Application Programming Interfaces for the Virtual Grid Execution System (vgES)
Java Language
Version 0.7
February 15, 2005
Andrew Chien, Henri Casanova, Yang-suk Kee, Ken Yocum,
Richard Huang, Jing Zhu, and Dionysios Logothetis
vgrads@cs.ucsd.edu

The Virtual Grid Execution System provides a simple programming interface thru three major classes, VGES (the overall execution system), VG (a virtual grid instance), and VGNode (a virtual grid node within a virtual grid). The current system exposes a Java language application programming interface, as described in this document. A C language programming interface has been designed, and is planned to be supported in a future release of the vgES system.

We use a simple notation to indicate the implemented, defined, and corresponding C API calls for the Java interface. In particular, the **implemented Java methods** are indicated in bold. The *future Java methods* are indicated in italics. And the corresponding C API calls are indicated in light blue, and marked as program comments. For example,

VGNode getChild (int idx) // VGnode vgChild (VgNode node, int idx)
VGNode getGrandChild (int idx, int idx2)

Indicates the getChild method, the corresponding method in the C API, and that the getGrandChild method is planned for future implementation (it's not!).

1. Basic Datatypes

These datatypes are use for status, handles, etc. in vgES programs.

A. Class VGStatus – bit vector return value for which 0 is success

```
public class VGStatus
{
    // status values for VGNode and VG
    static final int VG_UNDEFINED
    static final int VG_NOT_FOUND
    static final int VG_FOUND
    static final int VG_PARTIAL_FOUND
    static final int VG_PARTIAL_BOUND
    static final int VG_FOUND_BOUND

    static final int VG_EMPTY
    static final int VG_TIMEOUT
    static final int VG_FAB_FAILURE

    // status values for VGCmdHandle
    static final int VG_RUNNING
```

```
    static final int VG_SLEEPING
    static final int VG_STOPPED
    static final int VG_WAIT
    static final int VG_FAILED
    static final int VG_SUCCESS
    static final int VG_INVOKING
    static final int VG_INVOKING_ERROR
    static final int VG_UNRECOGNIZED

    int get ();
}
```

A.1. Core Interface

- **int get ()** // No correspondence in C API;
  Returns the value of status.

B. Class VGCmdHandle – reference to an active command on a VGNode or VG

```
public class VGCmdHandle
{
    boolean done ();
    boolean terminate ();
    int getStatus ();
}
```

B.1. Core Interface

- **boolean done ()** // VGResult vgCmdDone (VGCmdHandle cmdHandle);
  Returns true if the command has completed, false otherwise.
- **boolean terminate ()** // VGResult vgTerminateCmd(VGCmdHandle cmdHandle);
  Terminates a command on all nodes. The resulting state from partially completed commands is undefined.
- **int getStatus ()** // No correspondence in C API;
  Gets the running status of processes associated with this command handle.

2. Major Classes

A. Class VGES

```
public class VGES
{
    VG createVG (String server_name, String description, int timeout, String script);
    boolean terminateVG (VG vg);
    VGCmdHandle runCmd (VGNode node, String command);
    VGCmdHandle runCmd (VG vg, String command);
    boolean copyToNode (VGNode node, String src, String dst);
    boolean copyFromNode (VGNode node, String src, String dst);
    VGNode getMyNode ();
    VG getMyVG ();
    String [][] readInterAttribute (VGNode node1, VGNode node2, String attribute);
    VG findVG (String server_name, String description, int timeout);
    VGStatus bindVG (VG vg, int timeout, String script);
    boolean setCallback (VGMonCallback callback);
```

