Application Web Services

Marlon Pierce, Choonhan Youn, Geoffrey Fox
Community Grid Labs, Indiana University
Bloomington, IN 47404

Introduction

This report gives an overview of web services, with a particular focus on how to deploy and use an application as a web service. This is a work in progress and part of the proposed Application Metadata Working Group [1] and the Grid Computing Environments Research Group [2] of the Global Grid Forum, so revisions and refinement will occur.

An *application* here means specifically some code developed by the scientific community. Examples would be finite element codes, grid generation codes, and visualization tools. These might be written in Fortran or C, may be parallelized with MPI, and so on. From the point of view of the portal interactions, these details are unimportant in our approach. We will treat all these applications as black boxes and will describe here how to wrap these applications in XML proxies. The wrappers can then be converted into Java data classes for manipulation by services. No modification of the application source code is required. We refer to this as *proxy wrapping,* as distinguished from the direct service wrapping one might generate with a tool such as SWIG.

The following figure summarizes our general architecture. An actual application (a scientific code or a database, for example) is wrapped by a Java program. For databases, this is well known: the Java application just makes a JDBC connection to the database and defines and implements an API for clients to interact with the database. Likewise, scientific applications can be wrapped by general purpose Java applications, which can be used to invoke the application, either directly or through submission to a queuing system.

The WSDL interface is the XML abstraction of this wrapper application interface and can be viewed as a list of instructions for building clients. The actual client (say, a JSP page) may be developed offline, or (as in the figure) generated automatically by mapping the application interface into visual components (like HTML form elements). These user interfaces to services further can be wrapped as *portlets* and aggregated into a single user interface using a technology such as Jetspeed. Thus in the grand scheme towards which we are building, a Web portal consists of client user interfaces to various services generated from the XML description of the service. Likewise, this XML description of the service describes how to invoke the service methods, which in turn are just proxies to some legacy application.

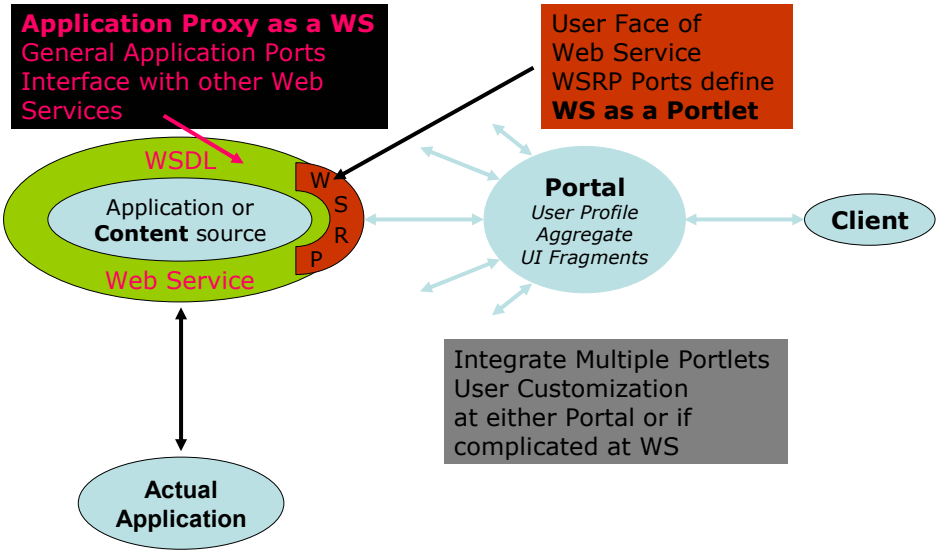


Figure 1 An overview of the proxy component architecture

Applications services are deployed into a computational web portal, a browser-based system that provides a user interface to applications and various services. A typical web portal allows a user to log in securely to some computing resource, submit jobs, view results, and manage files on the remote resources. The web interface is desirable since it allows the user to potentially log in from anywhere with no software on the client besides the web browser.

We believe the process we introduce here for transforming applications into services is generally applicable to any portal designed to support multiple codes, particularly when the applications to be deployed are not known *a priori* by the portal developer. To make our arguments concrete, we refer to our computational web portal project, Gateway [3], in many examples.

Message-Based Architectures and Coarse Grained Components

A general characteristic of the Application Web services architecture that we propose is that it supports (at least at a coarse-grained level) an environment in which distributed components communicate via XML messages. Components needing higher performance communication may negotiate down to a lower level protocol, but more typically we want to define the scope of our components by their communication requirements. Network speeds of XML messages may be on the order of milliseconds, so components should be chosen based in part on this communication speed. Typically this is not an issue for sequentially accessed codes: a component is a proxy wrapper to an application, which may take minutes, hours, or even days to run and which may sit in queues for longer periods of time. An event service for checking the job's status, stopping the job, or launching a subsequent application likewise does not need microsecond response times. There is substantial work building support for this model, and a set of recent articles can be found in Ref [4].

The message-based architecture allows us to incorporate other frameworks, such as security and collaboration, into our architecture. Security is naturally implemented at the message level in a way that can be re-used between multiple web services [5] [6]. Collaboration corresponds to sharing either applications or their state between multiple clients. This again can be largely implemented at message level without detailed application-specific work [7].

Web Service Review

Web services refer to the invocation of remote methods (functions) using an XML-based protocol and method interface definitions. The protocol (usually SOAP [8]) is attached to an HTTP message and contains, for instance, the name of the method and necessary parameters needed to invoke the remote service. The SOAP service is typically deployed as an application in a web server. For example, Apache Axis [9] runs as a Java servlet in an Apache Tomcat server. The method interface (in WSDL [10]) is an agreed-upon set of methods, parameters and return types for a particular service and is implemented in some Web-friendly programming language (such as Java or Python) and can be used as a guideline for writing clients, or it may be used to generate client stubs: classes that can be used locally by a client but that are actually wrappers around SOAP invocation calls to a remote service.

Core Portal Web Services

While it is possible (and in future versions desirable) to write the interface and control portions of an application in a Web-friendly language such as Java or Python, this is typically not the case for most legacy applications. C/C++ and FORTRAN codes can be wrapped inside Python (using SWIG [11]) or Java (using the Java Native Interface, JNI) but this is an invasive procedure that requires access to some application source code, which is often not available.

We therefore do not develop application-specific web services directly with WSDL and SOAP. Instead, we use SOAP and WSDL (along with Java) to develop general purpose *core web services* that perform the following tasks:

1. Run an arbitrary application on some computer as an external process. Running the application on the same computer as the SOAP server is straightforward. Executing an application on a different machine requires one of the following: a) yet another SOAP server on that second machine, b) some remote shell capability (rsh or ssh), or c) some computational grid technology such as Globus. The latter cases introduce complicated installation and security issues, so for now we will assume that the SOAP service runs on the same machine that will execute the application.
2. Move, copy, rename, delete files on some computer (either directly in Java or else as an external call to Unix commands).
3. Upload, download, or crossload files between computers. Upload/download refers to file transfers between the user's desktop and some remote destination. Because browsers do not directly support SOAP clients, this is not entirely a web service. Crossload refers to transfers of files between computers using web services. File transfer may be implemented entirely in Java or may use some external helper application (such as rcp).
4. Generate a batch script request for a particular queuing system, such as PBS.
5. Monitor the execution of a job running in a queuing system. This may be done by periodic status queries (running qstat on a host) or may be event-driven (such as by an email handling system). The monitoring service may also allow the queued job to be deleted or suspended.

6. Authenticate and authorize users. This is currently highly dependent on the desired underlying security mechanism (Kerberos or Globus GSI for example).
7. Provide information services. Here the web services are a set of WSDL interfaces that wrap heterogeneous backend information services. These might include general purpose LDAP servers, Globus MDS, XML databases, and UDDI.

To use these services, one needs to develop clients for each service using the WSDL interface. These clients generate SOAP requests to the associated service. A client might typically be a JSP page. SOAP clients and servers can be on the same Tomcat server or on separate servers.

Proxy Service Wrapping versus Direct Service Wrapping

We refer to the above as a *proxy wrapper* approach. The web service is not actually the application but is instead a proxy that invokes the application. The proxy wrapper is implemented in a web friendly language that can easily be converted to a web service. We may similarly label the alternate approach of directly interacting with the application as the *direct wrapping* approach. The advantage of the proxy wrapping approach is that any application or legacy service (such as Kerberos client commands) can be wrapped in this manner. Thus we can define rules (in XML) for describing an arbitrary application. We can then use these rules to define a particular application service, and bind the application service to underlying core portal services. This allows us to build a general purpose framework for working with applications, but sacrifices highly sophisticated interfaces for a particular application.

Alternate Web Service Protocols

"Web Services" are very loosely defined. SOAP services, for example, can be invoked without using WSDL and WSDL method invocations do not need SOAP as the message protocol. There are several instances when the latter is desirable. First, the system might rely on frequent high-performance messaging (relatively small communication-to-execution ratios). This can always be alleviated by increasing the scope of the service to reduce the frequency of communication. Second, the service may only execute infrequently but still be better served by a more appropriate protocol. For example, transferring large datasets requires a service for high-performance file transfer that will need to use some other protocol besides SOAP over HTTP for the actual data transfer, although SOAP may be used for setting up the call to the underlying service (FTP). Thirdly, one may already have a legacy distributed object system that makes use of a protocol such as CORBA's IIOP. Direct support within WSDL for this protocol is not currently available and would need to be developed. In the interim, protocol bridges between SOAP and IIOP are a relatively simple solution. Finally, one may wish to use a protocol with features not currently found in SOAP. This may include reliable protocols that guarantee once-and-only-once execution of a service. Again, an interim solution for this is to use SOAP clients and servers to initialize the use of the more feature-rich protocol.

Application Service Lifecycle and XML Descriptors

Applications can exist in four stages, as defined by the Application Metadata Working Group:

1. Application abstraction: a general description of how to run the application (it takes 1 input file, generates 4 output files, lives on computers A, B, and C).

2. Prepared instance: this is a specific instance of the application. A user has provided most or all of the information needed to run an application, but has not yet actually submitted it.
3. Submitted instance: This is job that has been committed to a set of resources (submitted to a queue, for example). Queuing systems represent several additional substates of existence, such as queued, running, sleeping, exiting, and completed.
4. Completed instance: The application has finished and we preserve metadata about it.

The important point here is that all of this can be described by a set of XML descriptions languages: application descriptions, host descriptions, and queue descriptions. We actually have two sets of XML for each description: one set for stage (1) above, and one set for stages (2)-(4) above. That is, the XML descriptions for (1) describe user options, while the descriptors for (2)-(4) contain a user's actual choices. We dub the first set of schemas as *abstract descriptors* and the second set as *abstract instances.*

The documentation for these schema sets can be found in the appendices. A key feature of our design is to divide things into modular containers. Thus an application descriptor contains one or more host descriptors, which in turn contain queue descriptors. The reason for this is that we want to keep these schemas pluggable. Many groups have developed or are developing extensive schema descriptions of queuing systems, for example, and if we find a better schema then our own, we would like to plug it in. This is straightforward for the queue descriptors, since they don't contain any other schema, but not straightforward for host descriptors, so in the next iteration we will need to define schema wrappers that encase the external schemas and provide the <anyType> hooks.

Deploying Application Web Services

We will first examine the descriptors associated with stage (1). Creating an XML instance of these schemas is done by the person deploying the web service.

We now use Gateway as an example of deploying an application web service, although the process should be similar with other proxy-based portals. Figure 1 below illustrates (based on an earlier version of the Application Descriptor) how to deploy an application on a particular host as a service. This form is used to edit the Application Descriptor XML file.

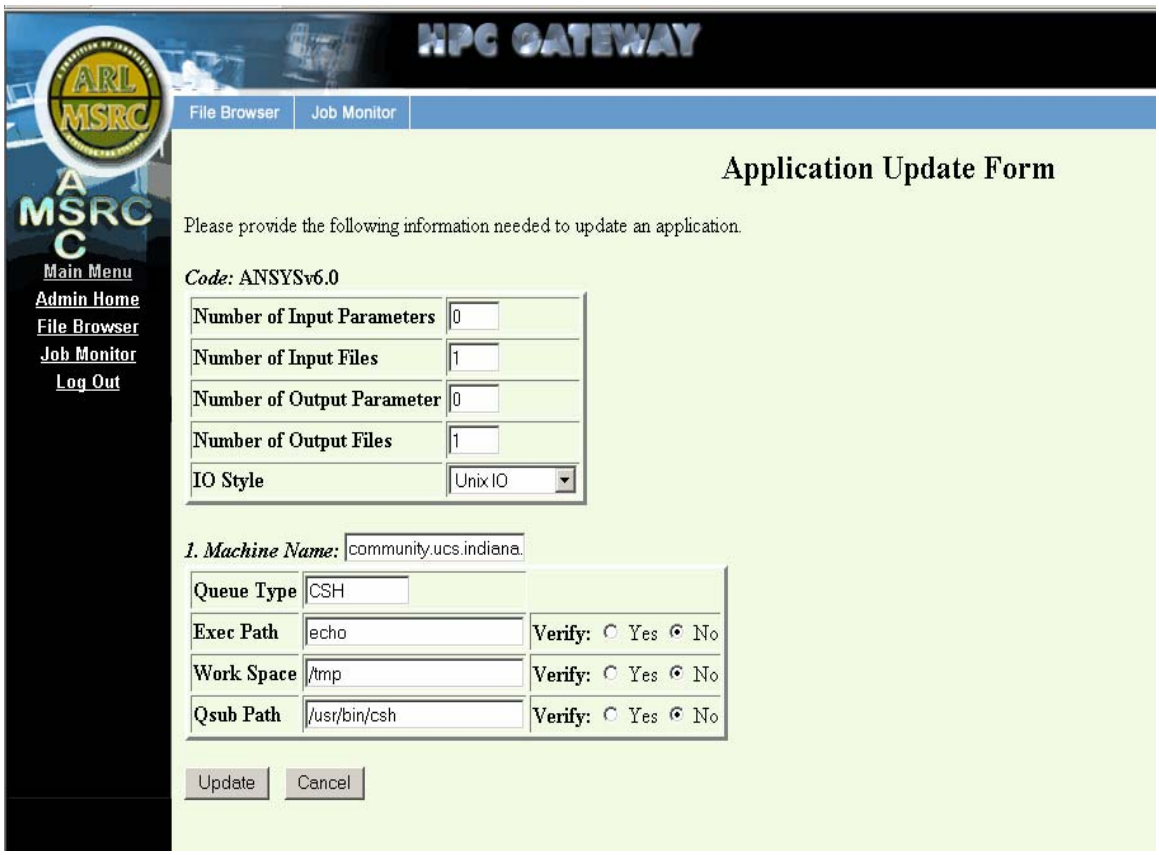


Figure 2 HTML forms for deploying an application

The essential idea is that the Application Web Service (AWS) presents an XML interface that can be used to build application clients and is composed of core web service clients. Initially we can think of all of these parts (the XML interface to AWS, the AWS implementation consisting of SOAP clients, the corresponding SOAP services, and the AWS client interface) as all living on a single server, but these pieces can eventually be decoupled and each live on a separate (Tomcat) server.

Initially, portal developers such as the Gateway group develop all parts: the SOAP services and clients, the general Application Web service (consisting of several core web service clients) and the AWS clients in JSP. An application server is then installed on some host resource.

A particular application web service is now ready to be deployed. This amounts to filling in a set of web forms to create an instance of the Application Descriptor XML file (following the schema in Appendix A). The person responsible for making the application into a service just fills in the forms and optionally confirms all of the information. The Application Descriptor is created (or appended to if it already exists) and the service is deployed.

More sophisticated systems can be built. An additional general capability of Web services is that they may be discovered dynamically (through UDDI or similar systems) and may be scripted in a workflow (through WSFL, for example). Thus the next step is that the AWS is initially decoupled from the specific instances of the core web services that it uses. The AWS is composed of services that are scripted in some workflow language. The AWS then discovers and binds to particular core web services dynamically.

Using Application Web Services

We described above how to deploy an application web service but not how to use it. A user of an AWS has a separate user interface that is created from the AWS descriptors. These AWS descriptors give the user various choices and are used to generate forms needed to collect the information needed to run the code. The user's particular choices constitute a separate XML document, the Application Instance. This contains all the metadata about a particular invocation of the application (such as the particular input file that was used, the particular set of resources, and so on).

