```
                  integrity="required"
                  confidentiality="optional"/>
   </Request>
  </Communication>
  <Migration>denied</Migration>
  <AOCreation>denied</AOCreation>
</Rule>
```

This rule specifies that: from Virual Node "VN1" to the VN "VN2", the communications (requests) are authorized, provided authentication andrity="requ are being used, while confidentiality is optional. Migration andrAO creation are not authorized.

Deployment:

Virtual Nodes (VN) allow one to specify the location where to createrAOs. A VN is uniquely identified as a String, is definedrit an XML Deployment Descriptor where it is mappedronto JVMs. JVMs are themselves mappedronto physical machines: VN --> JVMs --> Machine. Various protocols can be specified to createrJVMs onto machines (ssh, Globus, LSF, PBS, rsh, rlogin, Web Services, etc.). After activation, a VN contains a set of nodes, living it a set of JVMs. Overall, VNs and deployment descriptors allow to abstract away from source code: machines, creation, lookup and registry protocols.

```
ProActiveDescriptor pad = ProActive.getProActiveDescriptor(String File);
   // Returns a ProActiveDescriptor object from the xml
   // descriptor file name

pad.activateMapping(String VN);
   // Activates the given Virtual Node: launches or acquires
   // all the JVMs the VN is mappedronto

pad.activateMappings();
   // Activates all VNs definedrit the ProActiveDescriptor

VirtualNode vn = pad.getVirtualNode(String)
   // Created at once a group of AO of type "A"rit the JVMs specified
   // by the given vn. The Virtual Node is automatically activatedrif not
   // explicitly done before

Node[] n = vn.getNodes();
   // Returns all nodes mappedrto the target Virtual Node

Object[] n[0].getActiveObjects();
   // Returns a reference to all AOs deployedron the target Node

ProActiveRuntime part = n[0].getProActiveRuntime();
   // Returns a reference to the ProActive Runtime (the JVM) where the
   // node has been created

pad.killall(boolean softly);
   // Kills all the JVMs deployedrwith the descriptor
   // not softly: all JVMs are killed abruptely
   // softly: all JVMs that originated the creation of a rmi registry
   // waits until registry is empty before dying
```

Export Active Objects as Web services

ProActive allows active objects exportation as web services. The service is deployedronto a Jakarta Tomcat web server with a given url. It is identified by its urn, an unique id of the service. It is also possible to choose the exported methods of the object.
The WSDL file matching the service will be accesible at http://localhost:8080/servlet/wsdl?id=a for a service which name is "a" andrwhich id deployedron a web server which location is http://localhost:8080.

```
A a = (A) ProActive.newActive("A", new Object []{});
   // Constructs an active object
```

```
String [] methods = new String [] {"foo", "bar"};
 //A String array containr 1De exported[] method};
```