A Common Job Description Markup Language written in XML

Stephen McGough et al.

London e-Science Centre
Imperial College of Science, Technology and Medicine
180 Queen's Gate, London SW7 2BZ, UK
lesc@ic.ac.uk
http://www.lesc.ic.ac.uk/

Abstract. In this document is a description of the XML Job Description Markup Language (JDML) used within ICENI is outlined. This Markup Language is based upon Condor ClassAds and may be used independent of the ICENI environment. The structure of the language and the motivation for this structure are described along with several examples given both in the original Job Description Language and the XML representation.

1 Introduction

This language comes from the need to develop an XML version of the Job Submission Language used within the EU-Datagrid. It was then developed further to be used within the ICENI grid middleware as a language for describing jobs to be launched onto heterogenious resources. The language was developed from the Condor ClassAds language using XML for the underlying structure. The language has basic type checking to enforce correctness of the document.

The document has been divided up into a number of sections each represented by a section equation, this allows for clear divisions between the parts of a job description and a hirarchical approach to the document allowing quicker access to the appropriate parts of the document.

Expand Like ClassAds and EU-Datagrid JSL the JDML is based upon attribute value pairs. Support is provided for Strings, lists of strings, booleans, reals, integer numbers and sections. Efforts have been made to retain the flexability in describing values that was introduced in ClassAds. Thus a String value may be more than just a simple string, it may be a collection of concatenated strings or a choice of two strings given some condition.

This document is split into three sections, in the first the structure of the XML document is described. The second section explains the attributes used within an ICENI JDML, Condor, SGE and Globus document while the last section gives the XML definitions for the language (in XSD format).

2 Structure of a JDML Document

The JDML document supports 6 types of attribute value pairs these are Section, String, StringList, Real, Integer and Boolean. Each attribute value pair is defined through an Equation. For example the attribute value pair Name=JDML would be represented by the following:

```
<StringEquation attribute="Name">
  <StringValue>JDML</StringValue>
<StringEquation>
```

The inclusion of the `<StringValue>` tag allows the 'Value' of the value to be more than a simple string. The following will produce the same result as the last example by concaternating two strings together:

```
<StringEquation attribute="Name">
  <StringAddition>
    <StringLHS>
      <StringValue>JD</StringValue>
    </StringLHS>
```

```
    <StringRHS>
      <StringValue>ML</StringValue>
    </StringRHS>
  </StringAddition>
<StringEquation>
```

The 'Value' of a attribute value pair can also be defined from that of another pair, using the name of the other pair as a variable:

```
<StringEquation attribute="Name">
  <StringValue>JDML</StringValue>
<StringEquation>
<StringEquation attribute="Name2">
  <StringVariable name="Name">
<StringEquation>
```

Rewrite This notion of String Operations, String Variables and String Values has been implemented for all types Sections, StringLists, Boolean, Real and Integer numbers. The tables below illustrate those operations which are valid for a given type, in that the operation makes sense to perform. For a full definition of these operations see the XML definitions (JDML.xsd). These are taken from the EU-DataGrid JDL document *ref*.

String Operation	Required Elements	Result type	Description
StringAddition	StringLHS StringRHS	String	Concatenates the strings together
StringLessThan	StringLHS StringRHS	Boolean	Compares two string and returns true if StringLHS would come before StringRHS in the alphabet.
StringLessThanOrEquals	StringLHS StringRHS	Boolean	Same as StringLessThan but will also return true if the two strings are the same.
StringGreaterThan	StringLHS StringRHS	Boolean	Compares the two strings and returns true if StringLHS would come after StringRHS in the alphabet.
StringGreaterThanOrEquals	StringLHS StringRHS	Boolean	Same as StringGreaterThan but will also return true if the two strings are the same.
StringEqual	StringLHS StringRHS	Boolean	Compares the two strings and returns true if the strings are the same.
StringNotEqual	StringLHS StringRHS	Boolean	Compares the two strings and returns true if the two strings differ.
StringIs	StringLHS StringRHS	Boolean	Same as StringEquals but doesn't fail in the case where either string is undefined or an error.
StringIsnt	StringLHS StringRHS	Boolean	Same as StringNotEquals but doesn't fail in the case where either string is undefined or an error.
ConditionalStringResult	BooleanTest StringTrueResult StringFalseResult	String	If the BooleanTest is true return the string StringTrueResult otherwise return the string StringFalseResult.
StringCompound	StringValue/ StringVariable/ StringOperation	String	Construct a compound string from the contained StringValue, StringVariable or String operation. Equivalent to placing brackets around the string.

The following table indicates the valid Section Operations.

Section Operation	Required Elements	Result type	Description
SectionAddition	SectionLHS SectionRHS	Section	Concatenates the sections together
SectionEqual	SectionLHS SectionRHS	Boolean	Compares the two sections and returns true if the sections contain the same information.
SectionNotEqual	SectionLHS SectionRHS	Boolean	Compares the two sections and returns true if the sections are different in some way.
SectionIs	SectionLHS SectionRHS	Boolean	Same as SectionEquals but doesn't fail in the case where either section is undefined or an error.
SectionIsnt	SectionLHS SectionRHS	Boolean	Same as SectionNotEquals but doesn't fail in the case where either section is undefined or an error.
ConditionalSectionResult	BooleanTest SectionTrueResult SectionFalseResult	Section	If the BooleanTest is true return the section SectionTrueResult otherwise return the section SectionFalseResult
SectionCompound	SectionValue/ SectionVariable/ SectionOperation	Section	Construct a compound Section from some other section - equivalent to placing brackets around the Section.

The following table indicates the valid StringList Operations.

The following table indicates the valid Real Operations.

The following table describes the valid Integer Operations.

The following table describes the Boolean operations that are available.

The EU-DataGrid JDL provides a number of built in functions and operations for both obtaining information and manipulating data. These have been replicated into the JDML.

StringList Operation	Required Elements	Result type	Description
StringListAddition	StringListLHS StringListRHS	StringList	Concatenates the StringLists together
StringListEqual	StringListLHS StringListRHS	Boolean	Compares the two string lists and returns true if they contain the same data.
StringListNotEqual	StringListLHS StringListRHS	Boolean	Compares the two string lists and returns true if they don't contain the same data.
StringListIs	StringListLHS StringListRHS	Boolean	Same as StringListEquals but doesn't fail in the case where either string list is undefined or an error.
StringListIsnt	StringListLHS StringListRHS	Boolean	Same as StringListEquals but doesn't fail in the case where either string list is undefined or an error.
ConditionalStringListResult	BooleanTest StringListTrueResult StringListFalseResult	StringList	If the BooleanTest is true return the string list StringListTrueResult otherwise return the string list StringListFalseResult.
StringListCompound	StringListValue/ StringListVariable/ StringListOperation	StringList	Construct a compound StringList from the given StringList. Equivalent to placing brackets around the StringList.

Operation	Result type	Required elements	Description
IsError	Boolean	Real/Integer/ String/StringList Section	Result is true if the element represents an error.
IsString	Boolean	Real/Integer/ String/StringList Section	True iff element is a string value
IsStringList	Boolean	Real/Integer/ String/StringList Section	True iff element is a string list value
IsClassad	Boolean	Real/Integer/ String/StringList Section	True iff element is a Section
IsBoolean	Boolean	Real/Integer/ String/StringList Section	True iff element is a boolean value
IsAbsTime	Boolean	Time	True iff element is an absolute time value
IsRelTime	Boolean	Time	True iff element is a relative time value.
Member	Boolean	StringSearch StringList	Returns true if StringSearch is found in StringList.
IsMember	Boolean	StringSearch StringList	Same as Member but uses is for comparison rather than Equals.
CurrentTime	Integer		Returns the current (absolute) time.
TimeZoneOffset	Real		Gets the time zone offset as a relative time.
Daytime	Integer		Get current time as relative time since midnight.
MakeDate	Integer	NumericMonth/ StringMonth NumericDay NumericYear	Create an absolute time value of midnight for the given day. Month can be either numeric or string (e.g., "jan").
MakeAbsTime	Integer	Time	Convert numeric value into an absolute time (number of seconds past UNIX epoch).
MakeRelTime	Integer	Time	Convert numeric value into a relative time (number of seconds in interval)

Real Operation	Required Elements	Result type	Description
RealUnaryPositive	RealValue/ RealVariable/ RealOperation	Real	Perform a unary positive operation on the given Real. Equivalent to +(Real).
RealUnaryNegative	RealValue/ RealVariable/ RealOperation	Real	Perform a unary negative operation on the given Real. Equivalent to −(Real).
RealMultiplication	RealLHS RealRHS	Real	Perform the multiplication RealLHS × RealRHS.
RealDivision	RealLHS RealRHS	Real	Perform the division RealLHS ÷ RealRHS.
RealAddition	RealLHS RealRHS	Real	Perform the addition RealLHS + RealRHS.
RealSubtraction	RealLHS RealRHS	Real	Perform the subtraction RealLHS − RealRHS.
RealLessThan	RealLHS RealRHS	Boolean	Compares two numbers and returns true if RealLHS comes before RealRHS.
RealLessThanOrEquals	RealLHS RealRHS	Boolean	Same as RealLessThan but will also return true if the two numbers are the same.
RealGreaterThan	RealLHS RealRHS	Boolean	Compares the two numbers and returns true if RealLHS comes after RealRHS.
RealGreaterThanOrEquals	RealLHS RealRHS	Boolean	Same as RealGreaterThan but will also return true if the two numbers are the same.
RealEqual	RealLHS RealRHS	Boolean	Compares the two Reals and returns true if they represent the same value.
RealNotEqual	RealLHS RealRHS	Boolean	Compares the two Reals and returns true if they represent different values.
RealIs	RealLHS RealRHS	Boolean	Same as RealEquals but doesn't fail in the case where either real is undefined or an error.
RealIsnt	RealLHS RealRHS	Boolean	Same as RealNotEquals but doesn't fail in the case where either real is undefined or an error.
ConditionalRealResult	BooleanTest RealTrueResult RealFalseResult	Real	If the BooleanTest is true return the real RealTrueResult otherwise return RealFalseResult.
RealCompound	RealValue/ RealVariable/ RealOperation	Real	Construct a compound Real from the given RealValue, RealVariable or RealOperatin. Equivalent to placing brackets around the Real.

Integer Operation	Required Elements	Result type	Description
IntegerUnaryPositive	IntegerValue/ IntegerVariable/ IntegerOperation	Integer	Perform a unary positive operation on the given Integer. Equivalent to +(Integer).
IntegerUnaryNegative	IntegerValue/ IntegerVariable/ IntegerOperation	Integer	Perform a unary negative operation on the given Integer. Equivalent to −(Integer).
IntegerOnesComplement	IntegerValue/ IntegerVariable/ IntegerOperation	Integer	Take the ones complement of the Integer.
IntegerMultiplication	IntegerLHS IntegerRHS	Integer	Perform the multiplication IntegerLHS × IntegerRHS.
IntegerDivision	IntegerLHS IntegerRHS	Integer	Perform the division IntegerLHS ÷ IntegerRHS, the result is the largest integer smaller than the result.
IntegerRemainder	IntegerLHS IntegerRHS	Integer	Perform the division IntegerLHS ÷ IntegerRHS with the result being the integer remainder.
IntegerAddition	IntegerLHS IntegerRHS	Integer	Perform the addition IntegerLHS + IntegerRHS
IntegerSubtraction	IntegerLHS IntegerRHS	Integer	Perform the subtraction IntegerLHS − IntegerRHS.
IntegerShiftLeft	IntegerLHS IntegerRHS	Integer	Shift the bits of the IntegerLHS to the left by IntegerRHS bits.
IntegerShiftRight	IntegerLHS IntegerRHS	Integer	Shift the bits of the IntegerLHS to the right by IntegerRHS bits.
IntegerUnsignedShiftRight	IntegerLHS IntegerRHS	Integer	Shift the bits of the IntegerLHS to the right by IntegerRHS bits ignoring the sign of IntegerLHS.
IntegerLessThan	IntegerLHS IntegerRHS	Boolean	Compares two numbers and returns true if IntegerLHS comes before IntegerRHS.
IntegerLessThanOrEquals	IntegerLHS IntegerRHS	Boolean	Same as IntegerLessThan but will also return true if the two numbers are the same.
IntegerGreaterThan	IntegerLHS IntegerRHS	Boolean	Compares the two numbers and returns true if IntegerLHS comes after IntegerRHS.
IntegerGreaterThanOrEquals	IntegerLHS IntegerRHS	Boolean	Same as IntegerGreaterThan but will also return true if the two numbers are the same.
IntegerEqual	IntegerLHS IntegerRHS	Boolean	Compare the two integers and return true if they both represent the same value.
IntegerNotEqual	IntegerLHS IntegerRHS	Boolean	Compare the two integers and return true if they represent different values.
IntegerIs	IntegerLHS IntegerRHS	Boolean	Same as IntegerEquals but doesn't fail in the case where either integer is undefined or an error.
IntegerIsnt	IntegerLHS IntegerRHS	Boolean	Same as IntegerNotEquals but doesn't fail in the case where either is undefined or an error.
IntegerBitwiseAND	IntegerLHS IntegerRHS	Integer	Perform a bitwise AND of the two integers.
IntegerBitwiseXOR	IntegerLHS IntegerRHS	Integer	Perform a bitwise XOR of the two integers.
IntegerBitwiseOR	IntegerLHS IntegerRHS	Integer	Perform a bitwise OR of the two integers.
ConditionalIntegerResult	BooleanTest IntegerTrueResult IntegerFalseResult	Integer	If the value of BooleanTest is true then the return is IntegerTrueResult, otherwise it is IntegerFalseResult.
IntegerCompound	IntegerValue/ IntegerVariable/ IntegerOperation	Integer	Construct a compound Integer from the given IntegerValue, IntegerVariable or IntegerOperation. Equivalent to placing brackets around the Integer.

Boolean Operation	Required Elements	Result type	Description
BooleanNot	BooleanValue/ BooleanVariable/ BooleanOperation	Boolean	Negates the value of the boolean.
BooleanEqual	BooleanLHS BooleanRHS	Boolean	Compares the two booleans, true if both have the same value otherwise false.
BooleanNotEqual	BooleanLHS BooleanRHS	Boolean	Compares the two booleans, true if both have different values otherwise false.
BooleanIs	BooleanLHS BooleanRHS	Boolean	Same as BooleanEqual but doesn't fail in the case where either boolean is undefined or an error.
BooleanIsnt	BooleanLHS BooleanRHS	Boolean	Same as BooleanNotEqual but doesn't fail in the case where either boolean is undefined or an error.
LogicalAND	BooleanLHS BooleanRHS	Boolean	Perform a logical AND of two booleans.
LogicalOR	BooleanLHS BooleanRHS	Boolean	Perform a logical OR of two booleans.
ConditionalBooleanResult	BooleanTest BooleanTrueResult BooleanFalseResult	Boolean	If the value of BooleanTest is true then return is BooleanTrueResult, otherwise it is BooleanFalseResult.
BooleanCompound	BooleanValue/ BooleanVariable/ BooleanOperation	Boolean	Construct a compound Boolean from the given BooleanValue, BooleanVariable or BooleanOperation. Equivalent to placing brackets around the Boolean.