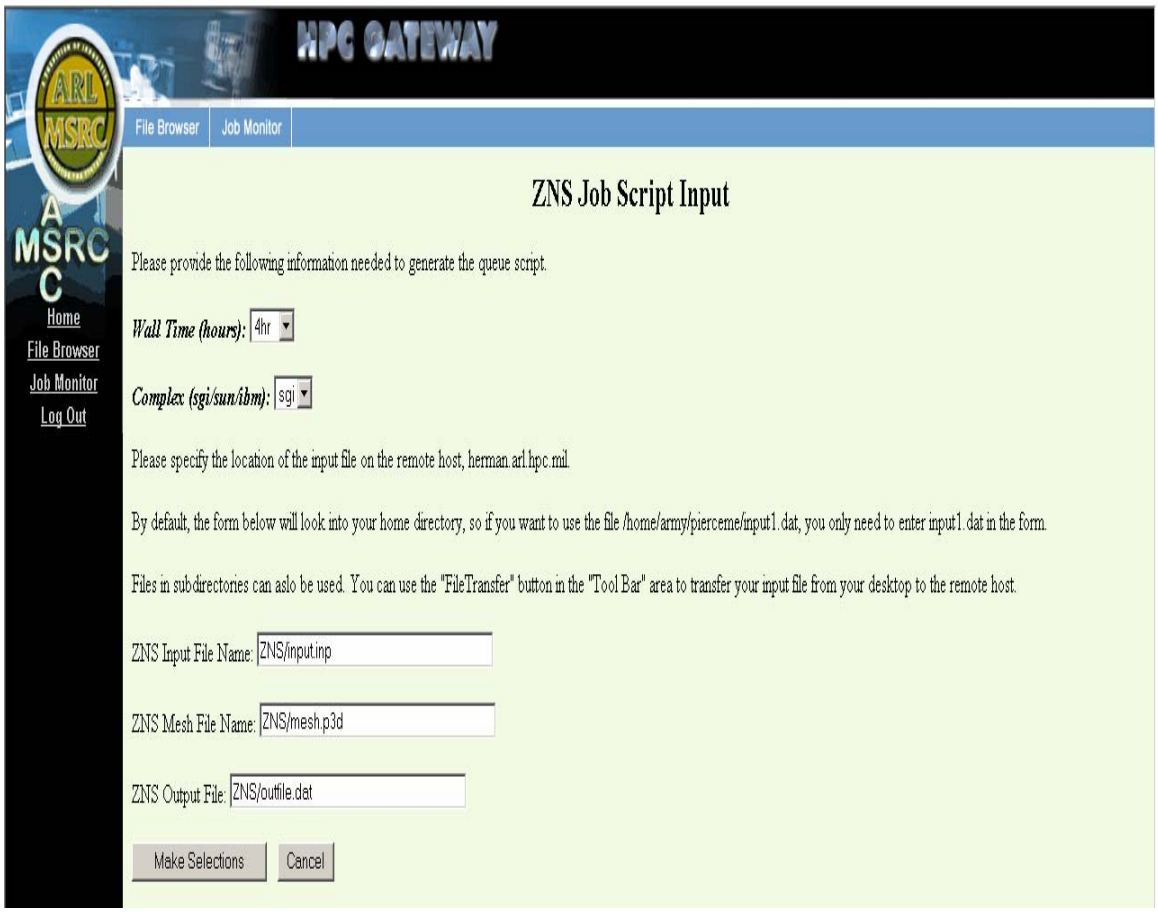


Figure 3 An input form for the application ZNS.

The Application Instance Descriptor (see Appendix D) serves as the guideline for building the user interface. In principal, this can be separated into client and server pieces. The client and server share the Application Instance interface, and the client collects information from the user about the application, which it then passes to the Application Instance server implementation (using SOAP, for example) where the application is invoked. Currently, we do not make this separation: the user interface also handles the application invocation directly.

Users fill out forms like the one shown in Figure 3. These forms are generated from the Application Descriptor for a particular application but result in Application Instance XML documents.

Portlets and Portals

As we have stated previously, the application web service defines an interface that describes how to create a client for that service. This client may be deployed on the same host computer as the application web service, but this is not required. Instead, the AWS interface definition should be viewed as a way to create clients (and associated user interfaces) on any host computer, just as WSDL defines how to create a client to invoke a remote method.

Previous web portals such as Gateway, as illustrated in Figures 2 and 3, did not distinguish between the code that processes the user interface and code that executes service requests. In the AWS architecture these can be decoupled, with one server responsible for creating and managing the user interface and user interaction. The user interface server implements clients to remote servers, which are invoked through SOAP.

The consequence of this is that user interfaces to both core web services (such as a job monitor or an LDAP browser) and application web services can be developed by a number of different groups, reusing a service deployed on some set of resources. The problem of how to manage all of these interfaces then arises. We believe that the next generation of computational portals will need to aggregate and manage these various user interfaces as components (or "portlets").

The key idea now is that a computational web portal actually is just a skeleton for holding and managing web interfaces to services, which may be delivered from either local or remote sources. The portal administrator picks the portlets that he or she wishes to make available, and the user customizes his interface to add the selected services. Thus a user may decorate his portal with the user interfaces to the core service "Job Monitoring on Computer A" and to "ANSYS Application Service," while another user may create a portal out of a completely different set of published user interfaces.

The portlet idea is already being realized by Jetspeed [8], an open source project from Apache. A common Java portlet API is currently being defined by the JSR 168 [9], and WSRP [10] (Web Services for Remote Portlets) is defining a web services based portlet framework.

References

[1] Application Metadata Working Group Web Site: http://ecs.erc.msstate.edu/AMD-WG/index.jsp.

[2] Grid Computing Environments Working Group Web Site: http://www.computingportals.org/.

[3] Gateway Computational Portal: http://www.gatewayportal.org.

[4] Grid 2002: http://www.grid2002.org

[5] http://www.nwfusion.com/news/2002/0627wssec.html

[6] http://www-106.ibm.com/developerworks/library/ws-secure/

[7] http://www.naradabrokering.org

[8] Simple Object Access Protocol (SOAP) 1.1: http://www.w3c.org/TR/SOAP/

[9] Apache Axis: http://xml.apache.org/axis/.

[10] Web Service Description Language (WSDL) 1.1: http://www.w3.org/TR/wsdl.

[11] Simple Wrapper Interface Generator (SWIG): http://www.swig.org/

[12] Jetspeed: http://jakarta.apache.org/jetspeed/site/index.html

[13] Portlet API Java Specification Request 168: http://www.jcp.org/jsr/detail/168.jsp

[14] Web Services for Remote Portlets (WSRP): http://www.oasis-open.org/committees/wsrp/

Appendix A: Application Descriptor Schema

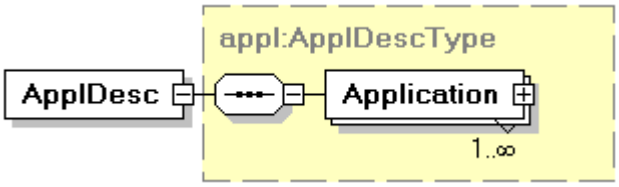
Schema **Appl.xsd**

schema location:	C:\castor\ApplDesc\Appl.xsd
targetNamespace:	http://grids.ucs.indiana.edu:8005/GCWS/Schema/Appl

Elements	Complex types	Simple types
ApplDesc	ApplDescType	mechanism
	ApplicationType	
	ErrorPortType	
	HostBindingType	
	InputPortType	
	OptionFlagType	
	OutputPortType	
	ParameterType	

element **ApplDesc**

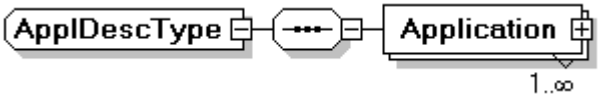
diagram



namespace	http://grids.ucs.indiana.edu:8005/GCWS/Schema/Appl
type	appl:ApplDescType
children	Application
source	<xsd:element name="ApplDesc" type="appl:ApplDescType"/>

complexType **ApplDescType**

diagram



namespace http://grids.ucs.indiana.edu:8005/GCWS/Schema/Appl

children Application

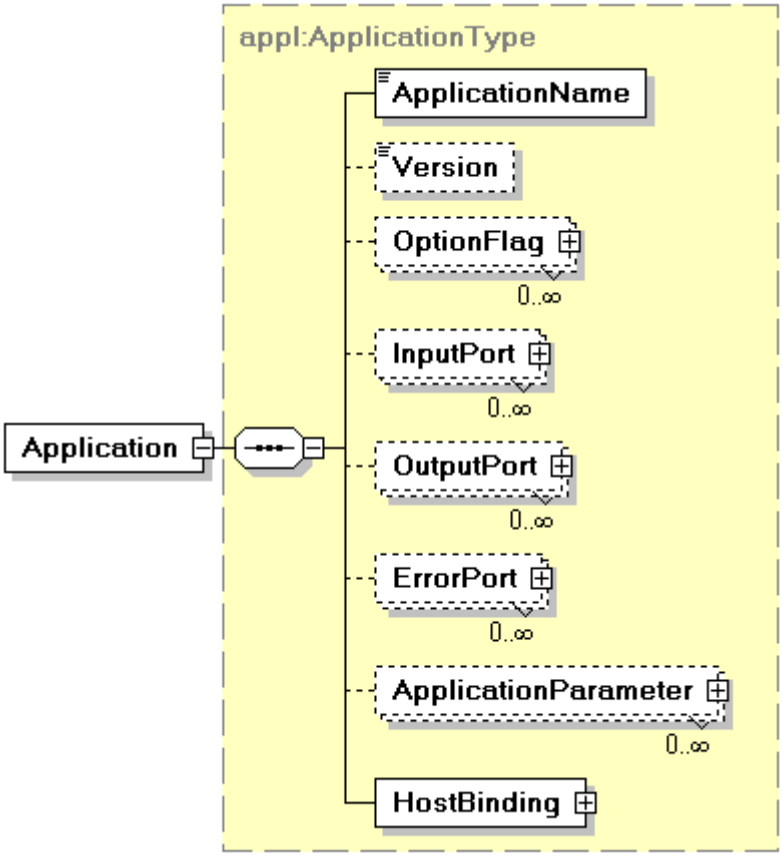
used by element ApplDesc

source

```
<xsd:complexType name="ApplDescType">
  <xsd:sequence>
    <xsd:element name="Application" type="appl:ApplicationType"
maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>
```

element **ApplDescType/Application**

diagram



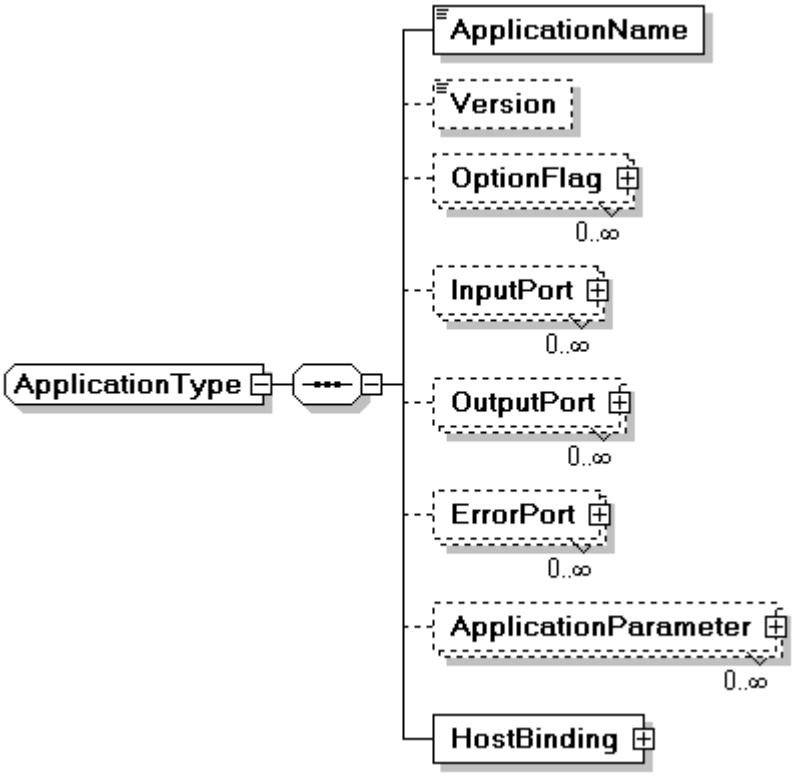
type appl:ApplicationType

children ApplicationName Version OptionFlag InputPort OutputPort ErrorPort ApplicationParameter HostBinding

source
```
<xsd:element name="Application" type="appl:ApplicationType" maxOccurs="unbounded"/>
```

complexType **ApplicationType**

diagram



namespace http://grids.ucs.indiana.edu:8005/GCWS/Schema/Appl

children ApplicationName Version OptionFlag InputPort OutputPort ErrorPort ApplicationParameter HostBinding

used by element ApplDescType/Application

source
```
<xsd:complexType name="ApplicationType">
  <xsd:sequence>
    <xsd:element name="ApplicationName" type="xsd:string"/>
    <xsd:element name="Version" type="xsd:string" minOccurs="0"/>
    <xsd:element name="OptionFlag" type="appl:OptionFlagType" minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element name="InputPort" type="appl:InputPortType" minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element name="OutputPort" type="appl:OutputPortType" minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element name="ErrorPort" type="appl:ErrorPortType" minOccurs="0" maxOccurs="unbounded"/>
```

```
    <xsd:element name="ApplicationParameter" type="appl:ParameterType"
minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element name="HostBinding" type="appl:HostBindingType"/>
  </xsd:sequence>
</xsd:complexType>
```

element **ApplicationType/ApplicationName**

diagram 

type **xsd:string**

source `<xsd:element name="ApplicationName" type="xsd:string"/>`

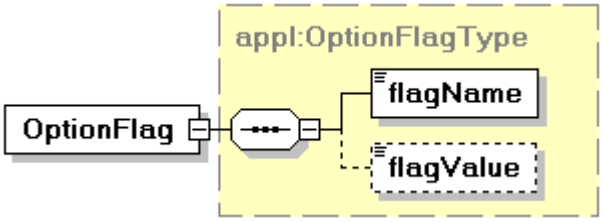
element **ApplicationType/Version**

diagram 

type **xsd:string**

source `<xsd:element name="Version" type="xsd:string" minOccurs="0"/>`

element **ApplicationType/OptionFlag**

diagram 

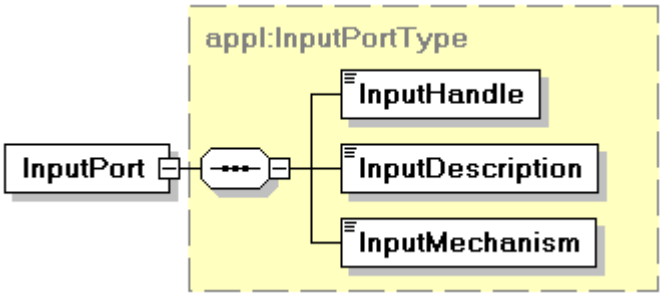
type appl:OptionFlagType

children flagName flagValue

source `<xsd:element name="OptionFlag" type="appl:OptionFlagType" minOccurs="0" maxOccurs="unbounded"/>`

element **ApplicationType/InputPort**

diagram



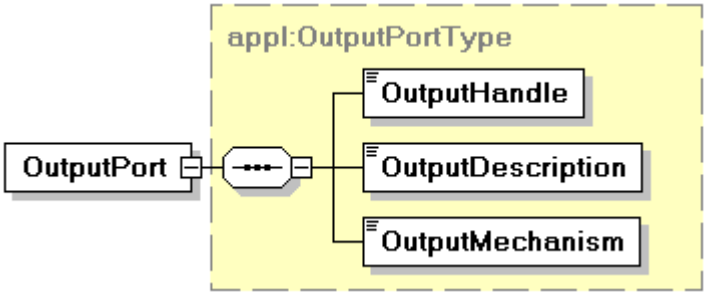
type appl:InputPortType

children InputHandle InputDescription InputMechanism

source
```
<xsd:element name="InputPort" type="appl:InputPortType" minOccurs="0"
maxOccurs="unbounded"/>
```

element **ApplicationType/OutputPort**

diagram



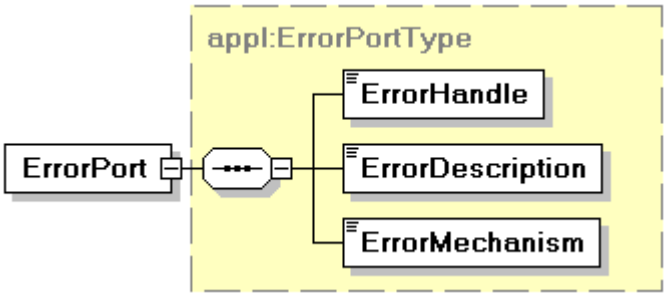
type appl:OutputPortType

children OutputHandle OutputDescription OutputMechanism

source
```
<xsd:element name="OutputPort" type="appl:OutputPortType" minOccurs="0"
maxOccurs="unbounded"/>
```

element **ApplicationType/ErrorPort**

diagram



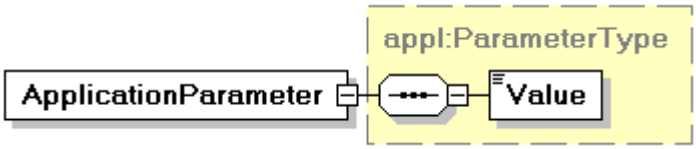
type appl:ErrorPortType

children ErrorHandle ErrorDescription ErrorMechanism

source
```
<xsd:element name="ErrorPort" type="appl:ErrorPortType" minOccurs="0" maxOccurs="unbounded"/>
```

element **ApplicationType/ApplicationParameter**

diagram



type appl:ParameterType

children Value

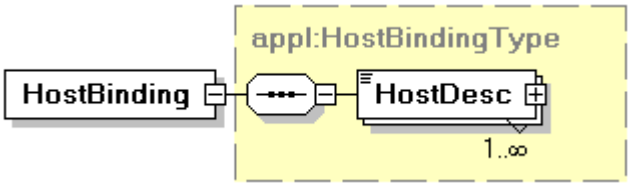
attributes

Name	Type	Use	Default	Fixed	Annotation
Name	xsd:string				

source
```
<xsd:element name="ApplicationParameter" type="appl:ParameterType" minOccurs="0" maxOccurs="unbounded"/>
```

