

# DataGRID

## EDG ATF BASELINE API DOCUMENT

### API DOCUMENTATION

---

Document identifier:	<b>baseline-api-1.0.0-1</b>
EDMS id:	...
Date:	April 14, 2003
Work package:	<b>ATF</b>
Partner(s):	<b>CERN</b>
Lead Partner:	<b>CERN</b>
Document status:	<b>WORKING DRAFT</b>
Author(s):	EDG ATF
File:	<b>refman</b>

---

Abstract: This document defines the baseline API specification for EDG release 2.0.

## CONTENTS

<b>1. WORKLOAD MANAGEMENT</b>	<b>3</b>
1.1. EVENT CLASS REFERENCE . . . . .	3
1.1.1. DETAILED DESCRIPTION . . . . .	4
1.1.2. MEMBER FUNCTION DOCUMENTATION . . . . .	4
1.1.3. MEMBER DATA DOCUMENTATION . . . . .	4
1.2. JOBAD CLASS REFERENCE . . . . .	8
1.2.1. DETAILED DESCRIPTION . . . . .	9
1.2.2. CONSTRUCTOR & DESTRUCTOR DOCUMENTATION . . . . .	9
1.2.3. MEMBER FUNCTION DOCUMENTATION . . . . .	9
1.2.4. MEMBER DATA DOCUMENTATION . . . . .	15
1.3. JOBCOLLECTION CLASS REFERENCE . . . . .	15
1.3.1. DETAILED DESCRIPTION . . . . .	16
1.3.2. CONSTRUCTOR & DESTRUCTOR DOCUMENTATION . . . . .	16
1.3.3. MEMBER FUNCTION DOCUMENTATION . . . . .	17
1.4. JOBID CLASS REFERENCE . . . . .	19
1.4.1. CONSTRUCTOR & DESTRUCTOR DOCUMENTATION . . . . .	19
1.4.2. MEMBER FUNCTION DOCUMENTATION . . . . .	20
1.5. JOBSTATE CLASS REFERENCE . . . . .	21
1.5.1. DETAILED DESCRIPTION . . . . .	21
1.5.2. MEMBER FUNCTION DOCUMENTATION . . . . .	21
1.5.3. MEMBER DATA DOCUMENTATION . . . . .	21
1.6. JOBSTATUS CLASS REFERENCE . . . . .	22
1.6.1. DETAILED DESCRIPTION . . . . .	23
1.6.2. MEMBER FUNCTION DOCUMENTATION . . . . .	23
1.6.3. MEMBER DATA DOCUMENTATION . . . . .	23
1.7. USERCREDENTIAL CLASS REFERENCE . . . . .	27
1.7.1. DETAILED DESCRIPTION . . . . .	27
1.7.2. CONSTRUCTOR & DESTRUCTOR DOCUMENTATION . . . . .	27
1.7.3. MEMBER FUNCTION DOCUMENTATION . . . . .	28
1.8. USERJOBS CLASS REFERENCE . . . . .	29
1.8.1. DETAILED DESCRIPTION . . . . .	29
1.8.2. CONSTRUCTOR & DESTRUCTOR DOCUMENTATION . . . . .	29
1.8.3. MEMBER FUNCTION DOCUMENTATION . . . . .	29



---

<b>2. BROKER INFORMATION</b>	<b>30</b>
2.1. BROKERINFO CLASS REFERENCE . . . . .	30
2.1.1. DETAILED DESCRIPTION . . . . .	31
2.1.2. CONSTRUCTOR & DESTRUCTOR DOCUMENTATION . . . . .	31
2.1.3. MEMBER FUNCTION DOCUMENTATION . . . . .	31
2.2. BROKERINFOEX CLASS REFERENCE . . . . .	33
<b>3. DATA MANAGEMENT</b>	<b>33</b>
3.1. REPLICAMANAGER INTERFACE REFERENCE . . . . .	33
3.1.1. DETAILED DESCRIPTION . . . . .	33
3.2. EDGLOCALREPLICACATALOG INTERFACE REFERENCE . . . . .	33
3.2.1. DETAILED DESCRIPTION . . . . .	33
3.2.2. MEMBER FUNCTION DOCUMENTATION . . . . .	34
3.3. EDGREPLICAMETADATACATALOG INTERFACE REFERENCE . . . . .	37
3.3.1. DETAILED DESCRIPTION . . . . .	37
3.3.2. MEMBER FUNCTION DOCUMENTATION . . . . .	37
3.4. EDGREPLICAOPTIMIZATION INTERFACE REFERENCE . . . . .	39
3.4.1. DETAILED DESCRIPTION . . . . .	39
3.4.2. MEMBER FUNCTION DOCUMENTATION . . . . .	39
<b>4. R-GMA</b>	<b>40</b>
4.1. APIBASE CLASS REFERENCE . . . . .	40
4.1.1. DETAILED DESCRIPTION . . . . .	40
4.1.2. CONSTRUCTOR & DESTRUCTOR DOCUMENTATION . . . . .	41
4.1.3. MEMBER FUNCTION DOCUMENTATION . . . . .	41
4.1.4. MEMBER DATA DOCUMENTATION . . . . .	45
4.2. ARCHIVER CLASS REFERENCE . . . . .	45
4.2.1. DETAILED DESCRIPTION . . . . .	46
4.2.2. CONSTRUCTOR & DESTRUCTOR DOCUMENTATION . . . . .	46
4.2.3. MEMBER FUNCTION DOCUMENTATION . . . . .	47
4.3. CANONICALPRODUCER CLASS REFERENCE . . . . .	49
4.3.1. DETAILED DESCRIPTION . . . . .	50
4.3.2. CONSTRUCTOR & DESTRUCTOR DOCUMENTATION . . . . .	50
4.3.3. MEMBER DATA DOCUMENTATION . . . . .	51
4.4. CIRCULARBUFFERPRODUCER CLASS REFERENCE . . . . .	51
4.4.1. DETAILED DESCRIPTION . . . . .	51