Operation	Result type	Required elements	Description
StrCat	String	String	Concatenates string representations of values together
ToUpper	String	String	Upcases string
ToLower	String	String	Downcases string
SubStr	String	String Offset [Length]	Returns substring of String. Negative offsets and lengths count from the end of the string.
RegExp	Boolean	Pattern String	Checks if String matches pattern Pattern
Int	Integer	<Any>	Converts to an integer. Time values are converted to number of seconds, strings are parsed, bools are mapped to 0 or 1. Other values result in error
Real	Real	<Any>	Similar to Int, but to a real value.
String	String	<Any>	Converts to its string representation
Bool	Boolean	<Any>	Converts to a boolean value. Empty strings, and zero values converted to false; non-empty strings and non-zero values converted to true.
AbsTime	Time	String/Real/ Integer	Converts to an absolute time. Numeric values treated as seconds past UNIX epoch, strings parsed as necessary.
RelTime	Time	String/Real/ Integer	Converts to a relative time. Numeric values treated as number of seconds, string parsed as necessary.
Floor	Integer	Real/Integer	Floor of numeric value
Ceil	Integer	Real/Integer	Ceiling of numeric value
Round	Integer	Real/Integer	Rounded value of numeric value

3 Representing different Jobs in JDML

The JDML is designed to hold descriptions of jobs for different job launching languages (eg Condor ClassAds, RSL, EU-Datagrid JDL). All common tags for describing a job are rolled into the Job section of the JDML. Other information which is only of relevence for a particular DRM is placed in a section named after that DRM. Any section or element within the JDML document which is not understood by the current system must be preserved in case another system can make use of it.

It should be stressed that all jobs should be executable from the Job section only. The Named DRM sections should be seen as ways of passing extra information wich can help a particular DRM.

4 Core JDML

In this section the core elements of a JDML document are defined. These elements can be defined in any JDML document irrespective of the language the JDML is representing. It should be possible to take a job described only in core terms and launch it on a resource.

4.1 JDML

Element Name:	JDML
Element Type:	SectionEquation
Required Elements:	Job, files

The JDML element represents the entire job submission. It has two required sub-elements. The Job section which contains detals of how to run the job and the files section which details how to obtain the files required for execution.

4.2 Job

Element Name:	Job
Element Type:	SectionEquation
Required Elements	StdInput, StdOutput, StdError, Executable, InputSandbox, OutputSandbox, Arguments, Environment, RetryCount, Requirements, Rank

This is the full description of the job to execute.

StdInput

Element Name:	StdInput
Element Type:	StringEquation

This represents the filename of the the file to use for standard input. See the “files” element for how to describe where this file is located.

The example below shows how to represent the file “subdir/file” for standard input.

```
<StringEquation attribute="StdInput">
  <StringValue>subdir/file</StringValue>
</StringEquation>
```

As can be seen from the example the filename may contain directory structure. This will be reproduced onto the destination computer.

StdOutput

Element Name:	StdOutput
Element Type:	StringEquation

This represents the filename of the file to be created to contain the standard output. See the “files” element for how to describe where this file should be sent once execution completes.

StdError

Element Name: StdError
Element Type: StringEquation

This represents the filename of the file to be created to contain the standard error. See the “files” element for how to describe where this file should be sent once execution completes.

Executable

Element Name: Executable
Element Type: StringEquation

This is the name of the file to execute on the resource. If the file is to be staged to the resource then it should be done so through the input sandbox. On a unix system if the name starts with a slash (’/’) then the path is assumed to be absolute, otherwise it is relative to the working directory on the resource. Under windows use <drive letter>: to denote absolute addresses.

EG1:

```
<StringEquation attribute="Executable">
  <StringValue>/bin/echo</StringValue>
</StringEquation>
```

This case will use the local binary /bin/echo.

EG2:

```
<StringEquation attribute="Executable">
  <StringValue>myBin</StringValue>
</StringEquation>
```

In this case the program myBin will be used. If it is not available on the resource before launch this binary must be included in the InputSandbox.

InputSandbox

Element Name: InputSandbox
Element Type: StringListEquation

The filenames of those files that need to be staged to the resource before the job can be started. Each of these filenames should appear as tags within the files section (see below).

For example to stage the three files one, two and subdir/three the following XML should be used.

```
<StringListEquation attribute="InputSandbox">
  <StringListValue>
    <StringValue>one</StringValue>
    <StringValue>two</StringValue>
    <StringValue>subdir/three</StringValue>
  </StringListValue>
</StringListEquation>
```

All files listed (one, two and subdir/three) must also appear as elements in the “files” section.

OutputSandbox

Element Name: OutputSandbox
Element Type: StringListEquation

The filenames of those files that need to be staged back from the resource once the job has completed. Each of these filenames should appear as tags within the files section (see above).

Arguments

Element Name: Arguments
Element Type: StringListEquation

A set of strings representing any arguments which need to be passes to the executable when it is started. Any arguments that represent files must either be alredy on the resource or staged over as part of the InputSandbox.

As an example if you wish to run a command with arguments "-a 1024 -p 55" the following should be used.

```
<StringListEquation attribute="Arguments">
  <StringListValue>
    <StringValue>-a</StringValue>
    <StringValue>1024</StringValue>
    <StringValue>-p</StringValue>
    <StringValue>55</StringValue>
  </StringListValue>
</StringListEquation>
```

Environment

Element Name: Environment
Element Type: SectionEquation
Required Elements <environmental attribute names>

A section containing a number of StringEquations, representing argument value pairs for setting up the environment. Each StringEquation has an argument tag which is used for the argument of the environment variable and contains a String used for the value of the environment variable.

EG: To represent:

```
ICENI_VERSION=1.0
PATH_TO_CODE=/homes/iceni/bin
```

The following XML would suffice:

```
<SectionEquation attribute="Environment">
  <StringEquation attribute="ICENI_VERSION">
    <StringValue>1.0</StringValue>
  </StringEquation>
  <StringEquation attribute="PATH_TO_CODE">
    <StringValue>/homes/iceni/bin</StringValue>
  </StringEquation>
</SectionEquation>
```

4.3 files

Element Name: files
Element Type: SectionEquation
Required Elements: <filenames of those files to be either staged to or from resource>

For each file name listed in the input and output sandboxes, or input, output, error files there is a Section equation (see below) describing how to obtain the file.

EG:

```
<SectionEquation attribute="files">
  <SectionEquation attribute="file1">
    <SectionEquation attribute="gridFTP">
```

```
      <StringEquation attribute="gFTPserver">
        <StringValue>gsiftp://server.icenigrid.org</StringValue>
      </StringEquation>
      <StringEquation attribute="path">
        <StringValue>some/depth/to/file</StringValue>
      </StringEquation>
    </SectionEquation>
  </SectionEquation>
  <SectionEquation attribute="file2">
    <SectionEquation attribute="gridFTP">
      <StringEquation attribute="gFTPserver">
        <StringValue>gsiftp://server2.icenigrid.org</StringValue>
      </StringEquation>
      <StringEquation attribute="path">
        <StringValue>not/so/deep</StringValue>
      </StringEquation>
    </SectionEquation>
  </SectionEquation>
</SectionEquation>
```

In this example two files have their methods for retrieving/storeing described. Both file1 and file2 are available via grid FTP.

<filename>

Element Name:	<filename>
Element Type:	SectionEquation
Required Elements	wget/copy/gridFTP, path

For each file the methods for obtaining the files are detailed. One transport method must be provided for each file.

wget

Element Name:	wget
Element Type:	SectionEquation
Required Elements	urlBase, path

This holds the relevent information for describing how to obtain a file using the Http protocol.

urlBase

Element Name:	urlBase
Element Type:	StringEquation

A string representing the location of a web server. Normally of the format http://<server name>/<some path part>.

path

Element Name:	path
Element Type:	StringEquation

A string representing the relative path to a file.

copy

Element Name:	copy
Element Type:	SectionEquation
Required Elements	nfsExport, path

This holds the relevent information for describing how to obtain a file available through a shared file space.

nfsExport

Element Name:	nfsExport
Element Type:	StringEquation

This is the normal nfs export name. Normally of the format <server name>:<path to dir>. This name will not be added into the filename but used to match to the same mount on the resource.

gridFTP

Element Name:	gridFTP
Element Type:	SectionEquation
Required Elements	gFTPserver, path

This holds the relevent information for describing how to obtain a file available through a grid FTP server.

gFTPserver

Element Name:	gFTPserver
Element Type:	StringEquation

JobId

Element Name:	JobId
Element Type:	SectionEquation
Required Elements	SystemJobID, UserJobID

This is a normal grid ftp locator. Normally of the format gsiftp://<server name>/<some path part>.

SystemJobID

Element Name:	SystemJobID
Element Type:	StringEquation

This is used to store the unique (as far as the system is concerned) identity for this job. The value is allocated by the system at the point where the job is submitted.

UserJobID

Element Name:	UserJobID
Element Type:	StringEquation

This is an arbitary String allocated by the user. The string is only used for identifying the job to the user.

5 Representing an ICENI job in JDML

As well as the required core elements, elements for ICENI and JobId need to be defined for an ICENI job. Thus the root element of an ICENI job description start with:

Element Name:	JDML
Element Type:	SectionEquation
Required Elements:	ICENI, Job, files, JobId

Below are the definitions of the ICENI and JobId sections.

5.1 ICENI

Element Name:	ICENI
Element Type:	SectionEquation
Required Elements:	LauncherClass, resourceId, IceniIdentityFile

This section holds information relevent to the launching of ICENI jobs.

LauncherClass

Element Name:	LauncherClass
Element Type:	StringEquation

This is used when submitting jobs to run on a Binary Component (a way of executing binary programms under ICENI) it is used to identify a suitable launcher for starting the job under.

6 Core Resource Description in JDML

The EU-DataGrid JDL used the same markup language to describe resources as was used to describe jobs. The JDML has also followed this ethos. Described below is the core set of features that should be present in any resource description.

7 Representing an ICENI Resource in JDML

Element Name:	JDML
Element Type:	SectionEquation
Required Elements:	Resource, Accounting, FileTransfer, ICENI

A resource description is a SectionEquation with attribute name “JDML”. The SectionEquation contains an element for Resource which describes the aspects of the the resouce common to all DRM’s. Again specific DRM’s can add an element (named after the DRM) to hold system specific information.

Resource

Element Name:	Resource
Element Type:	SectionEquation
Required Elements	CPUSpeed, FreeCPUs, ResourceName

Accounting

Element Name:	Accounting
Element Type:	SectionEquation
Required Elements	Price

FileTransfer

Element Name:	FileTransfer
Element Type:	SectionEquation
Required Elements	CanCopy, CanGridFTP, CanGridFTPServe, CanWebGet, CanWebServe

ICENI

Element Name:	ICENI
Element Type:	SectionEquation
Required Elements	GridContainer, LauncherID

CPUSpeed

Element Name:	CPUSpeed
Element Type:	RealEquation

This holds the speed of each CPU on the resource.

FreeCPUs

Element Name:	FreeCPUs
Element Type:	IntegerEquation

This holds the number of CPUs that are currently free to run jobs.

ResourceName

Element Name:	ResourceName
Element Type:	StringEquation

The name of the resource.

Price

Element Name:	Price
Element Type:	RealEquation

The price the resource owner will charge for using this resource per second.

CanCopy

Element Name:	CanCopy
Element Type:	BooleanEquation

Idicates if a resource has the ability to copy files (potentially from network shared file spaces).

CanGridFTP

Element Name:	CanGridFTP
Element Type:	BooleanEquation

Idicates if a resource has the ability to copy files using grid FTP.

CanGridFTPServe

Element Name:	CanGridFTPServe
Element Type:	BooleanEquation

Inidcates that the resource is running a grid FTP server.

CanWebGet

Element Name:	CanWebGet
Element Type:	BooleanEquation

Idicates if a resource has the ability to copy files using web (http) protocols.

CanWebServe

Element Name: CanWebServe
Element Type: BooleanEquation

Idicates if a resource is running a webserver which can be used to make files available.

GridContainer

Element Name: GridContainer
Element Type: StringEquation

The name of the ICENI grid container available on this resource.

LauncherID

Element Name: LauncherID
Element Type: StringEquation

The name of the launcher being used for this resource.

8 Representing an EU-Datagrid job in JDML

RetryCount

Element Name: RetryCount
Element Type: RealEquation

For those systems that allow more than one attempt at running the code, if failures occur, this sets the count. This should evaluate to a Real value.

Requirements

Element Name: Requirements
Element Type: BooleanEquation

This tag must evaluate to a Boolean value that indicates that the resource can be used to launch the code. This is based on the ClassAd Requirements tag and can be as descriptive.

Rank

Element Name: Rank
Element Type: RealEquation

This tag must evaluate to a Real value that indicates how highly the current resource matches the requirements of the job. This is based on the ClassAd Rank tag and can be as descriptive.

9 Representing an EU-Datagrid Resource in JDML

10 Representing a Condor ClassAds job in JDML

11 Representing a Condor ClassAds resource in JDML

12 Representing a RSL job in JDML

A Example XML files for ICENI

```
<?xml version="1.0" encoding="UTF-8"?>
<SectionEquation xmlns="http://www.icenigrid.org/JDML"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      attribute="JDML"

      xsi:schemaLocation=
               "http://www.icenigrid.org/JDML
                http://www.lesc.ic.ac.uk/iceni/xml/jdml-1.1/JDML.xsd">
  <SectionEquation attribute="ICENI">
    <StringEquation attribute="LauncherClass">
      <StringValue>
        icpc.broker.launcher.BashScriptLauncher
      </StringValue>
    </StringEquation>
  </SectionEquation>
  <SectionEquation attribute="Job">
    <StringEquation attribute="StdInput">
      <StringValue>std.in</StringValue>
    </StringEquation>
    <StringEquation attribute="StdOutput">
      <StringValue>std.out</StringValue>
    </StringEquation>
    <StringEquation attribute="Executable">
      <StringValue>jobR</StringValue>
    </StringEquation>
    <StringListEquation attribute="Arguments">
      <StringListValue>
        <StringValue>hello</StringValue>
        <StringValue>world</StringValue>
      </StringListValue>
    </StringListEquation>
    <StringListEquation attribute="InputSandbox">
      <StringListValue>
        <StringValue>jobR</StringValue>
      </StringListValue>
    </StringListEquation>
    <RealEquation attribute="RetryCount">
      <RealValue>6</RealValue>
    </RealEquation>
    <RealEquation attribute="Rank">
        <RealValue>0.9</RealValue>
    </RealEquation>
    <BooleanEquation attribute="Requirements">
      <BooleanValue>
        true
      </BooleanValue>
    </BooleanEquation>
  </SectionEquation>
  <SectionEquation attribute="files">
    <SectionEquation attribute="jobR">
      <SectionEquation attribute="gridFTP">
        <StringEquation attribute="gFTPserver">
          <StringValue>gsiftp://titan.doc.ic.ac.uk//homes/asm100</StringValue>
        </StringEquation>
        <StringEquation attribute="path">
          <StringValue>jobdata/first</StringValue>
```

```
        </StringEquation>
    </SectionEquation>
    <SectionEquation attribute="std.in">
      <SectionEquation attribute="gridFTP">
        <StringEquation attribute="gFTPserver">
          <StringValue>gsiftp://titan.doc.ic.ac.uk/</StringValue>
        </StringEquation>
        <StringEquation attribute="path">
          <StringValue>/tmp</StringValue>
        </StringEquation>
      </SectionEquation>
    </SectionEquation>
    <SectionEquation attribute="std.out">
      <SectionEquation attribute="gridFTP">
        <StringEquation attribute="gFTPserver">
          <StringValue>gsiftp://titan.doc.ic.ac.uk/</StringValue>
        </StringEquation>
        <StringEquation attribute="path">
          <StringValue>/tmp</StringValue>
        </StringEquation>
      </SectionEquation>
    </SectionEquation>
  </SectionEquation>
</SectionEquation>
```

