# ETSI TS 102 828 V2.1.1 (2010-03)

*Technical Specification*

**GRID;**
**Grid Component Model (GCM);**
**GCM Application Description**

Reference
DTS/GRID-0004-2[2]

Keywords
application, architecture, interoperability, network, service

# Contents

## Intellectual Property Rights

## Foreword

This Technical Specification (TS) has been produced by ETSI Technical Committee GRID (GRID).

The present document is related to document TS 102 827 [1] (GCM Interoperability Deployment).

## Introduction

The purpose of the present document is to offer a standard, uniform way of describing grid-based applications, and the resources they need.

When building a grid, one is confronted with several problems, among which:

- choosing a grid framework, as many are available;
- when running a grid-enabled application, one must rewrite configuration directives when switching to a different grid framework;
- a grid can be built on widely different hardware infrastructures, from an heterogeneous set of desktop machines to a dedicated cluster, or even any combination of those;
- grid infrastructures can also be very dynamic, it should be easy to add or remove machines, and such changes should be transparent to users.

To simplify this, the GCM Interoperability Deployment standard (TS 102 827 [1]) offers a uniform way to describe grid resources. On the application side, the present document describes how various resources can be used by an application, providing gridification of applications in a very portable manner.

# 1 Scope

The present document describes an XML schema for describing application constraints and resources to be used for deploying these applications on distributed and parallel infrastructures, like enterprise and scientific grids or job schedulers.

The present document will help enterprises and laboratories to manage their IT software within large-scale computer and telecom infrastructures with the necessary virtualization.

Its primary audience are users of grid-based applications who will need to write application descriptors so they can run them on the grid for their grid.

# 2 References

References are either specific (identified by date of publication and/or edition number or version number) or non-specific.

- For a specific reference, subsequent revisions do not apply.
- Non-specific reference may be made only to a complete document or a part thereof and only in the following cases:
  - if it is accepted that it will be possible to use all future changes of the referenced document for the purposes of the referring document;
  - for informative references.

Referenced documents which are not found to be publicly available in the expected location might be found at http://docbox.etsi.org/Reference.

NOTE: While any hyperlinks included in this clause were valid at the time of publication ETSI cannot guarantee their long term validity.

## 2.1 Normative references

The following referenced documents are indispensable for the application of the present document. For dated references, only the edition cited applies. For non-specific references, the latest edition of the referenced document (including any amendments) applies.

[1] ETSI TS 102 827: "GRID; Grid Component Model (GCM); GCM Interoperability Deployment".

## 2.2 Informative references

The following referenced documents are not essential to the use of the present document but they assist the user with regard to a particular subject area. For non-specific references, the latest version of the referenced document (including any amendments) applies.

[i.1] Proactive, INRIA.

NOTE: Available at http://proactive.inria.fr.

[i.2] MPI Forum.

NOTE: Available at http://www.mpi-forum.org/docs/.

[i.3] Fractal, objectweb.

NOTE: Available at http://fractal.objectweb.org/.

# 3 Definitions

For the purposes of the present document, the terms and definitions given in TS 102 827 [1] and the following apply:

**component:** abstraction of a software entity with a well-defined interface for both server and client parts

**component model:** specification of how components are defined and interact together

**executable:** arbitrary program which can be the launch target of an GCM Application Descriptor

**fractal:** modular and extensible component model

NOTE: See http://fractal.objectweb.org/ [i.3].

**Message Passing Interface (MPI):** computer specification and is an implementation that allows many computers to communicate with one another

NOTE: See http://www.mpi-forum.org/docs/ [i.2].

**ProActive:** Java Grid middleware for parallel, distributed and multi-threaded computing

NOTE 1: See http://proactive.inria.fr [i.1].

NOTE 2: ProActive is part of the ObjectWeb/OW2 consortium and developed by INRIA, CNRS and University of Nice Sophia Antipolis, with Open Source code under the GPL license.

# 4 Overall Structure of the GCM Application Description

## 4.1 Principles

Like a deployment descriptor, an application descriptor is written in XML. Each element of the descriptor is defined as a single node, which can have attributes and child nodes.

Also, see clause 6.1 of TS 102 827 [1].

## 4.2 Overall XML structure

A deployment descriptor has the following structure:

```
<environment>
        <descriptorVariable …/>
        <programVariable …/>
        <descriptorDefaultVariable …/>
        <programDefaultVariable …/>
        <includePropertyFile …>
        …
</environment>

<application>
…
</application>

<resources>
        <nodeProvider>
            <file …/>
            …
        </nodeProvider>
        …
</resources>
```

The elements must be specified in this order. The `<environment>` element can be omitted, while the other ones are mandatory.

The `<environment>` element is a table of variable names and their values, which is used to give some amount of flexibility to the descriptor.

The `<application>` element describes the application that will be run on the grid, and the amount of resources it will use (the number of processing nodes).

Finally, the `<resources>` element describes the resources made available by the grid, by referencing deployment descriptors through `<nodeProvider>` elements.

### 4.2.1 Environment

The `<environment>` element can have the following children elements:

- **descriptorVariable:** the value has to be set in the descriptor, and cannot be specified in the program;
- **descriptorDefaultVariable:** a default value must be specified in the descriptor. The program has the ability to change the value at execution. If the value is changed in the program, then this new value will have precedence over the one defined in the descriptor;
- **programVariable:** the value must be set in the program, and cannot be specified in the descriptor;
- **programDefaultVariable:** a default value must be specified by the program as a constant, or computed (based on input size, for instance). The descriptor has the ability to change the value. If the value is changed in the descriptor, then this new value will have precedence over the one defined in the program;
- **includePropertyFile:** read variable definitions from a property file. A property file contains variable definitions as "NAME = VALUE" lines, for example:
  - variable1 = value1;
  - variable2 = value2.