element **ApplicationType/HostBinding**

diagram



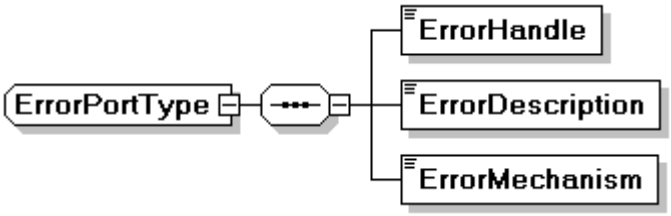
type appl:HostBindingType

children HostDesc

source
```
<xsd:element name="HostBinding" type="appl:HostBindingType"/>
```

complexType **ErrorPortType**

diagram



namespace http://grids.ucs.indiana.edu:8005/GCWS/Schema/Appl

children ErrorHandle ErrorDescription ErrorMechanism

used by element ApplicationType/ErrorPort

source

```
<xsd:complexType name="ErrorPortType">
  <xsd:sequence>
    <xsd:element name="ErrorHandle" type="xsd:string"/>
    <xsd:element name="ErrorDescription" type="xsd:string"/>
    <xsd:element name="ErrorMechanism" type="appl:mechanism"/>
  </xsd:sequence>
</xsd:complexType>
```

element **ErrorPortType/ErrorHandle**

diagram

type **xsd:string**

source

```
<xsd:element name="ErrorHandle" type="xsd:string"/>
```

element **ErrorPortType/ErrorDescription**

diagram

type **xsd:string**

source

```
<xsd:element name="ErrorDescription" type="xsd:string"/>
```

element **ErrorPortType/ErrorMechanism**

diagram

type appl:mechanism

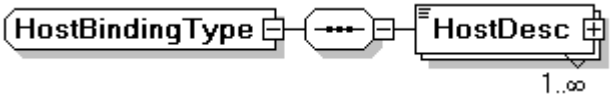
facets
enumeration StandardIO
enumeration CArgument

source

```
<xsd:element name="ErrorMechanism" type="appl:mechanism"/>
```

complexType **HostBindingType**

diagram



namespace http://grids.ucs.indiana.edu:8005/GCWS/Schema/Appl

children HostDesc

used by element ApplicationType/HostBinding

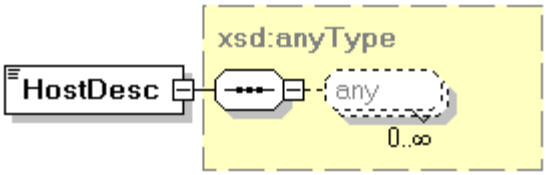
source

```
<xsd:complexType name="HostBindingType">
  <xsd:sequence>
```

```
    <xsd:element name="HostDesc" type="xsd:anyType" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>
```

element **HostBindingType/HostDesc**

diagram



type **xsd:anyType**

attributes

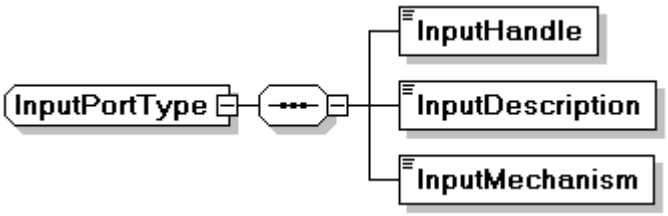
Name	Type	Use	Default	Fixed	Annotation

source

```
<xsd:element name="HostDesc" type="xsd:anyType" maxOccurs="unbounded"/>
```

complexType **InputPortType**

diagram



namespace http://grids.ucs.indiana.edu:8005/GCWS/Schema/Appl

children InputHandle InputDescription InputMechanism

used by element ApplicationType/InputPort

source

```
<xsd:complexType name="InputPortType">
  <xsd:sequence>
    <xsd:element name="InputHandle" type="xsd:string"/>
    <xsd:element name="InputDescription" type="xsd:string"/>
    <xsd:element name="InputMechanism" type="appl:mechanism"/>
  </xsd:sequence>
</xsd:complexType>
```

element **InputPortType/InputHandle**

diagram

type **xsd:string**

source

```
<xsd:element name="InputHandle" type="xsd:string"/>
```

element **InputPortType/InputDescription**

diagram 

type **xsd:string**

source
```
<xsd:element name="InputDescription" type="xsd:string"/>
```

element **InputPortType/InputMechanism**

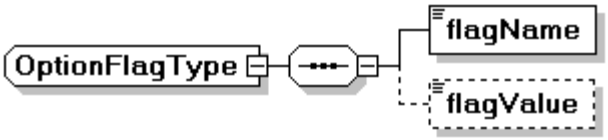
diagram 

type appl:mechanism

facets
enumeration StandardIO
enumeration CArgument

source
```
<xsd:element name="InputMechanism" type="appl:mechanism"/>
```

complexType **OptionFlagType**

diagram 

namespace http://grids.ucs.indiana.edu:8005/GCWS/Schema/Appl

children flagName flagValue

used by element ApplicationType/OptionFlag

source
```
<xsd:complexType name="OptionFlagType">
  <xsd:sequence>
    <xsd:element name="flagName" type="xsd:string"/>
    <xsd:element name="flagValue" type="xsd:boolean" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>
```

element **OptionFlagType/flagName**

diagram 

type **xsd:string**

source
```
<xsd:element name="flagName" type="xsd:string"/>
```

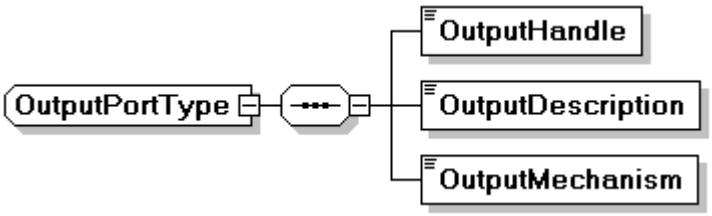
element **OptionFlagType/flagValue**

diagram 

type **xsd:boolean**

source
```
<xsd:element name="flagValue" type="xsd:boolean" minOccurs="0"/>
```

complexType **OutputPortType**

diagram 

namespace http://grids.ucs.indiana.edu:8005/GCWS/Schema/Appl

children OutputHandle OutputDescription OutputMechanism

used by element ApplicationType/OutputPort

source
```
<xsd:complexType name="OutputPortType">
  <xsd:sequence>
    <xsd:element name="OutputHandle" type="xsd:string"/>
    <xsd:element name="OutputDescription" type="xsd:string"/>
    <xsd:element name="OutputMechanism" type="appl:mechanism"/>
  </xsd:sequence>
</xsd:complexType>
```

element **OutputPortType/OutputHandle**

diagram 

type **xsd:string**

source
```
<xsd:element name="OutputHandle" type="xsd:string"/>
```

element **OutputPortType/OutputDescription**

diagram 

type **xsd:string**

source
```
<xsd:element name="OutputDescription" type="xsd:string"/>
```

element **OutputPortType/OutputMechanism**

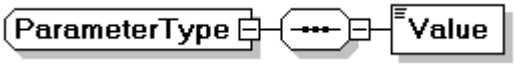
diagram 

type appl:mechanism

facets
enumeration StandardIO
enumeration CArgument

source
```
<xsd:element name="OutputMechanism" type="appl:mechanism"/>
```

complexType **ParameterType**

diagram 

namespace http://grids.ucs.indiana.edu:8005/GCWS/Schema/Appl

children Value

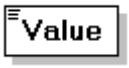
used by
element ApplicationType/ApplicationParameter

attributes

Name	Type	Use	Default	Fixed	Annotation
Name	xsd:string				

source
```
<xsd:complexType name="ParameterType">
  <xsd:sequence>
    <xsd:element name="Value" type="xsd:string"/>
  </xsd:sequence>
  <xsd:attribute name="Name" type="xsd:string"/>
</xsd:complexType>
```

element **ParameterType/Value**

diagram 

type **xsd:string**

source
```
<xsd:element name="Value" type="xsd:string"/>
```

simpleType **mechanism**

namespace http://grids.ucs.indiana.edu:8005/GCWS/Schema/Appl

type restriction of **xsd:string**

used by
elements
ErrorPortType/ErrorMechanism
InputPortType/InputMechanism
OutputPortType/OutputMechanism

facets
enumeration StandardIO
enumeration CArgument

source

```
<xsd:simpleType name="mechanism">
 <xsd:restriction base="xsd:string">
  <xsd:enumeration value="StandardIO"/>
  <xsd:enumeration value="CArgument"/>
 </xsd:restriction>
</xsd:simpleType>
```

XML Schema documentation generated with XML Spy Schema Editor www.xmlspy.com

```
<?xml version="1.0"?>
<!-- edited with XML Spy v4.4 U (http://www.xmlspy.com) by Choonhan
Youn (Florida State University) -->
<xsd:schema
targetNamespace="http://grids.ucs.indiana.edu:8005/GCWS/Schema/Appl"
xmlns:appl="http://grids.ucs.indiana.edu:8005/GCWS/Schema/Appl"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
      <xsd:annotation>
            <xsd:documentation>
        This schema is used to describe an application.
        </xsd:documentation>
      </xsd:annotation>
      <!--
   ApplDesc is the root element, with any number of applications.
   -->
      <xsd:element name="ApplDesc" type="appl:ApplDescType"/>
      <xsd:complexType name="ApplDescType">
            <xsd:sequence>
                  <xsd:element name="Application"
type="appl:ApplicationType" maxOccurs="unbounded"/>
            </xsd:sequence>
      </xsd:complexType>
      <!--
   Each application's info is contained by the tag ApplicationType.
   Applications have some specific required elements and optionally
   can be described by additional Parameters.  Examples of
   parameters would include environmental variables that
   might need to be set.
   -->
      <xsd:complexType name="ApplicationType">
            <xsd:sequence>
                  <xsd:element name="ApplicationName"
type="xsd:string"/>
                  <xsd:element name="Version" type="xsd:string"
minOccurs="0"/>
                  <xsd:element name="OptionFlag"
type="appl:OptionFlagType" minOccurs="0" maxOccurs="unbounded"/>
                  <xsd:element name="InputPort"
type="appl:InputPortType" minOccurs="0" maxOccurs="unbounded"/>
                  <xsd:element name="OutputPort"
type="appl:OutputPortType" minOccurs="0" maxOccurs="unbounded"/>
```

```
                  <xsd:element name="ErrorPort"
type="appl:ErrorPortType" minOccurs="0" maxOccurs="unbounded"/>
                  <xsd:element name="ApplicationParameter"
type="appl:ParameterType" minOccurs="0" maxOccurs="unbounded"/>
                  <xsd:element name="HostBinding"
type="appl:HostBindingType"/>
            </xsd:sequence>
      </xsd:complexType>
      <!--
   Code option flags.  These might be followed by arguments, depeding
   on the flag.
   -->
      <xsd:complexType name="OptionFlagType">
            <xsd:sequence>

                  <xsd:element name="flagName" type="xsd:string"/>
                  <xsd:element name="flagValue" type="xsd:boolean"
minOccurs="0"/>
            </xsd:sequence>
      </xsd:complexType>
      <!--
   InputPortType is used to describe the input methods (ie a file
   on local disk).

      0 InputHandle is a short name for a particular input type
      0 InputDescription is a long description of what type
        of input is expected (ie an ANSYS Prep7 file). Put
        instructions to users here.
      0 Mechanism is the way that file file takes input (such
        as by Unix-style standard input with a "<".
   -->
      <xsd:complexType name="InputPortType">
            <xsd:sequence>
                  <xsd:element name="InputHandle" type="xsd:string"/>
                  <xsd:element name="InputDescription"
type="xsd:string"/>
                  <xsd:element name="InputMechanism"
type="appl:mechanism"/>
            </xsd:sequence>
      </xsd:complexType>
      <!--
   OutputPortType is used to describe the output files (ie a file
   on local disk).  You need one of these tags for each
   output file the code generates.

      0 OutputHandle is a short name for a particular output type
      0 OutputDescription is a long description of a particular
        output file and is application-specific.
   -->
      <xsd:complexType name="OutputPortType">
            <xsd:sequence>
                  <xsd:element name="OutputHandle" type="xsd:string"/>
                  <xsd:element name="OutputDescription"
type="xsd:string"/>
                  <xsd:element name="OutputMechanism"
type="appl:mechanism"/>
            </xsd:sequence>
```

```
      </xsd:complexType>
      <!--
   ErrorPortType is used to describe the error files that may be
   generated.
      0 ErrorHandle is a short name for a particular error type
      0 ErrorDescription is a long description of what type
        of error.
   -->
      <xsd:complexType name="ErrorPortType">
            <xsd:sequence>
                  <xsd:element name="ErrorHandle" type="xsd:string"/>
                  <xsd:element name="ErrorDescription"
type="xsd:string"/>
                  <xsd:element name="ErrorMechanism"
type="appl:mechanism"/>
            </xsd:sequence>
      </xsd:complexType>
      <xsd:simpleType name="mechanism">
            <xsd:restriction base="xsd:string">
                  <xsd:enumeration value="StandardIO"/>
                  <xsd:enumeration value="CArgument"/>
            </xsd:restriction>
      </xsd:simpleType>
      <!--
   ParameterType is used for name/value pairs.
   -->
      <xsd:complexType name="ParameterType">
            <xsd:sequence>
                  <xsd:element name="Value" type="xsd:string"/>
            </xsd:sequence>
            <xsd:attribute name="Name" type="xsd:string"/>
      </xsd:complexType>
      <xsd:complexType name="HostBindingType">
            <xsd:sequence>
                  <xsd:element name="HostDesc" type="xsd:anyType"
maxOccurs="unbounded"/>
            </xsd:sequence>
      </xsd:complexType>
</xsd:schema>
```

Appendix B: Host Descriptor Schema

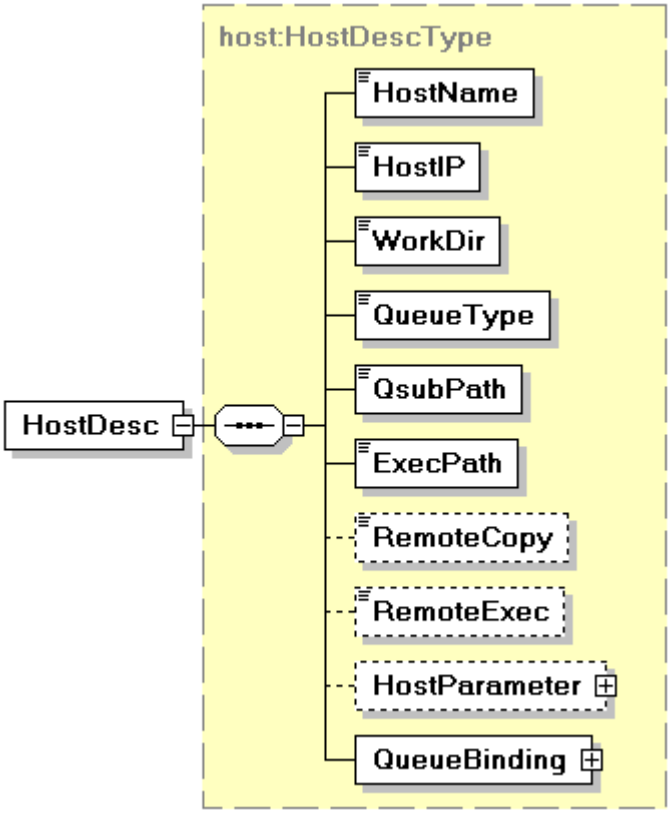
Schema **Host.xsd**

schema location:	C:\castor\ApplDesc\Host.xsd
targetNamespace:	http://grids.ucs.indiana.edu:8005/GCWS/Schema/Host