A second example:

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

<SectionEquation xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://www.icenigrid.org/JDML"
  xsi:schemaLocation="http://www.icenigrid.org/JDML http://www.lesc.ic.ac.uk/iceni/xml/jdml-1.1/JDML
  attribute="JDML">
    <SectionEquation attribute="JobID">
        <StringEquation attribute="SystemJobID">
            <StringValue>plan-0++48904c9c-dd02-4d1b-b2b5-e3b6f71f2992++DumbLister</StringValue>
        </StringEquation>
    </SectionEquation>
    <SectionEquation attribute="ICENI">
        <StringEquation attribute="resourceId">
            <StringValue>48904c9c-dd02-4d1b-b2b5-e3b6f71f2992++DumbLister</StringValue>
        </StringEquation>
        <StringEquation attribute="IceniIdentityFile">
            <StringValue>IceniIdentity_andrewstephenmcgough</StringValue>
        </StringEquation>
    </SectionEquation>
    <SectionEquation attribute="files">
        <SectionEquation attribute="48904c9c-dd02-4d1b-b2b5-e3b6f71f2992++DumbLister-plan-0data.in">
            <SectionEquation attribute="copy">
                <StringEquation attribute="nfsExport">
                    <StringValue>buzzard:/export1/users/a/asm100</StringValue>
                </StringEquation>
                <StringEquation attribute="path">
                    <StringValue>/ictmp</StringValue>
                </StringEquation>
```

```
    </SectionEquation>
    <SectionEquation attribute="wget">
        <StringEquation attribute="urlBase">
            <StringValue>http://www.doc.ic.ac.uk/~asm100/ep/</StringValue>
        </StringEquation>
        <StringEquation attribute="path">
            <StringValue></StringValue>
        </StringEquation>
    </SectionEquation>
    <SectionEquation attribute="gridFTP">
        <StringEquation attribute="gFTPserver">
            <StringValue>gsiftp://titan.doc.ic.ac.uk</StringValue>
        </StringEquation>
        <StringEquation attribute="path">
            <StringValue>/homes/asm100/ictmp</StringValue>
        </StringEquation>
    </SectionEquation>
</SectionEquation>
<SectionEquation attribute="48904c9c-dd02-4d1b-b2b5-e3b6f71f2992++DumbLister-plan-0data.out">
    <SectionEquation attribute="copy">
        <StringEquation attribute="nfsExport">
            <StringValue>buzzard:/export1/users/a/asm100</StringValue>
        </StringEquation>
        <StringEquation attribute="path">
            <StringValue>/ictmp</StringValue>
        </StringEquation>
    </SectionEquation>
    <SectionEquation attribute="wget">
        <StringEquation attribute="urlBase">
            <StringValue>http://www.doc.ic.ac.uk/~asm100/ep/</StringValue>
        </StringEquation>
        <StringEquation attribute="path">
            <StringValue></StringValue>
        </StringEquation>
    </SectionEquation>
    <SectionEquation attribute="gridFTP">
        <StringEquation attribute="gFTPserver">
            <StringValue>gsiftp://titan.doc.ic.ac.uk</StringValue>
        </StringEquation>
        <StringEquation attribute="path">
            <StringValue>/homes/asm100/ictmp</StringValue>
        </StringEquation>
    </SectionEquation>
</SectionEquation>
<SectionEquation attribute="48904c9c-dd02-4d1b-b2b5-e3b6f71f2992++DumbLister-plan-0data.err">
    <SectionEquation attribute="copy">
        <StringEquation attribute="nfsExport">
            <StringValue>buzzard:/export1/users/a/asm100</StringValue>
        </StringEquation>
        <StringEquation attribute="path">
            <StringValue>/ictmp</StringValue>
        </StringEquation>
    </SectionEquation>
    <SectionEquation attribute="wget">
        <StringEquation attribute="urlBase">
            <StringValue>http://www.doc.ic.ac.uk/~asm100/ep/</StringValue>
```

```
        </StringEquation>
        <StringEquation attribute="path">
            <StringValue></StringValue>
        </StringEquation>
    </SectionEquation>
    <SectionEquation attribute="gridFTP">
        <StringEquation attribute="gFTPserver">
            <StringValue>gsiftp://titan.doc.ic.ac.uk</StringValue>
        </StringEquation>
        <StringEquation attribute="path">
            <StringValue>/homes/asm100/ictmp</StringValue>
        </StringEquation>
    </SectionEquation>
</SectionEquation>
<SectionEquation attribute="plan-0_1071247948135.xml">
    <SectionEquation attribute="copy">
        <StringEquation attribute="nfsExport">
            <StringValue>buzzard:/export1/users/a/asm100</StringValue>
        </StringEquation>
        <StringEquation attribute="path">
            <StringValue>/ictmp</StringValue>
        </StringEquation>
    </SectionEquation>
    <SectionEquation attribute="wget">
        <StringEquation attribute="urlBase">
            <StringValue>http://www.doc.ic.ac.uk/~asm100/ep/</StringValue>
        </StringEquation>
        <StringEquation attribute="path">
            <StringValue></StringValue>
        </StringEquation>
    </SectionEquation>
    <SectionEquation attribute="gridFTP">
        <StringEquation attribute="gFTPserver">
            <StringValue>gsiftp://titan.doc.ic.ac.uk</StringValue>
        </StringEquation>
        <StringEquation attribute="path">
            <StringValue>/homes/asm100/ictmp</StringValue>
        </StringEquation>
    </SectionEquation>
</SectionEquation>
<SectionEquation attribute="IceniIdentity_andrewstephenmcgough">
    <SectionEquation attribute="copy">
        <StringEquation attribute="nfsExport">
            <StringValue>buzzard:/export1/users/a/asm100</StringValue>
        </StringEquation>
        <StringEquation attribute="path">
            <StringValue>/ictmp</StringValue>
        </StringEquation>
    </SectionEquation>
    <SectionEquation attribute="wget">
        <StringEquation attribute="urlBase">
            <StringValue>http://www.doc.ic.ac.uk/~asm100/ep/</StringValue>
        </StringEquation>
        <StringEquation attribute="path">
            <StringValue></StringValue>
        </StringEquation>
```

```
        </SectionEquation>
        <SectionEquation attribute="gridFTP">
            <StringEquation attribute="gFTPserver">
                <StringValue>gsiftp://titan.doc.ic.ac.uk</StringValue>
            </StringEquation>
            <StringEquation attribute="path">
                <StringValue>/homes/asm100/ictmp</StringValue>
            </StringEquation>
        </SectionEquation>
    </SectionEquation>
</SectionEquation>
<SectionEquation attribute="Job">
    <SectionEquation attribute="Environment">
        <StringEquation attribute="ICENI_VERSION">
            <StringValue>0.01</StringValue>
        </StringEquation>
        <StringEquation attribute="LD_LIBRARY_PATH">
            <StringValue>/homes/asm100/iceni/iceni-external/build/dist/iceni-external-1.0/iceni-e
        </StringEquation>
    </SectionEquation>
    <StringEquation attribute="Executable">
        <StringValue>plan-0_1071247948135.xml</StringValue>
    </StringEquation>
    <StringEquation attribute="RunCommand">
        <StringValue>gridContainer.sh</StringValue>
    </StringEquation>
    <StringEquation attribute="StdInput">
        <StringValue>48904c9c-dd02-4d1b-b2b5-e3b6f71f2992++DumbLister-plan-0data.in</StringValue>
    </StringEquation>
    <StringEquation attribute="StdOutput">
        <StringValue>48904c9c-dd02-4d1b-b2b5-e3b6f71f2992++DumbLister-plan-0data.out</StringValue
    </StringEquation>
    <StringEquation attribute="StdError">
        <StringValue>48904c9c-dd02-4d1b-b2b5-e3b6f71f2992++DumbLister-plan-0data.err</StringValue
    </StringEquation>
    <StringListEquation attribute="Arguments">
        <StringListValue>
            <StringValue>48904c9c-dd02-4d1b-b2b5-e3b6f71f2992++DumbLister</StringValue>
            <StringValue>IceniIdentity_andrewstephenmcgough</StringValue>
        </StringListValue>
    </StringListEquation>
    <StringListEquation attribute="InputSandbox">
        <StringListValue>
            <StringValue>plan-0_1071247948135.xml</StringValue>
            <StringValue>IceniIdentity_andrewstephenmcgough</StringValue>
        </StringListValue>
    </StringListEquation>
    <RealEquation attribute="RetryCount">
        <RealValue>3.0</RealValue>
    </RealEquation>
    <RealEquation attribute="Rank">
        <RealValue>100.0</RealValue>
    </RealEquation>
    <BooleanEquation attribute="Requirements">
        <BooleanValue>True</BooleanValue>
    </BooleanEquation>
```

```
    </SectionEquation>
</SectionEquation>
```

B Example XML files for EUDG

The EU-Datagrid places a number of restrictions on the general ClassAds language by enforcing certin elements to exist within the document. At present these elements are not forced to be within the document, though this feature may be added in the future.

In this appendix the examples in the original EU-Datagrid JDL documet are presented first in their original format and then in the XML representation JDML.

B.1 Example 1

This is a simple submission request taken from the EU-DataGrid documentation.

```
[
CertificateSubject = "/O=Grid/O=UKHEP/OU=hep.ph.ac.uk/CN=Tom Scott";
  Executable = "WP1testF";
  Arguments = "datafile1.in 5.56 1024";
  StdInput = "sim.dat" ;
  StdOutput = "sim.out" ;
  StdError = "sim.err" ;
  InputSandbox = "/home/fpacini/DATA/datafile1.in",
                 "/home/fpacini/DATA/sim.dat",
                 "/home/fpacini/exe/WP1testF",
                 "/home/fpacini/DATA/file2";
  OutputSandbox = "sim.err","sim.out";
  InputData = "LF:test10096-0009" , "LF:test100960010",
      "PF:testbed002.cern.ch/home/flavia/ffiles/test10096-0011";
  ReplicaCatalog = "ldap://sunlab2g.cnaf.infn.it:2010/rc=WP2
                    INFN Test Replica Catalog,dc=sunlab2g,
                    dc=cnaf, dc=infn, dc=ita" ;
  DataAccessProtocol = "gridftp";
  OutputSE = "lx11.hep.ph.ic.ac.uk";
  RetryCount = 6;
  Rank = other.FreeCPUs;
  Requirements = other.Architecture == "INTEL" &&
                 (other.OpSys == "RH 6.2" ||
                 other.OpSys == "Solaris 2.6") &&
                 other.MinPhysicalMemory >= 200  &&
                 other.OutboundIP == TRUE;
]
```

```
<?xml version="1.0" encoding="UTF-8"?>
<SectionEquation xmlns="http://www.icenigrid.org/JDML"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      attribute="JDML"

      xsi:schemaLocation=
               "http://www.icenigrid.org/JDML
                http://www.lesc.ic.ac.uk/iceni/xml/jdml-1.1/JDML.xsd">
```

```
<SectionEquation atribute="JobUI">
  <StringEquation attribute="CertificateSubject">
    <StringValue>
      "/O=Grid/O=UKHEP/OU=hep.ph.ac.uk/CN=Tom Scott"
    </StringValue>
  </StringEquation>
</SectionEquation>
<SectionEquation attribute="Job">
  <StringEquation attribute="Executable">
    <StringValue>"WP1testF"</StringValue>
  </StringEquation>
  <StringEquation attribute="StdInput">
    <StringValue>"sim.dat"</StringValue>
  </StringEquation>
  <StringEquation attribute="StdOutput">
    <StringValue>"sim.out"</StringValue>
  </StringEquation>
  <StringEquation attribute="SetStdError">
    <StringValue>"sim.err"</StringValue>
  </StringEquation>
  <StringEquation attribute="OutputSE">
    <StringValue>"lx11.hep.ph.ic.ac.uk"</StringValue>
  </StringEquation>
  <StringListEquation attributes="Arguments">
    <StringListValue>
      <StringValue>"datafile1.in"</StringValue>
      <StringValue>"5.56"</StringValue>
      <StringValue>"1024"</StringValue>
    </StringListValue>
  </StringListEquation>
  <StringListEquation attribute="InputSandbox">
    <StringListValue>
      <StringValue>"/home/fpacini/DATA/datafile1.in"</StringValue>
      <StringValue>"/home/fpacini/DATA/sim.dat"</StringValue>
      <StringValue>"/home/fpacini/exe/WP1testF"</StringValue>
      <StringValue>"/home/fpacini/DATA/file2"</StringValue>
    </StringListValue>
  </StringListEquation>
  <StringListEquation attribute="OutputSandbox">
    <StringListValue>
      <StringValue>"sim.err"</StringValue>
      <StringValue>"sim.out"</StringValue>
    </StringListValue>
  </StringListEquation>
  <SectionEquation attribute="InputDataSet">
    <StringListEquation attribute="InputData">
      <StringListValue>
        <StringValue>"LF:test10096-0009"</StringValue>
        <StringValue>"LF:test100960010"</StringValue>
        <StringValue>
          "PF:testbed002.cern.ch/home/flavia/ffiles/test10096-0011"
        </StringValue>
      </StringListValue>
    </StringListEquation>
    <StringListEquation attribute="DataAccessProtocol">
      <StringListValue>
```

```
      <StringValue>"gridftp"</StringValue>
    </StringListValue>
  </StringListEquation>
  <StringEquation attribute="ReplicaCatalog">
    <StringValue>
      "ldap://sunlab2g.cnaf.infn.it:2010/rc=WP2
       INFN Test Replica Catalog,dc=sunlab2g,
       dc=cnaf, dc=infn, dc=ita"
    </StringValue>
  </StringEquation>
</SectionEquation>
<RealEquation attribute="RetryCount">
  <RealValue>6</RealValue>
</RealEquation>
<RealEquation attribute="Rank">
  <RealVariable name="FreeCPUs" context="other"/>
</RealEquation>
<BooleanEquation attribute="Requirements">
  <LogicalAND>
    <BooleanLHS>
      <LogicalAND>
        <BooleanLHS>
          <LogicalAND>
            <BooleanLHS>
              <StringEquals>
                <StringLHS>
                  <StringVariable name="Resource:Architecture" context="other"/>
                </StringLHS>
                <StringRHS>
                  <StringValue>"INTEL"</StringValue>
                </StringRHS>
              </StringEquals>
            </BooleanLHS>
            <BooleanRHS>
              <BooleanCompound>
                <LogicalOR>
                  <BooleanLHS>
                    <StringEquals>
                      <StringLHS>
                        <StringVariable name="Resource:OpSys" context="other"/>
                      </StringLHS>
                      <StringRHS>
                        <StringValue>"RH 6.2"</StringValue>
                      </StringRHS>
                    </StringEquals>
                  </BooleanLHS>
                  <BooleanRHS>
                    <StringEquals>
                      <StringLHS>
                        <StringVariable name="Resource:OpSys" context="other"/>
                      </StringLHS>
                      <StringRHS>
                        <StringValue>
                          "Solaris 2.6"
                        </StringValue>
                      </StringRHS>
```

```
                    </StringEquals>
                  </BooleanRHS>
                </LogicalOR>
              </BooleanCompound>
            </BooleanRHS>
          </LogicalAND>
        </BooleanLHS>
        <BooleanRHS>
          <RealGreaterThanOrEquals>
            <RealLHS>
              <RealVariable name="Resource:MinPhysicalMemory" context="other"/>
            </RealLHS>
            <RealRHS>
              <RealValue>200</RealValue>
            </RealRHS>
          </RealGreaterThanOrEquals>
        </BooleanRHS>
      </LogicalAND>
    </BooleanLHS>
    <BooleanRHS>
      <BooleanEquals>
        <BooleanLHS>
          <BooleanVariable name="Resource:OutboundIP" context="other"/>
        </BooleanLHS>
        <BooleanRHS>
          <BooleanValue>true</BooleanValue>
        </BooleanRHS>
      </BooleanEquals>
    </BooleanRHS>
  </LogicalAND>
</BooleanEquation>
</SectionEquation>
```

B.2 Example 2

This is a slightly more complex submission request taken from the EU-DataGrid documentation.