There can be any number of these children elements. They cannot have child elements.

`<descriptorVariable>`, `<descriptorDefaultVariable>` and `<programDefaultVariable>` have two attributes, "name" and "value", each accepting string values.

`<programVariable>` only has a single "name" attribute (since it can only be set in the program).

`<includePropertyFile>` only has a single "location" attribute.

EXAMPLE:

```
<environment>
    <descriptorVariable name="usertype" value="admin" />
  <descriptorDefaultVariable name="username" value="jsmith" />
  <programVariable name="nbNodes" />
  <programDefaultVariable name="nbJobs" value="10" />
    <includePropertyFile location="otherVariableDefs.property" />
</environment>
```

# 5 Application Specification

The `<application>` element is the core of the descriptor, as it describes the application which will be run on a GCM-described grid. It contains a single child which type determines the kind of application that will be launched on the grid. The types currently defined are:

1) Executable.

2) ProActive.

## 5.1 Executable

An executable defines an application which is actually a simple command to be run.

It can have the following child elements:

- **nodeProvider** (empty element with a single "refId" attribute)**:** the id of a node provider (defined in the `<resources>` part). There can be any number of such element;
- **command** (command)**:** the command which will be run on the portion of the grid defined by the specified node providers.

This element can have the following attribute:

- **instances** (one of "onePerHost", "onePerVM", "onePerCapacity")**:** the number of instances of the command which will be run.

The `<command>` element can have the following children (in this specified order):

- **path** (path string)**:** the path of the executable;
- **arg** (string)**:** the arg string which will be passed to the command. There can be any number of such element;
- **filetransfer** (file transfer)**:** the files which should be transferred prior to running the command.

It can have the following attribute:

- **name** (string)**:** name of the executable. If a *<path>* child element is present, the value of this attribute will be appended to the value of the *<path>* child element.

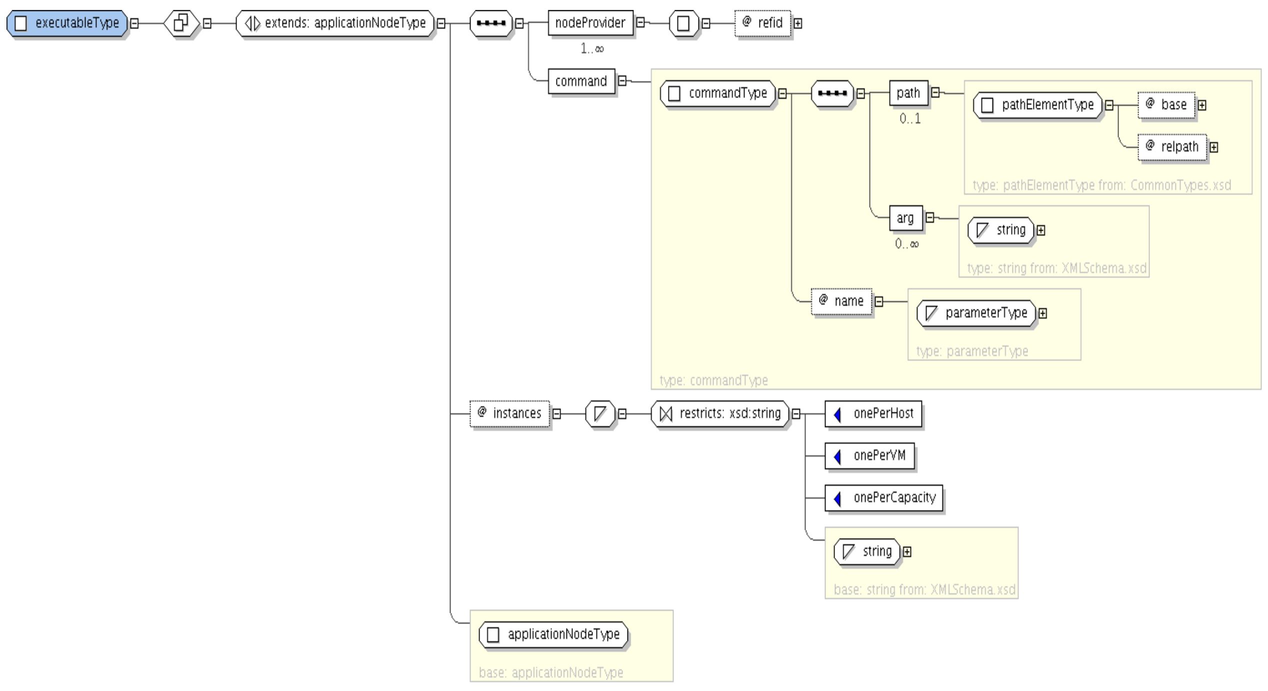

**Figure 1**

Example of a call to the "ls" Unix command:

```
<executable>
    <command name="ls">
        <arg>-l</arg>
        <arg>-h</arg>
        <arg>--sort=time</arg>
        <arg>-r</arg>
    </command>
    <nodeProvider refid="INRIA_LAN" />
</executable>
```

## 5.2 Extension of applicationNodeType (ProActive)

### 5.2.1 Overall structure

The proactiveType is a more complex extension of applicationNodeType than executableType.It uses the technicalServices and virtualNode XML types as follows:

```
<proactive>
    <technicalServices>
    </technicalServices>

    <virtualNode>
        <nodeProvider/>
        …
    </virtualNode>
    <virtualNode>
    …
    </virtualNode>

    …
</proactive>
```

The technicalServices elements specify the set of technical services global to the instance of ProActive. Technical Services are described in clause 5.2.3.

