystem	HP-UX	PG_UnixProcess	713
CIM_UnitaryComputerSystem	goreme..	CIM_OperatingSystem	HP-UX	PG_UnixProcess	9
CIM_UnitaryComputerSystem	goreme..	CIM_OperatingSystem	HP-UX	PG_UnixProcess	10
...					

EnumerateInstances Example



EnumerateInstances

CIM Operation

EnumerateInstances

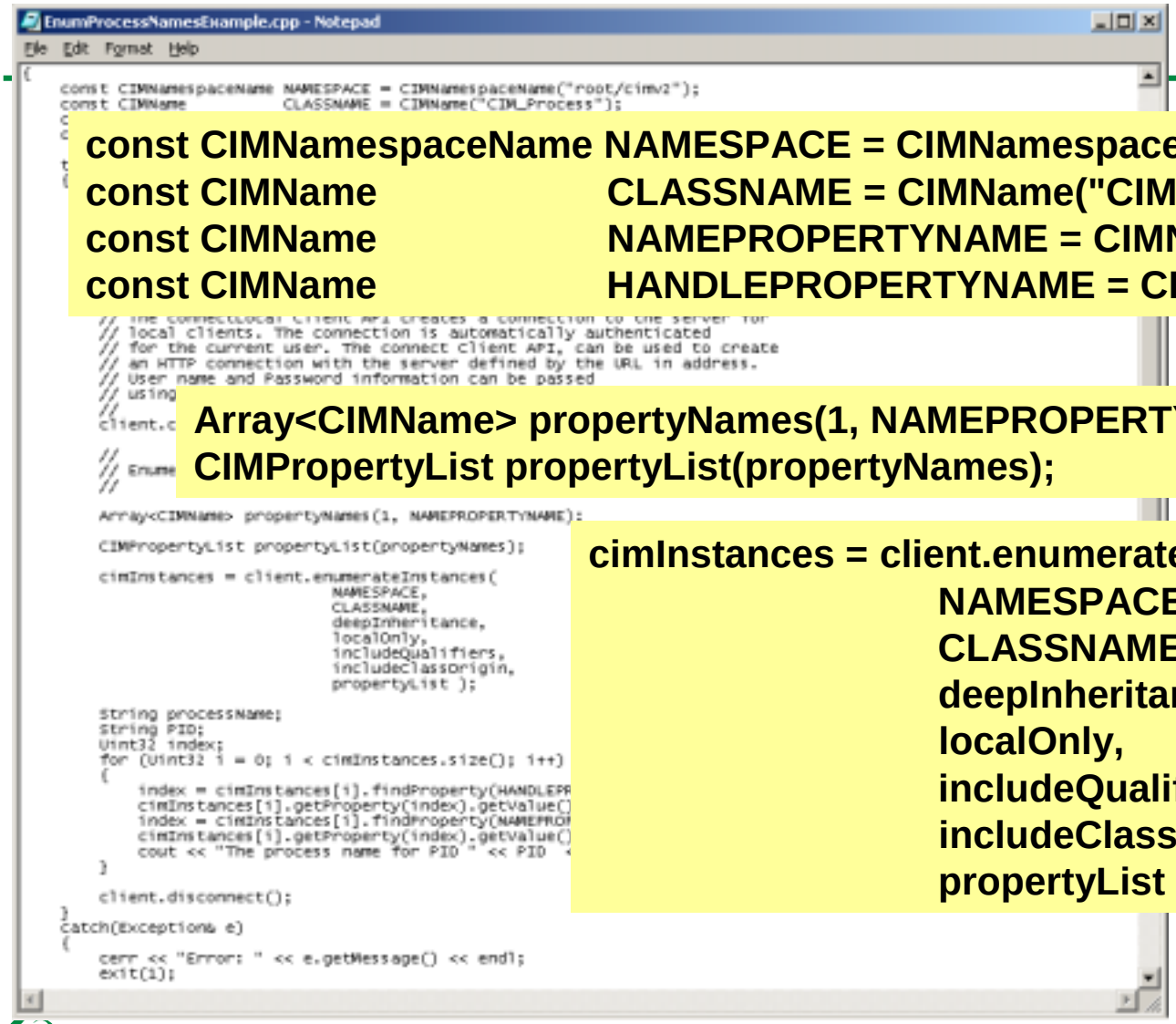
```
<namedInstance>*EnumerateInstances (
        [IN] <className> ClassName,
        [IN,OPTIONAL] boolean LocalOnly = true,
        [IN,OPTIONAL] boolean DeepInheritance = true,
        [IN,OPTIONAL] boolean IncludeQualifiers = false,
        [IN,OPTIONAL] boolean IncludeClassOrigin = false,
      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

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

OpenPegasus C++ Client API

enumerateInstances

```
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()
 );
```



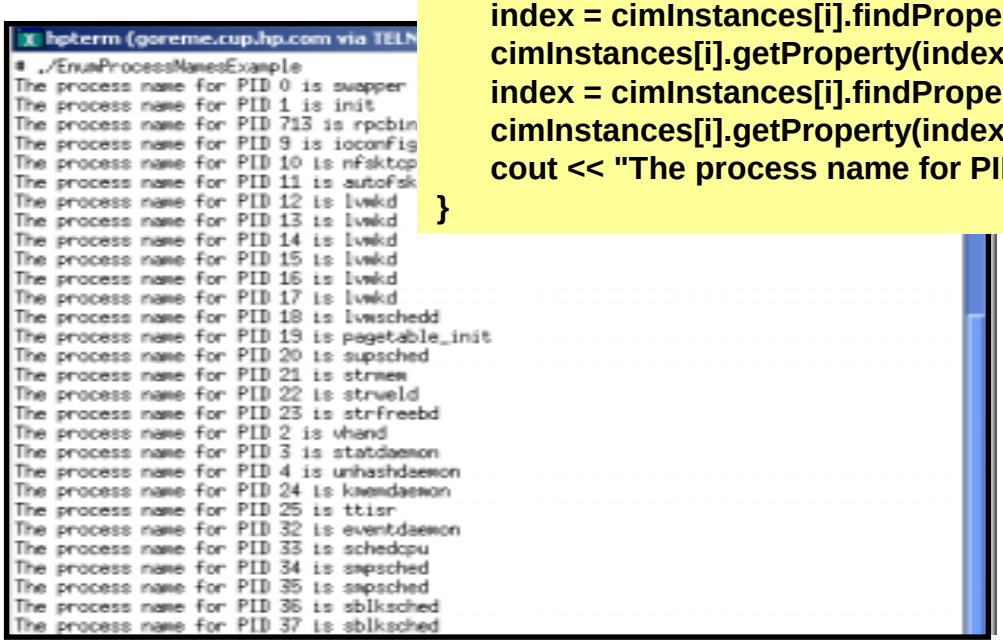

```
const CIMNamespaceName NAMESPACE = CIMNamespaceName("root/cimv2");