```
[
Executable = "/opt/edg/WP1testC";
StdInput = "sim.dat" ;
StdOutput = "sim.out" ;
StdError = "sim.err" ;
InputSandbox = "/home/fpacini/DATA/file1",
               "/home/fpacini/DATA/sim.dat",
               "/home/fpacini/DATA/file2";
OutputSandbox = "sim.err","sim.out","datafile1.out";
InputData = "PF:testbed001.cern.ch/home/ffiles/test10096-0009",
            "PF:testbed002.cern.ch/home/ffiles/test10096-0011";
DataAccessProtocol = "file";
RetryCount = 3;
Rank = other.MaxRunningJobs + (other.AFSAvailabe == True ? 10 : 5);
Requirements = other.Architecture == "INTEL" && other.OpSys == "RH 6.1 &&
               Member(other.RunTimeEnvironment , "EO4.2")  &&
```

```
               other.LRMSType == "PBS";
]
```

B.3 Example 2

```
<?xml version="1.0" encoding="UTF-8"?>
<SectionEquation xmlns="http://www.icenigrid.org/JDML"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      attribute="JDML"

      xsi:schemaLocation=
               "http://www.icenigrid.org/JDML
                http://www.lesc.ic.ac.uk/iceni/xml/jdml-1.1/JDML.xsd">
  <SectionEquation attribute="Job">
    <StringEquation attribute="Executable">
      <StringValue>"/opt/edg/WP1testC"</StringValue>
    </StringEquation>
    <StringEquation attribute="StdInput">
      <StringValue>"sim.dat"</StringValue>
    </StringEquation>
    <StringEquation attribute="StdOutput">
      <StringValue>"sim.out"</StringValue>
    </StringEquation>
    <StringEquation attribute="StdError">
      <StringValue>"sim.err"</StringValue>
    </StringEquation>
    <StringListEquation attribute="InputSandbox">
      <StringListValue>
        <StringValue>"/home/fpacini/DATA/file1"</StringValue>
        <StringValue>"/home/fpacini/DATA/sim.dat"</StringValue>
        <StringValue>"/home/fpacini/DATA/file2"</StringValue>
      </StringListValue>
    </StringListEquation>
    <StringListEquation attribute="OutputSandbox">
      <StringListValue>
        <StringValue>"sim.err"</StringValue>
        <StringValue>"sim.out"</StringValue>
        <StringValue>"datafile1.out"</StringValue>
      </StringListValue>
    </StringListEquation>
    <SectionEquation attribute="InputDataSet">
      <StringListEquation attribute="InputData">
        <StringListValue>
          <StringValue>
            "PF:testbed001.cern.ch/home/ffiles/test10096-0009"
          </StringValue>
          <StringValue>
            "PF:testbed002.cern.ch/home/ffiles/test10096-0011"
          </StringValue>
        </StringListValue>
      </StringListEquation>
      <StringListEquation attribute="DataAccessProtocol">
        <StringListValue>
          <StringValue>"file"</StringValue>
```

```
    </StringListValue>
  </StringListEquation>
  <StringEquation attribute="ReplicaCatalog">
    <StringValue>
      "ldap://sunlab2g.cnaf.infn.it:2010/rc=WP2
       INFN Test Replica Catalog,dc=sunlab2g,
       dc=cnaf, dc=infn, dc=ita"
    </StringValue>
  </StringEquation>
</SectionEquation>
<RealEquation attribute="RetryCount">
  <RealValue>3</RealValue>
</RealEquation>
<RealEquation attribute="Rank">
  <RealAddition>
    <RealLHS>
      <RealVariable name="Resource:MaxRunningJobs" context="other" />
    </RealLHS>
    <RealRHS>
      <RealCompound>
        <ConditionalRealResult>
          <BooleanTest>
            <BooleanEquals>
              <BooleanLHS>
                <BooleanVariable name="Resource:AFSAvailable" context="other" />
              </BooleanLHS>
              <BooleanRHS>
                <BooleanValue>true</BooleanValue>
              </BooleanRHS>
            </BooleanEquals>
          </BooleanTest>
          <RealTrueResult>
            <RealValue>10</RealValue>
          </RealTrueResult>
          <RealFalseResult>
            <RealValue>5</RealValue>
          </RealFalseResult>
        </ConditionalRealResult>
      </RealCompound>
    </RealRHS>
  </RealAddition>
</RealEquation>
<BooleanEquation attribute="Requirements">
  <LogicalAND>
    <BooleanLHS>
      <StringEquals>
        <StringLHS>
          <StringVariable name="Resource:Architecture" context="other" />
        </StringLHS>
        <StringRHS>
          <StringValue>"INTEL"</StringValue>
        </StringRHS>
      </StringEquals>
    </BooleanLHS>
    <BooleanRHS>
      <LogicalAND>
```