A <virtualNode> element can have the following children:

- **nodeProvider** (reference to a node provider)**:** the node provider for this virtual node - see below for description;
- **technicalServices** (technical service)**:** a technical service specific to the virtual node. There can be any number of such child elements.

A virtualNode element can also have the following attributes:

- **id** (string)**:** a string identifying the virtual node;
- **capacity** (positive integer)**:** the capacity requested by this virtual node (that is, the total number of nodes it will request from the node providers which are associated with it). If not specified, the virtual node gets as many node as possible. If multiple virtual nodes without a capacity are defined, a fair share allocation must be performed. Virtual nodes with specified capacity must be fulfilled before virtual nodes without a specified capacity.

A <nodeProvider> within a <virtualNode> can only have <technicalServices> child elements. These describe technical services specific to the node provider. A <nodeProvider> can also have the following attributes:

- **refid** (string)**:** the id of the node provider (as defined in the resources element);
- **capacity** (positive integer)**:** the capacity of the node provider (that is, the number of nodes which may be requested.

### 5.2.2 Definition of proactiveType

In addition to using standard GCM types, elements of type proactiveType can have the following children:

- **configuration:** various configuration parameters. This element can have the following children:
  - **bootClasspath** (simple classpath)**:** the boot classpath for the JVM;
  - **java** (path string)**:** the path to the Java executable;
  - **jvmarg** (string)**:** arguments passed to the JVM;
  - **applicationClasspath** (classpath)**:** classpath for the application;
  - **proactiveClasspath** (classpath)**:** classpath used to override the standard ProActive classpath computed from its installation location;
  - **securityPolicy** (relative path)**:** path to the Java security policy file;
  - **proactiveSecurity:** security policy for application and runtime. This element has two children:
    - **applicationPolicy** (relative path)**:** path to Java security policy file that will be applied on the application's objects deployed at runtime, like nodes and active objects;
    - **runtimePolicy** (relative path)**:** path to Java security policy file that will be applied on the ProActive Runtime.
  - **log4jProperties** (relative path)**:** path to the Java log4j configuration file;
  - **userProperties** (relative path)**:** path to the Java properties file;
  - **virtualNode** (virtual node)**:** description of a virtual node. There can be any number of such element.

The `<proactive>` element can have the following attributes:

- **relpath** (path string)**:** the location of the ProActive installation;
- **base** (one of "HOME", "ROOT")**:** base location of the ProActive installation: HOME is the user's home directory, ROOT is the root directory of the system.