---

4.4.2.	CONSTRUCTOR & DESTRUCTOR DOCUMENTATION . . . . .	52
4.4.3.	MEMBER FUNCTION DOCUMENTATION . . . . .	52
4.4.4.	MEMBER DATA DOCUMENTATION . . . . .	53
4.5.	CLEANABLE CLASS REFERENCE . . . . .	53
4.5.1.	DETAILED DESCRIPTION . . . . .	53
4.5.2.	CONSTRUCTOR & DESTRUCTOR DOCUMENTATION . . . . .	53
4.5.3.	MEMBER FUNCTION DOCUMENTATION . . . . .	54
4.6.	CONSUMER CLASS REFERENCE . . . . .	54
4.6.1.	DETAILED DESCRIPTION . . . . .	55
4.6.2.	CONSTRUCTOR & DESTRUCTOR DOCUMENTATION . . . . .	56
4.6.3.	MEMBER FUNCTION DOCUMENTATION . . . . .	57
4.6.4.	MEMBER DATA DOCUMENTATION . . . . .	62
4.7.	DATABASEPRODUCER CLASS REFERENCE . . . . .	62
4.7.1.	DETAILED DESCRIPTION . . . . .	63
4.7.2.	CONSTRUCTOR & DESTRUCTOR DOCUMENTATION . . . . .	63
4.7.3.	MEMBER FUNCTION DOCUMENTATION . . . . .	63
4.7.4.	MEMBER DATA DOCUMENTATION . . . . .	64
4.8.	DECLARABLE CLASS REFERENCE . . . . .	65
4.8.1.	DETAILED DESCRIPTION . . . . .	65
4.8.2.	CONSTRUCTOR & DESTRUCTOR DOCUMENTATION . . . . .	65
4.8.3.	MEMBER FUNCTION DOCUMENTATION . . . . .	65
4.9.	FIXEDPOINT CLASS REFERENCE . . . . .	66
4.9.1.	DETAILED DESCRIPTION . . . . .	67
4.9.2.	CONSTRUCTOR & DESTRUCTOR DOCUMENTATION . . . . .	67
4.9.3.	MEMBER FUNCTION DOCUMENTATION . . . . .	68
4.10.	INSERTABLE CLASS REFERENCE . . . . .	70
4.10.1.	CONSTRUCTOR & DESTRUCTOR DOCUMENTATION . . . . .	70
4.10.2.	MEMBER FUNCTION DOCUMENTATION . . . . .	71
4.11.	LATESTPRODUCER CLASS REFERENCE . . . . .	72
4.11.1.	DETAILED DESCRIPTION . . . . .	72
4.11.2.	CONSTRUCTOR & DESTRUCTOR DOCUMENTATION . . . . .	72
4.12.	NETLOGGERFACTORY CLASS REFERENCE . . . . .	72
4.13.	PRODUCERCONNECTION CLASS REFERENCE . . . . .	72
4.13.1.	DETAILED DESCRIPTION . . . . .	72
4.13.2.	CONSTRUCTOR & DESTRUCTOR DOCUMENTATION . . . . .	73
4.13.3.	MEMBER FUNCTION DOCUMENTATION . . . . .	73



---

4.14. PROPERTYGETTER CLASS REFERENCE . . . . .	73
4.14.1. DETAILED DESCRIPTION . . . . .	73
4.14.2. CONSTRUCTOR & DESTRUCTOR DOCUMENTATION . . . . .	73
4.14.3. MEMBER FUNCTION DOCUMENTATION . . . . .	73
4.15. REGISTRY CLASS REFERENCE . . . . .	73
4.15.1. DETAILED DESCRIPTION . . . . .	74
4.15.2. CONSTRUCTOR & DESTRUCTOR DOCUMENTATION . . . . .	74
4.15.3. MEMBER FUNCTION DOCUMENTATION . . . . .	74
4.16. REPLICACONFIGREADER CLASS REFERENCE . . . . .	75
4.16.1. DETAILED DESCRIPTION . . . . .	76
4.16.2. CONSTRUCTOR & DESTRUCTOR DOCUMENTATION . . . . .	76
4.16.3. MEMBER FUNCTION DOCUMENTATION . . . . .	76
4.16.4. MEMBER DATA DOCUMENTATION . . . . .	77
4.17. RESILIENTSTREAMPRODUCER CLASS REFERENCE . . . . .	77
4.17.1. DETAILED DESCRIPTION . . . . .	77
4.17.2. CONSTRUCTOR & DESTRUCTOR DOCUMENTATION . . . . .	77
4.17.3. MEMBER FUNCTION DOCUMENTATION . . . . .	78
4.18. RESULTSET CLASS REFERENCE . . . . .	78
4.18.1. DETAILED DESCRIPTION . . . . .	79
4.18.2. MEMBER FUNCTION DOCUMENTATION . . . . .	79
4.19. RESULTSETMETADATA CLASS REFERENCE . . . . .	80
4.19.1. DETAILED DESCRIPTION . . . . .	80
4.19.2. MEMBER FUNCTION DOCUMENTATION . . . . .	80
4.20. RGMAERRORHANDLER CLASS REFERENCE . . . . .	80
4.20.1. MEMBER FUNCTION DOCUMENTATION . . . . .	81
4.21. RGMAEXCEPTION CLASS REFERENCE . . . . .	81
4.21.1. CONSTRUCTOR & DESTRUCTOR DOCUMENTATION . . . . .	81
4.21.2. MEMBER FUNCTION DOCUMENTATION . . . . .	82
4.22. SCHEMA CLASS REFERENCE . . . . .	82
4.22.1. DETAILED DESCRIPTION . . . . .	83
4.22.2. CONSTRUCTOR & DESTRUCTOR DOCUMENTATION . . . . .	83
4.22.3. MEMBER FUNCTION DOCUMENTATION . . . . .	83
4.22.4. MEMBER DATA DOCUMENTATION . . . . .	86
4.23. SERVLETCONNECTION CLASS REFERENCE . . . . .	86
4.23.1. DETAILED DESCRIPTION . . . . .	87
4.23.2. CONSTRUCTOR & DESTRUCTOR DOCUMENTATION . . . . .	87



---

4.23.3. MEMBER FUNCTION DOCUMENTATION . . . . .	87
4.23.4. MEMBER DATA DOCUMENTATION . . . . .	90
4.24. STREAMPRODUCER CLASS REFERENCE . . . . .	90
4.24.1. DETAILED DESCRIPTION . . . . .	91
4.24.2. CONSTRUCTOR & DESTRUCTOR DOCUMENTATION . . . . .	91
4.24.3. MEMBER FUNCTION DOCUMENTATION . . . . .	91
4.25. STREAMREQUEST CLASS REFERENCE . . . . .	93
4.25.1. MEMBER FUNCTION DOCUMENTATION . . . . .	93
4.26. TIME CLASS REFERENCE . . . . .	93
4.26.1. DETAILED DESCRIPTION . . . . .	94
4.26.2. CONSTRUCTOR & DESTRUCTOR DOCUMENTATION . . . . .	94
4.26.3. MEMBER FUNCTION DOCUMENTATION . . . . .	95
4.27. TIMECONVERTER CLASS REFERENCE . . . . .	102
4.27.1. DETAILED DESCRIPTION . . . . .	102
4.27.2. CONSTRUCTOR & DESTRUCTOR DOCUMENTATION . . . . .	102
4.27.3. MEMBER FUNCTION DOCUMENTATION . . . . .	103
4.27.4. MEMBER DATA DOCUMENTATION . . . . .	104
4.28. TIMEDETAILS CLASS REFERENCE . . . . .	104
4.28.1. DETAILED DESCRIPTION . . . . .	104
4.28.2. MEMBER FUNCTION DOCUMENTATION . . . . .	104
4.29. TIMEINTERVAL CLASS REFERENCE . . . . .	105
4.29.1. DETAILED DESCRIPTION . . . . .	105
4.29.2. CONSTRUCTOR & DESTRUCTOR DOCUMENTATION . . . . .	105
4.29.3. MEMBER FUNCTION DOCUMENTATION . . . . .	105
4.30. XMLCONVERTER CLASS REFERENCE . . . . .	106
4.30.1. DETAILED DESCRIPTION . . . . .	106
4.30.2. CONSTRUCTOR & DESTRUCTOR DOCUMENTATION . . . . .	106
4.30.3. MEMBER FUNCTION DOCUMENTATION . . . . .	106
<b>5. FABRIC</b>	<b>107</b>
5.1. METRIC STRUCT REFERENCE . . . . .	107
5.1.1. DETAILED DESCRIPTION . . . . .	107
5.1.2. MEMBER DATA DOCUMENTATION . . . . .	107
5.2. METRICBASE CLASS REFERENCE . . . . .	107
5.2.1. DETAILED DESCRIPTION . . . . .	108
5.2.2. CONSTRUCTOR & DESTRUCTOR DOCUMENTATION . . . . .	108