const CIMName          CLASSNAME = CIMName("CIM_Process");
const CIMName          NAMEPROPERTYNAME = CIMName("Name");
const CIMName          HANDLEPROPERTYNAME = CIMName("Handle");
```

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

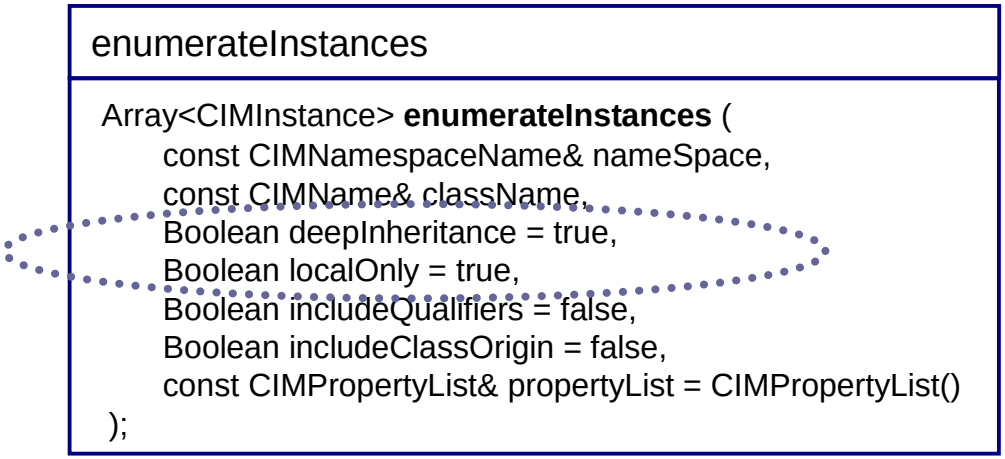
```
cimInstances = client.enumerateInstances(
                        NAMESPACE,
                        CLASSNAME,
                        deepInheritance,
                        localOnly,
                        includeQualifiers,
                        includeClassOrigin,
                     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

EnumerateInstances

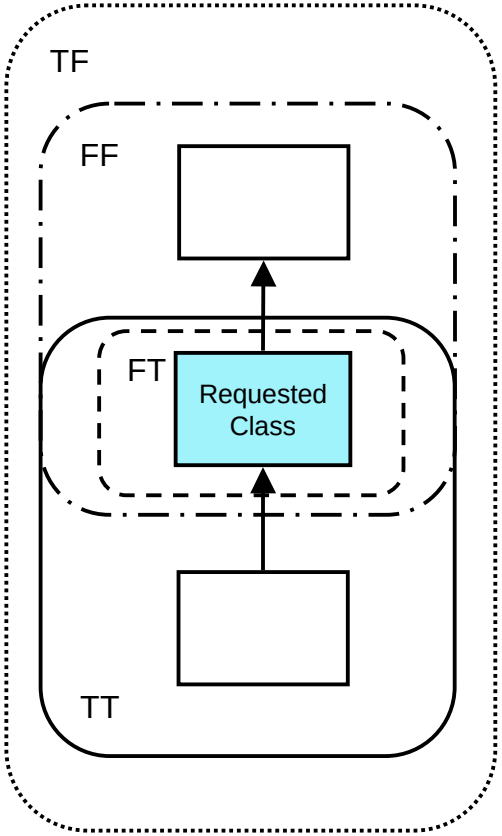
```
String processName;
String PID;
Uint32 index;
for (Uint32 i = 0; i < cimInstances.size(); i++)
{
    index = cimInstances[i].findProperty(HANDLEPROPERTYNAME);
    cimInstances[i].getProperty(index).getValue().get(PID);
    index = cimInstances[i].findProperty(NAMEPROPERTYNAME);