}

A.1. Core VGES Interface

- **VG createVG (String server_name, String description, int timeout, String script)**
  // VGID vgCreate (ADDRESS server_address, VGDL desc, tsec timeout, char* script, VGCmdHandle *cmdHandle)
  User (Client) contacts the virtual grid execution system (vgES) designated by server_address[1] and requests creation of a Virtual Grid (VG). The vgES selects, binds, and then launches the application script on the newly created VG's resources. It also creates a vgMON which monitors the VG. The VGID returned is a handle for the VG. This routine is synchronous and can be called with a timeout. If the call times out, a partially instantiated VG may be returned. In all cases, users can check if resource selection and binding have succeeded by invoking the vgGetStatus() routine. The cmdHandle provides users the capability of status and job control for the running application scripts.
- **boolean terminateVG (VG vg)** // VGResult vgTerminate (VGID vg)
  User (client) requests termination of the VG instance. The vgES terminates the application and the monitor, unbinds the resources, and destroys the VG instance. Any subsequent calls made against this VGID will return undefined results.
- **VGCmdHandle runCmd (VGNode node, String command)**
  // VGResult vgRunCmdOnNode (VGNode node, char* cmd, VGCmdHandle *cmdHandle)
  Runs the specified command on the specified node and returns a handle to the command. If the node is an aggregator (i.e. ClusterOf() or BagOf), then it runs the command on each of the members of that aggregate.
- **VGCmdHandle runCmd (VG vg, String command)**
  // VGResult vgRunCmd (VGID VG, char* cmd, VGCmdHandle *cmdHandle)
  Runs a command on all nodes in the VG and returns a command handle.
- **boolean copyToNode (VGNode node, String src, String dst)**
  // VGResult vgCopyToNode (VGNode node, char* srcFname, char* destFname)
  Copies a local file with qualified path name srcFname to the specified node, and saves the new copy with qualified path name destFname. No specific mechanism is implied.
- **boolean copyFromNode (VGNode node, String src, String dst)**
  // VGResult vgCopyFromNode (VGNode node, char* srcFname, char* destFname)
  Copies a file with qualified path name srfFname from the node, and saves a local copy with qualified path name destFname. No specific mechanism is implied.
- *VGNode getMyNode ()* // VGNode vgGetMyNode()
  Returns a VGNode reference for the VG resource on which the function was called. This function allows an application convenient access to its virtual grid state.
- *VG getMyVG ()* // VGID vgGetMyVG()
  Returns the vgID (virtual grid reference) that corresponds to the resource on which the function was called. This function allows an application convenient access to its virtual grid state.
- **String [][] readInterAttribute (VGNode node1, VGNode node2, String attribute)**
  // No correspondence in C APIs;
  Returns the value associated with the attribute between two VGNodes.

A.2. Expanded Interface (Dynamic VG's, Fault-tolerance, Regeneration)

- *VG findVG (String server_name, String description, int timeout)*

---

[1] The vgES address may correspond to the actual machine running vgES, or in some cases only the attribute server.

// VGID vgFind (ADDRESS server_address, VGDL desc, tsec timeout)

Selects one set of resources appropriate for the vgDL description and returns a virtual grid which is decorated with this set of unbound resources. No guarantee is made that these resources are bindable.

- *VGStatus bindVG (VG vg, int timeout, String script)*
  // VGStatus vgBind (VGID vg, tsec timeout, char* script)
  Bind the resources in the virtual grid that are not already bound. This function applies this hierarchically to the entire virtual grid.
- *boolean setCallback (VGMonCallback callback)*
  // VGResult vgMonSetCallback (VGID vg, vgMonCallback callback)
  Sets the callback function for vgMON. This function will be called by vgMON to notify the application when relevant resource changes occur. For example, vgMON will call this function when it detects a violation of the vgDL description.

B. Class VG – a virtual grid instance

```
public class VG
{
    VGStatus getStatus ()
    VGNode getRoot ()
    VGStatus addNode (String description, VGNode parent, int timeout, String script)
    VGStatus addNode (VGNode node, VGNode parent)
    boolean removeNode (VGNode node, boolean savevgDL)
    String getDesc ()
}
```

B.1. Core VG Interfaces

- **VGStatus getStatus ()** // VGStatus vgGetStatus(VGID vg);
  Returns the status of VG. The status can be parsed as indicated in Section 1.
- **VGNode getRoot ()** // VGNode vgRoot (VGID vg);
  Returns the root node of the VG. VGNode contains dynamic and static information about the resource.