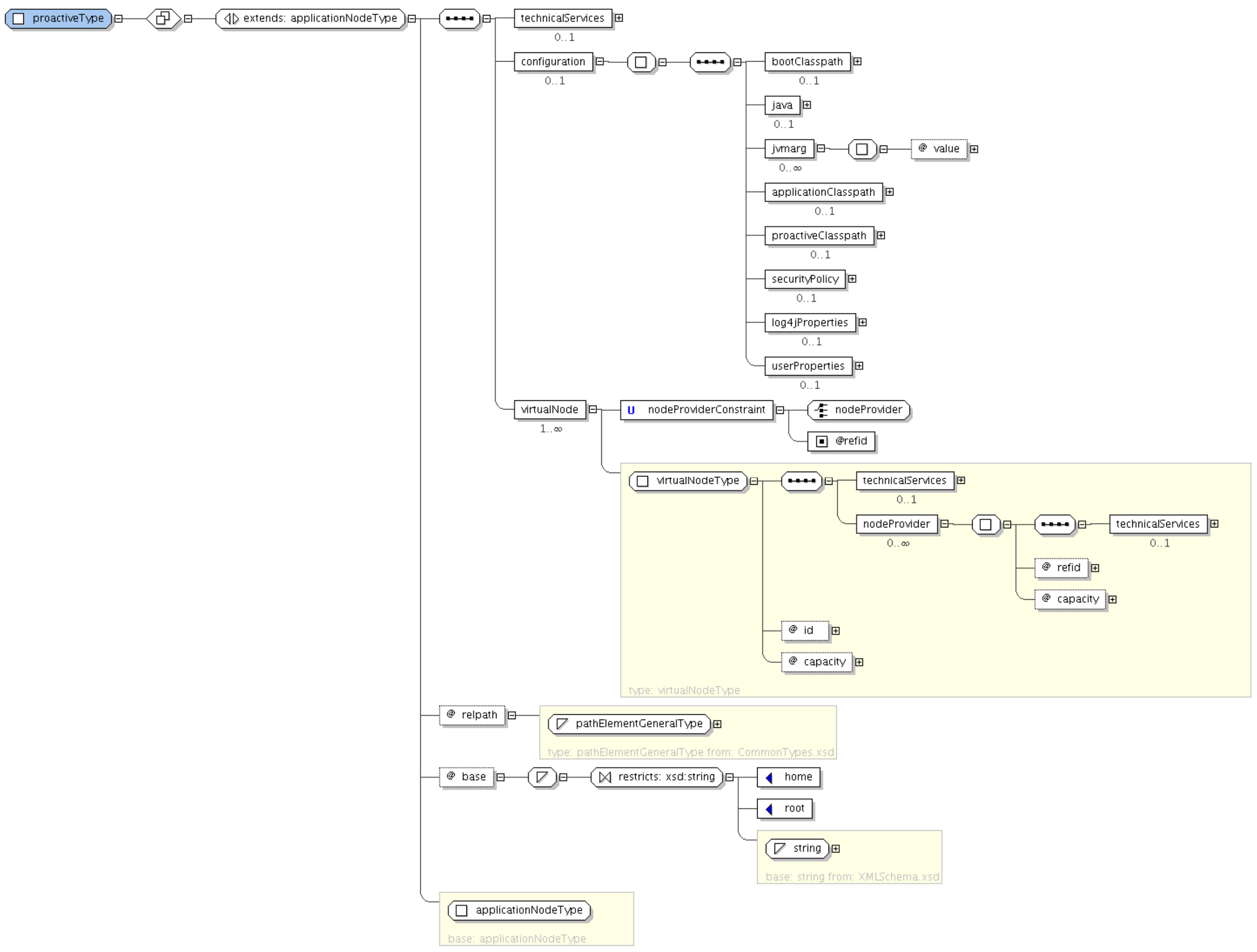

Figure 2

EXAMPLE:

```
<proactive base="root" relpath="/usr/local/proactive">
    <technicalServices>
        <class name="proactiveservices">
            <property name="myprop" value="avalue"/>
        </class>
        <class name="otherservices">
            <property name="x" value="12"/>
            <property name="y" value="15"/>
        </class>
    </technicalServices>

    <virtualNode id="master" capacity="1">
        <technicalServices>
            <class name="vnodeservices">
                <property name="myprop" value="avalue"/>
            </class>
        </technicalServices>

        <nodeProvider refid="INRIA_LAN">
            <technicalServices>
                <class name="nodeprovidertechservices">
                    <property name="myprop" value="avalue"/>
                </class>
            </technicalServices>
        </nodeProvider>
    </virtualNode>

    <virtualNode id="slaves" >
        <nodeProvider refid="INRIA_LAN"/>
        <nodeProvider refid="INRIA_CLUSTERS"/>
    </virtualNode>
    <virtualNode id="slaves2" >
        <nodeProvider refid="INRIA_LAN"/>
        <nodeProvider refid="INRIA_CLUSTERS"/>
    </virtualNode>

</proactive>
```

## 5.2.3 Technical Services

A technical service is a non-functional requirement that may be dynamically fulfilled at runtime by adapting the configuration of selected resources. Such services are specified with a `<technicalServices>` element. This element holds a sequence of `<class>` children elements, which themselves hold a sequence of `<property>` elements.

A `<class>` element has a single attribute, "name", of type string. A `<property>` element has two attributes, "name" and "value", both of type string.

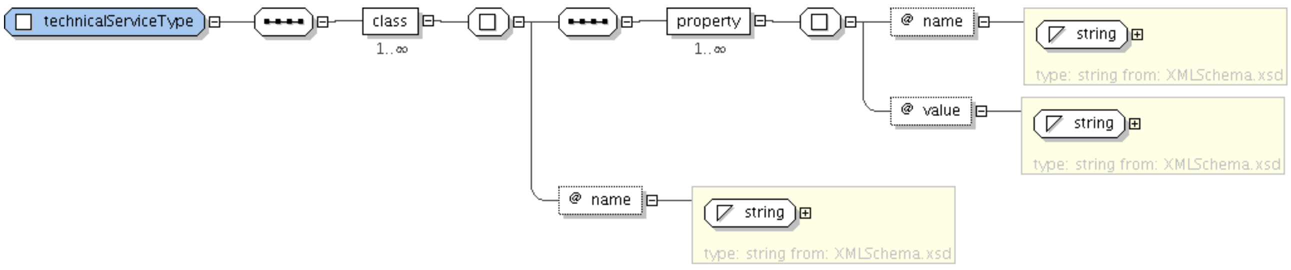
technicalServiceType
class
1..∞
property
1..∞
name
string
type: string from: XMLSchema.xsd
value
string
type: string from: XMLSchema.xsd
name
string
type: string from: XMLSchema.xsd

**Figure 3**

EXAMPLE:

```
<technicalServices>
    <class name="proactiveservice">
        <property name="myprop" value="avalue"/>
    </class>
    <class name="otherservice">
        <property name="x" value="12"/>
        <property name="y" value="15"/>
    </class>
</technicalServices>
```

- Technical services can be defined at several levels of the application definition:
    - globally;
    - for a given virtual node;
    - for a given node provider for a virtual node.

This is better explained with the following skeleton:

```
<proactive>
    <technicalServices>
        <class name="proactiveFaultTolerance">
            <property name="enable" value="true" />
        </class>
    </technicalServices>

    <virtualNode>
        <technicalServices>
        <class name="…">
            <property name="…" value="…" />
        </class>

        </technicalServices>
        <nodeProvider>
            <technicalServices>
                <class name="…">
                    <property name="…" value="…" />
                </class>
            </technicalServices>
        </nodeProvider>
        …
    </virtualNode>
    …
</proactive>
```

### 5.2.4 Data

The data tag allows describing the input and output data stores required by an application. Each data store has a unique identifier. The default input and output data stores are automatically tagged with the "`default`" identifier. Grid middleware are expected to provide an interface to upload and download data from the remote data stores. This element is optional and can appear only once.