    cimInstances[i].getProperty(index).getValue().get(processName);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



DeepInheritance & LocalOnly

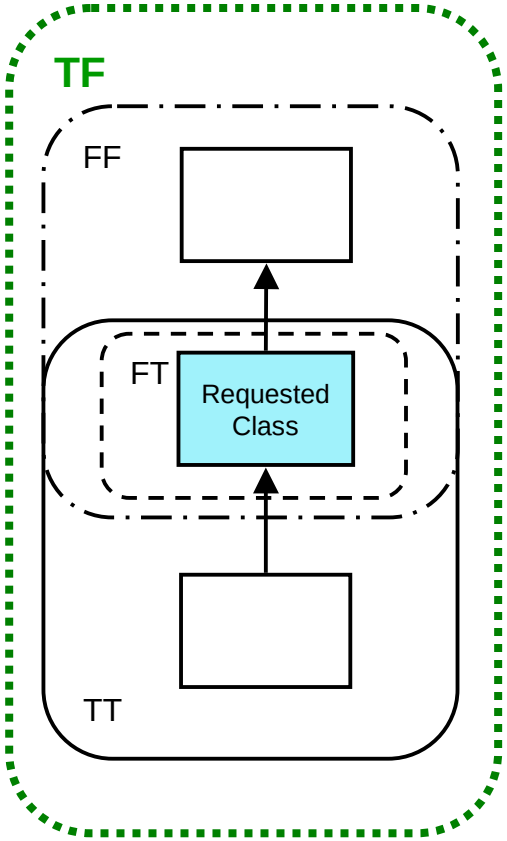


Deep Inheritance and Local Only Flags



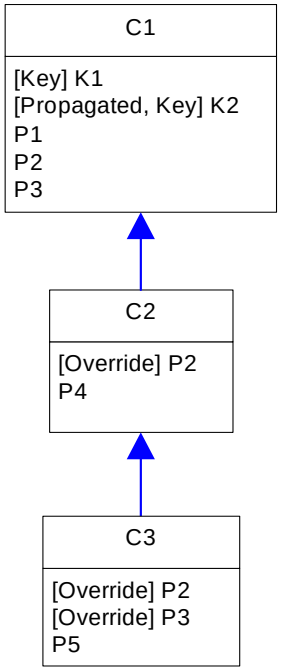
Deep	Local	Use Model	Description
T	F	"Tell me everything you know"	The elements for all defined Properties are returned. This includes elements for Properties explicitly defined in the designated class, elements for inherited Properties, and elements for any Properties defined in subclasses of the designated class.
T	T	"I'm not interested in seeing inherited properties."	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.
F	T	"I'm only interested in the properties explicitly defined in the class I specified."	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 includes elements for Properties explicitly defined in the designated class.
F	F	"I'm not interested in any properties defined in the subclasses."	Only elements for Properties defined in the designated class or one 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.