Elements	Complex types
HostDesc	HostDescType
	ParameterType
	QueueDescType

element **HostDesc**

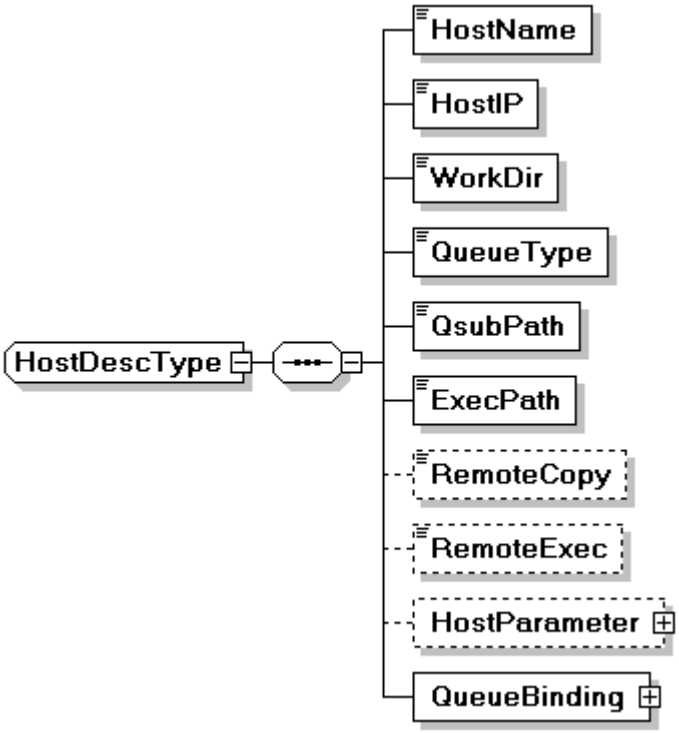
diagram



namespace	http://grids.ucs.indiana.edu:8005/GCWS/Schema/Host
type	host:HostDescType
children	HostName HostIP WorkDir QueueType QsubPath ExecPath RemoteCopy RemoteExec HostParameter QueueBinding
source	<xsd:element name="HostDesc" type="host:HostDescType"/>

complexType **HostDescType**

diagram



namespace http://grids.ucs.indiana.edu:8005/GCWS/Schema/Host

children HostName HostIP WorkDir QueueType QsubPath ExecPath RemoteCopy RemoteExec HostParameter QueueBinding

used by element HostDesc

source

```
<xsd:complexType name="HostDescType">
  <xsd:sequence>
    <xsd:element name="HostName" type="xsd:string"/>
    <xsd:element name="HostIP" type="xsd:string"/>
    <xsd:element name="WorkDir" type="xsd:string"/>
    <xsd:element name="QueueType" type="xsd:string"/>
    <xsd:element name="QsubPath" type="xsd:string"/>
    <xsd:element name="ExecPath" type="xsd:string"/>
    <xsd:element name="RemoteCopy" type="xsd:string" minOccurs="0"/>
    <xsd:element name="RemoteExec" type="xsd:string" minOccurs="0"/>
    <xsd:element name="HostParameter" type="host:ParameterType" minOccurs="0"/>
    <xsd:element name="QueueBinding" type="host:QueueDescType"/>
  </xsd:sequence>
</xsd:complexType>
```

element **HostDescType/HostName**

diagram 

type **xsd:string**

source <xsd:element name="HostName" type="xsd:string"/>

element **HostDescType/HostIP**

diagram 

type **xsd:string**

source <xsd:element name="HostIP" type="xsd:string"/>

element **HostDescType/WorkDir**

diagram 

type **xsd:string**

source <xsd:element name="WorkDir" type="xsd:string"/>

element **HostDescType/QueueType**

diagram 

type **xsd:string**

source <xsd:element name="QueueType" type="xsd:string"/>

element **HostDescType/QsubPath**

diagram 

type **xsd:string**

source <xsd:element name="QsubPath" type="xsd:string"/>

element **HostDescType/ExecPath**

diagram 

type **xsd:string**

source <xsd:element name="ExecPath" type="xsd:string"/>

element **HostDescType/RemoteCopy**

diagram 

type **xsd:string**

source <xsd:element name="RemoteCopy" type="xsd:string" minOccurs="0"/>

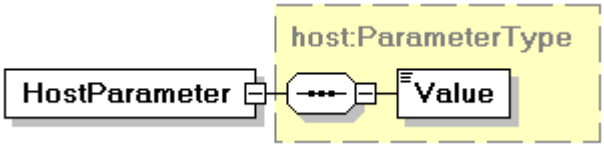
element **HostDescType/RemoteExec**

diagram 

type **xsd:string**

source <xsd:element name="RemoteExec" type="xsd:string" minOccurs="0"/>

element **HostDescType/HostParameter**

diagram 

type host:ParameterType

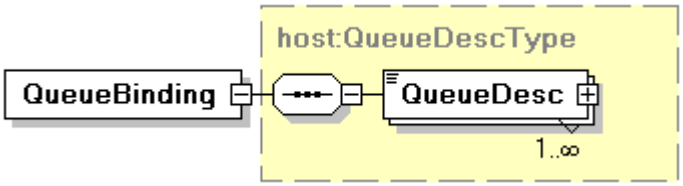
children Value

attributes

Name	Type	Use	Default	Fixed	Annotation
Name	xsd:string				

source <xsd:element name="HostParameter" type="host:ParameterType" minOccurs="0"/>

element **HostDescType/QueueBinding**

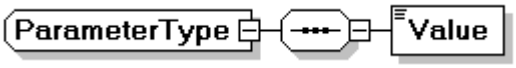
diagram 

type host:QueueDescType

children QueueDesc

source <xsd:element name="QueueBinding" type="host:QueueDescType"/>

complexType **ParameterType**

diagram 

namespace http://grids.ucs.indiana.edu:8005/GCWS/Schema/Host

children Value

used by element HostDescType/HostParameter

attributes

Name	Type	Use	Default	Fixed	Annotation
Name	xsd:string				

source

```
<xsd:complexType name="ParameterType">
  <xsd:sequence>
    <xsd:element name="Value" type="xsd:string"/>
  </xsd:sequence>
  <xsd:attribute name="Name" type="xsd:string"/>
</xsd:complexType>
```

element **ParameterType/Value**

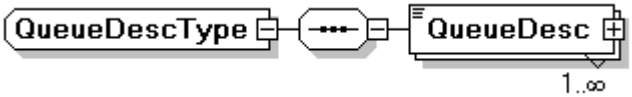
diagram 

type **xsd:string**

source

```
<xsd:element name="Value" type="xsd:string"/>
```

complexType **QueueDescType**

diagram 

namespace http://grids.ucs.indiana.edu:8005/GCWS/Schema/Host

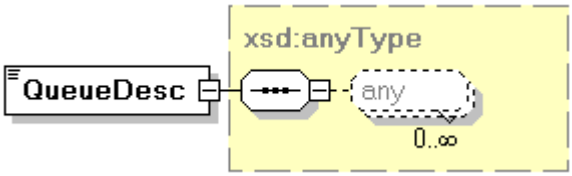
children QueueDesc

used by element HostDescType/QueueBinding

source

```
<xsd:complexType name="QueueDescType">
  <xsd:sequence>
    <xsd:element name="QueueDesc" type="xsd:anyType" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>
```

element **QueueDescType/QueueDesc**

diagram 

type **xsd:anyType**

attributes	Name	Type	Use	Default	Fixed	Annotation

source <xsd:element name="QueueDesc" type="xsd:anyType" maxOccurs="unbounded"/>

XML Schema documentation generated with XML Spy Schema Editor www.xmlspy.com

```
<?xml version="1.0"?>
<!-- edited with XML Spy v4.4 U (http://www.xmlspy.com) by Choonhan
Youn (Florida State University) -->
<xsd:schema
targetNamespace="http://grids.ucs.indiana.edu:8005/GCWS/Schema/Host"
xmlns:host="http://grids.ucs.indiana.edu:8005/GCWS/Schema/Host"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
      <xsd:annotation>
            <xsd:documentation>
        This schema describes host machine bindings for applications.
        </xsd:documentation>
      </xsd:annotation>
      <!--
   HostDesc is the root element, with any number of applications.
   HostDesc contains a description of a host machine. It contains
   several required fields and can be extended by an arbitrary
   number of parameters.  Optionally, you can provide commands
   for remote copy and execution (rsh, rcp, globusrun).  This must
   be set up externally.

   Hosts also contain one or more bindings for queue execution.  This
   could be a queuing system (PBS, GRD) or it could be
   -->
      <xsd:element name="HostDesc" type="host:HostDescType"/>
      <xsd:complexType name="HostDescType">
            <xsd:sequence>
                  <xsd:element name="HostName" type="xsd:string"/>
                  <xsd:element name="HostIP" type="xsd:string"/>
                  <xsd:element name="WorkDir" type="xsd:string"/>
                  <xsd:element name="QueueType" type="xsd:string"/>
                  <xsd:element name="QsubPath" type="xsd:string"/>
                  <xsd:element name="ExecPath" type="xsd:string"/>
                  <xsd:element name="RemoteCopy" type="xsd:string"
minOccurs="0"/>
                  <xsd:element name="RemoteExec" type="xsd:string"
minOccurs="0"/>
```

```
                  <xsd:element name="HostParameter"
type="host:ParameterType" minOccurs="0"/>
                  <xsd:element name="QueueBinding"
type="host:QueueBindingType"/>
            </xsd:sequence>
      </xsd:complexType>
      <xsd:complexType name="ParameterType">
            <xsd:sequence>
                  <xsd:element name="Value" type="xsd:string"/>
            </xsd:sequence>
            <xsd:attribute name="Name" type="xsd:string"/>
      </xsd:complexType>
      <xsd:complexType name="QueueBindingType">
            <xsd:sequence>
                  <xsd:element name="QueueDesc" type="xsd:anyType"
maxOccurs="unbounded"/>
            </xsd:sequence>
      </xsd:complexType>
</xsd:schema>
```

Appendix C: Queue Descriptor Schema

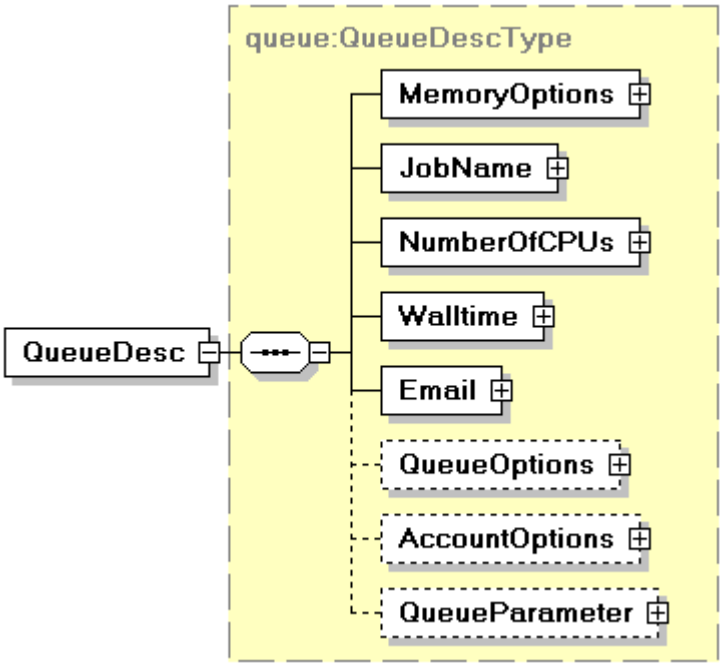
Schema **Queue.xsd**

schema location:	C:\castor\ApplDesc\Queue.xsd
targetNamespace:	http://grids.ucs.indiana.edu:8005/GCWS/Schema/Queue

Elements	Complex types
QueueDesc	EmailOptType
	OptionType
	ParameterType
	QueueDescType

element **QueueDesc**

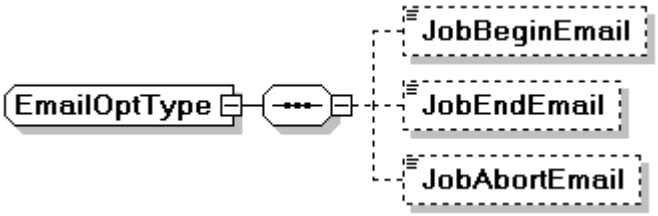
diagram



namespace	http://grids.ucs.indiana.edu:8005/GCWS/Schema/Queue
type	queue:QueueDescType
children	MemoryOptions JobName NumberOfCPUs Walltime Email QueueOptions AccountOptions QueueParameter
source	`<xsd:element name="QueueDesc" type="queue:QueueDescType"/>`

complexType **EmailOptType**

diagram



namespace		http://grids.ucs.indiana.edu:8005/GCWS/Schema/Queue
children		JobBeginEmail JobEndEmail JobAbortEmail
used by	element	QueueDescType/Email

source

```
<xsd:complexType name="EmailOptType">
  <xsd:sequence>
    <xsd:element name="JobBeginEmail" type="xsd:string" minOccurs="0"/>
    <xsd:element name="JobEndEmail" type="xsd:string" minOccurs="0"/>
    <xsd:element name="JobAbortEmail" type="xsd:string" minOccurs="0"/>
```

```
  </xsd:sequence>
</xsd:complexType>
```

element **EmailOptType/JobBeginEmail**

diagram 

type **xsd:string**

source
```
<xsd:element name="JobBeginEmail" type="xsd:string" minOccurs="0"/>
```

element **EmailOptType/JobEndEmail**

diagram 

type **xsd:string**

source
```
<xsd:element name="JobEndEmail" type="xsd:string" minOccurs="0"/>
```

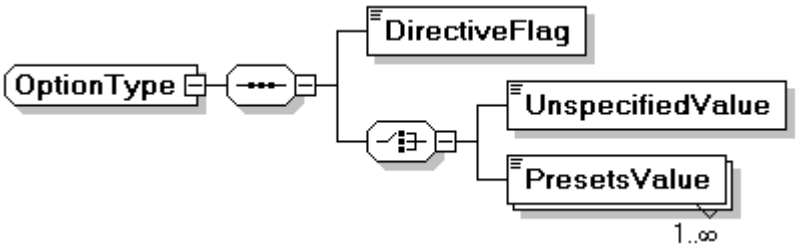
element **EmailOptType/JobAbortEmail**

diagram 

type **xsd:string**

source
```
<xsd:element name="JobAbortEmail" type="xsd:string" minOccurs="0"/>
```

complexType **OptionType**

diagram 

namespace http://grids.ucs.indiana.edu:8005/GCWS/Schema/Queue

children DirectiveFlag UnspecifiedValue PresetsValue

used by elements QueueDescType/AccountOptions QueueDescType/JobName QueueDescType/MemoryOptions QueueDescType/NumberOfCPUs QueueDescType/QueueOptions QueueDescType/Walltime

source
```
<xsd:complexType name="OptionType">
  <xsd:sequence>
    <xsd:element name="DirectiveFlag" type="xsd:string"/>
    <xsd:choice>
```

```
    <xsd:element name="UnspecifiedValue" type="xsd:string"/>
    <xsd:element name="PresetsValue" type="xsd:string" maxOccurs="unbounded"/>
   </xsd:choice>
  </xsd:sequence>
</xsd:complexType>
```

element **OptionType/DirectiveFlag**

diagram 

type **xsd:string**

source
```
<xsd:element name="DirectiveFlag" type="xsd:string"/>
```

element **OptionType/UnspecifiedValue**

diagram 

type **xsd:string**

source
```
<xsd:element name="UnspecifiedValue" type="xsd:string"/>
```

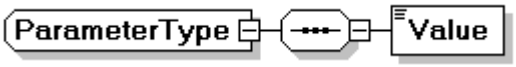
element **OptionType/PresetsValue**

diagram 

type **xsd:string**

source
```
<xsd:element name="PresetsValue" type="xsd:string" maxOccurs="unbounded"/>
```

complexType **ParameterType**

diagram 

namespace http://grids.ucs.indiana.edu:8005/GCWS/Schema/Queue

children Value

used by element QueueDescType/QueueParameter

attributes

Name	Type	Use	Default	Fixed	Annotation
Name	xsd:string				

source
```
<xsd:complexType name="ParameterType">
 <xsd:sequence>
  <xsd:element name="Value" type="xsd:string"/>
 </xsd:sequence>
 <xsd:attribute name="Name" type="xsd:string"/>
```

```
</xsd:complexType>
```

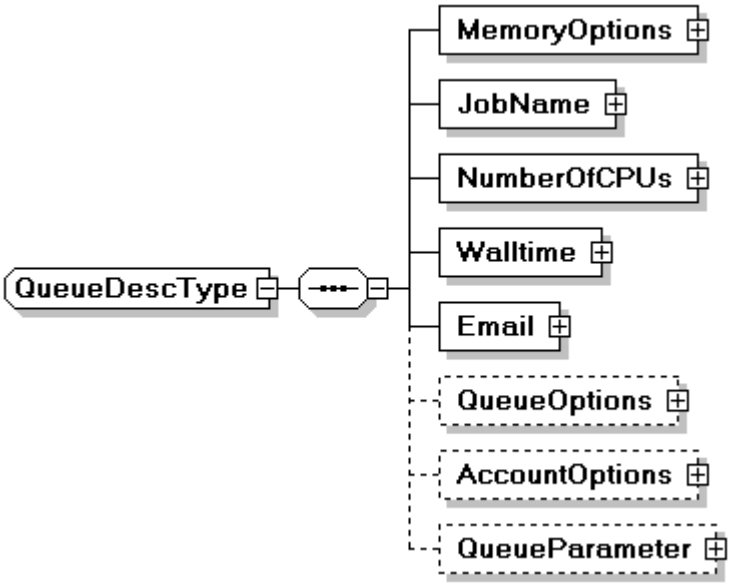
element **ParameterType/Value**

diagram 

type **xsd:string**

source
```
<xsd:element name="Value" type="xsd:string"/>
```

complexType **QueueDescType**

diagram 

namespace http://grids.ucs.indiana.edu:8005/GCWS/Schema/Queue

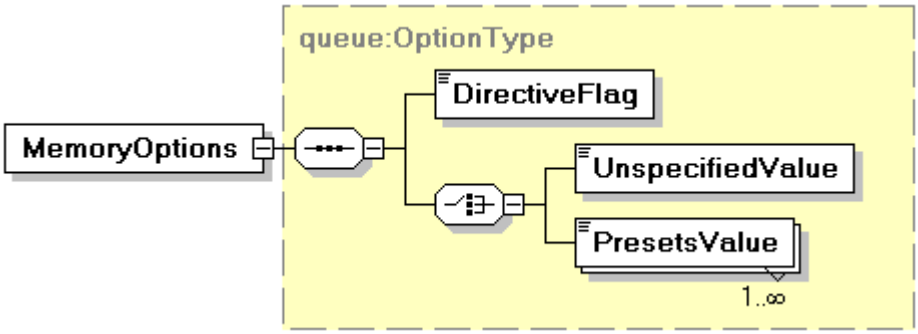
children MemoryOptions JobName NumberOfCPUs Walltime Email QueueOptions AccountOptions QueueParameter