```
            <BooleanLHS>
              <StringEquals>
                <StringLHS>
                  <StringVariable name="Resource:OpSys" context="other" />
                </StringLHS>
                <StringRHS>
                  <StringValue>"RH 6.1"</StringValue>
                </StringRHS>
              </StringEquals>
            </BooleanLHS>
            <BooleanRHS>
              <LogicalAND>
                <BooleanLHS>
                  <Member>
                    <StringSearch>
                      <StringVariable name="Resource:RunTimeEnvironment" context="other" />
                    </StringSearch>
                    <StringList>
                      <StringListValue>
                        <StringValue>"EO4.2"</StringValue>
                      </StringListValue>
                    </StringList>
                  </Member>
                </BooleanLHS>
                <BooleanRHS>
                  <StringEquals>
                    <StringLHS>
                      <StringVariable name="Resource:LRMSType" context="other" />
                    </StringLHS>
                    <StringRHS>
                      <StringValue>"PBS"</StringValue>
                    </StringRHS>
                  </StringEquals>
                </BooleanRHS>
              </LogicalAND>
            </BooleanRHS>
          </LogicalAND>
        </BooleanRHS>
      </LogicalAND>
    </BooleanEquation>
  </SectionEquation>
</SectionEquation>
```

B.4 JDML.xsd

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

<!--=================================================================-->
<!-- SECTIONS -->
<!--=================================================================-->
```

```
  <xsd:complexType name="SectionElement">
    <xsd:choice minOccurs="1" maxOccurs="1">
      <xsd:element ref="SectionValue"     minOccurs="1" maxOccurs="1" />
      <xsd:element ref="SectionVariable"  minOccurs="1" maxOccurs="1" />
      <xsd:element ref="SectionOperation" minOccurs="1" maxOccurs="1" />
    </xsd:choice>
  </xsd:complexType>

  <xsd:complexType name="SectionValueElement">
    <xsd:sequence>
      <xsd:element ref="SectionEquation"    minOccurs="0" maxOccurs="unbounded" />
      <xsd:element ref="StringEquation"     minOccurs="0" maxOccurs="unbounded" />
      <xsd:element ref="StringListEquation" minOccurs="0" maxOccurs="unbounded" />
      <xsd:element ref="RealEquation"       minOccurs="0" maxOccurs="unbounded" />
      <xsd:element ref="IntegerEquation"    minOccurs="0" maxOccurs="unbounded" />
      <xsd:element ref="BooleanEquation"    minOccurs="0" maxOccurs="unbounded" />
    </xsd:sequence>
  </xsd:complexType>

  <xsd:element name="SectionEquation">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="SectionValueElement">
          <xsd:attribute name="attribute" type="xsd:string" use="required" />
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="SectionValue" type="SectionValueType" />

  <xsd:complexType name="SectionValueType">
    <xsd:complexContent>
      <xsd:extension base="SectionValueElement" />
    </xsd:complexContent>
  </xsd:complexType>

  <xsd:element name="SectionVariable" type="SectionVariableType" />

  <xsd:complexType name="SectionVariableType">
    <xsd:attribute name="context" type="ContextType" default="self"/>
    <xsd:attribute name="name" type="xsd:string" /> <!-- use="required" /> -->
  </xsd:complexType>

  <xsd:element name="SectionOperation" type="SectionOperationType" />

  <xsd:complexType name="SectionOperationType">
  </xsd:complexType>

<!-- String Operations -->

<!-- Unary operations (-+!~) - make no sense for sections -->

<!-- Multiplicative (*/%) - make no sense for sections -->
```

```
<!-- Addative (+-) - only + makes sense (section concatination) -->

  <xsd:element name="SectionAddition" substitutionGroup="SectionOperation">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="SectionOperationType">
          <xsd:sequence>
            <xsd:element name="SectionLHS" type="SectionElement"
                 minOccurs="1" maxOccurs="1" />
            <xsd:element name="SectionRHS" type="SectionElement"
                 minOccurs="1" maxOccurs="1" />
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>

<!-- Shift (<< >> >>>) - make no sense for sections -->

<!-- Relational (< <= > >=) - no sense for sections -->

<!-- Equality (== != is isnt) - all make sense for sections -->

  <xsd:element name="SectionEquals" substitutionGroup="BooleanOperation">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="BooleanOperationType">
          <xsd:sequence>
            <xsd:element name="SectionLHS" type="SectionElement"
                 minOccurs="1" maxOccurs="1" />
            <xsd:element name="SectionRHS" type="SectionElement"
                 minOccurs="1" maxOccurs="1" />
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="SectionNotEqual" substitutionGroup="BooleanOperation">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="BooleanOperationType">
          <xsd:sequence>
            <xsd:element name="SectionLHS" type="SectionElement"
                 minOccurs="1" maxOccurs="1" />
            <xsd:element name="SectionRHS" type="SectionElement"
                 minOccurs="1" maxOccurs="1" />
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="SectionIs" substitutionGroup="BooleanOperation">
    <xsd:complexType>
      <xsd:complexContent>
```

```
        <xsd:extension base="BooleanOperationType">
          <xsd:sequence>
            <xsd:element name="SectionLHS" type="SectionElement"
                 minOccurs="1" maxOccurs="1" />
            <xsd:element name="SectionRHS" type="SectionElement"
                 minOccurs="1" maxOccurs="1" />
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="SectionIsnt" substitutionGroup="BooleanOperation">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="BooleanOperationType">
          <xsd:sequence>
            <xsd:element name="SectionLHS" type="SectionElement"
                 minOccurs="1" maxOccurs="1" />
            <xsd:element name="SectionRHS" type="SectionElement"
                 minOccurs="1" maxOccurs="1" />
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>

<!-- Bitwise AND - no sense -->

<!-- Bitwise XOR - no sense -->

<!-- Bitwise OR - no sense -->

<!-- Logical AND - no sense -->

<!-- Logical OR - no sense -->

<!-- Conditional - makes sense - well kind of... -->

  <xsd:element name="ConditionalSectionResult"
       substitutionGroup="SectionOperation">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="SectionOperationType">
          <xsd:sequence>
            <xsd:element name="BooleanTest"       type="BooleanElement"
                 minOccurs="1" maxOccurs="1" />
            <xsd:element name="SectionTrueResult"  type="SectionElement"
                 minOccurs="1" maxOccurs="1" />
            <xsd:element name="SectionFalseResult" type="SectionElement"
                 minOccurs="1" maxOccurs="1" />
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>
```

```
<!-- Compound () statment -->

  <xsd:element name="SectionCompound" substitutionGroup="SectionOperation">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="SectionOperationType">
          <xsd:sequence>
            <xsd:choice minOccurs="1" maxOccurs="1">
              <xsd:element ref="SectionValue"    minOccurs="1" maxOccurs="1"/>
              <xsd:element ref="SectionVariable" minOccurs="1" maxOccurs="1"/>
              <xsd:element ref="SectionOperation"
                   minOccurs="1" maxOccurs="1" />
            </xsd:choice>
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>

<!--============================================================================-->
<!-- STRINGS -->
<!--============================================================================-->

  <xsd:complexType name="StringElement">
    <xsd:choice minOccurs="1" maxOccurs="1">
      <xsd:element ref="StringValue"     minOccurs="1" maxOccurs="1" />
      <xsd:element ref="StringVariable"  minOccurs="1" maxOccurs="1" />
      <xsd:element ref="StringOperation" minOccurs="1" maxOccurs="1" />
    </xsd:choice>
  </xsd:complexType>

  <xsd:element name="StringEquation">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="StringElement">
          <xsd:attribute name="attribute" type="xsd:string" use="required" />
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="StringValue" type="StringValueType" />

  <xsd:complexType name="StringValueType">
    <xsd:simpleContent>
      <xsd:extension base="xsd:string">
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>

  <xsd:element name="StringVariable" type="StringVariableType" />

  <xsd:complexType name="StringVariableType">
    <xsd:attribute name="context" type="ContextType" default="self"/>
    <xsd:attribute name="name" type="xsd:string" /> <!-- use="required" /> -->
```

```
  </xsd:complexType>

  <xsd:element name="StringOperation" type="StringOperationType" />

  <xsd:complexType name="StringOperationType">
  </xsd:complexType>

<!-- String Operations -->

<!-- Unary operations (-+!~) - make no sense for strings -->

<!-- Multiplicative (*/%) - make no sense for strings -->

<!-- Addative (+-) - only + makes sense (string concatination) -->

  <xsd:element name="StringAddition" substitutionGroup="StringOperation">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="StringOperationType">
          <xsd:sequence>
            <xsd:element name="StringLHS" type="StringElement"
                 minOccurs="1" maxOccurs="1" />
            <xsd:element name="StringRHS" type="StringElement"
                 minOccurs="1" maxOccurs="1" />
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>

<!-- Shift (<< >> >>>) - make no sense for strings -->

<!-- Relational (< <= > >=) - some sense if comparing strings -->

  <xsd:element name="StringLessThan" substitutionGroup="BooleanOperation">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="BooleanOperationType">
          <xsd:sequence>
            <xsd:element name="StringLHS" type="StringElement"
                 minOccurs="1" maxOccurs="1" />
            <xsd:element name="StringRHS" type="StringElement"
                 minOccurs="1" maxOccurs="1" />
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="StringLessThanOrEqual" substitutionGroup="BooleanOperation">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="BooleanOperationType">
          <xsd:sequence>
            <xsd:element name="StringLHS" type="StringElement"
                 minOccurs="1" maxOccurs="1" />
```

```
            <xsd:element name="StringRHS" type="StringElement"
                 minOccurs="1" maxOccurs="1" />
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="StringGreaterThan" substitutionGroup="BooleanOperation">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="BooleanOperationType">
          <xsd:sequence>
            <xsd:element name="StringLHS" type="StringElement"
                 minOccurs="1" maxOccurs="1" />
            <xsd:element name="StringRHS" type="StringElement"
                 minOccurs="1" maxOccurs="1" />
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="StringGreaterThanOrEqual" substitutionGroup="BooleanOperation">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="BooleanOperationType">
          <xsd:sequence>
            <xsd:element name="StringLHS" type="StringElement"
                 minOccurs="1" maxOccurs="1" />
            <xsd:element name="StringRHS" type="StringElement"
                 minOccurs="1" maxOccurs="1" />
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>

<!-- Equality (== != is isnt) - all make sense for strings -->

  <xsd:element name="StringEquals" substitutionGroup="BooleanOperation">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="BooleanOperationType">
          <xsd:sequence>
            <xsd:element name="StringLHS" type="StringElement"
                 minOccurs="1" maxOccurs="1" />
            <xsd:element name="StringRHS" type="StringElement"
                 minOccurs="1" maxOccurs="1" />
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="StringNotEqual" substitutionGroup="BooleanOperation">
```

```
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="BooleanOperationType">
          <xsd:sequence>
            <xsd:element name="StringLHS" type="StringElement"
                 minOccurs="1" maxOccurs="1" />
            <xsd:element name="StringRHS" type="StringElement"
                 minOccurs="1" maxOccurs="1" />
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="StringIs" substitutionGroup="BooleanOperation">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="BooleanOperationType">
          <xsd:sequence>
            <xsd:element name="StringLHS" type="StringElement"
                 minOccurs="1" maxOccurs="1" />
            <xsd:element name="StringRHS" type="StringElement"
                 minOccurs="1" maxOccurs="1" />
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="StringIsnt" substitutionGroup="BooleanOperation">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="BooleanOperationType">
          <xsd:sequence>
            <xsd:element name="StringLHS" type="StringElement"
                 minOccurs="1" maxOccurs="1" />
            <xsd:element name="StringRHS" type="StringElement"
                 minOccurs="1" maxOccurs="1" />
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>

<!-- Bitwise AND - no sense -->

<!-- Bitwise XOR - no sense -->

<!-- Bitwise OR - no sense -->

<!-- Logical AND - no sense -->

<!-- Logical OR - no sense -->

<!-- Conditional - makes sense-->
```

```
  <xsd:element name="ConditionalStringResult"
       substitutionGroup="StringOperation">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="StringOperationType">
          <xsd:sequence>
            <xsd:element name="BooleanTest"       type="BooleanElement"
                 minOccurs="1" maxOccurs="1" />
            <xsd:element name="StringTrueResult"  type="StringElement"
                 minOccurs="1" maxOccurs="1" />
            <xsd:element name="StringFalseResult" type="StringElement"
                 minOccurs="1" maxOccurs="1" />
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>

<!-- Compound () statment -->

  <xsd:element name="StringCompound" substitutionGroup="StringOperation">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="StringOperationType">
          <xsd:sequence>
            <xsd:choice minOccurs="1" maxOccurs="1">
              <xsd:element ref="StringValue"     minOccurs="1" maxOccurs="1"/>
              <xsd:element ref="StringVariable"  minOccurs="1" maxOccurs="1"/>
              <xsd:element ref="StringOperation" minOccurs="1" maxOccurs="1"/>
            </xsd:choice>
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>

<!--=========================================================================-->
<!-- STRING LISTS -->
<!--=========================================================================-->

  <xsd:complexType name="StringListElement">
    <xsd:choice minOccurs="1" maxOccurs="1">
      <xsd:element ref="StringListVariable"  minOccurs="1" maxOccurs="1" />
      <xsd:element ref="StringListOperation" minOccurs="1" maxOccurs="1" />
      <xsd:element ref="StringListValue"     minOccurs="1" maxOccurs="1" />
    </xsd:choice>
  </xsd:complexType>

  <xsd:element name="StringListEquation">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="StringListElement">
          <xsd:attribute name="attribute" type="xsd:string" use="required" />
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
```

```
  </xsd:element>

  <xsd:complexType name="StringListVariableElement">
    <xsd:choice minOccurs="1" maxOccurs="1">
      <xsd:element ref="StringListVariable" minOccurs="1" maxOccurs="1"/>
    </xsd:choice>
  </xsd:complexType>

  <xsd:element name="StringListValue" type="StringListValueType" />

  <xsd:complexType name="StringListValueType">
    <xsd:choice minOccurs="1" maxOccurs="unbounded">
        <xsd:element ref="StringValue"     minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element ref="StringVariable"  minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element ref="StringOperation" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:choice>
  </xsd:complexType>

  <xsd:element name="StringListVariable" type="StringListVariableType" />

  <xsd:complexType name="StringListVariableType">
    <xsd:attribute name="context" type="ContextType" default="self" />
    <xsd:attribute name="name" type="xsd:string" />
        <!--use="required"/> -->
  </xsd:complexType>

  <xsd:element name="StringListOperation" type="StringListOperationType" />

  <xsd:complexType name="StringListOperationType">
  </xsd:complexType>

<!-- String List Operations -->

<!-- Unary operations (-+!~) - make no sense for string Lists -->

<!-- Multiplicative (*/%) - make no sense for string lists -->

<!-- Addative (+-) - only + makes sense (string list concatination) -->

  <xsd:element name="StringListAddition" substitutionGroup="StringListOperation">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="StringListOperationType">
          <xsd:sequence>
            <xsd:element name="StringListLHS" type="StringListElement"
                 minOccurs="1" maxOccurs="1" />
            <xsd:element name="StringListRHS" type="StringListElement"
                 minOccurs="1" maxOccurs="1" />
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>

<!-- Shift (<< >> >>>) - make no sense for string lists -->
```

```
<!-- Relational (< <= > >=) - makes no sense if comparing string lists -->

<!-- Equality (== != is isnt) - all make sense for string lists -->

  <xsd:element name="StringListEquals" substitutionGroup="BooleanOperation">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="BooleanOperationType">
          <xsd:sequence>
            <xsd:element name="StringListLHS" type="StringListElement"
                 minOccurs="1" maxOccurs="1" />
            <xsd:element name="StringListRHS" type="StringListElement"
                 minOccurs="1" maxOccurs="1" />
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="StringListNotEqual" substitutionGroup="BooleanOperation">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="BooleanOperationType">
          <xsd:sequence>
            <xsd:element name="StringListLHS" type="StringListElement"
                 minOccurs="1" maxOccurs="1" />
            <xsd:element name="StringListRHS" type="StringListElement"
                 minOccurs="1" maxOccurs="1" />
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="StringListIs" substitutionGroup="BooleanOperation">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="BooleanOperationType">
          <xsd:sequence>
            <xsd:element name="StringListLHS" type="StringListElement"
                 minOccurs="1" maxOccurs="1" />
            <xsd:element name="StringListRHS" type="StringListElement"
                 minOccurs="1" maxOccurs="1" />
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="StringListIsnt" substitutionGroup="BooleanOperation">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="BooleanOperationType">
          <xsd:sequence>
            <xsd:element name="StringListLHS" type="StringListElement"
                 minOccurs="1" maxOccurs="1" />
```

```
            <xsd:element name="StringListRHS" type="StringListElement"
                 minOccurs="1" maxOccurs="1" />
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>

<!-- Bitwise AND - no sense -->

<!-- Bitwise XOR - no sense -->

<!-- Bitwise OR - no sense -->

<!-- Logical AND - no sense -->

<!-- Logical OR - no sense -->

<!-- Conditional - makes sense-->

  <xsd:element name="ConditionalStringListResult"
       substitutionGroup="StringListOperation">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="StringListOperationType">
          <xsd:sequence>
            <xsd:element name="BooleanTest"       type="BooleanElement"
                 minOccurs="1" maxOccurs="1" />
            <xsd:element name="StringListTrueResult"  type="StringListElement"
                 minOccurs="1" maxOccurs="1" />
            <xsd:element name="StringListFalseResult" type="StringListElement"
                 minOccurs="1" maxOccurs="1" />
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>

<!-- Compound () statment -->

  <xsd:element name="StringListCompound" substitutionGroup="StringListOperation">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="StringListOperationType">
          <xsd:sequence>
            <xsd:choice minOccurs="1" maxOccurs="1">
              <xsd:element ref="StringListValue"     minOccurs="1" maxOccurs="1" />
              <xsd:element ref="StringListVariable"  minOccurs="1" maxOccurs="1" />
              <xsd:element ref="StringListOperation" minOccurs="1" maxOccurs="1" />
            </xsd:choice>
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>
```

```
<!--=========================================================================-->
<!-- REAL -->
<!--=========================================================================-->

  <xsd:complexType name="RealElement">
    <xsd:choice minOccurs="1" maxOccurs="1">
      <xsd:element ref="RealVariable"  minOccurs="1" maxOccurs="1" />
      <xsd:element ref="RealValue"     minOccurs="1" maxOccurs="1" />
      <xsd:element ref="RealOperation" minOccurs="1" maxOccurs="1" />
    </xsd:choice>
  </xsd:complexType>

  <xsd:element name="RealEquation">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="RealElement">
          <xsd:attribute name="attribute" type="xsd:string" use="required" />
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="RealValue" type="RealValueType" />

  <xsd:complexType name="RealValueType">
    <xsd:simpleContent>
      <xsd:extension base="xsd:double">
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>

  <xsd:element name="RealVariable" type="RealVariableType" />

  <xsd:complexType name="RealVariableType">
    <xsd:attribute name="context" type="ContextType" default="self" />
    <xsd:attribute name="name" type="xsd:string" /> <!-- use="required" /> -->
  </xsd:complexType>

  <xsd:element name="RealOperation" type="RealOperationType" />

  <xsd:complexType name="RealOperationType">
  </xsd:complexType>

<!-- Real Operations -->

<!-- Unary operations (-+!~) - +/-/~ make sense for doubles -->

  <xsd:element name="RealUnaryPositive" substitutionGroup="RealOperation">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="RealOperationType">
          <xsd:choice minOccurs="1" maxOccurs="1">
            <xsd:element ref="RealVariable"  minOccurs="1" maxOccurs="1" />
            <xsd:element ref="RealOperation" minOccurs="1" maxOccurs="1" />
            <xsd:element ref="RealValue"     minOccurs="1" maxOccurs="1" />
          </xsd:choice>
```

```
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:element>

<xsd:element name="RealUnaryNegative" substitutionGroup="RealOperation">
  <xsd:complexType>
    <xsd:complexContent>
      <xsd:extension base="RealOperationType">
        <xsd:choice minOccurs="1" maxOccurs="1">
          <xsd:element ref="RealVariable"  minOccurs="1" maxOccurs="1" />
          <xsd:element ref="RealOperation" minOccurs="1" maxOccurs="1" />
          <xsd:element ref="RealValue"     minOccurs="1" maxOccurs="1" />
        </xsd:choice>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:element>

<xsd:element name="RealOnesComplement" substitutionGroup="RealOperation">
  <xsd:complexType>
    <xsd:complexContent>
      <xsd:extension base="RealOperationType">
        <xsd:choice minOccurs="1" maxOccurs="1">
          <xsd:element ref="RealVariable"  minOccurs="1" maxOccurs="1" />
          <xsd:element ref="RealOperation" minOccurs="1" maxOccurs="1" />
          <xsd:element ref="RealValue"     minOccurs="1" maxOccurs="1" />
        </xsd:choice>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:element>

<!-- Multiplicative (*/%) - make sense for doubles -->

<xsd:element name="RealMultiplication" substitutionGroup="RealOperation">
  <xsd:complexType>
    <xsd:complexContent>
      <xsd:extension base="RealOperationType">
        <xsd:sequence>
          <xsd:element name="RealLHS" type="RealElement"
               minOccurs="1" maxOccurs="1" />
          <xsd:element name="RealRHS" type="RealElement"
               minOccurs="1" maxOccurs="1" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:element>

<xsd:element name="RealDivision" substitutionGroup="RealOperation">
  <xsd:complexType>
    <xsd:complexContent>
      <xsd:extension base="RealOperationType">
        <xsd:sequence>
          <xsd:element name="RealLHS" type="RealElement"
```

```
                minOccurs="1" maxOccurs="1" />
            <xsd:element name="RealRHS" type="RealElement"
                minOccurs="1" maxOccurs="1" />
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="RealRemainder" substitutionGroup="RealOperation">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="RealOperationType">
          <xsd:sequence>
            <xsd:element name="RealLHS" type="RealElement"
                minOccurs="1" maxOccurs="1" />
            <xsd:element name="RealRHS" type="RealElement"
                minOccurs="1" maxOccurs="1" />
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>

<!-- Addative (+-) - makes sense for doubles -->

  <xsd:element name="RealAddition" substitutionGroup="RealOperation">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="RealOperationType">
          <xsd:sequence>
            <xsd:element name="RealLHS" type="RealElement"
                minOccurs="1" maxOccurs="1" />
            <xsd:element name="RealRHS" type="RealElement"
                minOccurs="1" maxOccurs="1" />
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="RealSubtraction" substitutionGroup="RealOperation">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="RealOperationType">
          <xsd:sequence>
            <xsd:element name="RealLHS" type="RealElement"
                minOccurs="1" maxOccurs="1" />
            <xsd:element name="RealRHS" type="RealElement"
                minOccurs="1" maxOccurs="1" />
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>
```

```
<!-- Shift (<< >> >>>) - make no sense for strings -->

  <xsd:element name="RealShiftLeft" substitutionGroup="RealOperation">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="RealOperationType">
          <xsd:sequence>
            <xsd:element name="RealLHS" type="RealElement"
                 minOccurs="1" maxOccurs="1" />
            <xsd:element name="RealRHS" type="RealElement"
                 minOccurs="1" maxOccurs="1" />
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="RealShiftRight" substitutionGroup="RealOperation">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="RealOperationType">
          <xsd:sequence>
            <xsd:element name="RealLHS" type="RealElement"
                 minOccurs="1" maxOccurs="1" />
            <xsd:element name="RealRHS" type="RealElement"
                 minOccurs="1" maxOccurs="1" />
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="RealUnsignedShiftRight" substitutionGroup="RealOperation">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="RealOperationType">
          <xsd:sequence>
            <xsd:element name="RealLHS" type="RealElement"
                 minOccurs="1" maxOccurs="1" />
            <xsd:element name="RealRHS" type="RealElement"
                 minOccurs="1" maxOccurs="1" />
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>

<!-- Relational (< <= > >=) - make sense for doubles -->

  <xsd:element name="RealLessThan" substitutionGroup="BooleanOperation">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="BooleanOperationType">
          <xsd:sequence>
            <xsd:element name="RealLHS" type="RealElement"
                 minOccurs="1" maxOccurs="1" />
```

```
            <xsd:element name="RealRHS" type="RealElement"
                 minOccurs="1" maxOccurs="1" />
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="RealLessThanOrEquals" substitutionGroup="BooleanOperation">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="BooleanOperationType">
          <xsd:sequence>
            <xsd:element name="RealLHS" type="RealElement"
                 minOccurs="1" maxOccurs="1" />
            <xsd:element name="RealRHS" type="RealElement"
                 minOccurs="1" maxOccurs="1" />
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="RealGreaterThan" substitutionGroup="BooleanOperation">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="BooleanOperationType">
          <xsd:sequence>
            <xsd:element name="RealLHS" type="RealElement"
                 minOccurs="1" maxOccurs="1" />
            <xsd:element name="RealRHS" type="RealElement"
                 minOccurs="1" maxOccurs="1" />
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="RealGreaterThanOrEquals" substitutionGroup="BooleanOperation">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="BooleanOperationType">
          <xsd:sequence>
            <xsd:element name="RealLHS" type="RealElement"
                 minOccurs="1" maxOccurs="1" />
            <xsd:element name="RealRHS" type="RealElement"
                 minOccurs="1" maxOccurs="1" />
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>

<!-- Equality (== != is isnt) - all make sense for doubles -->

  <xsd:element name="RealEquals" substitutionGroup="BooleanOperation">
```

```
  <xsd:complexType>
    <xsd:complexContent>
      <xsd:extension base="BooleanOperationType">
        <xsd:sequence>
          <xsd:element name="RealLHS" type="RealElement"
               minOccurs="1" maxOccurs="1" />
          <xsd:element name="RealRHS" type="RealElement"
               minOccurs="1" maxOccurs="1" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:element>

<xsd:element name="RealNotEqual" substitutionGroup="BooleanOperation">
  <xsd:complexType>
    <xsd:complexContent>
      <xsd:extension base="BooleanOperationType">
        <xsd:sequence>
          <xsd:element name="RealLHS" type="RealElement"
               minOccurs="1" maxOccurs="1" />
          <xsd:element name="RealRHS" type="RealElement"
               minOccurs="1" maxOccurs="1" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:element>

<xsd:element name="RealIs" substitutionGroup="BooleanOperation">
  <xsd:complexType>
    <xsd:complexContent>
      <xsd:extension base="BooleanOperationType">
        <xsd:sequence>
          <xsd:element name="RealLHS" type="RealElement"
               minOccurs="1" maxOccurs="1" />
          <xsd:element name="RealRHS" type="RealElement"
               minOccurs="1" maxOccurs="1" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:element>

<xsd:element name="RealIsnt" substitutionGroup="BooleanOperation">
  <xsd:complexType>
    <xsd:complexContent>
      <xsd:extension base="BooleanOperationType">
        <xsd:sequence>
          <xsd:element name="RealLHS" type="RealElement"
               minOccurs="1" maxOccurs="1" />
          <xsd:element name="RealRHS" type="RealElement"
               minOccurs="1" maxOccurs="1" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
```