B.2. Expanded Interface (Dynamic VG's, Fault-tolerance, Regeneration)

- *VGStatus addNode (String description, VGNode parent, int timeout, String script)*
  // VGStatus vgAddtoVG (VGID vg, VGDL desc, VGNode parentNode, tsec timeout, char* script);
  This command uses the vgDL description to find, bind, launch and extend vgMON (all in analogy to vgCreate()), and then adds them under the parentNode
- *VGStatus addNode (VGNode node, VGNode parent)*
  // VGStatus vgAddNodetoVG (VGID vg, VGNode addNode, VGNode parentNode);
  This command adds the addNode under the parentNode. No automatic application launching or monitor adjustment is included.
- *boolean removeNode (VGNode node, boolean savevgDL)*
  // VGResult vgRemoveNodeVG(VGID vg, VGNode removeNode, vgFlag savevgDL) ;
  This command kills the application and monitoring and then unbinds the resources for the VGNode and all its children, and then deletes this node and all of its children.
- *String getDesc ()*
  // VGDL vgGetDesc (VGID vg);
  This routine returns vgDL that corresponds to the Virtual Grid. For merged or modified Virtual Grids, the vgDL may not be clearly specified. The implementation will generate a "reasonable" vgDL based on the original vgDL used to create the virtual grid(s) and the modification calls made.

C. Class VGNode – a node in the virtual grid instance, typically corresponding to a single resource

```
public class VGNode
{
    int getType ()
    VGNode getParent ()
    VGNode getChild (int idx)
    int getNumChildren ()
    String[] getAttributes ()
    String readAttribute (String attribute)
    boolean writeAttribute (String attribute, String value)
    String[] getInterAttributes();
    String[][] readInterAttribute (String attribute)
    boolean writeInterAttribute (String attribute, String[][] value)
    String getDesc ()
    VGStatus getStatus ()
}
```

C.1. Core VGNode Interface

- **int getType ()** // No correspondence in C APIs;
  Returns the type of this VGNode.
- **VGNode getParent ()** // VGNode vgParent (VGNode node);
  Returns the parent of this VGNode.
- **VGNode getChild (int idx)** // VGNode vgChild (VGNode node, int idx);
  Returns the idx-th child of this VGNode.
- **int getNumChildren ()** // int vgNumChildren (VGNode node);
  Returns the number of children of this VGNode.
- **String[] getAttributes ()** // char** vgGetAttributes (VGNode node);
  Return the current set of attributes describing this VGNode as a null-terminated array of null-terminated strings.
- **String readAttribute (String attribute)** // char* vgReadAttribute (VGNode node, char* AttribName);
  Returns the value associated with the attribute for this VGNode.
- **boolean writeAttribute (String attribute, String value)**
  // VGResult vgWriteAttribute (VGNode node, char* AttribName, char* AttribValue);
  Writes the value associated with the attribute for this VGNode.
- **String[] getInterAttributes()**
  Returns the current set of InterAttributes defined on this node as an array of strings.
- **String[][] readInterAttribute (String attribute)** // No correspondence in C APIs;
  Returns the value associated with the inter-node attribute for this VGNode.
- **boolean writeInterAttribute (String attribute, String[][] value)**
  Writes the array of values as the definition of the InterAttribute for this VGNode.
- **VGStatus getStatus ()** // No correspondence in C APIs;
  Returns the status of this VGNode.

C.2. Expanded Interface (Dynamic VG's, Fault-tolerance, Regeneration)

- *public String getDesc()* // VGDL vgGetDescNode (VGNode node);
  This routine returns the current vgDL for this node and its children.

3. Summary

The current implementation includes the static elements of the Virtual grid system, encompassing selection, binding, and the static and dynamic information systems. Major extensions in the future will include the following areas:

- Dynamic Virtual Grids: change, evolution, modification to support both Fault tolerance and application adaptation,
- vgDL and explicit expectation-based monitoring of Virtual grids,
- Virtual grid to vgDL conversion, enabling VG to provide a full of application resource management from specification, allocation, and evolution back to a captured description which can be stored and used to begin the process again.
- Advanced implementations of the vgES components and capabilities, particularly vgFAB, vgMON, etc.