used by element QueueDesc

source
```
<xsd:complexType name="QueueDescType">
  <xsd:sequence>
    <xsd:element name="MemoryOptions" type="queue:OptionType"/>
    <xsd:element name="JobName" type="queue:OptionType"/>
    <xsd:element name="NumberOfCPUs" type="queue:OptionType"/>
    <xsd:element name="Walltime" type="queue:OptionType"/>
    <xsd:element name="Email" type="queue:EmailOptType"/>
    <xsd:element name="QueueOptions" type="queue:OptionType" minOccurs="0"/>
    <xsd:element name="AccountOptions" type="queue:OptionType" minOccurs="0"/>
    <xsd:element name="QueueParameter" type="queue:ParameterType"
minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>
```

element **QueueDescType/MemoryOptions**

diagram



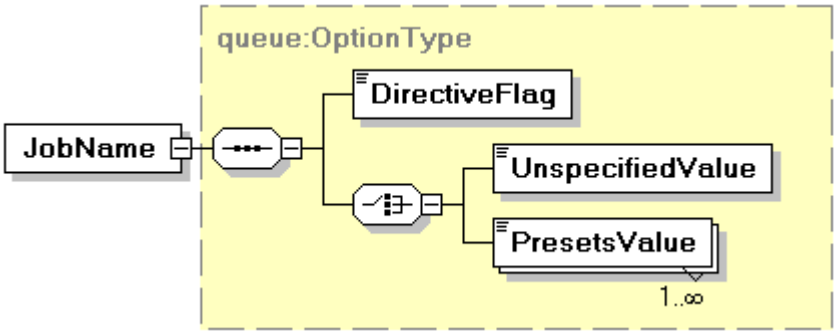
type queue:OptionType

children DirectiveFlag UnspecifiedValue PresetsValue

source `<xsd:element name="MemoryOptions" type="queue:OptionType"/>`

element **QueueDescType/JobName**

diagram



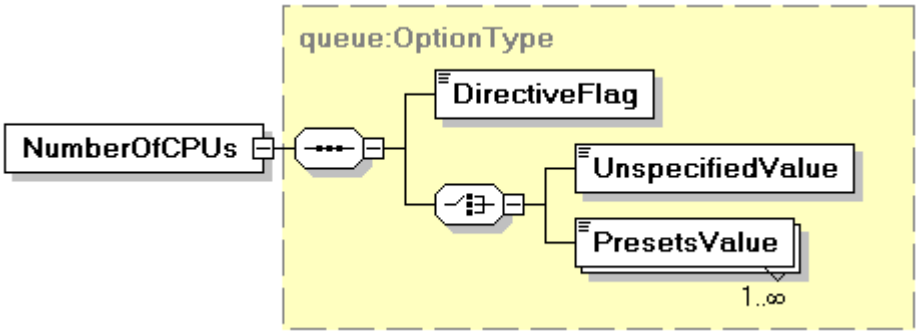
type queue:OptionType

children DirectiveFlag UnspecifiedValue PresetsValue

source `<xsd:element name="JobName" type="queue:OptionType"/>`

element **QueueDescType/NumberOfCPUs**

diagram



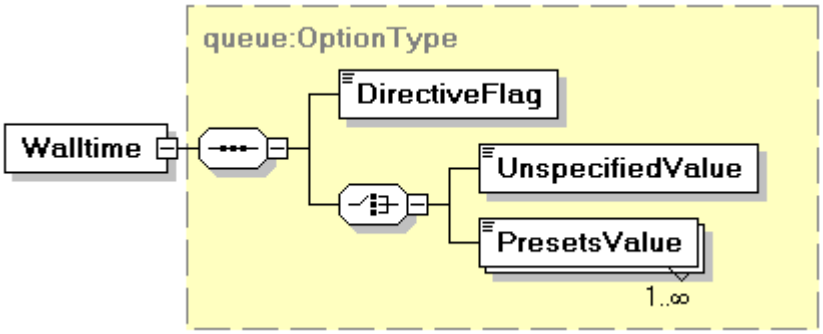
type queue:OptionType

children DirectiveFlag UnspecifiedValue PresetsValue

source <xsd:element name="NumberOfCPUs" type="queue:OptionType"/>

element **QueueDescType/Walltime**

diagram



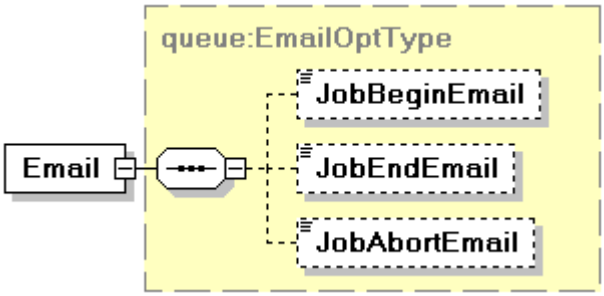
type queue:OptionType

children DirectiveFlag UnspecifiedValue PresetsValue

source <xsd:element name="Walltime" type="queue:OptionType"/>

element **QueueDescType/Email**

diagram



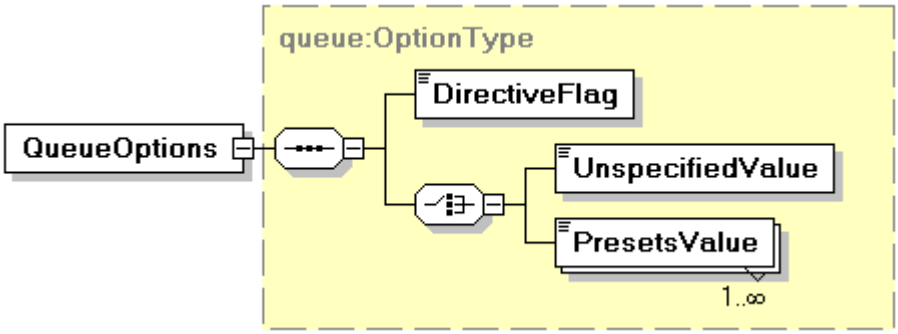
type queue:EmailOptType

children JobBeginEmail JobEndEmail JobAbortEmail

source <xsd:element name="Email" type="queue:EmailOptType"/>

element **QueueDescType/QueueOptions**

diagram



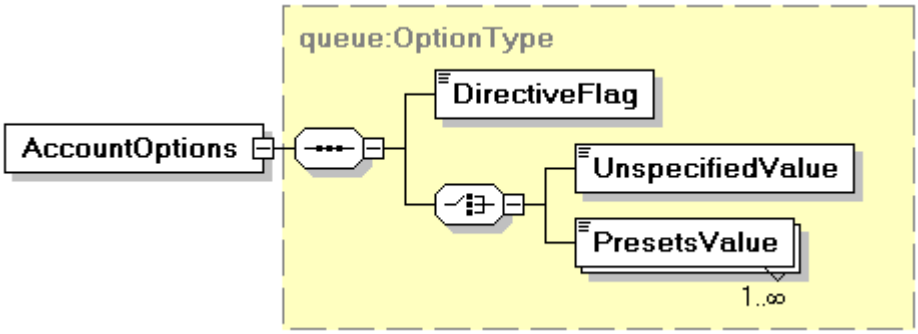
type queue:OptionType

children DirectiveFlag UnspecifiedValue PresetsValue

source `<xsd:element name="QueueOptions" type="queue:OptionType" minOccurs="0"/>`

element **QueueDescType/AccountOptions**

diagram



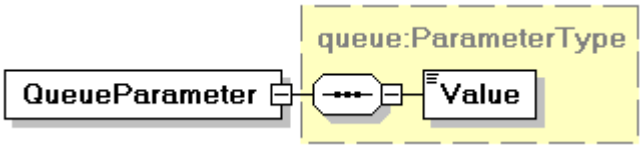
type queue:OptionType

children DirectiveFlag UnspecifiedValue PresetsValue

source `<xsd:element name="AccountOptions" type="queue:OptionType" minOccurs="0"/>`

element **QueueDescType/QueueParameter**

diagram



type queue:ParameterType

children Value

attributes

Name	Type	Use	Default	Fixed	Annotation
Name	xsd:string				

source <xsd:element name="QueueParameter" type="queue:ParameterType" minOccurs="0"/>

XML Schema documentation generated with XML Spy Schema Editor www.xmlspy.com

```
<?xml version="1.0"?>
<!-- edited with XML Spy v4.4 U (http://www.xmlspy.com) by Choonhan
Youn (Florida State University) -->
<xsd:schema
targetNamespace="http://grids.ucs.indiana.edu:8005/GCWS/Schema/Queue"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:queue="http://grids.ucs.indiana.edu:8005/GCWS/Schema/Queue">
      <xsd:annotation>
            <xsd:documentation>
        This schema describes queuing systems on a host.
        </xsd:documentation>
      </xsd:annotation>
      <!--
   QueueDesc is the root element and describes a queuing system
   parameters.  An instance of this schema describes a particular
   queuing system (such as PBS).  Queue systems that need
   additional values not explicitly stated in the
   -->
      <xsd:element name="QueueDesc" type="queue:QueueDescType"/>
      <xsd:complexType name="QueueDescType">
            <xsd:sequence>
                  <xsd:element name="MemoryOptions"
type="queue:OptionType"/>
                  <xsd:element name="JobName" type="queue:OptionType"/>
                  <xsd:element name="NumberOfCPUs"
type="queue:OptionType"/>
                  <xsd:element name="Walltime"
type="queue:OptionType"/>
                  <xsd:element name="Email" type="queue:EmailOptType"/>
                  <xsd:element name="QueueOptions"
type="queue:OptionType" minOccurs="0"/>
                  <xsd:element name="AccountOptions"
type="queue:OptionType" minOccurs="0"/>
                  <xsd:element name="QueueParameter"
type="queue:ParameterType" minOccurs="0"/>
            </xsd:sequence>
      </xsd:complexType>
      <!--
   This allows us to describe the queue's memory options (ie
   1 GB, 4 GB, etc.)
   -->
      <xsd:complexType name="EmailOptType">
            <xsd:sequence>
                  <xsd:element name="JobBeginEmail" type="xsd:string"
minOccurs="0"/>
                  <xsd:element name="JobEndEmail" type="xsd:string"
minOccurs="0"/>
```

```
                  <xsd:element name="JobAbortEmail" type="xsd:string"
minOccurs="0"/>
            </xsd:sequence>
      </xsd:complexType>
      <!--
   This allows different subqueues to be supported (ie default,
   priority, large, medium, small).
   -->
      <xsd:complexType name="ParameterType">
            <xsd:sequence>
                  <xsd:element name="Value" type="xsd:string"/>
            </xsd:sequence>
            <xsd:attribute name="Name" type="xsd:string"/>
      </xsd:complexType>
      <xsd:complexType name="OptionType">
            <xsd:sequence>
                  <xsd:element name="DirectiveFlag" type="xsd:string"/>
                  <xsd:choice>
                        <xsd:element name="UnspecifiedValue"
type="xsd:string"/>
                        <xsd:element name="PresetsValue"
type="xsd:string" maxOccurs="unbounded"/>
                  </xsd:choice>
            </xsd:sequence>
      </xsd:complexType>
</xsd:schema>
```

Appendix D: Application Instance Descriptor Schema

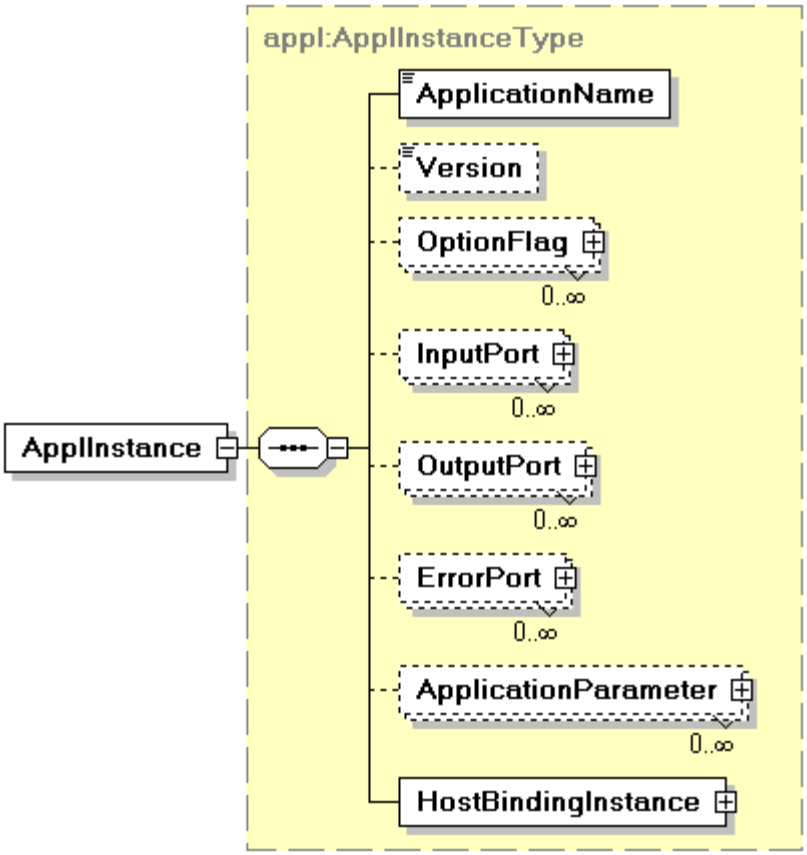
Schema **Applins.xsd**

schema location:	C:\castor\ApplDesc\Applins.xsd
targetNamespace:	http://grids.ucs.indiana.edu:8005/GCWS/Schema/Appll

Elements	Complex types	Simple types
ApplInstance	ApplInstanceType	mechanism
	ErrorPortType	
	HostBindingType	
	InputPortType	
	OptionFlagType	
	OutputPortType	
	ParameterType	

element **AppIInstance**

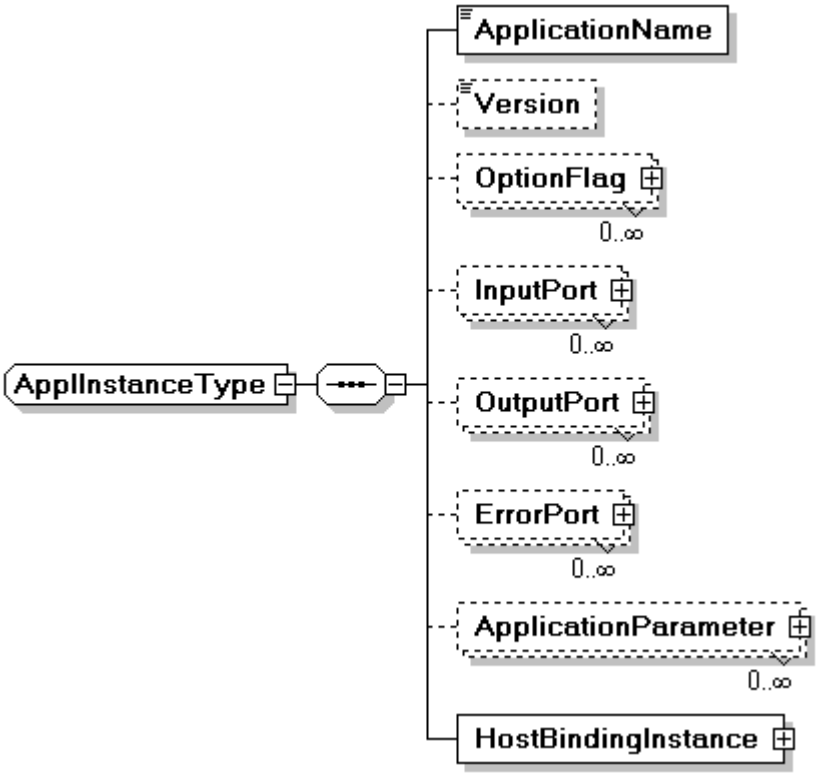
diagram



namespace	http://grids.ucs.indiana.edu:8005/GCWS/Schema/AppII
type	appl:AppIInstanceType
children	ApplicationName Version OptionFlag InputPort OutputPort ErrorPort ApplicationParameter HostBindingInstance
source	<xsd:element name="AppIInstance" type="appl:AppIInstanceType"/>

complexType **ApplInstanceType**

diagram



namespace http://grids.ucs.indiana.edu:8005/GCWS/Schema/ApplI

children ApplicationName Version OptionFlag InputPort OutputPort ErrorPort ApplicationParameter HostBindingInstance

used by element ApplInstance

source

```
<xsd:complexType name="ApplInstanceType">
  <xsd:sequence>
    <xsd:element name="ApplicationName" type="xsd:string"/>
    <xsd:element name="Version" type="xsd:string" minOccurs="0"/>
    <xsd:element name="OptionFlag" type="appl:OptionFlagType" minOccurs="0"
maxOccurs="unbounded"/>
    <xsd:element name="InputPort" type="appl:InputPortType" minOccurs="0"
maxOccurs="unbounded"/>
    <xsd:element name="OutputPort" type="appl:OutputPortType" minOccurs="0"
maxOccurs="unbounded"/>
    <xsd:element name="ErrorPort" type="appl:ErrorPortType" minOccurs="0"
maxOccurs="unbounded"/>
    <xsd:element name="ApplicationParameter" type="appl:ParameterType"
minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element name="HostBindingInstance" type="appl:HostBindingType"/>
  </xsd:sequence>
</xsd:complexType>
```