---

5.2.3.	MEMBER FUNCTION DOCUMENTATION . . . . .	108
5.2.4.	MEMBER DATA DOCUMENTATION . . . . .	109
5.3.	METRICPARAMS CLASS REFERENCE . . . . .	110
5.3.1.	DETAILED DESCRIPTION . . . . .	110
5.3.2.	CONSTRUCTOR & DESTRUCTOR DOCUMENTATION . . . . .	110
5.3.3.	MEMBER FUNCTION DOCUMENTATION . . . . .	110
5.4.	METRICVALUE STRUCT REFERENCE . . . . .	111
5.4.1.	DETAILED DESCRIPTION . . . . .	111
5.4.2.	MEMBER DATA DOCUMENTATION . . . . .	111
5.5.	NODE STRUCT REFERENCE . . . . .	111
5.5.1.	DETAILED DESCRIPTION . . . . .	111
5.5.2.	MEMBER DATA DOCUMENTATION . . . . .	112
5.6.	SAMPLE STRUCT REFERENCE . . . . .	112
5.6.1.	DETAILED DESCRIPTION . . . . .	112
5.6.2.	MEMBER DATA DOCUMENTATION . . . . .	112
<b>6.</b>	<b>STORAGE ELEMENT</b>	<b>113</b>
6.1.	FILECONTROL INTERFACE REFERENCE . . . . .	113
6.1.1.	DETAILED DESCRIPTION . . . . .	113
6.2.	FILEHANDLE INTERFACE REFERENCE . . . . .	115
6.2.1.	DETAILED DESCRIPTION . . . . .	115
6.3.	FILEINFO CLASS REFERENCE . . . . .	115
6.3.1.	DETAILED DESCRIPTION . . . . .	115
6.3.2.	CONSTRUCTOR & DESTRUCTOR DOCUMENTATION . . . . .	115
6.3.3.	MEMBER FUNCTION DOCUMENTATION . . . . .	116
6.4.	MODE CLASS REFERENCE . . . . .	117
6.4.1.	DETAILED DESCRIPTION . . . . .	117
6.4.2.	CONSTRUCTOR & DESTRUCTOR DOCUMENTATION . . . . .	117
6.4.3.	MEMBER FUNCTION DOCUMENTATION . . . . .	117
6.4.4.	MEMBER DATA DOCUMENTATION . . . . .	118
6.5.	STORAGEELEMENT INTERFACE REFERENCE . . . . .	118
6.5.1.	DETAILED DESCRIPTION . . . . .	118
6.6.	TRANSFERFILEHANDLE INTERFACE REFERENCE . . . . .	118
6.6.1.	DETAILED DESCRIPTION . . . . .	118
<b>7.</b>	<b>NETWORK</b>	<b>118</b>



---

7.1.	NETWORKCOST CLASS REFERENCE . . . . .	118
7.1.1.	DETAILED DESCRIPTION . . . . .	119
7.1.2.	CONSTRUCTOR & DESTRUCTOR DOCUMENTATION . . . . .	119
7.1.3.	MEMBER FUNCTION DOCUMENTATION . . . . .	119
7.2.	NETWORKCOSTCONTEXTIF INTERFACE REFERENCE . . . . .	119
7.2.1.	DETAILED DESCRIPTION . . . . .	120
7.2.2.	MEMBER FUNCTION DOCUMENTATION . . . . .	120
7.2.3.	MEMBER DATA DOCUMENTATION . . . . .	120
<b>8.</b>	<b>JAVA SECURITY</b>	<b>120</b>
8.1.	SECURITYINFO INTERFACE REFERENCE . . . . .	120
8.1.1.	DETAILED DESCRIPTION . . . . .	121
8.1.2.	MEMBER FUNCTION DOCUMENTATION . . . . .	121
8.2.	SECURITYINFOCONTAINER CLASS REFERENCE . . . . .	122
8.2.1.	DETAILED DESCRIPTION . . . . .	122
8.2.2.	MEMBER FUNCTION DOCUMENTATION . . . . .	123
8.3.	VOMSEXTENSION CLASS REFERENCE . . . . .	123
8.3.1.	DETAILED DESCRIPTION . . . . .	123
8.3.2.	CONSTRUCTOR & DESTRUCTOR DOCUMENTATION . . . . .	124
8.3.3.	MEMBER FUNCTION DOCUMENTATION . . . . .	124
8.3.4.	MEMBER DATA DOCUMENTATION . . . . .	125
<b>9.</b>	<b>VOMS</b>	<b>125</b>
9.1.	VOMS STRUCT REFERENCE . . . . .	125
9.1.1.	MEMBER DATA DOCUMENTATION . . . . .	125
9.2.	DATA STRUCT REFERENCE . . . . .	126
9.2.1.	MEMBER DATA DOCUMENTATION . . . . .	126
9.3.	EXT_VOMS STRUCT REFERENCE . . . . .	126
9.3.1.	MEMBER DATA DOCUMENTATION . . . . .	127
9.4.	VOMS STRUCT REFERENCE . . . . .	127
9.4.1.	MEMBER DATA DOCUMENTATION . . . . .	127
9.5.	VOMSDATA STRUCT REFERENCE . . . . .	128
9.5.1.	MEMBER DATA DOCUMENTATION . . . . .	128
9.6.	VOMSADMIN INTERFACE REFERENCE . . . . .	128
9.6.1.	DETAILED DESCRIPTION . . . . .	128
9.7.	VOMSCOMPATIBILITY INTERFACE REFERENCE . . . . .	129