A `<data>` element can have the following children:

- **inputDefault:** The default input location.
- **input:** Additional input locations. There can be any number of such children.
- **outputDefault:** The default output location.
- **output:** Additional output locations. There can be any number of such children.

Each one of the `<inputDefault>`, `<input>`, `<outputDefault>`, `<output>` elements can have the following children:

- **remoteAccess:** access URL to this data store. Used for accessing it from remote nodes. URL defines which protocol is used to access the data from remote node, and some additional information for protocol like path, username, password etc. This element is mandatory and can appear only once.
- **location:** describes where the data are physically stored. If specified it allows local access to the data when possible. This element is optional.

A `<location>` element can have the following attributes:

- **hostname:** hostname of host where data are stored. Data can be accessed locally on hostname with that name.
- **path:** local path to data. This path is local to host with hostname specified in hostname attribute.

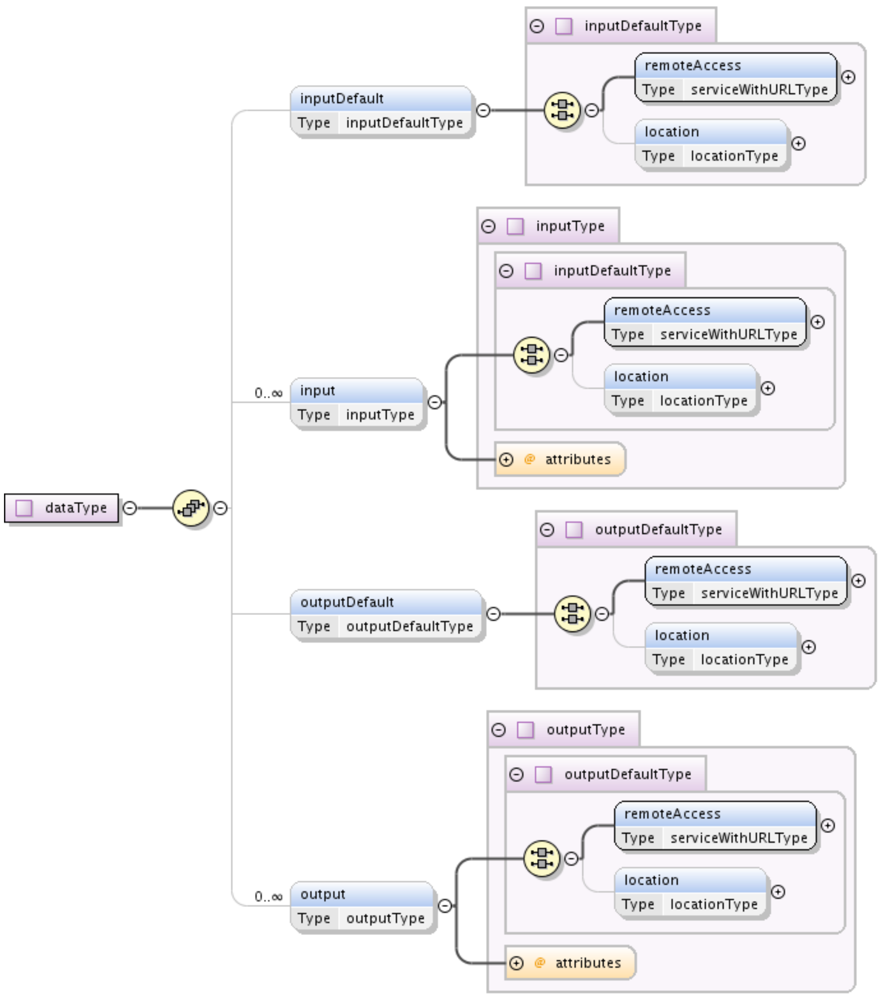

**Figure 4**

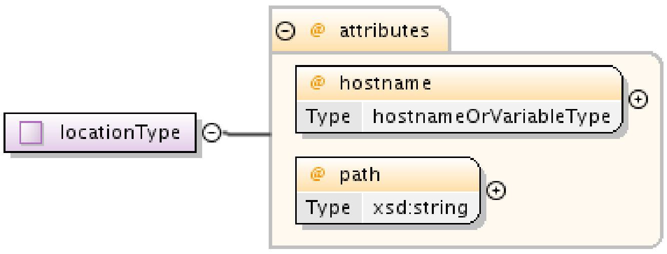

**Figure 5**

## 6 Resource Specification

The `<resources>` element has no attributes and only contains a sequence of `<nodeProvider>` elements. There can be any number of such elements.

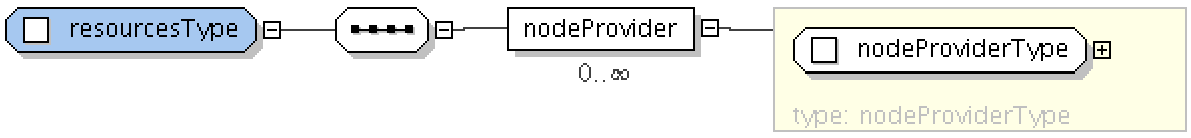

**Figure 6**

## 6.1 Node Provider

A `<nodeProvider>` can have the follow children:

- **file** (file)**:** this element can only have a single attribute ("path") which contains the path to the deployment descriptor file which the node provider uses. There can be any number of such elements.

It can also have the following attribute:

- **id** (id)**:** a string identifying the ProActive node provider.