element **ApplInstanceType/ApplicationName**

diagram 

type **xsd:string**

source <xsd:element name="ApplicationName" type="xsd:string"/>

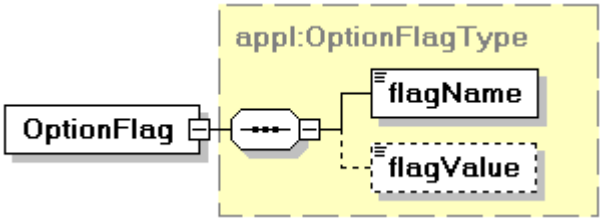
element **ApplInstanceType/Version**

diagram 

type **xsd:string**

source <xsd:element name="Version" type="xsd:string" minOccurs="0"/>

element **ApplInstanceType/OptionFlag**

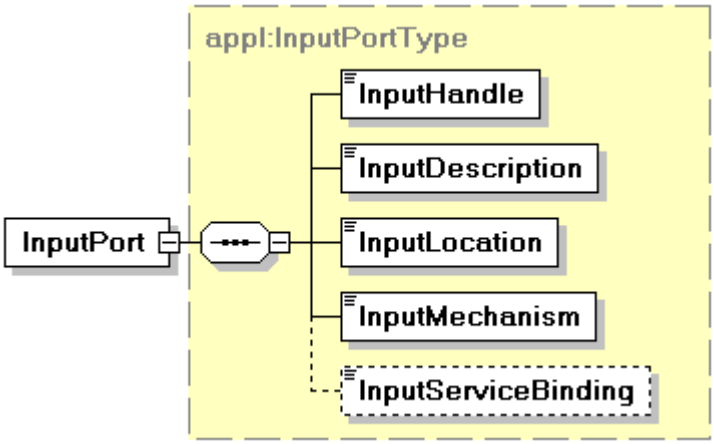
diagram 

type appl:OptionFlagType

children flagName flagValue

source <xsd:element name="OptionFlag" type="appl:OptionFlagType" minOccurs="0" maxOccurs="unbounded"/>

element **ApplInstanceType/InputPort**

diagram 

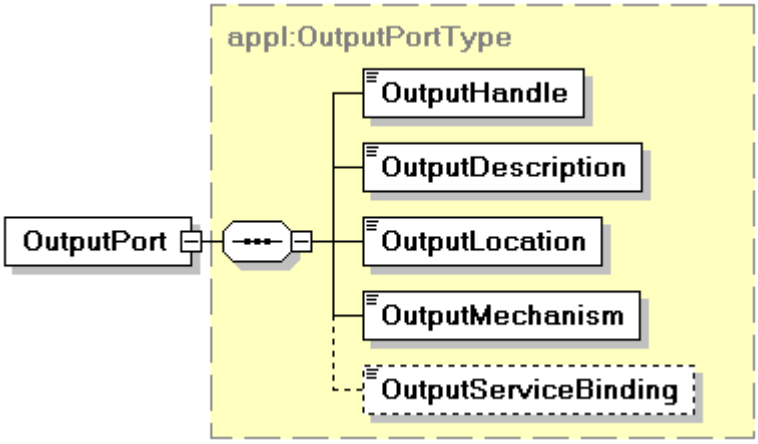
type appl:InputPortType

children InputHandle InputDescription InputLocation InputMechanism InputServiceBinding

source <xsd:element name="InputPort" type="appl:InputPortType" minOccurs="0" maxOccurs="unbounded"/>

element ApplInstanceType/OutputPort

diagram



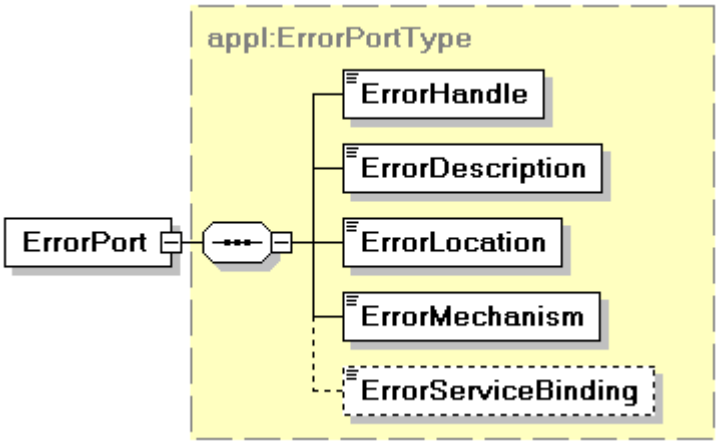
type appl:OutputPortType

children OutputHandle OutputDescription OutputLocation OutputMechanism OutputServiceBinding

source <xsd:element name="OutputPort" type="appl:OutputPortType" minOccurs="0" maxOccurs="unbounded"/>

element ApplInstanceType/ErrorPort

diagram



type appl:ErrorPortType

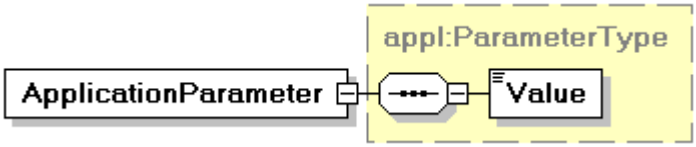
children ErrorHandle ErrorDescription ErrorLocation ErrorMechanism ErrorServiceBinding

source <xsd:element name="ErrorPort" type="appl:ErrorPortType" minOccurs="0"

```
maxOccurs="unbounded"/>
```

element **ApplInstanceType/ApplicationParameter**

diagram



type appl:ParameterType

children Value

attributes

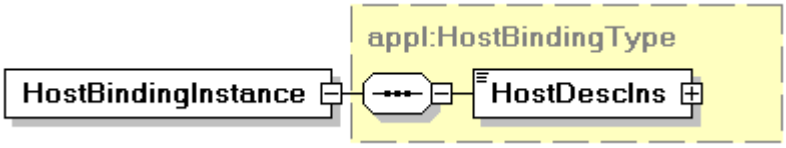
Name	Type	Use	Default	Fixed	Annotation
Name	xsd:string				

source

```
<xsd:element name="ApplicationParameter" type="appl:ParameterType"
minOccurs="0" maxOccurs="unbounded"/>
```

element **ApplInstanceType/HostBindingInstance**

diagram



type appl:HostBindingType

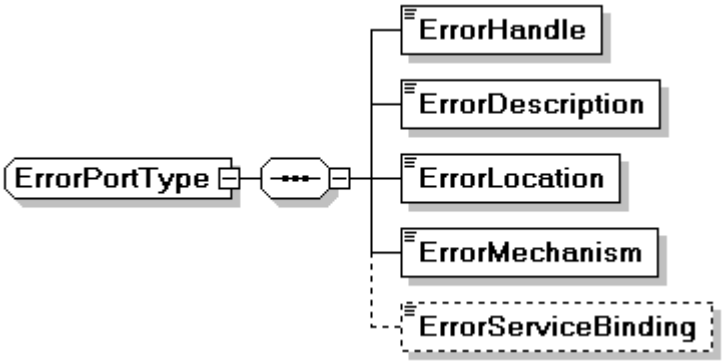
children HostDescIns

source

```
<xsd:element name="HostBindingInstance" type="appl:HostBindingType"/>
```

complexType **ErrorPortType**

diagram



namespace http://grids.ucs.indiana.edu:8005/GCWS/Schema/ApplI

children ErrorHandle ErrorDescription ErrorLocation ErrorMechanism ErrorServiceBinding

used by element ApplInstanceType/ErrorPort

source

```
<xsd:complexType name="ErrorPortType">
  <xsd:sequence>
    <xsd:element name="ErrorHandle" type="xsd:string"/>
    <xsd:element name="ErrorDescription" type="xsd:string"/>
    <xsd:element name="ErrorLocation" type="xsd:anyURI"/>
    <xsd:element name="ErrorMechanism" type="appl:mechanism"/>
    <xsd:element name="ErrorServiceBinding" type="xsd:string" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>
```

element **ErrorPortType/ErrorHandle**

diagram 

type **xsd:string**

source

```
<xsd:element name="ErrorHandle" type="xsd:string"/>
```

element **ErrorPortType/ErrorDescription**

diagram 

type **xsd:string**

source

```
<xsd:element name="ErrorDescription" type="xsd:string"/>
```

element **ErrorPortType/ErrorLocation**

diagram 

type **xsd:anyURI**

source

```
<xsd:element name="ErrorLocation" type="xsd:anyURI"/>
```

element **ErrorPortType/ErrorMechanism**

diagram 

type appl:mechanism

facets
enumeration StandardIO
enumeration CArgument

source

```
<xsd:element name="ErrorMechanism" type="appl:mechanism"/>
```

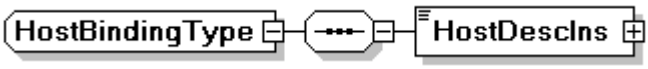
element **ErrorPortType/ErrorServiceBinding**

diagram 

type **xsd:string**

source
```
<xsd:element name="ErrorServiceBinding" type="xsd:string" minOccurs="0"/>
```

complexType **HostBindingType**

diagram 

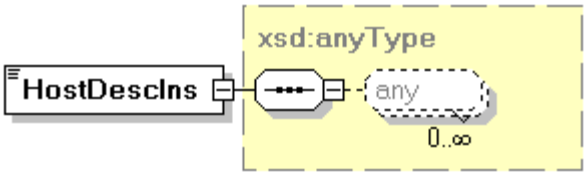
namespace http://grids.ucs.indiana.edu:8005/GCWS/Schema/AppII

children HostDescIns

used by element AppIInstanceType/HostBindingInstance

source
```
<xsd:complexType name="HostBindingType">
  <xsd:sequence>
    <xsd:element name="HostDescIns" type="xsd:anyType"/>
  </xsd:sequence>
</xsd:complexType>
```

element **HostBindingType/HostDescIns**

diagram 

type **xsd:anyType**

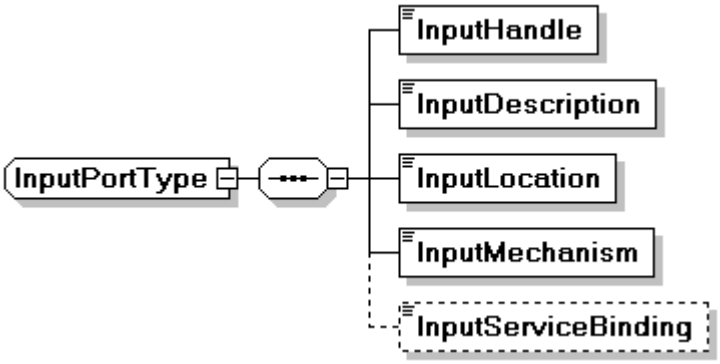
attributes

Name	Type	Use	Default	Fixed	Annotation

source
```
<xsd:element name="HostDescIns" type="xsd:anyType"/>
```

complexType **InputPortType**

diagram



namespace http://grids.ucs.indiana.edu:8005/GCWS/Schema/AppII

children InputHandle InputDescription InputLocation InputMechanism InputServiceBinding

used by element AppIInstanceType/InputPort

source

```
<xsd:complexType name="InputPortType">
  <xsd:sequence>
    <xsd:element name="InputHandle" type="xsd:string"/>
    <xsd:element name="InputDescription" type="xsd:string"/>
    <xsd:element name="InputLocation" type="xsd:anyURI"/>
    <xsd:element name="InputMechanism" type="appl:mechanism"/>
    <xsd:element name="InputServiceBinding" type="xsd:string" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>
```

element **InputPortType/InputHandle**

diagram

type **xsd:string**

source

```
<xsd:element name="InputHandle" type="xsd:string"/>
```

element **InputPortType/InputDescription**

diagram

type **xsd:string**

source

```
<xsd:element name="InputDescription" type="xsd:string"/>
```

element **InputPortType/InputLocation**

diagram 

type **xsd:anyURI**

source
```
<xsd:element name="InputLocation" type="xsd:anyURI"/>
```

element **InputPortType/InputMechanism**

diagram 

type appl:mechanism

facets
enumeration StandardIO
enumeration CArgument

source
```
<xsd:element name="InputMechanism" type="appl:mechanism"/>
```

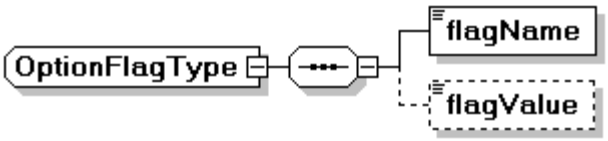
element **InputPortType/InputServiceBinding**

diagram 

type **xsd:string**

source
```
<xsd:element name="InputServiceBinding" type="xsd:string" minOccurs="0"/>
```

complexType **OptionFlagType**

diagram 

namespace http://grids.ucs.indiana.edu:8005/GCWS/Schema/AppII

children flagName flagValue

used by element AppIInstanceType/OptionFlag

source
```
<xsd:complexType name="OptionFlagType">
  <xsd:sequence>
    <xsd:element name="flagName" type="xsd:string"/>
    <xsd:element name="flagValue" type="xsd:string" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>
```

element **OptionFlagType/flagName**

diagram 

type **xsd:string**

source
```
<xsd:element name="flagName" type="xsd:string"/>
```

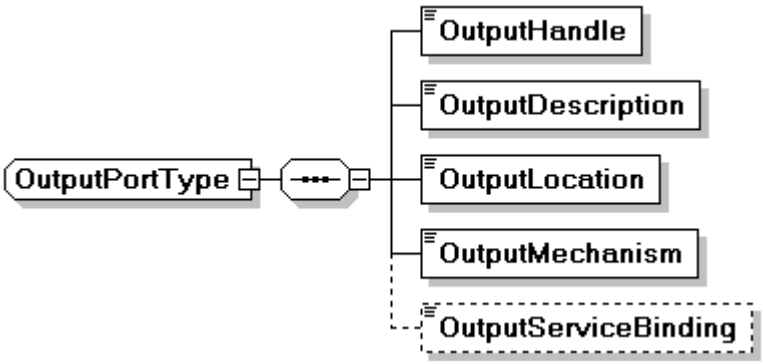
element **OptionFlagType/flagValue**

diagram 

type **xsd:string**

source
```
<xsd:element name="flagValue" type="xsd:string" minOccurs="0"/>
```

complexType **OutputPortType**

diagram 

namespace http://grids.ucs.indiana.edu:8005/GCWS/Schema/AppII

children OutputHandle OutputDescription OutputLocation OutputMechanism OutputServiceBinding

used by element AppIInstanceType/OutputPort

source
```
<xsd:complexType name="OutputPortType">
  <xsd:sequence>
    <xsd:element name="OutputHandle" type="xsd:string"/>
    <xsd:element name="OutputDescription" type="xsd:string"/>
    <xsd:element name="OutputLocation" type="xsd:anyURI"/>
    <xsd:element name="OutputMechanism" type="appl:mechanism"/>
    <xsd:element name="OutputServiceBinding" type="xsd:string" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>
```

element **OutputPortType/OutputHandle**

diagram OutputHandle

type **xsd:string**

source
```
<xsd:element name="OutputHandle" type="xsd:string"/>
```

element **OutputPortType/OutputDescription**

diagram OutputDescription

type **xsd:string**

source
```
<xsd:element name="OutputDescription" type="xsd:string"/>
```

element **OutputPortType/OutputLocation**

diagram OutputLocation

type **xsd:anyURI**

source
```
<xsd:element name="OutputLocation" type="xsd:anyURI"/>
```

element **OutputPortType/OutputMechanism**

diagram OutputMechanism

type appl:mechanism

facets
enumeration StandardIO
enumeration CArgument

source
```
<xsd:element name="OutputMechanism" type="appl:mechanism"/>
```

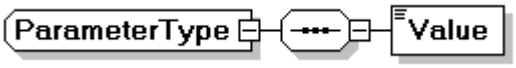
element **OutputPortType/OutputServiceBinding**

diagram OutputServiceBinding

type **xsd:string**

source
```
<xsd:element name="OutputServiceBinding" type="xsd:string" minOccurs="0"/>
```

complexType **ParameterType**

diagram 

namespace http://grids.ucs.indiana.edu:8005/GCWS/Schema/AppII

children Value

used by element AppIInstanceType/ApplicationParameter

attributes

Name	Type	Use	Default	Fixed	Annotation
Name	xsd:string				

source

```
<xsd:complexType name="ParameterType">
  <xsd:sequence>
    <xsd:element name="Value" type="xsd:string"/>
  </xsd:sequence>
  <xsd:attribute name="Name" type="xsd:string"/>
</xsd:complexType>
```

element **ParameterType/Value**

diagram 

type **xsd:string**

source

```
<xsd:element name="Value" type="xsd:string"/>
```

simpleType **mechanism**

namespace http://grids.ucs.indiana.edu:8005/GCWS/Schema/AppII

type restriction of **xsd:string**

used by elements ErrorPortType/ErrorMechanism
InputPortType/InputMechanism
OutputPortType/OutputMechanism

facets
enumeration StandardIO
enumeration CArgument

source

```
<xsd:simpleType name="mechanism">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="StandardIO"/>
    <xsd:enumeration value="CArgument"/>
  </xsd:restriction>
</xsd:simpleType>
```