---

9.7.1. DETAILED DESCRIPTION . . . . .	129
9.8. VOMSCORE INTERFACE REFERENCE . . . . .	129
9.8.1. DETAILED DESCRIPTION . . . . .	129
9.9. VOMSHISTORY INTERFACE REFERENCE . . . . .	129
9.9.1. DETAILED DESCRIPTION . . . . .	129
9.10. VOMSREQUEST INTERFACE REFERENCE . . . . .	129
9.10.1. DETAILED DESCRIPTION . . . . .	129
9.11. ACLENTY CLASS REFERENCE . . . . .	130
9.11.1. DETAILED DESCRIPTION . . . . .	130
9.11.2. CONSTRUCTOR & DESTRUCTOR DOCUMENTATION . . . . .	130
9.11.3. MEMBER FUNCTION DOCUMENTATION . . . . .	130
9.12. QUALIFIEDROLE CLASS REFERENCE . . . . .	131
9.12.1. DETAILED DESCRIPTION . . . . .	131
9.12.2. MEMBER FUNCTION DOCUMENTATION . . . . .	131
9.13. CREATEUSERREQUEST CLASS REFERENCE . . . . .	132
9.13.1. DETAILED DESCRIPTION . . . . .	132
9.13.2. MEMBER FUNCTION DOCUMENTATION . . . . .	132
9.14. REQUEST CLASS REFERENCE . . . . .	132
9.14.1. DETAILED DESCRIPTION . . . . .	133
9.14.2. MEMBER FUNCTION DOCUMENTATION . . . . .	133
9.15. USER CLASS REFERENCE . . . . .	134
9.15.1. DETAILED DESCRIPTION . . . . .	134
9.15.2. CONSTRUCTOR & DESTRUCTOR DOCUMENTATION . . . . .	134
9.15.3. MEMBER FUNCTION DOCUMENTATION . . . . .	134
<b>10. LCAS</b>	<b>135</b>
10.1. LCAS_CRED_ID_S STRUCT REFERENCE . . . . .	135
10.1.1. DETAILED DESCRIPTION . . . . .	135
10.1.2. MEMBER DATA DOCUMENTATION . . . . .	135
<b>11. LCMAPS</b>	<b>136</b>
11.1. CRED_DATA_S STRUCT REFERENCE . . . . .	136
11.1.1. MEMBER DATA DOCUMENTATION . . . . .	136
11.2. LCMAPS_ARGUMENT_S STRUCT REFERENCE . . . . .	136
11.2.1. MEMBER DATA DOCUMENTATION . . . . .	136
11.3. LCMAPS_CRED_ID_S STRUCT REFERENCE . . . . .	137



---

11.3.1. DETAILED DESCRIPTION . . . . .	137
11.3.2. MEMBER DATA DOCUMENTATION . . . . .	137
<b>12. SLASHGRID</b>	<b>137</b>
12.1. DIRENTRY STRUCT REFERENCE . . . . .	137
12.1.1. MEMBER DATA DOCUMENTATION . . . . .	137
12.2. FSPLUGIN STRUCT REFERENCE . . . . .	138
12.2.1. MEMBER DATA DOCUMENTATION . . . . .	139
12.3. PLUGPARAM STRUCT REFERENCE . . . . .	140
12.3.1. MEMBER DATA DOCUMENTATION . . . . .	140
12.4. UIDENTRY STRUCT REFERENCE . . . . .	141
12.4.1. MEMBER DATA DOCUMENTATION . . . . .	141
<b>13. GRID ACCESS CONTROL</b>	<b>141</b>
13.1. _GACLA CL STRUCT REFERENCE . . . . .	141
13.1.1. MEMBER DATA DOCUMENTATION . . . . .	141
13.2. _GACL CRED STRUCT REFERENCE . . . . .	142
13.2.1. MEMBER DATA DOCUMENTATION . . . . .	142
13.3. _GACLE NTRY STRUCT REFERENCE . . . . .	142
13.3.1. MEMBER DATA DOCUMENTATION . . . . .	142
13.4. _GACL NAMEVALUE STRUCT REFERENCE . . . . .	142
13.4.1. MEMBER DATA DOCUMENTATION . . . . .	142
13.5. _GACL USER STRUCT REFERENCE . . . . .	143
13.5.1. MEMBER DATA DOCUMENTATION . . . . .	143
<b>14. GRIDSITE</b>	<b>143</b>
14.1. BODYPTR STRUCT REFERENCE . . . . .	143
14.1.1. MEMBER DATA DOCUMENTATION . . . . .	143
14.2. LISTCHAR STRUCT REFERENCE . . . . .	143
14.2.1. MEMBER DATA DOCUMENTATION . . . . .	143

## 1. WORKLOAD MANAGEMENT

### 1.1. EVENT CLASS REFERENCE

#### PUBLIC METHODS

- String `name ()`
- String `toString ()`

#### STATIC PUBLIC ATTRIBUTES

- final String `code []`
- final String `attrNames []`
- final int `CLASSAD = 0`
- final int `DESCR = 1`
- final int `DEST_HOST = 2`
- final int `DEST_ID = 3`
- final int `DEST_INSTANCE = 4`
- final int `DEST_JOBID = 5`
- final int `DESTINATION = 6`
- final int `EXIT_CODE = 7`
- final int `FROM = 8`
- final int `FROM_HOST = 9`
- final int `FROM_INSTANCE = 10`
- final int `HELPER_NAME = 11`
- final int `HELPER_PARAMS = 12`
- final int `HOST = 13`
- final int `JDL = 14`
- final int `JOB = 15`
- final int `JOBID = 16`
- final int `LEVEL = 17`
- final int `LOCAL_JOBID = 18`
- final int `NAME = 19`
- final int `NODE = 20`
- final int `NS = 21`
- final int `NSUBJOBS = 22`
- final int `PARENT = 23`
- final int `PRIORITY = 24`
- final int `QUEUE = 25`
- final int `REASON = 26`
- final int `RESULT = 27`
- final int `RETVAL = 28`
- final int `SEED = 29`
- final int `SEQCODE = 30`
- final int `SOURCE = 31`
- final int `SRC_INSTANCE = 32`

- final int SRC\_ROLE = 33
- final int STATUS\_CODE = 34
- final int SVC\_HOST = 35
- final int SVC\_NAME = 36
- final int SVC\_PORT = 37
- final int TAG = 38
- final int TIMESTAMP = 39
- final int USER = 40
- final int VALUE = 41
- final int EVENT = 42

### 1.1.1. DETAILED DESCRIPTION

LB Event Class Wrapped

**Version:**

0.1

**Author:**

Alessandro Maraschini <alessandro.maraschini@datamat.it>

### 1.1.2. MEMBER FUNCTION DOCUMENTATION

#### 1.1.2.1 String Event::name ()

#### 1.1.2.2 String Event::toString ()

### 1.1.3. MEMBER DATA DOCUMENTATION

#### 1.1.3.1 final String Event::attrNames[] [static]

#### 1.1.3.2 final int Event::CLASSAD = 0 [static]

Event attribute - Chkpt: checkpoint value

#### 1.1.3.3 final String Event::code[] [static]

**Initial value:**

```
{  
    "Undefined",  
    "Transfer",  
    "Accepted",  
    "Refused",  
    "EnQueued",  
    "DeQueued",  
    "HelperCall",  
    "HelperReturn",  
    "Running",  
}
```

```
"Resubmission",  
"Done",  
"Cancel",  
"Abort",  
"Clear",  
"Purge",  
"Match",  
"Pending",  
"RegJob",  
"Chkpt",  
"Listener",  
"CurDescr",  
"UserTag",  
}
```

#### 1.1.3.4 final int Event::DESCR = 1 [static]

Event attribute - CurDescr: description of current job transformation (output of helper)