Deep Inheritance and Local Only



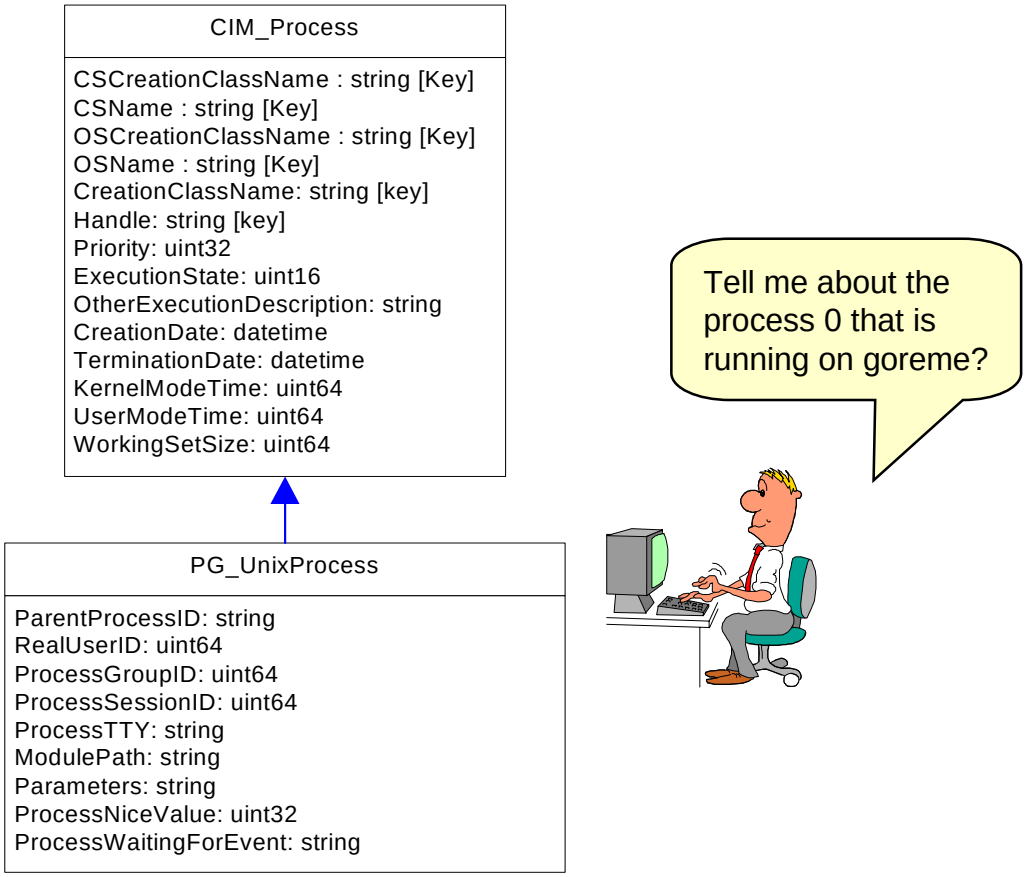
Key Fact: This release of the product supports only the values of true for DeepInheritance and false for LocalOnly (i.e., refer to TF in the diagram),

Deep Inheritance and Local Only Flags



Deep	Local	Class	C1 Instances	C2 Instances	C3 Instances
T	T	C1	K1, K2, P1, C1.P2, C1.P3	K1, K2, P1, C2.P2, C1.P3, P4	K1, K2, P1, C3.P2, C3.P3, P4, P5
T	F	C1	K1, K2, P1, C1.P2, C1.P3	K1, K2, P1, C2.P2, C1.P3, P4	K1, K2, P1, C3.P2, C3.P3, P4, P5
F	T	C1	K1, K2, P1, C1.P2, C1.P3	K1, K2, P1, C2.P2, C1.P3	K1, K2, P1, C3.P2, C3.P3
F	F	C1	K1, K2, P1, C1.P2, C1.P3	K1, K2, P1, C2.P2, C1.P3	K1, K2, P1, C3.P2, C3.P3
T	T	C2		C2.P2, P4	C3.P2, C3.P3, P4, P5
T	F	C2		K1, K2, P1, C2.P2, C1.P3, P4	K1, K2, P1, C3.P2, C3.P3, P4, P5
F	T	C2		C2.P2, P4	C3.P2, P4
F	F	C2		K1, K2, P1, C2.P2, C1.P3, P4	K1, K2, P1, C3.P2, C3.P3, P4
T	T	C3			C3.P2, C3.P3, P5