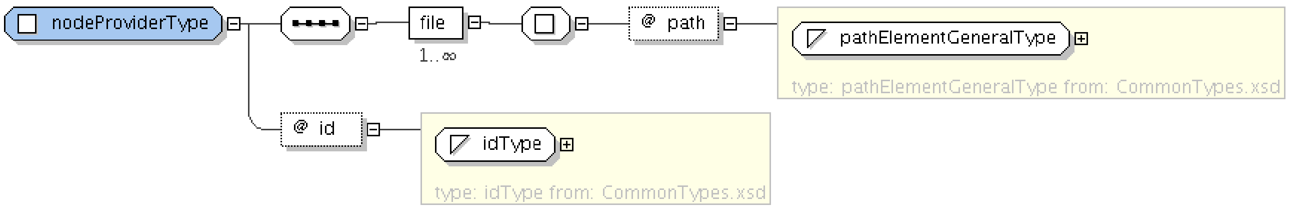
nodeProviderType
file
1..∞
@ path
pathElementGeneralType
type: pathElementGeneralType from: CommonTypes.xsd
@ id
idType
type: idType from: CommonTypes.xsd

**Figure 7**

# Annex A (normative): Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" targetNamespace="urn:gcm:application:1.0"
xmlns="urn:gcm:application:1.0"
    elementFormDefault="qualified">

    <xsd:include schemaLocation="./CommonTypes.xsd" />

    <xsd:attribute name="OSAttr">
        <xsd:simpleType>
            <xsd:restriction base="xsd:string">
                <xsd:enumeration value="Unix"></xsd:enumeration>
                <xsd:enumeration value="Windows"></xsd:enumeration>
            </xsd:restriction>
        </xsd:simpleType>
    </xsd:attribute>

    <xsd:attribute name="PriorityAttr">
        <xsd:simpleType>
            <xsd:restriction base="xsd:string">
                <xsd:enumeration value="Low"></xsd:enumeration>
                <xsd:enumeration value="Normal"></xsd:enumeration>
                <xsd:enumeration value="High"></xsd:enumeration>
            </xsd:restriction>
        </xsd:simpleType>
    </xsd:attribute>


    <xsd:simpleType name="parameterType">
        <xsd:restriction base="xsd:string"></xsd:restriction>
    </xsd:simpleType>


    <xsd:complexType name="jreType">
        <xsd:sequence>
            <xsd:element name="parameter" type="parameterType" />
        </xsd:sequence>
        <xsd:attribute name="path" type="xsd:string"></xsd:attribute>
    </xsd:complexType>

    <xsd:complexType name="valueType">
        <xsd:attribute name="value" type="xsd:string" />
    </xsd:complexType>

    <xsd:simpleType name="DataSpacesInputOutputIdType">
        <xsd:restriction base="idType">
            <xsd:pattern
value="[^/]{1,6}|[^/]{8,}|[^d][^/]{6}|[^/][^e][^/]{5}|[^/]{2}[^f][^/]{4}|[^/]{3}[^a][^/]{3}|[^/]{4}[
^u][^/]{2}|[^/]{5}[^l][^/]|[^/]{6}[^t]"/>
        </xsd:restriction>
    </xsd:simpleType>

    <xsd:simpleType name="DataSpacesInputOutputIdOrVariableType">
        <xsd:union memberTypes="DataSpacesInputOutputIdType variableRefType"/>
    </xsd:simpleType>

    <xsd:complexType name="namingServiceType">
        <xsd:attribute name="url" type="xsd:anyURI" use="required"/>
    </xsd:complexType>

    <xsd:complexType name="remoteAccessType">
        <xsd:attribute name="url" type="xsd:anyURI" use="required"/>
    </xsd:complexType>

    <xsd:complexType name="locationType">
        <xsd:attribute name="hostname" type="hostnameOrVariableType" use="required"/>
        <xsd:attribute name="path"     type="xsd:string" use="required"/>
    </xsd:complexType>


    <xsd:complexType name="inputDefaultType">
        <xsd:all>
```

```
         <xsd:element  name="remoteAccess" type="remoteAccessType" minOccurs="1" />
        <xsd:element  name="location" type="locationType"         minOccurs="0" />
        </xsd:all>
    </xsd:complexType>

    <xsd:complexType name="inputType">
        <xsd:complexContent>
            <xsd:extension base="inputDefaultType">
                <xsd:attribute name="id" use="required"
type="DataSpacesInputOutputIdOrVariableType"/>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="outputDefaultType">
        <xsd:all minOccurs="1">
         <xsd:element  name="remoteAccess" type="remoteAccessType" minOccurs="1" />
         <xsd:element  name="location" type="locationType"         minOccurs="0" />
        </xsd:all>
    </xsd:complexType>