#### 1.1.3.5 final int Event::DEST\_HOST = 2 [static]

Event attribute - Transfer: destination hostname

#### 1.1.3.6 final int Event::DEST\_ID = 3 [static]

Event attribute - Match: Id of the destination CE/queue

#### 1.1.3.7 final int Event::DEST\_INSTANCE = 4 [static]

Event attribute - Transfer: destination instance

#### 1.1.3.8 final int Event::DEST\_JOBID = 5 [static]

Event attribute - Transfer: destination internal jobid

#### 1.1.3.9 final int Event::DESTINATION = 6 [static]

Event attribute - Transfer: destination where the job is being transferred to

#### 1.1.3.10 final int Event::EVENT = 42 [static]

#### 1.1.3.11 final int Event::EXIT\_CODE = 7 [static]

Event attribute - Done: process exit code

#### 1.1.3.12 final int Event::FROM = 8 [static]

Event attribute - Accepted: where was the job received from \* Refused: where was the job received from

#### 1.1.3.13 final int Event::FROM\_HOST = 9 [static]

Event attribute - Accepted: sending component hostname \* Refused: sending component hostname

#### 1.1.3.14 final int Event::FROM\_INSTANCE = 10 [static]

Event attribute - Accepted: sending component instance \* Refused: sending component instance

**1.1.3..15 final int Event::HELPER\_NAME = 11 [static]**

Event attribute - HelperCall: name of the called component \* HelperReturn: name of the called component

**1.1.3..16 final int Event::HELPER\_PARAMS = 12 [static]**

Event attribute - HelperCall: parameters of the call

**1.1.3..17 final int Event::HOST = 13 [static]**

Event attribute - common: hostname of the machine where the event was generated

**1.1.3..18 final int Event::JDL = 14 [static]**

Event attribute - RegJob: job description

**1.1.3..19 final int Event::JOB = 15 [static]**

Event attribute - EnQueued: job description in receiver language \* Transfer: job description in receiver language

**1.1.3..20 final int Event::JOBID = 16 [static]**

Event attribute - common: DataGrid job id of the source job

**1.1.3..21 final int Event::LEVEL = 17 [static]**

Event attribute - common: logging level (system, debug, ...)

**1.1.3..22 final int Event::LOCAL\_JOBID = 18 [static]**

Event attribute - Accepted: new jobId (Condor, Globus ...) assigned by the receiving component \* DeQueued: new jobId assigned by the receiving component

**1.1.3..23 final int Event::NAME = 19 [static]**

Event attribute - UserTag: tag name

**1.1.3..24 final int Event::NODE = 20 [static]**

Event attribute - Running: worker node where the executable is run

**1.1.3..25 final int Event::NS = 21 [static]**

Event attribute - RegJob: NetworkServer handling the job

**1.1.3..26 final int Event::NSUBJOBS = 22 [static]**

Event attribute - RegJob: number of subjobs

**1.1.3..27 final int Event::PARENT = 23 [static]**

Event attribute - RegJob: jobid of parent job

**1.1.3..28 final int Event::PRIORITY = 24 [static]**

Event attribute - common: message priority (yet 0 for asynchronous and 1 for synchronous transfers)

**1.1.3..29 final int Event::QUEUE = 25 [static]**

Event attribute - DeQueued: queue name \* EnQueued: destination queue

**1.1.3..30 final int Event::REASON = 26** [static]

Event attribute - Abort: reason of abort \* Cancel: detailed description \* Clear: why the job was cleared \* Done: reason for the change \* EnQueued: detailed description of transfer, especially reason of failure \* Pending: why matching CE cannot be found \* Refused: reason of refusal \* Resubmission: reason for the decision \* Transfer: detailed description of transfer, especially reason of failure

**1.1.3..31 final int Event::RESULT = 27** [static]

Event attribute - EnQueued: result of the attempt \* Resubmission: result code \* Transfer: result of the attempt

**1.1.3..32 final int Event::RETVL = 28** [static]

Event attribute - HelperReturn: returned data

**1.1.3..33 final int Event::SEED = 29** [static]

Event attribute - RegJob: seed for subjob id generation

**1.1.3..34 final int Event::SEQCODE = 30** [static]

Event attribute - common: sequence code assigned to the event

**1.1.3..35 final int Event::SOURCE = 31** [static]

Event attribute - common: source (WMS component) which generated this event

**1.1.3..36 final int Event::SRC\_INSTANCE = 32** [static]

Event attribute - common: instance of WMS component (e.g. service communication endpoint)

**1.1.3..37 final int Event::SRC\_ROLE = 33** [static]

Event attribute - HelperCall: whether the logging component is called or calling one \* HelperReturn: whether the logging component is called or calling one

**1.1.3..38 final int Event::STATUS\_CODE = 34** [static]

Event attribute - Cancel: classification of the cancel \* Done: way of termination

**1.1.3..39 final int Event::SVC\_HOST = 35** [static]

Event attribute - Listener: hostname

**1.1.3..40 final int Event::SVC\_NAME = 36** [static]

Event attribute - Listener: port instance name

**1.1.3..41 final int Event::SVC\_PORT = 37** [static]

Event attribute - Listener: port number

**1.1.3..42 final int Event::TAG = 38** [static]

Event attribute - Chkpt: checkpoint tag \* Resubmission: value of the attribute on which the decision is based

**1.1.3..43 final int Event::TIMESTAMP = 39** [static]

Event attribute - common: timestamp of event generation

**1.1.3..44 final int Event::USER = 40** [static]

Event attribute - common: identity (cert. subj.) of the generator

**1.1.3..45 final int Event::VALUE = 41** [static]

Event attribute - UserTag: tag value

**1.2. JOBAD CLASS REFERENCE****INSERTION METHODS**

- void `setAttribute` (String attrName, Expr attrValue) throws Exception
- void `setAttributeAd` (String attrName, String attrValue) throws Exception
- void `setAttributeExpr` (String attrName, String attrValue) throws Exception
- void `setAttribute` (String attrName, int attrValue) throws Exception
- void `setAttribute` (String attrName, String attrValue) throws Exception
- void `setAttribute` (String attrName, double attrValue) throws Exception
- void `setAttribute` (String attrName, boolean attrValue) throws Exception
- void `addAttribute` (String attrName, Ad attrValue) throws Exception
- void `addAttribute` (String attrName, int attrValue) throws Exception
- void `addAttribute` (String attrName, double attrValue) throws Exception
- void `addAttribute` (String attrName, boolean attrValue) throws Exception
- void `addAttribute` (String attrName, String attrValue) throws Exception

**MISCELLANEOUS METHODS**

- void `delAttribute` (String attrName) throws NoSuchFieldException
- void `clear` ()
- void `check` () throws Exception
- void `checkAll` () throws Exception
- void `checkAll` (String attributes[]) throws Exception
- void `check` (String attributes[]) throws Exception

**PUBLIC METHODS**

- JobAd `copy` ()
- RecordExpr `copyAd` ()
- void `setAttributeDefault` (int attrName, String attrValue)