T	F	C3			K1, K2, P1, C3.P2, C3.P3, P4, P5
F	T	C3			C3.P2, C3.P3, P5
F	F	C3			K1, K2, P1, C3.P2, C3.P3, P4, P5

GetInstance Example



GetInstance Example

GetInstance — **CIM Operation**

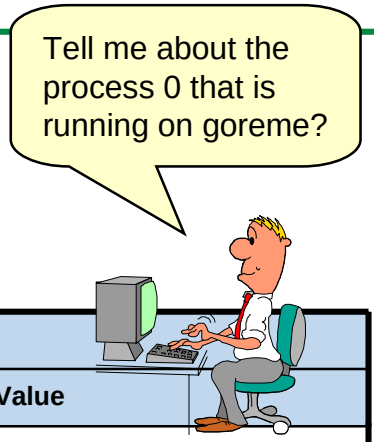
```
<instance> GetInstance (
        [IN] <instanceName> InstanceName,
        [IN,OPTIONAL] boolean LocalOnly = true,
        [IN,OPTIONAL] boolean IncludeQualifiers = false,
        [IN,OPTIONAL] boolean IncludeClassOrigin = false,
        [IN,OPTIONAL,NULL] string PropertyList [] = NULL
)
```

getInstance — **OpenPegasus C++ Client API**

```
CIMInstance getInstance (
      const CIMNamespaceName& nameSpace,
      const CIMObjectPath& instanceName,
      Boolean localOnly = true,
      Boolean includeQualifiers = false,
      Boolean includeClassOrigin = false,
      const CIMPropertyList& propertyList = CIMPropertyList()
 );
```

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

GetInstances Example



GetInstance (Input Values)		