    <xsd:complexType name="outputType">
        <xsd:complexContent>
            <xsd:extension base="outputDefaultType">
                <xsd:attribute name="id" type="DataSpacesInputOutputIdOrVariableType"
use="required"/>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="dataType">
        <xsd:sequence>
            <xsd:element minOccurs="0" maxOccurs="1" name="namingService" type="namingServiceType"
/>
            <xsd:element minOccurs="0" maxOccurs="1" name="inputDefault"  type="inputDefaultType"/>
            <xsd:element minOccurs="0" maxOccurs="unbounded" name="input" type="inputType"/>
            <xsd:element minOccurs="0" maxOccurs="1" name="outputDefault" type="outputDefaultType"/>
            <xsd:element minOccurs="0" maxOccurs="unbounded" name="output" type="outputType"/>
        </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="technicalServiceType">
        <xsd:sequence>
            <xsd:element name="class" maxOccurs="unbounded">
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element name="property" maxOccurs="unbounded">
                            <xsd:complexType>
                                <xsd:attribute name="name" type="xsd:string" use="required" />
                                <xsd:attribute name="value" type="xsd:string" use="required" />
                            </xsd:complexType>
                        </xsd:element>
                    </xsd:sequence>
                    <xsd:attribute name="name" type="xsd:string" use="required" />
                </xsd:complexType>
            </xsd:element>
        </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="nodeProviderType">
        <xsd:sequence>
            <xsd:element name="file" maxOccurs="unbounded">
                <xsd:complexType>
                    <xsd:attribute name="path" type="pathElementGeneralType"></xsd:attribute>
                </xsd:complexType>
            </xsd:element>
        </xsd:sequence>
        <xsd:attribute name="id" type="idType"></xsd:attribute>
    </xsd:complexType>

    <xsd:complexType name="virtualNodeType">
        <xsd:sequence>
            <xsd:element name="technicalServices" type="technicalServiceType" minOccurs="0" />
            <xsd:element name="nodeProvider" minOccurs="0" maxOccurs="unbounded">
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element name="technicalServices" type="technicalServiceType"
minOccurs="0" />
```

```
                    </xsd:sequence>
                    <xsd:attribute name="refid" type="idType" use="required" />
                    <xsd:attribute name="capacity" type="xsd:nonNegativeInteger" />
                </xsd:complexType>
            </xsd:element>
        </xsd:sequence>
        <xsd:attribute name="id" type="xsd:string" use="required"></xsd:attribute>
        <xsd:attribute name="capacity" type="xsd:nonNegativeInteger"></xsd:attribute>
    </xsd:complexType>

    <xsd:complexType name="commandType">
        <xsd:sequence>
            <xsd:element name="path" type="pathElementType" minOccurs="0" />
            <xsd:element name="arg" type="xsd:string" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
        <xsd:attribute name="name" type="parameterType"></xsd:attribute>
    </xsd:complexType>

    <xsd:complexType name="applicationNodeType" abstract="true"></xsd:complexType>

    <xsd:complexType name="proactiveType">
        <xsd:complexContent>
            <xsd:extension base="applicationNodeType">
                <xsd:sequence>
                    <xsd:element name="technicalServices" type="technicalServiceType" minOccurs="0"
/>
                    <xsd:element name="configuration" minOccurs="0">
                        <xsd:complexType>
                            <xsd:sequence>
                                <xsd:element name="bootClasspath" type="pathType" minOccurs="0" />
                                <xsd:element name="java" type="pathElementType" minOccurs="0"
maxOccurs="1" />
                                <xsd:element name="jvmarg" minOccurs="0" maxOccurs="unbounded">
                                    <xsd:complexType>
                                        <xsd:attribute name="value" type="xsd:string" />
                                    </xsd:complexType>
                                </xsd:element>
                                <xsd:element name="applicationClasspath" type="classpathType"
minOccurs="0" maxOccurs="1" />
                                <xsd:element name="proactiveClasspath" type="classpathType"
minOccurs="0" maxOccurs="1" />
                                <xsd:element name="securityPolicy" type="pathElementType"
minOccurs="0" maxOccurs="1" />
                                <xsd:element name="log4jProperties" type="pathElementType"
minOccurs="0" maxOccurs="1" />
                                <xsd:element name="userProperties" type="pathElementType"
minOccurs="0" maxOccurs="1" />
                            </xsd:sequence>
                        </xsd:complexType>
                    </xsd:element>

                    <xsd:element name="virtualNode" type="virtualNodeType" maxOccurs="unbounded">
                        <xsd:unique name="nodeProviderConstraint">
                            <xsd:selector xpath="nodeProvider"></xsd:selector>
                            <xsd:field xpath="@refid"></xsd:field>
                        </xsd:unique>
                    </xsd:element>

                    <xsd:element name=”data” type=”dataType” minOccurs=”0” maxOccurs=”1” />
                </xsd:sequence>
                <xsd:attribute name="relpath" type="pathElementGeneralType" use="required" />
                <xsd:attribute name="base" use="required">
                    <xsd:simpleType>
                        <xsd:restriction base="xsd:string">
                            <xsd:enumeration value="home" />
                            <xsd:enumeration value="root" />
                        </xsd:restriction>
                    </xsd:simpleType>
                </xsd:attribute>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="executableType">
        <xsd:complexContent>
            <xsd:extension base="applicationNodeType">
                <xsd:sequence>
                    <xsd:element name="command" type="commandType"></xsd:element>
                    <xsd:element name="nodeProvider" maxOccurs="unbounded">
                        <xsd:complexType>
                            <xsd:attribute name="refid" type="xsd:string" use="required" />
                        </xsd:complexType>
                    </xsd:element>
                </xsd:sequence>
                <xsd:attribute name="instances">
                    <xsd:simpleType>
                        <xsd:restriction base="xsd:string">
                            <xsd:enumeration value="onePerHost"></xsd:enumeration>
                            <xsd:enumeration value="onePerVM"></xsd:enumeration>
                            <xsd:enumeration value="onePerCapacity"></xsd:enumeration>
                        </xsd:restriction>
                    </xsd:simpleType>
                </xsd:attribute>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:element name="abstractApplicationElement" type="applicationNodeType" />