**Constructors**

- JobAd ()
- JobAd (String ad) throws Exception

**String and Stream Constructor/Destructor**

- boolean `hasAttribute` (String attrName, String attrValue)
- String `toLines` ()
- String `toString` (boolean multiLines, boolean multiLists)
- String `toSubmissionString` () throws Exception
- String `toString` ()
- void `ToFile` (String filePath) throws Exception
- void `setLocalAccess` (boolean lookInto)

**Get Methods**

- String `getAttributeExpr` (String attrName) throws Exception
- JobAd `getJobAdValue` (String attrName) throws Exception
- String `getString` (String attrName) throws Exception
- int `getInt` (String attrName) throws Exception
- boolean `getBoolean` (String attrName) throws Exception
- double `getDouble` (String attrName) throws Exception

## STATIC PUBLIC METHODS

- JobAd `copy` (RecordExpr re) throws Exception

## STATIC PUBLIC ATTRIBUTES

- final int `JOBTYPE_MPICH_REQ_RTE` = 0
- final int `JOBTYPE_MPICH_REQ_CPU` = 1
- final int `JOBTYPE_MPICH_RANK_FREE` = 2
- final int `REQ_DEFAULT` = 3
- final int `RANK_DEFAULT` = 4

### 1.2.1. DETAILED DESCRIPTION

Provides a representation of the job description in the JDL language and the functions for building and manipulating it. Basically the JDL is the Condor ClassAd language, so it is legitimate the direct use of the Condor API library for creating, modifying, deleting a job description. However the JobAd class extends the ClassAd class of the Condor ClassAd library additionally providing some helper methods that ease the construction of job descriptions being fully compliant to WP1 WMS specification.

#### Version:

0.1

#### Author:

Alessandro Maraschini <[alessandro.maraschini@datamat.it](mailto:alessandro.maraschini@datamat.it)>

### 1.2.2. CONSTRUCTOR & DESTRUCTOR DOCUMENTATION

#### 1.2.2.1 JobAd::JobAd ()

Instantiates an empty JobAd object

#### 1.2.2.2 JobAd::JobAd (String ad) throws Exception

### 1.2.3. MEMBER FUNCTION DOCUMENTATION

#### 1.2.3.1 void JobAd::addAttribute (String attrName, String attrValue) throws Exception

Allow adding a value to an already set attribute of the JobAd instance (i.e. it transforms it in a list attribute). if used on a non-set attribute the corresponding setAttribute method is automatically called.

#### Parameters:

*attrName* a String representing the attribute name

*attrValue* - The value of the attribute to be added

#### Exceptions:

*IllegalArgumentException* - The specified type is not allowed for the attribute value

*InvalidAttributeValueException* - The value has not the right format for the attribute

#### 1.2.3.2 void JobAd::addAttribute (String attrName, boolean attrValue) throws Exception

Allow adding a value to an already set attribute of the JobAd instance (i.e. it transforms it in a list attribute). if used on a non-set attribute the corresponding setAttribute method is automatically called.

#### Parameters:

*attrName* a String representing the attribute name

*attrValue* - The value of the attribute to be added

#### Exceptions:

*IllegalArgumentException* - The specified value is not allowed for the attribute

*InvalidAttributeValueException* - the value has not the right format for the attribute

### 1.2.3.3 void JobAd::addAttribute (String attrName, double attrValue) throws Exception

Allow adding a value to an already set attribute of the JobAd instance (i.e. it transforms it in a list attribute). if used on a non-set attribute the corresponding setAttribute method is automatically called.

**Parameters:**

*attrName* a String representing the attribute name

*attrValue* - The value of the attribute to be added

**Exceptions:**

*IllegalArgumentException* - The specified value is not allowed for the attribute

*InvalidAttributeValueException* - the has not the right format for the attribute

### 1.2.3.4 void JobAd::addAttribute (String attrName, int attrValue) throws Exception

Allow adding a value to an already set attribute of the JobAd instance (i.e. it transforms it in a list attribute). if used on a non-set attribute the corresponding setAttribute method is automatically called.

**Parameters:**

*attrName* a String representing the attribute name

*attrValue* - The value of the attribute to be added

**Exceptions:**

*IllegalArgumentException* - The specified value is not allowed for the attribute

*InvalidAttributeValueException* - A value has not the right type for an attribute

### 1.2.3.5 void JobAd::addAttribute (String attrName, Ad attrValue) throws Exception

Allow adding a value to an already set attribute of the JobAd instance (i.e. it transforms it in a list attribute). if used on a non-set attribute the corresponding setAttribute method is automatically called.

**Parameters:**

*attrName* a String representing the attribute name

*attrValue* - The value of the attribute to be added

**Exceptions:**

*IllegalArgumentException* - The specified value is not allowed for the attribute

*InvalidAttributeValueException* - A value has not the right type for an attribute

### 1.2.3.6 void JobAd::check (String attributes[]) throws Exception

Check the JobAd instance for both syntax and semantic errors

**Exceptions:**

*IllegalArgumentException* - A value is not of the right type/format/value for an attribute name in the JobAd

*NoSuchFieldException* - Unable to find a JobAd mandatory attribute/value

*JobAdException* - one or more values do not match with jobad semantic rule

### 1.2.3.7 void JobAd::check () throws Exception

### 1.2.3.8 void JobAd::checkAll (String attributes[]) throws Exception

### 1.2.3.9 void JobAd::checkAll () throws Exception

### 1.2.3.10 void JobAd::clear ()

Reset the JobAd Instance. All the previous existing attributes will be deleted

### 1.2.3..11 JobAd JobAd::copy ()

Copy all the attributes of the instance into a new JobAd

**Returns:**

a new JobAd with a copy of all the attributes contained in the current JobAd instance

### 1.2.3..12 JobAd JobAd::copy (RecordExpr *re*) throws Exception [static]

Copy all the attributes of the instance into a new JobAd

**Exceptions:**

*JobAdException* - all the error occurred while inserting the attributes

**Returns:**

a new JobAd with a copy of all the attributes contained in the passed RecordExpr instance

### 1.2.3..13 RecordExpr JobAd::copyAd ()

Copy the attributes of the classAd into a new Record Expression

### 1.2.3..14 void JobAd::delAttribute (String *attrName*) throws NoSuchFieldException

Delete an Attribute. It fails if the attribute doesn't exist

**Parameters:**

*attrName* The name of the attribute to be deleted

**Exceptions:**

*NoSuchFieldException* - The attribute has not been set yet

### 1.2.3..15 String JobAd::getAttributeExpr (String *attrName*) throws Exception

Retrieve the value of the specified Expression attribute as a string

**Parameters:**

*attrName* The name of the attribute name to be retrieved

**Returns:**

the String representng the value of the specified attribute

**Exceptions:**

*InvalidAttributeValueException* - Not an Expression value is allowed for the specified attribute

*NoSuchFieldException* - The attribute is not present in the JobAd

### 1.2.3..16 boolean JobAd::getBoolean (String *attrName*) throws Exception

Retrieve the value of the specified attribute, only if it is of non-list type

**Parameters:**

*attrName* The name of the attribute to be retrieved

**See also:**

getBooleanValue

**Returns:**

the value of the specified attribute as in the JobAd @InvalidAttributeValueException The specified attribute is of list type