XML Schema documentation generated with XML Spy Schema Editor www.xmlspy.com

```
<?xml version="1.0"?>
<!-- edited with XML Spy v4.4 U (http://www.xmlspy.com) by Choonhan
Youn (Florida State University) -->
<xsd:schema
targetNamespace="http://grids.ucs.indiana.edu:8005/GCWS/Schema/ApplI"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:appl="http://grids.ucs.indiana.edu:8005/GCWS/Schema/ApplI">
      <xsd:annotation>
            <xsd:documentation>
        This schema is used to describe an application instance.
        </xsd:documentation>
      </xsd:annotation>
      <!--
   ApplInstance is the root element and describes exactly one
application
   running on one host.
   -->
      <xsd:element name="ApplInstance" type="appl:ApplInstanceType"/>
      <!--
   Each application's info is contained by the tag ApplicationType.
   An application instance can have several input and output files, but
   only binds to one host.
   -->
      <xsd:complexType name="ApplInstanceType">
            <xsd:sequence>
                  <xsd:element name="ApplicationName"
type="xsd:string"/>
                  <xsd:element name="Version" type="xsd:string"
minOccurs="0"/>
                  <xsd:element name="OptionFlag"
type="appl:OptionFlagType" minOccurs="0" maxOccurs="unbounded"/>
                  <xsd:element name="InputPort"
type="appl:InputPortType" minOccurs="0" maxOccurs="unbounded"/>
                  <xsd:element name="OutputPort"
type="appl:OutputPortType" minOccurs="0" maxOccurs="unbounded"/>
                  <xsd:element name="ErrorPort"
type="appl:ErrorPortType" minOccurs="0" maxOccurs="unbounded"/>
                  <xsd:element name="ApplicationParameter"
type="appl:ParameterType" minOccurs="0" maxOccurs="unbounded"/>
                  <xsd:element name="HostBindingInstance"
type="appl:HostBindingType"/>
            </xsd:sequence>
      </xsd:complexType>
      <!--
   Code option flags.  These might be followed by arguments
   -->
      <xsd:complexType name="OptionFlagType">
            <xsd:sequence>
                  <xsd:element name="flagName" type="xsd:string"/>
                  <xsd:element name="flagValue" type="xsd:string"
minOccurs="0"/>
```

```
            </xsd:sequence>
      </xsd:complexType>
      <!--
   InputPortType is used to describe the input methods (ie a file
   on local disk).
      0 InputHandle is a short name for a particular input type
      0 InputInstanceription is a long description of what type
        of input is expected (ie an ANSYS Prep7 file). Put
        instructions to users here.
        0 InputLocation is a URI that describes where the input file
          is.  If this is other than a local file, you must define
        also the service that will perform this function.
      0 Input mechanism:  this is either Standard Input (with a "<") or
        as a C-style command line arguement.
      0 InputServiceBinding is a place holder for describing
        the service you want to use to get the file.
   -->
      <xsd:complexType name="InputPortType">
            <xsd:sequence>
                  <xsd:element name="InputHandle" type="xsd:string"/>
                  <xsd:element name="InputDescription"
type="xsd:string"/>
                  <xsd:element name="InputLocation" type="xsd:anyURI"/>
                  <xsd:element name="InputMechanism"
type="appl:mechanism"/>
                  <xsd:element name="InputServiceBinding"
type="xsd:string" minOccurs="0"/>
            </xsd:sequence>
      </xsd:complexType>
      <!--
   OutputPortType is used to describe the output files (ie a file
   on local disk).  You need one of these tags for each
   output file the code generates.
      0 OutputHandle is a short name for a particular output type
      0 OutputDescription is a long description of a particular
        output file and is application-specific.
        0 OutputLocation is a URI that describes where the output file
          is to be written.
      0 OutputServiceBinding is a placeholder for the service
        you want to move the output files, if other than to local
        disk.
   -->
      <xsd:complexType name="OutputPortType">
            <xsd:sequence>
                  <xsd:element name="OutputHandle" type="xsd:string"/>
                  <xsd:element name="OutputDescription"
type="xsd:string"/>
                  <xsd:element name="OutputLocation"
type="xsd:anyURI"/>
                  <xsd:element name="OutputMechanism"
type="appl:mechanism"/>
                  <xsd:element name="OutputServiceBinding"
type="xsd:string" minOccurs="0"/>
            </xsd:sequence>
      </xsd:complexType>
      <!--
   ErrorPortType is used to describe the error files that may be
```

```
   generated.
      0 ErrorHandle is a short name for a particular error type
      0 ErrorDescription is a long description of what type
        of error.
        0 ErrorLocation is a URI that describes where the error file
          is to be written.
      0 ErrorServiceBinding is a placeholder for the service
        you want to move the error files, if other than to local
        disk.
   -->
      <xsd:complexType name="ErrorPortType">
            <xsd:sequence>

                  <xsd:element name="ErrorHandle" type="xsd:string"/>
                  <xsd:element name="ErrorDescription"
type="xsd:string"/>
                  <xsd:element name="ErrorLocation" type="xsd:anyURI"/>
                  <xsd:element name="ErrorMechanism"
type="appl:mechanism"/>
                  <xsd:element name="ErrorServiceBinding"
type="xsd:string" minOccurs="0"/>
            </xsd:sequence>
      </xsd:complexType>
      <xsd:simpleType name="mechanism">
            <xsd:restriction base="xsd:string">
                  <xsd:enumeration value="StandardIO"/>
                  <xsd:enumeration value="CArgument"/>
            </xsd:restriction>
      </xsd:simpleType>
      <!--
   ParameterType is used for name/value pairs.
   -->
      <xsd:complexType name="ParameterType">
            <xsd:sequence>
                  <xsd:element name="Value" type="xsd:string"/>
            </xsd:sequence>
            <xsd:attribute name="Name" type="xsd:string"/>
      </xsd:complexType>
      <!--
   Extend to include host information from HostInstance.xml
   -->
      <xsd:complexType name="HostBindingType">
            <xsd:sequence>
                  <xsd:element name="HostDescIns" type="xsd:anyType"/>
            </xsd:sequence>
      </xsd:complexType>
</xsd:schema>
```

Appendix E: Host Instance Descriptor Schema

Schema **Hostins.xsd**

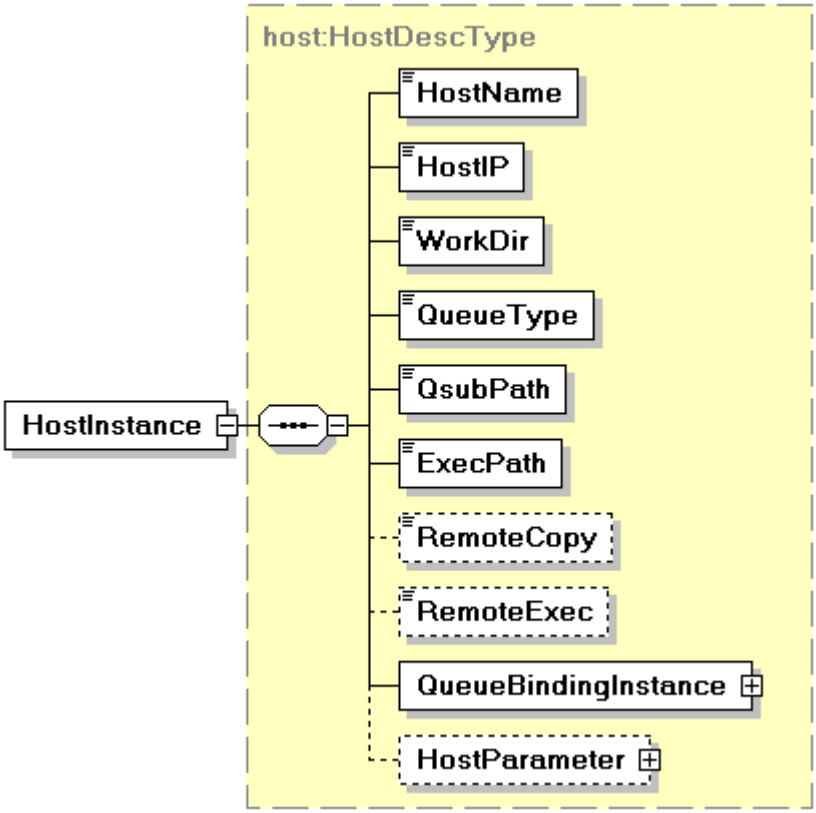
schema location: C:\castor\ApplDesc\Hostins.xsd

targetNamespace: http://grids.ucs.indiana.edu:8005/GCWS/Schema/Host

Elements	Complex types
HostInstance	HostDescType ParameterType QueueBindingType

element **HostInstance**

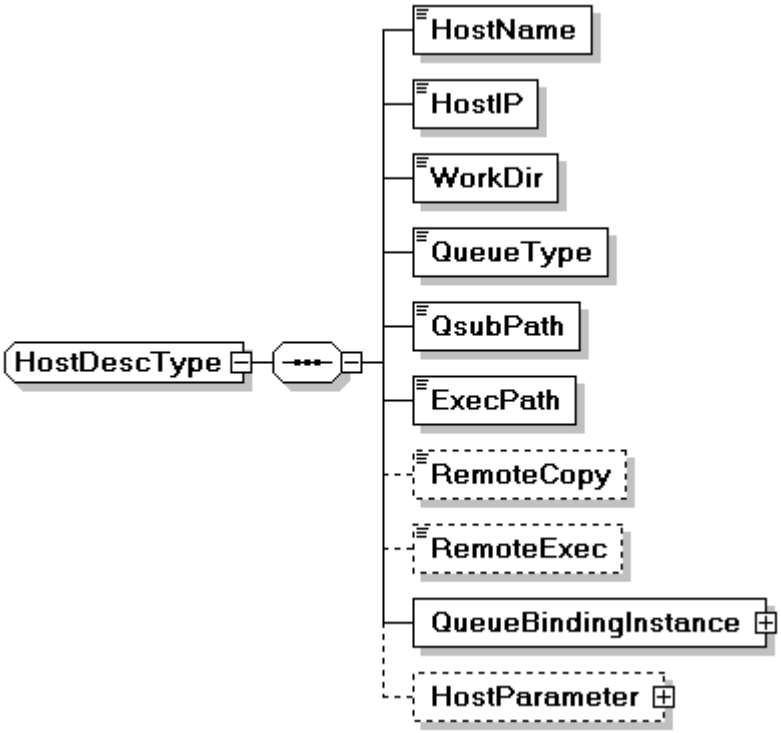
diagram



namespace	http://grids.ucs.indiana.edu:8005/GCWS/Schema/Host
type	host:HostDescType
children	HostName HostIP WorkDir QueueType QsubPath ExecPath RemoteCopy RemoteExec QueueBindingInstance HostParameter
source	<xsd:element name="HostInstance" type="host:HostDescType"/>

complexType **HostDescType**

diagram



namespace http://grids.ucs.indiana.edu:8005/GCWS/Schema/Host

children HostName HostIP WorkDir QueueType QsubPath ExecPath RemoteCopy RemoteExec QueueBindingInstance HostParameter

used by element HostInstance

source

```
<xsd:complexType name="HostDescType">
  <xsd:sequence>
    <xsd:element name="HostName" type="xsd:string"/>
    <xsd:element name="HostIP" type="xsd:string"/>
    <xsd:element name="WorkDir" type="xsd:string"/>
    <xsd:element name="QueueType" type="xsd:string"/>
    <xsd:element name="QsubPath" type="xsd:string"/>
    <xsd:element name="ExecPath" type="xsd:string"/>
    <xsd:element name="RemoteCopy" type="xsd:string" minOccurs="0"/>
    <xsd:element name="RemoteExec" type="xsd:string" minOccurs="0"/>
    <xsd:element name="QueueBindingInstance" type="host:QueueBindingType"/>
    <xsd:element name="HostParameter" type="host:ParameterType" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>
```

element **HostDescType/HostName**

diagram 

type **xsd:string**

source <xsd:element name="HostName" type="xsd:string"/>

element **HostDescType/HostIP**

diagram 

type **xsd:string**

source <xsd:element name="HostIP" type="xsd:string"/>

element **HostDescType/WorkDir**

diagram 

type **xsd:string**

source <xsd:element name="WorkDir" type="xsd:string"/>

element **HostDescType/QueueType**

diagram 

type **xsd:string**

source <xsd:element name="QueueType" type="xsd:string"/>

element **HostDescType/QsubPath**

diagram 

type **xsd:string**

source <xsd:element name="QsubPath" type="xsd:string"/>

element **HostDescType/ExecPath**

diagram 

type **xsd:string**

source <xsd:element name="ExecPath" type="xsd:string"/>

element **HostDescType/RemoteCopy**

diagram 

type **xsd:string**

source <xsd:element name="RemoteCopy" type="xsd:string" minOccurs="0"/>

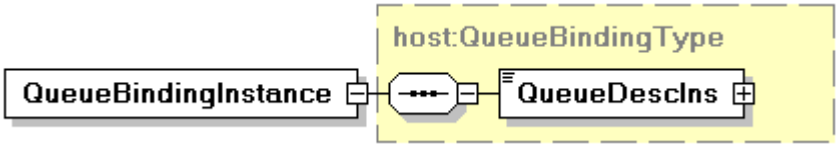
element **HostDescType/RemoteExec**

diagram 

type **xsd:string**

source <xsd:element name="RemoteExec" type="xsd:string" minOccurs="0"/>

element **HostDescType/QueueBindingInstance**

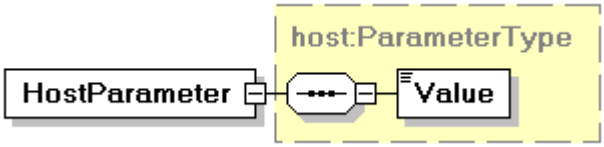
diagram 

type host:QueueBindingType

children QueueDescIns

source <xsd:element name="QueueBindingInstance" type="host:QueueBindingType"/>

element **HostDescType/HostParameter**

diagram 

type host:ParameterType

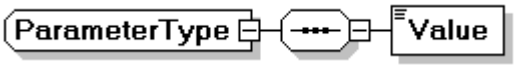
children Value

attributes

Name	Type	Use	Default	Fixed	Annotation
Name	xsd:string				

source <xsd:element name="HostParameter" type="host:ParameterType" minOccurs="0"/>

complexType **ParameterType**

diagram 

namespace http://grids.ucs.indiana.edu:8005/GCWS/Schema/Host

children Value

used by element HostDescType/HostParameter

attributes

Name	Type	Use	Default	Fixed	Annotation
Name	xsd:string				

source

```
<xsd:complexType name="ParameterType">
  <xsd:sequence>
    <xsd:element name="Value" type="xsd:string"/>
  </xsd:sequence>
  <xsd:attribute name="Name" type="xsd:string"/>
</xsd:complexType>
```

element **ParameterType/Value**

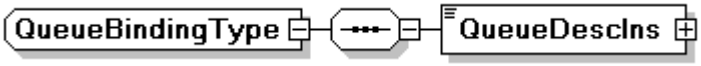
diagram 

type **xsd:string**

source

```
<xsd:element name="Value" type="xsd:string"/>
```

complexType **QueueBindingType**

diagram 

namespace http://grids.ucs.indiana.edu:8005/GCWS/Schema/Host

children QueueDescIns

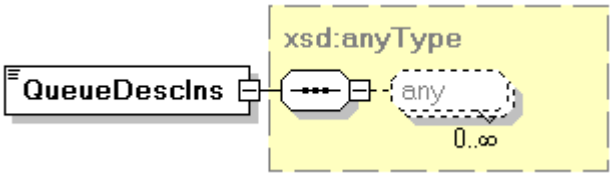
used by element HostDescType/QueueBindingInstance

source

```
<xsd:complexType name="QueueBindingType">
  <xsd:sequence>
    <xsd:element name="QueueDescIns" type="xsd:anyType"/>
  </xsd:sequence>
</xsd:complexType>
```

element **QueueBindingType/QueueDescIns**

diagram



type **xsd:anyType**

attributes	Name	Type	Use	Default	Fixed	Annotation

source <xsd:element name="QueueDescIns" type="xsd:anyType"/>

XML Schema documentation generated with [XML Spy](www.xmlspy.com) Schema Editor www.xmlspy.com

```
<?xml version="1.0"?>
<!-- edited with XML Spy v4.4 U (http://www.xmlspy.com) by Choonhan
Youn (Florida State University) -->
<xsd:schema
targetNamespace="http://grids.ucs.indiana.edu:8005/GCWS/Schema/Host"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:host="http://grids.ucs.indiana.edu:8005/GCWS/Schema/Host">
      <xsd:annotation>
            <xsd:documentation>
        This schema describes host machine bindings for applications.
        </xsd:documentation>
      </xsd:annotation>
      <!--
   HostDesc is the root element, with any number of applications.
   HostDesc contains a description of a host machine. It contains
   several required fields and can be extended by an arbitrary
   number of parameters.  Optionally, you can provide commands
   for remote copy and execution (rsh, rcp, globusrun).  This must
   be set up externally.

   Hosts also contain one or more bindings for queue execution.  This
   could be a queuing system (PBS, GRD) or it could be
   -->
      <xsd:element name="HostInstance" type="host:HostDescType"/>
      <xsd:complexType name="HostDescType">
            <xsd:sequence>
                  <xsd:element name="HostName" type="xsd:string"/>
                  <xsd:element name="HostIP" type="xsd:string"/>
                  <xsd:element name="WorkDir" type="xsd:string"/>
                  <xsd:element name="QueueType" type="xsd:string"/>
                  <xsd:element name="QsubPath" type="xsd:string"/>
                  <xsd:element name="ExecPath" type="xsd:string"/>
                  <xsd:element name="RemoteCopy" type="xsd:string"
minOccurs="0"/>
                  <xsd:element name="RemoteExec" type="xsd:string"
minOccurs="0"/>
```

```
                  <xsd:element name="QueueBindingInstance"
type="host:QueueBindingType"/>
                  <xsd:element name="HostParameter"
type="host:ParameterType" minOccurs="0"/>
            </xsd:sequence>
      </xsd:complexType>
      <xsd:complexType name="ParameterType">
            <xsd:sequence>
                  <xsd:element name="Value" type="xsd:string"/>
            </xsd:sequence>
            <xsd:attribute name="Name" type="xsd:string"/>
      </xsd:complexType>
      <!--
   Extend to include host information from QueueInstance.xml
   -->
      <xsd:complexType name="QueueBindingType">
            <xsd:sequence>
                  <xsd:element name="QueueDescIns" type="xsd:anyType"/>
            </xsd:sequence>
      </xsd:complexType>
</xsd:schema>
```