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

<!-- Bitwise AND - makes sense -->

  <xsd:element name="RealBitwiseAND" substitutionGroup="RealOperation">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="RealOperationType">
          <xsd:sequence>
            <xsd:element name="RealLHS" type="RealElement"
                 minOccurs="1" maxOccurs="1" />
            <xsd:element name="RealRHS" type="RealElement"
                 minOccurs="1" maxOccurs="1" />
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>

<!-- Bitwise XOR - no sense -->

  <xsd:element name="RealBitwiseXOR" substitutionGroup="RealOperation">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="RealOperationType">
          <xsd:sequence>
            <xsd:element name="RealLHS" type="RealElement"
                 minOccurs="1" maxOccurs="1" />
            <xsd:element name="RealRHS" type="RealElement"
                 minOccurs="1" maxOccurs="1" />
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>

<!-- Bitwise OR - no sense -->

  <xsd:element name="RealBitwiseOR" substitutionGroup="RealOperation">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="RealOperationType">
          <xsd:sequence>
            <xsd:element name="RealLHS" type="RealElement"
                 minOccurs="1" maxOccurs="1" />
            <xsd:element name="RealRHS" type="RealElement"
                 minOccurs="1" maxOccurs="1" />
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>

<!-- Logical AND - no sense -->
```

```
<!-- Logical OR - no sense -->

<!-- Conditional - makes sense-->

  <xsd:element name="ConditionalRealResult"
       substitutionGroup="RealOperation">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="RealOperationType">
          <xsd:sequence>
            <xsd:element name="BooleanTest"     type="BooleanElement"
                 minOccurs="1" maxOccurs="1" />
            <xsd:element name="RealTrueResult"  type="RealElement"
                 minOccurs="1" maxOccurs="1" />
            <xsd:element name="RealFalseResult" type="RealElement"
                 minOccurs="1" maxOccurs="1" />
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>

<!-- Compound () statment -->

  <xsd:element name="RealCompound" substitutionGroup="RealOperation">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="RealOperationType">
          <xsd:sequence>
            <xsd:choice minOccurs="1" maxOccurs="1">
              <xsd:element ref="RealValue"     minOccurs="1" maxOccurs="1" />
              <xsd:element ref="RealVariable"  minOccurs="1" maxOccurs="1" />
              <xsd:element ref="RealOperation" minOccurs="1" maxOccurs="1" />
            </xsd:choice>
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>

<!--=========================================================================-->
<!-- INTEGER -->
<!--=========================================================================-->

  <xsd:complexType name="IntegerElement">
    <xsd:choice minOccurs="1" maxOccurs="1">
      <xsd:element ref="IntegerVariable"  minOccurs="1" maxOccurs="1" />
      <xsd:element ref="IntegerValue"     minOccurs="1" maxOccurs="1" />
      <xsd:element ref="IntegerOperation" minOccurs="1" maxOccurs="1" />
    </xsd:choice>
  </xsd:complexType>

  <xsd:element name="IntegerEquation">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="IntegerElement">
```

```
          <xsd:attribute name="attribute" type="xsd:string" use="required" />
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="IntegerValue" type="IntegerValueType" />

  <xsd:complexType name="IntegerValueType">
    <xsd:simpleContent>
      <xsd:extension base="xsd:integer">
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>

  <xsd:element name="IntegerVariable" type="IntegerVariableType" />

  <xsd:complexType name="IntegerVariableType">
    <xsd:attribute name="context" type="ContextType" default="self" />
    <xsd:attribute name="name" type="xsd:string" /> <!-- use="required" /> -->
  </xsd:complexType>

  <xsd:element name="IntegerOperation" type="IntegerOperationType" />

  <xsd:complexType name="IntegerOperationType">
  </xsd:complexType>

<!-- Integer Operations -->

<!-- Unary operations (-+!~) - +/-/~ make sense for doubles -->

  <xsd:element name="IntegerUnaryPositive" substitutionGroup="IntegerOperation">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="IntegerOperationType">
          <xsd:choice minOccurs="1" maxOccurs="1">
            <xsd:element ref="IntegerVariable"  minOccurs="1" maxOccurs="1" />
            <xsd:element ref="IntegerOperation" minOccurs="1" maxOccurs="1" />
            <xsd:element ref="IntegerValue"     minOccurs="1" maxOccurs="1" />
          </xsd:choice>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="IntegerUnaryNegative" substitutionGroup="IntegerOperation">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="IntegerOperationType">
          <xsd:choice minOccurs="1" maxOccurs="1">
            <xsd:element ref="IntegerVariable"  minOccurs="1" maxOccurs="1" />
            <xsd:element ref="IntegerOperation" minOccurs="1" maxOccurs="1" />
            <xsd:element ref="IntegerValue"     minOccurs="1" maxOccurs="1" />
          </xsd:choice>
        </xsd:extension>
      </xsd:complexContent>
```

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

  <xsd:element name="IntegerOnesComplement" substitutionGroup="IntegerOperation">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="IntegerOperationType">
          <xsd:choice minOccurs="1" maxOccurs="1">
            <xsd:element ref="IntegerVariable"  minOccurs="1" maxOccurs="1" />
            <xsd:element ref="IntegerOperation" minOccurs="1" maxOccurs="1" />
            <xsd:element ref="IntegerValue"     minOccurs="1" maxOccurs="1" />
          </xsd:choice>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>

<!-- Multiplicative (*/%) - make sense for doubles -->

  <xsd:element name="IntegerMultiplication" substitutionGroup="IntegerOperation">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="IntegerOperationType">
          <xsd:sequence>
            <xsd:element name="IntegerLHS" type="IntegerElement"
                 minOccurs="1" maxOccurs="1" />
            <xsd:element name="IntegerRHS" type="IntegerElement"
                 minOccurs="1" maxOccurs="1" />
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="IntegerDivision" substitutionGroup="IntegerOperation">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="IntegerOperationType">
          <xsd:sequence>
            <xsd:element name="IntegerLHS" type="IntegerElement"
                 minOccurs="1" maxOccurs="1" />
            <xsd:element name="IntegerRHS" type="IntegerElement"
                 minOccurs="1" maxOccurs="1" />
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="IntegerRemainder" substitutionGroup="IntegerOperation">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="IntegerOperationType">
          <xsd:sequence>
            <xsd:element name="IntegerLHS" type="IntegerElement"
                 minOccurs="1" maxOccurs="1" />
```

```
            <xsd:element name="IntegerRHS" type="IntegerElement"
                 minOccurs="1" maxOccurs="1" />
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>

<!-- Addative (+-) - makes sense for doubles -->

  <xsd:element name="IntegerAddition" substitutionGroup="IntegerOperation">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="IntegerOperationType">
          <xsd:sequence>
            <xsd:element name="IntegerLHS" type="IntegerElement"
                 minOccurs="1" maxOccurs="1" />
            <xsd:element name="IntegerRHS" type="IntegerElement"
                 minOccurs="1" maxOccurs="1" />
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="IntegerSubtraction" substitutionGroup="IntegerOperation">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="IntegerOperationType">
          <xsd:sequence>
            <xsd:element name="IntegerLHS" type="IntegerElement"
                 minOccurs="1" maxOccurs="1" />
            <xsd:element name="IntegerRHS" type="IntegerElement"
                 minOccurs="1" maxOccurs="1" />
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>

<!-- Shift (<< >> >>>) - make no sense for strings -->

  <xsd:element name="IntegerShiftLeft" substitutionGroup="IntegerOperation">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="IntegerOperationType">
          <xsd:sequence>
            <xsd:element name="IntegerLHS" type="IntegerElement"
                 minOccurs="1" maxOccurs="1" />
            <xsd:element name="IntegerRHS" type="IntegerElement"
                 minOccurs="1" maxOccurs="1" />
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>
```

```
  <xsd:element name="IntegerShiftRight" substitutionGroup="IntegerOperation">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="IntegerOperationType">
          <xsd:sequence>
            <xsd:element name="IntegerLHS" type="IntegerElement"
                 minOccurs="1" maxOccurs="1" />
            <xsd:element name="IntegerRHS" type="IntegerElement"
                 minOccurs="1" maxOccurs="1" />
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="IntegerUnsignedShiftRight" substitutionGroup="IntegerOperation">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="IntegerOperationType">
          <xsd:sequence>
            <xsd:element name="IntegerLHS" type="IntegerElement"
                 minOccurs="1" maxOccurs="1" />
            <xsd:element name="IntegerRHS" type="IntegerElement"
                 minOccurs="1" maxOccurs="1" />
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>

<!-- Relational (< <= > >=) - make sense for doubles -->

  <xsd:element name="IntegerLessThan" substitutionGroup="BooleanOperation">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="BooleanOperationType">
          <xsd:sequence>
            <xsd:element name="IntegerLHS" type="IntegerElement"
                 minOccurs="1" maxOccurs="1" />
            <xsd:element name="IntegerRHS" type="IntegerElement"
                 minOccurs="1" maxOccurs="1" />
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="IntegerLessThanOrEquals" substitutionGroup="BooleanOperation">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="BooleanOperationType">
          <xsd:sequence>
            <xsd:element name="IntegerLHS" type="IntegerElement"
                 minOccurs="1" maxOccurs="1" />
            <xsd:element name="IntegerRHS" type="IntegerElement"
```

```
                minOccurs="1" maxOccurs="1" />
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="IntegerGreaterThan" substitutionGroup="BooleanOperation">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="BooleanOperationType">
          <xsd:sequence>
            <xsd:element name="IntegerLHS" type="IntegerElement"
                 minOccurs="1" maxOccurs="1" />
            <xsd:element name="IntegerRHS" type="IntegerElement"
                 minOccurs="1" maxOccurs="1" />
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="IntegerGreaterThanOrEquals" substitutionGroup="BooleanOperation">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="BooleanOperationType">
          <xsd:sequence>
            <xsd:element name="IntegerLHS" type="IntegerElement"
                 minOccurs="1" maxOccurs="1" />
            <xsd:element name="IntegerRHS" type="IntegerElement"
                 minOccurs="1" maxOccurs="1" />
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>

<!-- Equality (== != is isnt) - all make sense for doubles -->

  <xsd:element name="IntegerEquals" substitutionGroup="BooleanOperation">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="BooleanOperationType">
          <xsd:sequence>
            <xsd:element name="IntegerLHS" type="IntegerElement"
                 minOccurs="1" maxOccurs="1" />
            <xsd:element name="IntegerRHS" type="IntegerElement"
                 minOccurs="1" maxOccurs="1" />
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="IntegerNotEqual" substitutionGroup="BooleanOperation">
    <xsd:complexType>
```

```
      <xsd:complexContent>
        <xsd:extension base="BooleanOperationType">
          <xsd:sequence>
            <xsd:element name="IntegerLHS" type="IntegerElement"
                 minOccurs="1" maxOccurs="1" />
            <xsd:element name="IntegerRHS" type="IntegerElement"
                 minOccurs="1" maxOccurs="1" />
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="IntegerIs" substitutionGroup="BooleanOperation">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="BooleanOperationType">
          <xsd:sequence>
            <xsd:element name="IntegerLHS" type="IntegerElement"
                 minOccurs="1" maxOccurs="1" />
            <xsd:element name="IntegerRHS" type="IntegerElement"
                 minOccurs="1" maxOccurs="1" />
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="IntegerIsnt" substitutionGroup="BooleanOperation">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="BooleanOperationType">
          <xsd:sequence>
            <xsd:element name="IntegerLHS" type="IntegerElement"
                 minOccurs="1" maxOccurs="1" />
            <xsd:element name="IntegerRHS" type="IntegerElement"
                 minOccurs="1" maxOccurs="1" />
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>

<!-- Bitwise AND - makes sense -->

  <xsd:element name="IntegerBitwiseAND" substitutionGroup="IntegerOperation">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="IntegerOperationType">
          <xsd:sequence>
            <xsd:element name="IntegerLHS" type="IntegerElement"
                 minOccurs="1" maxOccurs="1" />
            <xsd:element name="IntegerRHS" type="IntegerElement"
                 minOccurs="1" maxOccurs="1" />
          </xsd:sequence>
        </xsd:extension>
```

```
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>

<!-- Bitwise XOR - no sense -->

  <xsd:element name="IntegerBitwiseXOR" substitutionGroup="IntegerOperation">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="IntegerOperationType">
          <xsd:sequence>
            <xsd:element name="IntegerLHS" type="IntegerElement"
                 minOccurs="1" maxOccurs="1" />
            <xsd:element name="IntegerRHS" type="IntegerElement"
                 minOccurs="1" maxOccurs="1" />
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>

<!-- Bitwise OR - no sense -->

  <xsd:element name="IntegerBitwiseOR" substitutionGroup="IntegerOperation">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="IntegerOperationType">
          <xsd:sequence>
            <xsd:element name="IntegerLHS" type="IntegerElement"
                 minOccurs="1" maxOccurs="1" />
            <xsd:element name="IntegerRHS" type="IntegerElement"
                 minOccurs="1" maxOccurs="1" />
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>

<!-- Logical AND - no sense -->

<!-- Logical OR - no sense -->

<!-- Conditional - makes sense-->

  <xsd:element name="ConditionalIntegerResult"
       substitutionGroup="IntegerOperation">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="IntegerOperationType">
          <xsd:sequence>
            <xsd:element name="BooleanTest"       type="BooleanElement"
                 minOccurs="1" maxOccurs="1" />
            <xsd:element name="IntegerTrueResult"  type="IntegerElement"
                 minOccurs="1" maxOccurs="1" />
            <xsd:element name="IntegerFalseResult" type="IntegerElement"
                 minOccurs="1" maxOccurs="1" />
```

```
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>

<!-- Compound () statment -->

  <xsd:element name="IntegerCompound" substitutionGroup="IntegerOperation">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="IntegerOperationType">
          <xsd:sequence>
            <xsd:choice minOccurs="1" maxOccurs="1">
              <xsd:element ref="IntegerValue"    minOccurs="1" maxOccurs="1"/>
              <xsd:element ref="IntegerVariable" minOccurs="1" maxOccurs="1"/>
              <xsd:element ref="IntegerOperation"
                   minOccurs="1" maxOccurs="1" />
            </xsd:choice>
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>

<!--=========================================================================-->
<!-- BOOLEAN -->
<!--=========================================================================-->

  <xsd:complexType name="BooleanElement">
    <xsd:choice minOccurs="1" maxOccurs="1">
      <xsd:element ref="BooleanVariable"  minOccurs="1" maxOccurs="1" />
      <xsd:element ref="BooleanValue"     minOccurs="1" maxOccurs="1" />
      <xsd:element ref="BooleanOperation" minOccurs="1" maxOccurs="1" />
    </xsd:choice>
  </xsd:complexType>

  <xsd:element name="BooleanEquation">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="BooleanElement">
          <xsd:attribute name="attribute" type="xsd:string" use="required" />
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="BooleanValue" type="BooleanValueType" />

  <xsd:complexType name="BooleanValueType">
    <xsd:simpleContent>
      <xsd:extension base="xsd:string">
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>
```

```
  <xsd:element name="BooleanVariable" type="BooleanVariableType" />

  <xsd:complexType name="BooleanVariableType">
    <xsd:attribute name="name" type="xsd:string" /> <!-- breaks use="required" /> -->
    <xsd:attribute name="context" type="ContextType" default="self"/>
  </xsd:complexType>

  <xsd:element name="BooleanOperation" type="BooleanOperationType" />

  <xsd:complexType name="BooleanOperationType">
  </xsd:complexType>


<!-- Boolean Operations -->

<!-- Unary operations (-+!~) - ! make sense for booleans -->

  <xsd:element name="BooleanNot" substitutionGroup="BooleanOperation">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="BooleanOperationType">
          <xsd:sequence>
            <xsd:choice minOccurs="1" maxOccurs="1">
              <xsd:element ref="BooleanValue"    minOccurs="1" maxOccurs="1"/>
              <xsd:element ref="BooleanVariable" minOccurs="1" maxOccurs="1"/>
              <xsd:element ref="BooleanOperation"
                   minOccurs="1" maxOccurs="1" />
            </xsd:choice>
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>

<!-- Multiplicative (*/%) - make no sense for booleans -->

<!-- Addative (+-) - no sense -->

<!-- Shift (<< >> >>>) - make no sense for booleans -->

<!-- Relational (< <= > >=) - No sense -->

<!-- Equality (== != is isnt) - all make sense for strings -->

  <xsd:element name="BooleanEquals" substitutionGroup="BooleanOperation">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="BooleanOperationType">
          <xsd:sequence>
            <xsd:element name="BooleanLHS" type="BooleanElement"
                 minOccurs="1" maxOccurs="1" />
            <xsd:element name="BooleanRHS" type="BooleanElement"
                 minOccurs="1" maxOccurs="1" />
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
```