**Exceptions:**

*IllegalArgumentException* - The type of retrieved value is not allowed for the specified attribute name

*NoSuchFieldException* - The requested attribute has not been set yet

### 1.2.3..17 double JobAd::getDouble (String attrName) throws Exception

Retrieve the value of the specified attribute, only if it is of non-list type

**Parameters:**

*attrName* The name of the attribute to be retrieved

**Returns:**

the value of the specified attribute as in the JobAd

**See also:**

getDoubleValue @InvalidAttributeValueException The specified attribute is of list type

**Exceptions:**

*IllegalArgumentException* - The type of retrieved value is not allowed for the specified attribute name

*NoSuchFieldException* - The requested attribute has not been set yet

### 1.2.3..18 int JobAd::getInt (String attrName) throws Exception

Retrieve the value of the specified attribute, only if it is of non-list type

**Parameters:**

*attrName* The name of the attribute to be retrieved

**See also:**

getIntValue

**Returns:**

the value of the specified attribute as in the JobAd @InvalidAttributeValueException The specified attribute is of list type

**Exceptions:**

*IllegalArgumentException* - The type of retrieved value is not allowed for the specified attribute name

*NoSuchFieldException* - The requested attribute has not been set yet

### 1.2.3..19 JobAd JobAd::getJobAdValue (String attrName) throws Exception

Retrieve the JobAd value of the specified attribute

**Parameters:**

*attrName* The name of the attribute name to be retrieved

**Returns:**

a Vector containing the values listed in the specified attribute , (1-size Vector if the attribute has a single value)

**Exceptions:**

*IllegalArgumentException* - The type of retrieved value is not allowed for the specified attribute name

*NoSuchFieldException* - The requested attribute has not been set yet

### 1.2.3..20 String JobAd::getString (String attrName) throws Exception

Retrieve the value of the specified attribute, only if it is of non-list type

**Parameters:**

*attrName* The name of the attribute to be retrieved

**See also:**

getStringValue

**Returns:**

the value of the specified attribute as in the JobAd @InvalidAttributeValueException The specified attribute is of list type

**Exceptions:**

*IllegalArgumentException* - The type of retrieved value is not allowed for the specified attribute name

*NoSuchFieldException* - The requested attribute has not been set yet

### 1.2.3..21 boolean JobAd::hasAttribute (String attrName, String attrValue)

### 1.2.3..22 void JobAd::setAttribute (String attrName, boolean attrValue) throws Exception

Add The specified String Attribute to the jdl instance

**Parameters:**

*attrName* - The Name of the attribute to be added

*attrValue* - The value of the attribute to be added

**Exceptions:**

*IllegalArgumentException* - The attribute attrName had been already set

*InvalidAttributeValueException* - A value has not the right type for an attribute

### 1.2.3..23 void JobAd::setAttribute (String attrName, double attrValue) throws Exception

Add The specified String Attribute to the jdl instance

**Parameters:**

*attrName* - The Name of the attribute to be added

*attrValue* - The value of the attribute to be added

**Exceptions:**

*IllegalArgumentException* - The attribute attrName had been

*InvalidAttributeValueException* - the value is out of limits for the specified attribute

### 1.2.3..24 void JobAd::setAttribute (String attrName, String attrValue) throws Exception

Add The specified String Attribute to the jdl instance

**Parameters:**

*attrName* - The Name of the attribute to be added

*attrValue* - The value of the attribute to be added

**Exceptions:**

*IllegalArgumentException* - The type of value is not allowed for the specified attribute name

*InvalidAttributeValueException* - the value has not the right format for the specified attribute

### 1.2.3..25 void JobAd::setAttribute (String attrName, int attrValue) throws Exception

Add The specified Integer Attribute to the jdl instance

**Parameters:**

*attrName* - The Name of the attribute to be added

*attrValue* - The value of the attribute to be added

**Exceptions:**

*IllegalArgumentException* - The type of value is not allowed for the specified attribute name

*InvalidAttributeValueException* - The specified value is out of limits for the specified attribute

### 1.2.3..26 void JobAd::setAttribute (String attrName, Expr attrValue) throws Exception

Add The specified Expression Attribute to the jdl instance

**Parameters:**

*attrName* - The Name of the attribute to be added

*attrValue* - The String expression of the attribute to be added

**Exceptions:**

*IllegalArgumentException* - The type of value is not allowed for the specified attribute name

*InvalidAttributeValueException* - The value has not the right format for the specified attribute

**1.2.3..27 void JobAd::setAttributeAd (String attrName, String attrValue) throws Exception****1.2.3..28 void JobAd::setAttributeDefault (int attrName, String attrValue)**

Default attribute value overriding Depending on othe schema to be used, the default value of some attributes might change

**Parameters:**

*attrName* the value you want to change

*attrValue* the new value

**See also:**

[JOBTYPE\\_MPICH\\_REQ RTE](#) , [JOBTYPE\\_MPICH\\_REQ\\_CPU](#) , [JOBTYPE\\_MPICH\\_RANK\\_FREE](#) , [REQ\\_DEFAULT](#) , [RANK\\_DEFAULT](#)

**1.2.3..29 void JobAd::setAttributeExpr (String attrName, String attrValue) throws Exception**

Add The specified Expression Attribute to the jdl instance

**Parameters:**

*attrName* - The Name of the attribute to be added

*attrValue* - The value of the attribute to be added

**Exceptions:**

*IllegalArgumentException* - The type of value is not allowed for the specified attribute name

*InvalidAttributeValueException* - the value has not the right format for the specified attribute

**1.2.3..30 void JobAd::setLocalAccess (boolean lookInto)**

If JobAd is used inside an applet, it is impossible to look into the local hard-disk by default this parameter is set to TRUE

**Parameters:**

*lookInto* allow all the check methods to access to the local hard disk (true) or skip the check (false)

**1.2.3..31 void JobAd::toFile (String filePath) throws Exception**

Print the JobAd instance into the specified file, with its multi-lines representation

**Parameters:**

*filePath* where to write the JobAd

**See also:**

[toLines\(\)](#)

**Exceptions:**

*IOException* - if unable to write the specified file

**1.2.3..32 String JobAd::toLines ()**

Convert the JobAd Instance into its String representation, one line per attribute, multi line for listed attribute active This method is the same as toString ( true , true )

**See also:**

[toSubmissionString\(\)](#) , [toString\(\)](#)

**1.2.3..33 String JobAd::toString ()**

Convert the JobAd Instance into a single-line String representation

**See also:**

[toSubmissionString\(\)](#) , [toLines\(\)](#)

### 1.2.3..34 String JobAd::toString (boolean *multiLines*, boolean *multiLists*)

Convert the JobAd Instance into its String representation, one line per attribute, multi line for listed attribute active

**Parameters:**

- multiLines* one-attribute per line representation enabling
- multiLists* list attributes splitted into multi line representation enabling

**See also:**

[toSubmissionString\(\)](#) , [toString\(\)](#)

### 1.2.3..35 String JobAd::toSubmissionString () throws Exception

Perform a check over the JobAd instance and if possible convert it into its String representation as it would be ready for a submission