Appendix F: Queue Instance Descriptor Schema

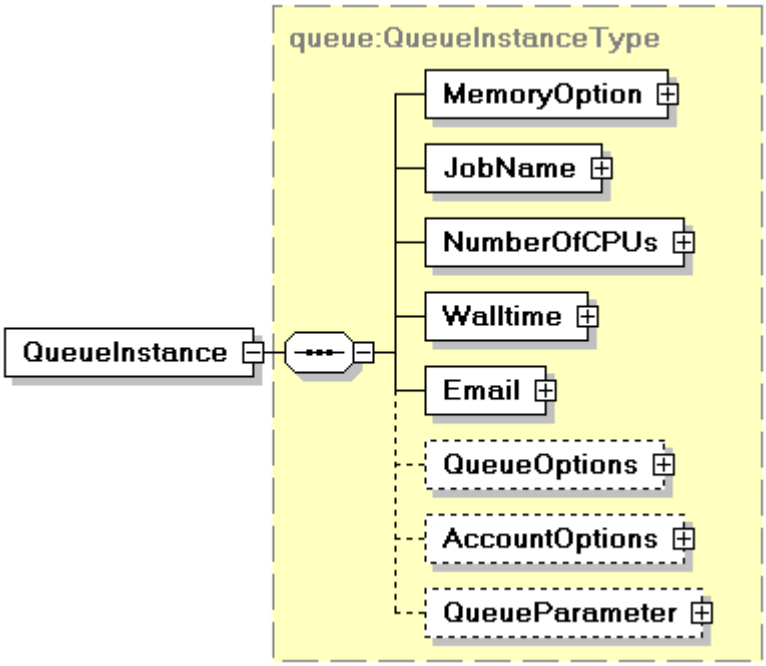
Schema **Queueins.xsd**

schema location:	C:\castor\ApplDesc\Queueins.xsd
targetNamespace:	http://grids.ucs.indiana.edu:8005/GCWS/Schema/Queue

Elements	Complex types
QueueInstance	EmailOptType
	OptionType
	ParameterType
	QueueInstanceType

element **QueueInstance**

diagram



namespace http://grids.ucs.indiana.edu:8005/GCWS/Schema/Queue

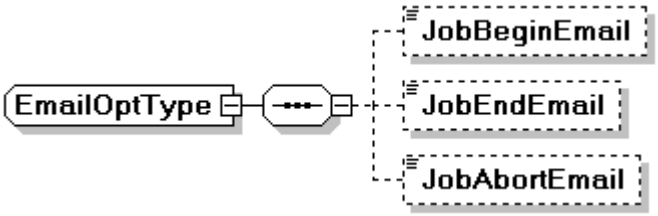
type queue:QueueInstanceType

children MemoryOption JobName NumberOfCPUs Walltime Email QueueOptions AccountOptions QueueParameter

source
```
<xsd:element name="QueueInstance" type="queue:QueueInstanceType"/>
```

complexType **EmailOptType**

diagram



namespace http://grids.ucs.indiana.edu:8005/GCWS/Schema/Queue

children JobBeginEmail JobEndEmail JobAbortEmail

used by element QueueInstanceType/Email

source
```
<xsd:complexType name="EmailOptType">
  <xsd:sequence>
    <xsd:element name="JobBeginEmail" type="xsd:string" minOccurs="0"/>
    <xsd:element name="JobEndEmail" type="xsd:string" minOccurs="0"/>
    <xsd:element name="JobAbortEmail" type="xsd:string" minOccurs="0"/>
```

```
  </xsd:sequence>
</xsd:complexType>
```

element **EmailOptType/JobBeginEmail**

diagram 

type **xsd:string**

source
```
<xsd:element name="JobBeginEmail" type="xsd:string" minOccurs="0"/>
```

element **EmailOptType/JobEndEmail**

diagram 

type **xsd:string**

source
```
<xsd:element name="JobEndEmail" type="xsd:string" minOccurs="0"/>
```

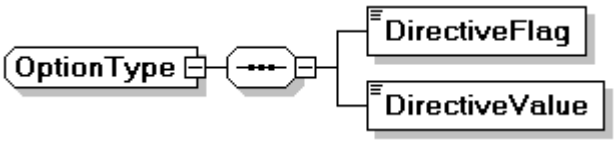
element **EmailOptType/JobAbortEmail**

diagram 

type **xsd:string**

source
```
<xsd:element name="JobAbortEmail" type="xsd:string" minOccurs="0"/>
```

complexType **OptionType**

diagram 

namespace http://grids.ucs.indiana.edu:8005/GCWS/Schema/Queue

children DirectiveFlag DirectiveValue

used by elements
- QueueInstanceType/AccountOptions
- QueueInstanceType/JobName
- QueueInstanceType/MemoryOption
- QueueInstanceType/NumberOfCPUs
- QueueInstanceType/QueueOptions
- QueueInstanceType/Walltime

source
```
<xsd:complexType name="OptionType">
  <xsd:sequence>
    <xsd:element name="DirectiveFlag" type="xsd:string"/>
    <xsd:element name="DirectiveValue" type="xsd:string"/>
```

```
  </xsd:sequence>
</xsd:complexType>
```

element **OptionType/DirectiveFlag**

diagram 

type **xsd:string**

source
```
<xsd:element name="DirectiveFlag" type="xsd:string"/>
```

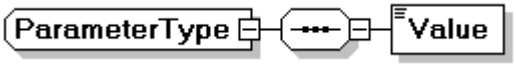
element **OptionType/DirectiveValue**

diagram 

type **xsd:string**

source
```
<xsd:element name="DirectiveValue" type="xsd:string"/>
```

complexType **ParameterType**

diagram 

namespace http://grids.ucs.indiana.edu:8005/GCWS/Schema/Queue

children Value

used by element QueueInstanceType/QueueParameter

attributes

Name	Type	Use	Default	Fixed	Annotation
Name	xsd:string				

source
```
<xsd:complexType name="ParameterType">
  <xsd:sequence>
    <xsd:element name="Value" type="xsd:string"/>
  </xsd:sequence>
  <xsd:attribute name="Name" type="xsd:string"/>
</xsd:complexType>
```

element **ParameterType/Value**

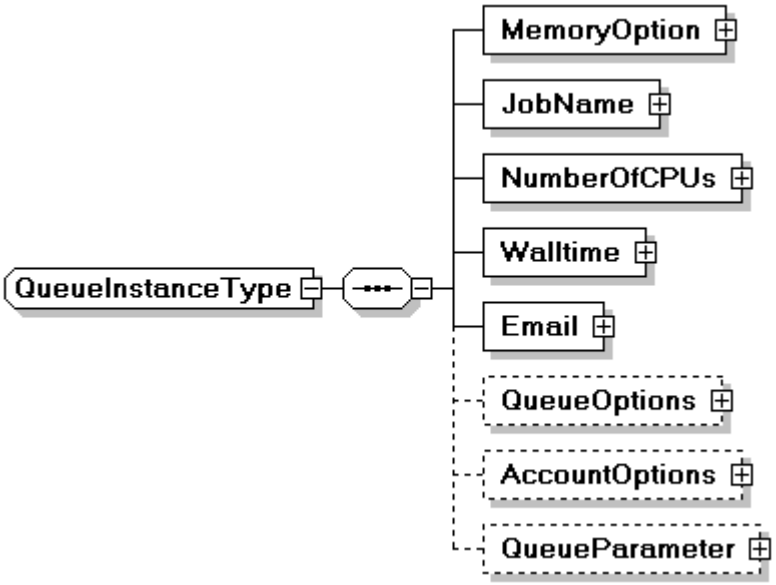
diagram 

type **xsd:string**

source
```
<xsd:element name="Value" type="xsd:string"/>
```

complexType **QueueInstanceType**

diagram



namespace http://grids.ucs.indiana.edu:8005/GCWS/Schema/Queue

children MemoryOption JobName NumberOfCPUs Walltime Email QueueOptions AccountOptions QueueParameter

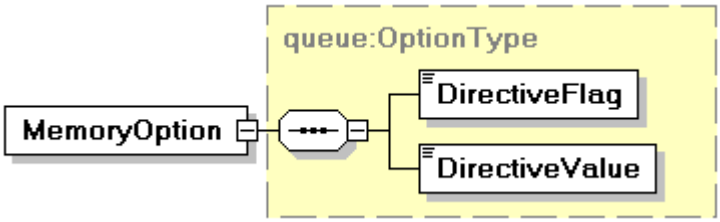
used by element QueueInstance

source

```
<xsd:complexType name="QueueInstanceType">
  <xsd:sequence>
    <xsd:element name="MemoryOption" type="queue:OptionType"/>
    <xsd:element name="JobName" type="queue:OptionType"/>
    <xsd:element name="NumberOfCPUs" type="queue:OptionType"/>
    <xsd:element name="Walltime" type="queue:OptionType"/>
    <xsd:element name="Email" type="queue:EmailOptType"/>
    <xsd:element name="QueueOptions" type="queue:OptionType" minOccurs="0"/>
    <xsd:element name="AccountOptions" type="queue:OptionType" minOccurs="0"/>
    <xsd:element name="QueueParameter" type="queue:ParameterType"
minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>
```

element **QueueInstanceType/MemoryOption**

diagram



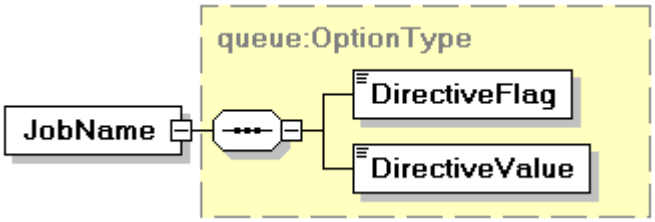
type queue:OptionType

children DirectiveFlag DirectiveValue

source <xsd:element name="MemoryOption" type="queue:OptionType"/>

element **QueueInstanceType/JobName**

diagram



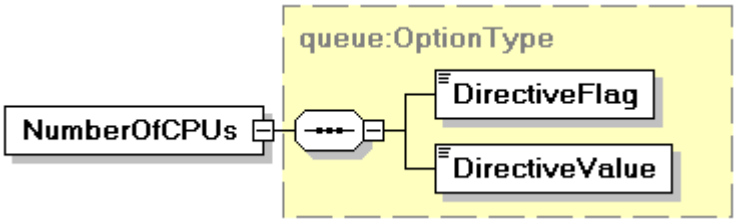
type queue:OptionType

children DirectiveFlag DirectiveValue

source <xsd:element name="JobName" type="queue:OptionType"/>

element **QueueInstanceType/NumberOfCPUs**

diagram



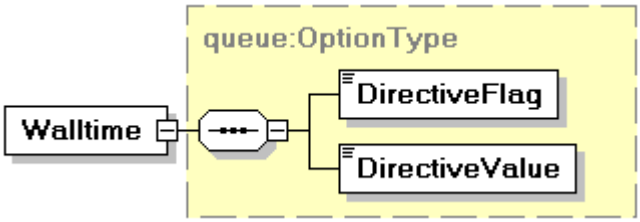
type queue:OptionType

children DirectiveFlag DirectiveValue

source <xsd:element name="NumberOfCPUs" type="queue:OptionType"/>

element **QueueInstanceType/Walltime**

diagram



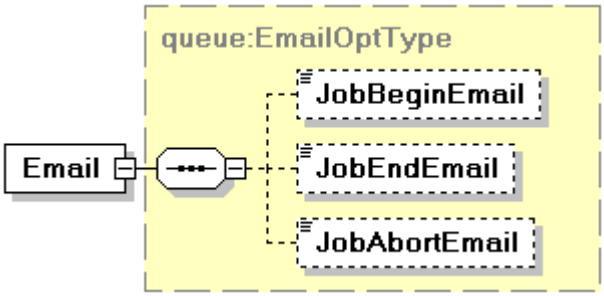
type queue:OptionType

children DirectiveFlag DirectiveValue

source <xsd:element name="Walltime" type="queue:OptionType"/>

element **QueueInstanceType/Email**

diagram



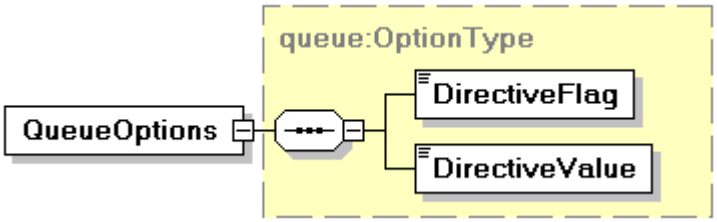
type queue:EmailOptType

children JobBeginEmail JobEndEmail JobAbortEmail

source <xsd:element name="Email" type="queue:EmailOptType"/>

element **QueueInstanceType/QueueOptions**

diagram



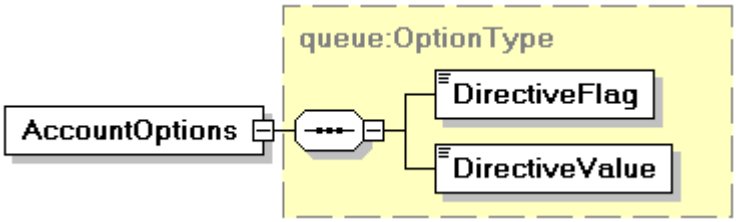
type queue:OptionType

children DirectiveFlag DirectiveValue

source <xsd:element name="QueueOptions" type="queue:OptionType" minOccurs="0"/>

element **QueueInstanceType/AccountOptions**

diagram



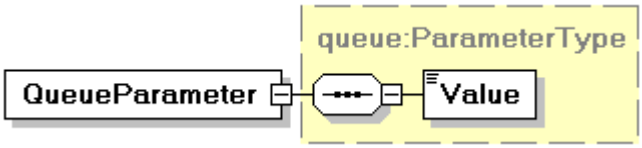
type queue:OptionType

children DirectiveFlag DirectiveValue

source <xsd:element name="AccountOptions" type="queue:OptionType" minOccurs="0"/>

element **QueueInstanceType/QueueParameter**

diagram



type queue:ParameterType

children Value

attributes

Name	Type	Use	Default	Fixed	Annotation
Name	xsd:string				

source <xsd:element name="QueueParameter" type="queue:ParameterType" minOccurs="0"/>

XML Schema documentation generated with XML Spy Schema Editor www.xmlspy.com

```
<?xml version="1.0"?>
<!-- edited with XML Spy v4.4 U (http://www.xmlspy.com) by Choonhan
Youn (Florida State University) -->
<xsd:schema
targetNamespace="http://grids.ucs.indiana.edu:8005/GCWS/Schema/Queue"
xmlns:queue="http://grids.ucs.indiana.edu:8005/GCWS/Schema/Queue"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
      <xsd:annotation>
            <xsd:documentation>
        This schema describes queuing systems on a host.
        </xsd:documentation>
      </xsd:annotation>
      <!--
   QueueInstance is the root element and describes a queuing system
   parameters.  An instance of this schema describes a particular
   queuing system (such as PBS).  Queue systems that need
```

```
   additional values not explicitly stated in the
   -->
      <xsd:element name="QueueInstance"
type="queue:QueueInstanceType"/>
      <xsd:complexType name="QueueInstanceType">
            <xsd:sequence>
                  <xsd:element name="MemoryOption"
type="queue:OptionType"/>
                  <xsd:element name="JobName" type="queue:OptionType"/>
                  <xsd:element name="NumberOfCPUs"
type="queue:OptionType"/>
                  <xsd:element name="Walltime"
type="queue:OptionType"/>
                  <xsd:element name="Email" type="queue:EmailOptType"/>
                  <xsd:element name="QueueOptions"
type="queue:OptionType" minOccurs="0"/>
                  <xsd:element name="AccountOptions"
type="queue:OptionType" minOccurs="0"/>
                  <xsd:element name="QueueParameter"
type="queue:ParameterType" minOccurs="0"/>
            </xsd:sequence>
      </xsd:complexType>
      <xsd:complexType name="ParameterType">
            <xsd:sequence>
                  <xsd:element name="Value" type="xsd:string"/>
            </xsd:sequence>
            <xsd:attribute name="Name" type="xsd:string"/>
      </xsd:complexType>
      <xsd:complexType name="OptionType">
            <xsd:sequence>
                  <xsd:element name="DirectiveFlag" type="xsd:string"/>
                  <xsd:element name="DirectiveValue"
type="xsd:string"/>
            </xsd:sequence>
      </xsd:complexType>
      <xsd:complexType name="EmailOptType">
            <xsd:sequence>
                  <xsd:element name="JobBeginEmail" type="xsd:string"
minOccurs="0"/>
                  <xsd:element name="JobEndEmail" type="xsd:string"
minOccurs="0"/>

                  <xsd:element name="JobAbortEmail" type="xsd:string"
minOccurs="0"/>
            </xsd:sequence>
      </xsd:complexType>
</xsd:schema>
```