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

  <xsd:element name="BooleanNotEqual" substitutionGroup="BooleanOperation">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="BooleanOperationType">
          <xsd:sequence>
            <xsd:element name="BooleanLHS" type="BooleanElement"
                 minOccurs="1" maxOccurs="1" />
            <xsd:element name="BooleanRHS" type="BooleanElement"
                 minOccurs="1" maxOccurs="1" />
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="BooleanIs" substitutionGroup="BooleanOperation">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="BooleanOperationType">
          <xsd:sequence>
            <xsd:element name="BooleanLHS" type="BooleanElement"
                 minOccurs="1" maxOccurs="1" />
            <xsd:element name="BooleanRHS" type="BooleanElement"
                 minOccurs="1" maxOccurs="1" />
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="BooleanIsnt" substitutionGroup="BooleanOperation">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="BooleanOperationType">
          <xsd:sequence>
            <xsd:element name="BooleanLHS" type="BooleanElement"
                 minOccurs="1" maxOccurs="1" />
            <xsd:element name="BooleanRHS" type="BooleanElement"
                 minOccurs="1" maxOccurs="1" />
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>

<!-- Bitwise AND - no sense -->

<!-- Bitwise XOR - no sense -->

<!-- Bitwise OR - no sense -->

<!-- Logical AND - sense -->
```

```
  <xsd:element name="LogicalAND" substitutionGroup="BooleanOperation">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="BooleanOperationType">
          <xsd:sequence>
            <xsd:element name="BooleanLHS" type="BooleanElement"
                 minOccurs="1" maxOccurs="1" />
            <xsd:element name="BooleanRHS" type="BooleanElement"
                 minOccurs="1" maxOccurs="1" />
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>

<!-- Logical OR - makes sense -->

  <xsd:element name="LogicalOR" substitutionGroup="BooleanOperation">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="BooleanOperationType">
          <xsd:sequence>
            <xsd:element name="BooleanLHS" type="BooleanElement"
                 minOccurs="1" maxOccurs="1" />
            <xsd:element name="BooleanRHS" type="BooleanElement"
                 minOccurs="1" maxOccurs="1" />
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>

<!-- Conditional - makes sense-->

  <xsd:element name="ConditionalBooleanResult"
       substitutionGroup="BooleanOperation">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="BooleanOperationType">
          <xsd:sequence>
            <xsd:element name="BooleanTest"       type="BooleanElement"
                 minOccurs="1" maxOccurs="1" />
            <xsd:element name="BooleanTrueResult"  type="BooleanElement"
                 minOccurs="1" maxOccurs="1" />
            <xsd:element name="BooleanFalseResult" type="BooleanElement"
                 minOccurs="1" maxOccurs="1" />
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>

<!-- Compound () statment -->

  <xsd:element name="BooleanCompound" substitutionGroup="BooleanOperation">
    <xsd:complexType>
```

```
      <xsd:complexContent>
        <xsd:extension base="BooleanOperationType">
          <xsd:sequence>
            <xsd:choice minOccurs="1" maxOccurs="1">
              <xsd:element ref="BooleanValue"     minOccurs="1" maxOccurs="1" />
              <xsd:element ref="BooleanVariable"  minOccurs="1" maxOccurs="1" />
              <xsd:element ref="BooleanOperation" minOccurs="1" maxOccurs="1" />
            </xsd:choice>
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>

<!--=========================================================================-->
<!-- Functions provided by ClassAdds -->
<!--=========================================================================-->
<!-- Type predicates (Non-Strict) -->
<!--  IsUndefined(V) - True iff V is the undefined value. -->

  <xsd:complexType name="CoreBooleanFunctionOnePeram" abstract="true">
    <xsd:complexContent>
      <xsd:extension base="BooleanOperationType">
        <xsd:sequence>
          <xsd:choice minOccurs="1" maxOccurs="1">
            <xsd:element ref="BooleanValue"     minOccurs="1" maxOccurs="1" />
            <xsd:element ref="BooleanVariable"  minOccurs="1" maxOccurs="1" />
            <xsd:element ref="BooleanOperation" minOccurs="1" maxOccurs="1" />
            <xsd:element ref="StringValue"      minOccurs="1" maxOccurs="1" />
            <xsd:element ref="StringVariable"   minOccurs="1" maxOccurs="1" />
            <xsd:element ref="StringOperation"  minOccurs="1" maxOccurs="1" />
            <xsd:element ref="RealValue"        minOccurs="1" maxOccurs="1" />
            <xsd:element ref="RealVariable"     minOccurs="1" maxOccurs="1" />
            <xsd:element ref="RealOperation"    minOccurs="1" maxOccurs="1" />
            <xsd:element ref="IntegerValue"     minOccurs="1" maxOccurs="1" />
            <xsd:element ref="IntegerVariable"  minOccurs="1" maxOccurs="1" />
            <xsd:element ref="IntegerOperation" minOccurs="1" maxOccurs="1" />
            <xsd:element ref="SectionValue"     minOccurs="1" maxOccurs="1" />
            <xsd:element ref="SectionVariable"  minOccurs="1" maxOccurs="1" />
            <xsd:element ref="SectionOperation" minOccurs="1" maxOccurs="1" />
          </xsd:choice>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>

  <xsd:complexType name="CoreBooleanFunctionOneTimePeram" abstract="true">
    <xsd:complexContent>
      <xsd:extension base="BooleanOperationType">
        <xsd:sequence>
          <xsd:choice minOccurs="1" maxOccurs="1">
            <xsd:element ref="StringValue"      minOccurs="1" maxOccurs="1" />
            <xsd:element ref="StringVariable"   minOccurs="1" maxOccurs="1" />
            <xsd:element ref="StringOperation"  minOccurs="1" maxOccurs="1" />
            <xsd:element ref="RealValue"        minOccurs="1" maxOccurs="1" />
            <xsd:element ref="RealVariable"     minOccurs="1" maxOccurs="1" />
```