    <xsd:element name="executable" type="executableType"
substitutionGroup="abstractApplicationElement" />
    <xsd:element name="proactive" type="proactiveType"
substitutionGroup="abstractApplicationElement" />


    <xsd:complexType name="applicationType">
        <xsd:sequence>
            <xsd:element ref="abstractApplicationElement" maxOccurs="1" minOccurs="1"></xsd:element>
        </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="resourcesType">
        <xsd:sequence>
            <xsd:element name="nodeProvider" type="nodeProviderType" minOccurs="0"
maxOccurs="unbounded"></xsd:element>
        </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="GCMApplicationType">
        <xsd:sequence>
            <xsd:element name="environment" type="environmentType" minOccurs="0" />
            <xsd:element name="application" type="applicationType" minOccurs="1"
maxOccurs="1"></xsd:element>
            <xsd:element name="resources" type="resourcesType" minOccurs="1"
maxOccurs="1"></xsd:element>
        </xsd:sequence>
    </xsd:complexType>

    <xsd:element name="GCMApplication" type="GCMApplicationType">
        <!-- resource providers refid/id key correctness -->
        <!--  -->
        <xsd:key name="idnodeProviders">
            <xsd:selector xpath="resources/nodeProvider" />
            <xsd:field xpath="@id" />
        </xsd:key>
        <xsd:keyref name="refIdKeynodeProvider" refer="idnodeProviders">
            <xsd:selector xpath="application/executable/nodeProvider"></xsd:selector>
            <xsd:field xpath="@refid"></xsd:field>
        </xsd:keyref>
        <xsd:keyref name="refIdKeynodeProvider2" refer="idnodeProviders">
            <xsd:selector xpath="application/proactive/virtualNode/nodeProvider"></xsd:selector>
            <xsd:field xpath="@refid"></xsd:field>
        </xsd:keyref>
    </xsd:element>

</xsd:schema>
```

# Annex B (informative): Example of application descriptor

```xml
<?xml version="1.0" encoding="UTF-8"?>

<GCMApplication
    xmlns="urn:gcm:application:1.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www-sop.inria.fr/oasis/ProActive/schemas
http://proactive.inria.fr/schemas/gcm/1.0/ApplicationDescriptorSchema.xsd">

    <application>
        <proactive relpath="Scratch/ProActive/">

            <configuration>
                <java relpath="myApps/Java/bin/java" />
                <proactiveClasspath>
                    <pathElement base="proactive"
                        relpath="lib/log4j.jar" />
                    <pathElement base="proactive"
                        relpath="lib/fractal.jar" />
                    <pathElement base="home"
                        relpath="myApps/ganymed-ssh.jar" />
                </proactiveClasspath>
                <applicationClasspath>
                    <pathElement relpath="myApps/nqueens.jar" />
                </applicationClasspath>
                <securityPolicy base="proactive"
                    relpath="config/security.policy" />
                <log4jProperties base="home" relpath="log4j.properties" />
            </configuration>

            <virtualNode id="master" capacity="1">
                <nodeProvider refid="COMPANY_LAN" />
            </virtualNode>

            <virtualNode id="slaves" capacity="max">
                <nodeProvider refid="COMPANY_LAN" />
            </virtualNode>

            <data>
                <inputDefault>
                    <remoteAccess url="http://remotehost.tld/path/dir/" />
                </inputDefault>

                <input id="images">
                    <remoteAccess url="ftp://san.tld/path/images/" />
                    <location hostname="san.tld" path="/user/data/images/" />
                </input>

                <outputDefault>
                    <remoteAccess url="scp://user@frontend/output/" />
                </outputDefault>
            </data>

        </proactive>
    </application>

    <resources>
        <nodeProvider id="COMPANY_LAN">
            <file path="deployment.xml" />
        </nodeProvider>
    </resources>

</GCMApplication>
```

# Annex C (informative): Bibliography

- (Programming, Composing, Deploying for the Grid Baude F., Baduel L., Caromel D., Contes A., Huet F., Morel M. and Quilici R.), in "GRID COMPUTING: Software Environments and Tools", Jose C. Cunha and Omer F. Rana (Eds), Springer Verlag, January 2006

- Snir, Marc; Otto, Steve; Huss-Lederman, Steven; Walker, David; Dongarra, Jack (1995): "MPI: The Complete Reference", MIT Press Cambridge, MA, USA. ISBN 0-262-69215-5

- Gropp, William; Lusk, Ewing; Skjellum, Anthony (1999): "Using MPI, 2nd Edition: portable Parallel Programming with the Message Passing Interface", MIT Press In Scientific And Engineering Computation Series, Cambridge, MA, USA. 395 pp. ISBN 978-0-262-57132-6.

- E. Bruneton, T. Coupaye, M. Leclercq, V. Quéma, and J.-B. Stefani: "The Fractal Component Model and Its Support in Java. Software Practice and Experience, special issue on Experiences with Auto-adaptive and Reconfigurable Systems", 36(11-12), 2006.

- E. Bruneton, T. Coupaye, M. Leclercq, V. Quéma, and J.-B. Stefani: "An Open Component Model and Its Support in Java. Seventh International Symposium on Component-Based Software Engineering (CBSE-7)", LNCS 3054, pp. 7-22, May 2004.

# History

| Document history | | |
|---|---|---|
| V1.1.1 | August 2008 | Publication |
| V2.1.1 | March 2010 | Publication |
| | | |
| | | |
| | | |