Parameter		**Value**
Namespace		root/cimv2
InstanceName	CSCreationClassName	CIM_UnitaryComputerSystem
InstanceName	CSName	goreme.cup.hp.com
InstanceName	OSCreationClassName	CIM_OperatingSystem
InstanceName	OSName	HP-UX
InstanceName	CreationClassName	PG_UnixProcess
InstanceName	Handle	0
LocalOnly		false

PID = 0

GetInstance

```
<?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">
      </INSTANCENAME>
     </IPARAMVALUE>
    </IMETHODCALL>
   </SIMPLEREQ>
  </MESSAGE>
 </CIM>
```

Operation

Namespace

Class of Interest

Instance Name

```
<KEYBINDING NAME="CSCreationClassName">
 <KEYVALUE VALUETYPE="string">
  CIM_UnitaryComputerSystem
 </KEYVALUE> </KEYBINDING>
<KEYBINDING NAME="CSName">
 <KEYVALUE VALUETYPE="string">
   goreme.cup.hp.com
 </KEYVALUE> </KEYBINDING>
<KEYBINDING NAME="OSCreationClassName">
 <KEYVALUE VALUETYPE="string">
  CIM_OperatingSystem
 </KEYVALUE> </KEYBINDING>
<KEYBINDING NAME="OSName">
 <KEYVALUE VALUETYPE="string">
  HP-UX
 </KEYVALUE> </KEYBINDING>