```
          <xsd:element ref="RealOperation"    minOccurs="1" maxOccurs="1" />
          <xsd:element ref="IntegerValue"     minOccurs="1" maxOccurs="1" />
          <xsd:element ref="IntegerVariable"  minOccurs="1" maxOccurs="1" />
          <xsd:element ref="IntegerOperation" minOccurs="1" maxOccurs="1" />
        </xsd:choice>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="CoreIntegerFunctionOneTimePeram" abstract="true">
  <xsd:complexContent>
    <xsd:extension base="IntegerOperationType">
      <xsd:sequence>
        <xsd:choice minOccurs="1" maxOccurs="1">
          <xsd:element ref="StringValue"      minOccurs="1" maxOccurs="1" />
          <xsd:element ref="StringVariable"   minOccurs="1" maxOccurs="1" />
          <xsd:element ref="StringOperation"  minOccurs="1" maxOccurs="1" />
          <xsd:element ref="RealValue"        minOccurs="1" maxOccurs="1" />
          <xsd:element ref="RealVariable"     minOccurs="1" maxOccurs="1" />
          <xsd:element ref="RealOperation"    minOccurs="1" maxOccurs="1" />
          <xsd:element ref="IntegerValue"     minOccurs="1" maxOccurs="1" />
          <xsd:element ref="IntegerVariable"  minOccurs="1" maxOccurs="1" />
          <xsd:element ref="IntegerOperation" minOccurs="1" maxOccurs="1" />
        </xsd:choice>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="CoreStringFunctionOneStringParam" abstract="true">
  <xsd:complexContent>
    <xsd:extension base="StringOperationType">
      <xsd:sequence minOccurs="1" maxOccurs="unbounded">
        <xsd:choice>
          <xsd:element ref="StringValue"    minOccurs="1" maxOccurs="1" />
          <xsd:element ref="StringVariable" minOccurs="1" maxOccurs="1" />
          <xsd:element ref="StringOperation" minOccurs="1" maxOccurs="1"/>
        </xsd:choice>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="CoreIntegerFunctionOneParam" abstract="true">
  <xsd:complexContent>
    <xsd:extension base="IntegerOperationType">
      <xsd:sequence minOccurs="1" maxOccurs="unbounded">
        <xsd:choice>
          <xsd:element ref="IntegerValue"     minOccurs="1" maxOccurs="1" />
          <xsd:element ref="IntegerVariable"  minOccurs="1" maxOccurs="1" />
          <xsd:element ref="IntegerOperation" minOccurs="1" maxOccurs="1"/>
          <xsd:element ref="RealValue"        minOccurs="1" maxOccurs="1" />
          <xsd:element ref="RealVariable"     minOccurs="1" maxOccurs="1" />
          <xsd:element ref="RealOperation"    minOccurs="1" maxOccurs="1"/>
        </xsd:choice>
```

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

  <xsd:element name="IsUndefined" type="CoreBooleanFunctionOnePeram"
       substitutionGroup="BooleanOperation" />

<!--  IsError(V) - True iff V is the error value. -->

  <xsd:element name="IsError" type="CoreBooleanFunctionOnePeram"
       substitutionGroup="BooleanOperation" />

<!--  IsString(V) - True iff V is a string value. -->

  <xsd:element name="IsString" type="CoreBooleanFunctionOnePeram"
       substitutionGroup="BooleanOperation" />

<!--  IsList(V) - True iff V is a list value. -->

  <xsd:element name="IsStringList" type="CoreBooleanFunctionOnePeram"
       substitutionGroup="BooleanOperation" />

<!--  IsClassad(V) - True iff V is a classad value. -->

  <xsd:element name="IsClassAdd" type="CoreBooleanFunctionOnePeram"
       substitutionGroup="BooleanOperation" />


<!--  IsBoolean(V) - True iff V is a boolean value. -->

  <xsd:element name="IsBoolean" type="CoreBooleanFunctionOnePeram"
       substitutionGroup="BooleanOperation" />

<!--  IsAbsTime(V) - True iff V is an absolute time value. -->

  <xsd:element name="IsAbsTime" type="CoreBooleanFunctionOneTimePeram"
       substitutionGroup="BooleanOperation" />

<!--  IsRelTime(V) - True iff V is a relative time value. -->

  <xsd:element name="IsRelTime" type="CoreBooleanFunctionOneTimePeram"
       substitutionGroup="BooleanOperation" />

<!-- List Membership -->
<!--  Member(V,L) - True iff scalar value V is a member of the list L. -->

  <xsd:element name="Member" substitutionGroup="BooleanOperation">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="BooleanOperationType">
          <xsd:sequence>
            <xsd:element name="StringSearch" type="StringElement"
                 minOccurs="1" maxOccurs="1" />
            <xsd:element name="StringList"   type="StringListElement"
                 minOccurs="1" maxOccurs="1" />
```

```
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>

<!--  IsMember(V,L) - Like Member, but uses is for comparison instead of ==.
      Not strict on first argument. -->

  <xsd:element name="IsMember" substitutionGroup="BooleanOperation">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="BooleanOperationType">
          <xsd:sequence>
            <xsd:element name="StringSearch" type="StringElement"
                 minOccurs="1" maxOccurs="1" />
            <xsd:element name="StringList"   type="StringListElement"
                 minOccurs="1" maxOccurs="1" />
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>

<!-- Time Queries   -->
<!-- CurrentTime() - Get current time (absolute time) -->

  <xsd:element name="CurrentTime" substitutionGroup="IntegerOperation">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="IntegerOperationType">
          <xsd:sequence>
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>

<!--  TimeZoneOffset() - Get time zone offset as a relative time -->

  <xsd:element name="TimeZoneOffset" substitutionGroup="RealOperation">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="RealOperationType">
          <xsd:sequence>
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>

<!--  DayTime() - Get current time as relative time since midnight. -->

  <xsd:element name="DayTime" substitutionGroup="IntegerOperation">
    <xsd:complexType>
      <xsd:complexContent>
```

```
        <xsd:extension base="IntegerOperationType">
          <xsd:sequence>
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>

<!-- Time Construction -->

<!--  MakeDate(M,D,Y) - Create an absolute time value of midnight for the given day. M
can be either numeric or string (e.g., "jan"). -->

  <xsd:element name="MakeDate" substitutionGroup="IntegerOperation">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="IntegerOperationType">
          <xsd:sequence>
            <xsd:choice>
              <xsd:element name="NumericMonth" type="MonthNum"
                   minOccurs="1" maxOccurs="1" />
              <xsd:element name="StringMonth" type="MonthName"
                   minOccurs="1" maxOccurs="1" />
            </xsd:choice>
            <xsd:element name="NumericDay" type="DayNum"
                 minOccurs="1" maxOccurs="1" />
            <xsd:element name="NumericYear" type="YearNum"
                 minOccurs="1" maxOccurs="1" />
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>

<xsd:simpleType name="MonthNum">
<xsd:restriction base="xsd:integer">
<xsd:minInclusive value="0" />
<xsd:maxInclusive value="11" />
</xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="DayNum">
<xsd:restriction base="xsd:positiveInteger">
<xsd:minInclusive value="1" />
<xsd:maxInclusive value="31" />
</xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="YearNum">
<xsd:restriction base="xsd:positiveInteger">
<xsd:minInclusive value="1901" />
<xsd:maxInclusive value="9999" />
</xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="MonthName">
```

```
<xsd:restriction base="xsd:NMTOKEN">
<xsd:enumeration value="jan" />
<xsd:enumeration value="feb" />
<xsd:enumeration value="mar" />
<xsd:enumeration value="apr" />
<xsd:enumeration value="may" />
<xsd:enumeration value="jun" />
<xsd:enumeration value="jul" />
<xsd:enumeration value="aug" />
<xsd:enumeration value="sep" />
<xsd:enumeration value="oct" />
<xsd:enumeration value="nov" />
<xsd:enumeration value="dec" />
<xsd:enumeration value="january" />
<xsd:enumeration value="february" />
<xsd:enumeration value="march" />
<xsd:enumeration value="april" />
<xsd:enumeration value="may" />
<xsd:enumeration value="june" />
<xsd:enumeration value="july" />
<xsd:enumeration value="august" />
<xsd:enumeration value="september" />
<xsd:enumeration value="october" />
<xsd:enumeration value="november" />
<xsd:enumeration value="december" />
</xsd:restriction>
</xsd:simpleType>

<!--  MakeAbsTime(N) - Convert numeric value N into an absolute time (number of
seconds past UNIX epoch). -->

  <xsd:element name="MakeAbsTime" type="CoreIntegerFunctionOneTimePeram"
       substitutionGroup="IntegerOperation" />

<!--  MakeRelTime(N) - Convert numeric value N into a relative time (number of seconds in
interval). -->

  <xsd:element name="MakeRelTime" type="CoreIntegerFunctionOneTimePeram"
       substitutionGroup="IntegerOperation" />

<!-- Absolute Time Component Extraction -->

<!--  GetYear(A) - Get integer year (A=absolute time) -->

  <xsd:element name="GetYear" type="CoreIntegerFunctionOneTimePeram"
       substitutionGroup="IntegerOperation" />

<!--  GetMonth(A) - 0 = jan; . . . .; 11 = dec -->

  <xsd:element name="GetMonth" type="CoreIntegerFunctionOneTimePeram"
       substitutionGroup="IntegerOperation" />

<!--  GetDayOfYear(A) - 0 . . . .365 (for leap year) -->

  <xsd:element name="GetDayOfYear" type="CoreIntegerFunctionOneTimePeram"
       substitutionGroup="IntegerOperation" />
```

```
<!--  GetDayOfMonth(A) - 1 . . . .31 -->

  <xsd:element name="GetDayOfMonth" type="CoreIntegerFunctionOneTimePeram"
       substitutionGroup="IntegerOperation" />

<!--  GetDayOfWeek(A) - 0 . . . .6 -->

  <xsd:element name="GetDayOfWeek" type="CoreIntegerFunctionOneTimePeram"
       substitutionGroup="IntegerOperation" />

<!--  GetHours(A) - 0 . . . .23 -->

  <xsd:element name="GetHours" type="CoreIntegerFunctionOneTimePeram"
       substitutionGroup="IntegerOperation" />

<!--  GetMinutes(A) - 0 . . . .59 -->

  <xsd:element name="GetMinutes" type="CoreIntegerFunctionOneTimePeram"
       substitutionGroup="IntegerOperation" />

<!--  GetSeconds(A) - 0 . . . .61 (for leap seconds) -->

  <xsd:element name="GetSeconds" type="CoreIntegerFunctionOneTimePeram"
       substitutionGroup="IntegerOperation" />

<!-- Relative Time Component Extraction -->

<!--  GetDays(R) - Get days component in the interval (R= relative time) -->

  <xsd:element name="RelativeGetDays" type="CoreIntegerFunctionOneTimePeram"
       substitutionGroup="IntegerOperation" />

<!--  GetHours(R) - 0 . . . .23 -->

  <xsd:element name="RelativeGetHours" type="CoreIntegerFunctionOneTimePeram"
       substitutionGroup="IntegerOperation" />

<!--  GetMinutes(R) - 0 . . . .59 -->

  <xsd:element name="RelativeGetMinutes" type="CoreIntegerFunctionOneTimePeram"
       substitutionGroup="IntegerOperation" />

<!--  GetSeconds(R)  - 0 . . . .59 -->

  <xsd:element name="RelativeGetSeconds" type="CoreIntegerFunctionOneTimePeram"
       substitutionGroup="IntegerOperation" />

<!-- Time Conversion  -->

<!--  InDays(T) - Convert time value into number of days -->

  <xsd:element name="InDays" type="CoreIntegerFunctionOneTimePeram"
       substitutionGroup="IntegerOperation" />

<!--  InHours(T) - Convert time value into number of hours -->
```

```
  <xsd:element name="InHours" type="CoreIntegerFunctionOneTimePeram"
       substitutionGroup="IntegerOperation" />

<!--  InMinutes(T) - Convert time value into number of minutes -->

  <xsd:element name="InMinutes" type="CoreIntegerFunctionOneTimePeram"
       substitutionGroup="IntegerOperation" />

<!--  InSeconds(T) - Convert time value into number of seconds  -->

  <xsd:element name="InSeconds" type="CoreIntegerFunctionOneTimePeram"
       substitutionGroup="IntegerOperation" />

<!-- String Functions -->

<!--  StrCat(V1, . . . , Vn) - Concatenates string representations of values V1
      through Vn -->

  <xsd:element name="StringCat" type="CoreStringFunctionOneStringParam"
       substitutionGroup="StringOperation" />

<!--  ToUpper(S) - Upcases string S -->

  <xsd:element name="ToUpper" type="CoreStringFunctionOneStringParam"
       substitutionGroup="StringOperation" />

<!--ToLower(S) - Downcases string S -->

  <xsd:element name="ToLower" type="CoreStringFunctionOneStringParam"
       substitutionGroup="StringOperation" />

<!--  SubStr(S,offset [,len]) - Returns substring of S. Negative offsets and lengths
      count from the end of the string. -->

  <xsd:element name="SubStr" substitutionGroup="StringOperation">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="StringOperationType">
          <xsd:sequence>
            <xsd:element name="String" type="StringElement"
                 minOccurs="1" maxOccurs="1" />
            <xsd:element name="Offset" type="IntegerElement"
                 minOccurs="1" maxOccurs="1" />
            <xsd:element name="Length" type="IntegerElement"
                 minOccurs="0" maxOccurs="1" />
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>

<!--  RegExp(P,S) - Checks if S matches pattern P (both args must be strings). -->

  <xsd:element name="RegExp" substitutionGroup="BooleanOperation">
    <xsd:complexType>
```

```
      <xsd:complexContent>
        <xsd:extension base="BooleanOperationType">
          <xsd:sequence>
            <xsd:element name="Pattern" type="StringElement"
                 minOccurs="1" maxOccurs="1" />
            <xsd:element name="String" type="StringElement"
                 minOccurs="1" maxOccurs="1" />
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>

<!-- Type Conversion Functions -->

<!--  Int(V) - Converts V to an integer. Time values are converted to number of
seconds, strings are parsed, bools are mapped to 0 or 1. Other values
result in error  -->

  <xsd:element name="Int" substitutionGroup="IntegerOperation">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="IntegerOperationType">
          <xsd:choice>
            <xsd:element ref="StringValue"      minOccurs="1" maxOccurs="1" />
            <xsd:element ref="StringVariable"   minOccurs="1" maxOccurs="1" />
            <xsd:element ref="StringOperation"  minOccurs="1" maxOccurs="1" />
            <xsd:element ref="RealValue"        minOccurs="1" maxOccurs="1" />
            <xsd:element ref="RealVariable"     minOccurs="1" maxOccurs="1" />
            <xsd:element ref="RealOperation"    minOccurs="1" maxOccurs="1" />
            <xsd:element ref="IntegerValue"     minOccurs="1" maxOccurs="1" />
            <xsd:element ref="IntegerVariable"  minOccurs="1" maxOccurs="1" />
            <xsd:element ref="IntegerOperation" minOccurs="1" maxOccurs="1" />
            <xsd:element ref="BooleanValue"     minOccurs="1" maxOccurs="1" />
            <xsd:element ref="BooleanVariable"  minOccurs="1" maxOccurs="1" />
            <xsd:element ref="BooleanOperation" minOccurs="1" maxOccurs="1" />
          </xsd:choice>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>

<!--  Real(V) - Similar to Int(V), but to a real value. -->

  <xsd:element name="Real" substitutionGroup="RealOperation">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="RealOperationType">
          <xsd:choice>
            <xsd:element ref="StringValue"      minOccurs="1" maxOccurs="1" />
            <xsd:element ref="StringVariable"   minOccurs="1" maxOccurs="1" />
            <xsd:element ref="StringOperation"  minOccurs="1" maxOccurs="1" />
            <xsd:element ref="RealValue"        minOccurs="1" maxOccurs="1" />
            <xsd:element ref="RealVariable"     minOccurs="1" maxOccurs="1" />
            <xsd:element ref="RealOperation"    minOccurs="1" maxOccurs="1" />
            <xsd:element ref="IntegerValue"     minOccurs="1" maxOccurs="1" />
```

```
            <xsd:element ref="IntegerVariable"  minOccurs="1" maxOccurs="1" />
            <xsd:element ref="IntegerOperation" minOccurs="1" maxOccurs="1" />
            <xsd:element ref="BooleanValue"     minOccurs="1" maxOccurs="1" />
            <xsd:element ref="BooleanVariable"  minOccurs="1" maxOccurs="1" />
            <xsd:element ref="BooleanOperation" minOccurs="1" maxOccurs="1" />
          </xsd:choice>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>

<!--  String(V) - Converts V to its string representation -->

  <xsd:element name="String" substitutionGroup="StringOperation">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="StringOperationType">
          <xsd:choice>
            <xsd:element ref="StringValue"         minOccurs="1" maxOccurs="1" />
            <xsd:element ref="StringVariable"      minOccurs="1" maxOccurs="1" />
            <xsd:element ref="StringOperation"     minOccurs="1" maxOccurs="1" />
            <xsd:element ref="RealValue"           minOccurs="1" maxOccurs="1" />
            <xsd:element ref="RealVariable"        minOccurs="1" maxOccurs="1" />
            <xsd:element ref="RealOperation"       minOccurs="1" maxOccurs="1" />
            <xsd:element ref="IntegerValue"        minOccurs="1" maxOccurs="1" />
            <xsd:element ref="IntegerVariable"     minOccurs="1" maxOccurs="1" />
            <xsd:element ref="IntegerOperation"    minOccurs="1" maxOccurs="1" />
            <xsd:element ref="BooleanValue"        minOccurs="1" maxOccurs="1" />
            <xsd:element ref="BooleanVariable"     minOccurs="1" maxOccurs="1" />
            <xsd:element ref="BooleanOperation"    minOccurs="1" maxOccurs="1" />
            <xsd:element ref="StringListValue"     minOccurs="1" maxOccurs="1" />
            <xsd:element ref="StringListVariable"  minOccurs="1" maxOccurs="1" />
            <xsd:element ref="StringListOperation" minOccurs="1" maxOccurs="1" />
          </xsd:choice>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>

<!-- Bool(V) - Converts V to a boolean value. Empty strings, and zero values
               converted to false; non-empty strings and non-zero values converted to
               true.  -->

  <xsd:element name="Boolean" substitutionGroup="BooleanOperation">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="BooleanOperationType">
          <xsd:choice>
            <xsd:element ref="StringValue"      minOccurs="1" maxOccurs="1" />
            <xsd:element ref="StringVariable"   minOccurs="1" maxOccurs="1" />
            <xsd:element ref="StringOperation"  minOccurs="1" maxOccurs="1" />
            <xsd:element ref="RealValue"        minOccurs="1" maxOccurs="1" />
            <xsd:element ref="RealVariable"     minOccurs="1" maxOccurs="1" />
            <xsd:element ref="RealOperation"    minOccurs="1" maxOccurs="1" />
            <xsd:element ref="IntegerValue"     minOccurs="1" maxOccurs="1" />
            <xsd:element ref="IntegerVariable"  minOccurs="1" maxOccurs="1" />
```

```
            <xsd:element ref="IntegerOperation" minOccurs="1" maxOccurs="1" />
            <xsd:element ref="BooleanValue"     minOccurs="1" maxOccurs="1" />
            <xsd:element ref="BooleanVariable"  minOccurs="1" maxOccurs="1" />
            <xsd:element ref="BooleanOperation" minOccurs="1" maxOccurs="1" />
          </xsd:choice>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>

<!--  AbsTime(V) - Converts V to an absolute time. Numeric values treated as seconds
past UNIX  epoch, strings parsed as necessary. -->

  <xsd:element name="AbsTime" substitutionGroup="IntegerOperation">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="IntegerOperationType">
          <xsd:choice>
            <xsd:element ref="StringValue"      minOccurs="1" maxOccurs="1" />
            <xsd:element ref="StringVariable"   minOccurs="1" maxOccurs="1" />
            <xsd:element ref="StringOperation"  minOccurs="1" maxOccurs="1" />
            <xsd:element ref="RealValue"        minOccurs="1" maxOccurs="1" />
            <xsd:element ref="RealVariable"     minOccurs="1" maxOccurs="1" />
            <xsd:element ref="RealOperation"    minOccurs="1" maxOccurs="1" />
            <xsd:element ref="IntegerValue"     minOccurs="1" maxOccurs="1" />
            <xsd:element ref="IntegerVariable"  minOccurs="1" maxOccurs="1" />
            <xsd:element ref="IntegerOperation" minOccurs="1" maxOccurs="1" />
          </xsd:choice>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>

<!-- RelTime(V) - Converts V to an relative time. Numeric values treated as
     number of seconds, string parsed as necessary. -->

  <xsd:element name="RelTime" substitutionGroup="IntegerOperation">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="IntegerOperationType">
          <xsd:choice>
            <xsd:element ref="StringValue"      minOccurs="1" maxOccurs="1" />
            <xsd:element ref="StringVariable"   minOccurs="1" maxOccurs="1" />
            <xsd:element ref="StringOperation"  minOccurs="1" maxOccurs="1" />
            <xsd:element ref="RealValue"        minOccurs="1" maxOccurs="1" />
            <xsd:element ref="RealVariable"     minOccurs="1" maxOccurs="1" />
            <xsd:element ref="RealOperation"    minOccurs="1" maxOccurs="1" />
            <xsd:element ref="IntegerValue"     minOccurs="1" maxOccurs="1" />
            <xsd:element ref="IntegerVariable"  minOccurs="1" maxOccurs="1" />
            <xsd:element ref="IntegerOperation" minOccurs="1" maxOccurs="1" />
          </xsd:choice>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>
```

```
<!-- Mathematical Functions -->

<!--Floor(N) - Floor of numeric value N -->

  <xsd:element name="Floor" type="CoreIntegerFunctionOneParam"
       substitutionGroup="IntegerOperation" />

<!--  Ceil(N) - Ceiling of numeric value N -->

  <xsd:element name="Ceil" type="CoreIntegerFunctionOneParam"
       substitutionGroup="IntegerOperation" />

<!--  Round(N) - Rounded value of numeric value N -->

  <xsd:element name="Round" type="CoreIntegerFunctionOneParam"
       substitutionGroup="IntegerOperation" />

<!-- CONTEXT TYPE -->

<xsd:simpleType name="ContextType">
<xsd:restriction base="xsd:NMTOKEN">
<xsd:enumeration value="self"/>
<xsd:enumeration value="other"/>
</xsd:restriction>
</xsd:simpleType>

</xsd:schema>
```