<KEYBINDING NAME="CreationClassName">
 <KEYVALUE VALUETYPE="string">
  PG_UnixProcess
 </KEYVALUE> </KEYBINDING>
<KEYBINDING NAME="Handle">
 <KEYVALUE VALUETYPE="string">
  0
</KEYVALUE> </KEYBINDING>
```

GetInstance

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

PID →

pst_status from pstat_getproc() →

pst_start from pstat_getproc() →

pst_stime from pstat_getproc() →

pst_utime from pstat_getproc() →

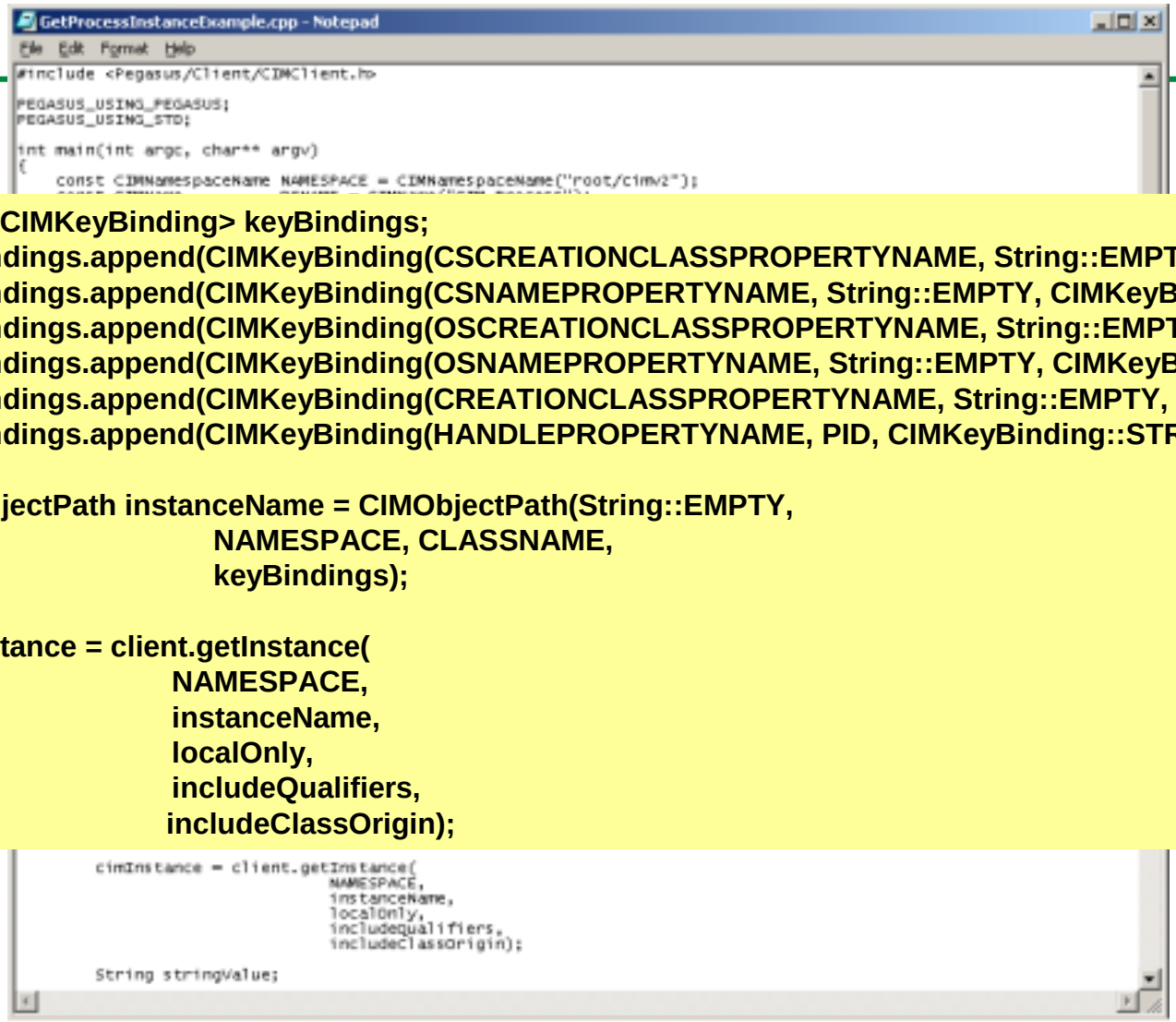
pst_ppid from pstat_getproc() →

pst_uid from pstat_getproc() →

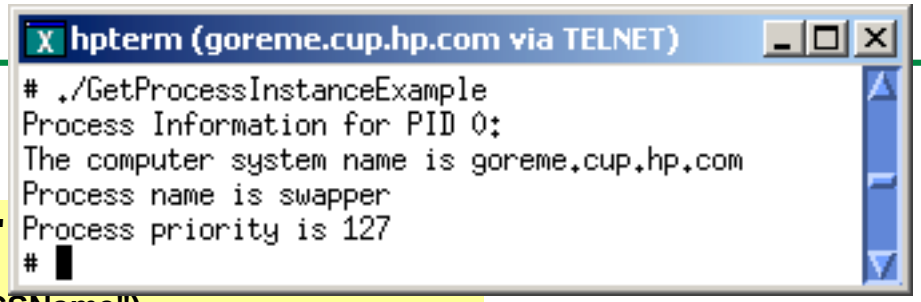
pst_gid from pstat_getproc() →

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, CIMKeyBinding::STRING));
keyBindings.append(CIMKeyBinding(OSCREATIONCLASSPROPERTYNAME, String::EMPTY, CIMKeyBinding::STRING));
keyBindings.append(CIMKeyBinding(OSNAMEPROPERTYNAME, String::EMPTY, CIMKeyBinding::STRING));
keyBindings.append(CIMKeyBinding(CREATIONCLASSPROPERTYNAME, 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;
    }
```