**Exceptions:**

- JobAdException* - one or more values do not match with jobad semantic rule
- IllegalArgumentException* - A value is not of the right type/format/value for an attribute name in the JobAd
- NoSuchFieldException* - Unable to find a JobAd mandatory attribute/value

**See also:**

[toString\(\)](#) , [toLines\(\)](#)

## 1.2.4. MEMBER DATA DOCUMENTATION

1.2.4..1 final int JobAd::JOBTYPE\_MPICH\_RANK\_FREE = 2 [static]

1.2.4..2 final int JobAd::JOBTYPE\_MPICH\_REQ\_CPU = 1 [static]

1.2.4..3 final int JobAd::JOBTYPE\_MPICH\_REQ\_RTE = 0 [static]

1.2.4..4 final int JobAd::RANK\_DEFAULT = 4 [static]

1.2.4..5 final int JobAd::REQ\_DEFAULT = 3 [static]

## 1.3. JOB COLLECTION CLASS REFERENCE

### PUBLIC METHODS

- Vector [getOutput](#) (String dirPath) throws Exception
- void [setMaxThreadNumber](#) (int maxThread)
- void [run](#) ()

### Constructors

- [JobCollection](#) ()
- [JobCollection](#) (Job job, int n) throws Exception
- [JobCollection](#) (Vector jobs) throws Exception
- [JobCollection](#) (Job[] jobs) throws Exception

### Jobs insertion/remotion handling

- boolean [empty](#) ()
- int [size](#) ()
- void [insertId](#) (JobId jobId) throws Exception
- void [insertAd](#) (JobAd jobAd) throws Exception
- void [insert](#) (Job job) throws Exception

- synchronized void `remove` (Job *job*) throws Exception
- void `clear` ()
- void `setCredPath` (File *cp*) throws Exception
- void `unsetCredPath` () throws Exception

#### Iteration action

- Iterator `jobs` ()

#### Submit

- Vector `submit` (Url *ns*, Vector *lbs*, String *ceId*) throws Exception

#### cancel

- Vector `cancel` (String *email*) throws Exception

#### getStatus

- Vector `getStatus` () throws Exception

### 1.3.1. DETAILED DESCRIPTION

The JobCollection Class is a container class for Job objects that has the main purpose of allowing the execution of collective operations on sets of independent jobs. The JobCollection class is just a logical container, and both not yet submitted and already submitted jobs can be inserted in it. A job collection is somehow orthogonal wrt a job cluster being a set of dependent jobs (e.g. all jobs spawned by the same father process).

#### Version:

0.1

#### Author:

Alessandro Maraschini <[alessandro.maraschini@datamat.it](mailto:alessandro.maraschini@datamat.it)>

### 1.3.2. CONSTRUCTOR & DESTRUCTOR DOCUMENTATION

#### 1.3.2.1 JobCollection::JobCollection ()

Instantiates an empty JobCollection object

#### 1.3.2.2 JobCollection::JobCollection (Job *job*, int *n*) throws Exception

Instantiates a collection with *n* copies of a job (the Job has to be of JOB\_AD type)

#### Parameters:

*job* - the source Job (of JOB\_TYPE) instances

*n* - the number of copies to be filled in the collection

#### Exceptions:

*JobCollectionException* - The job is not of JOB\_AD type

#### 1.3.2.3 JobCollection::JobCollection (Vector *jobs*) throws Exception

Instantiates a JobCollection object from a vector of Job

#### Parameters:

*jobs* - the vector of Job instances that have to be inserted

#### Exceptions:

*JobCollectionException* - two (or more) job ids are equals

*ClassCastException* if the Vector contains a non-Job instance

#### 1.3.2..4 JobCollection::JobCollection (Job jobs[]) throws Exception

Instantiates a JobCollection object from an array of Job

**Parameters:**

*jobs* - the array of Job instances that have to be inserted

**Exceptions:**

*JobCollectionException* - two (or more) job ids are equals

### 1.3.3. MEMBER FUNCTION DOCUMENTATION

#### 1.3.3..1 Vector JobCollection::cancel (String email) throws Exception

Cancel the job from the NS (SYNC)

**Parameters:**

*email* -The E-mail address where to send the cancellation notification result

#### 1.3.3..2 void JobCollection::clear ()

Deletes all elements from the collection.

#### 1.3.3..3 boolean JobCollection::empty ()

Check the size of the collection

**Returns:**

true if the collection's size is empty (length = 0)

#### 1.3.3..4 Vector JobCollection::getOutput (String dirPath) throws Exception

Get the output files of the jobs (SYNC)

**Parameters:**

*dir\_path* - the path where to retrieve the OutputSandbox files

#### 1.3.3..5 Vector JobCollection::getStatus () throws Exception

Retrieve the status information from the LB

#### 1.3.3..6 void JobCollection::insert (Job job) throws Exception

Insert a new Job to the collection

**Parameters:**

*job* - tht Job instance that has to be inserted

**Exceptions:**

*JobCollectionException* - Unable to insert the Job

#### 1.3.3..7 void JobCollection::insertAd (JobAd jobAd) throws Exception

Insert a new Job (of JobAd type) into the collection

**Parameters:**

*jobAd* - the JobAd instance that has to be inserted

**Exceptions:**

*JobCollectionException* - Unable to insert the Job

### 1.3.3..8 void JobCollection::insertId (JobId jobId) throws Exception

Insert a new Job (of JobId type) into the collection

**Parameters:**

*jobId* - the JobId instance that has to be inserted

**Exceptions:**

*JobCollectionException* - Unable to insert the Job

### 1.3.3..9 Iterator JobCollection::jobs ()

**Returns:**

an iterator pointing to the beginning of the collection

### 1.3.3..10 synchronized void JobCollection::remove (Job job) throws Exception

Remove a specified Job from the collection Delete the specified job from the collection (if the id has been set) Delete the last occurrence of the job from the collection (if the ad has not been set)

**Parameters:**

*job* - the Job that has to be removed

**Exceptions:**

*JobCollectionException* - Unable to remove the Job

### 1.3.3..11 void JobCollection::run ()

Thread Execution - This method execute the passed function as a separate thread

### 1.3.3..12 void JobCollection::setCredPath (File cp) throws Exception

Set a different Proxy certificate from the default one

**Parameters:**

*cp* - The full path of the proxy certificate file to be set

**Exceptions:**

*GlobusProxyException* - Unable to get the specified proxy certificate

### 1.3.3..13 void JobCollection::setMaxThreadNumber (int maxThread)

This method is used to override the MAX.THREAD.NUMBER macro variable

**Parameters:**

*maxThread* - the max number or simultaneous threads allowed (unless the -DWITHOUT.THREAD option is specified while compiling)

### 1.3.3..14 int JobCollection::size ()

**Returns:**

the length of the inserted Jobs

### 1.3.3.15 Vector JobCollection::submit (Url ns, Vector lbs, String ceId) throws Exception

Synchronous Submit method

**Parameters:**

- ns* The Network Server full address given in the form <NS host>:<NS port>
- ce\_id* The Computing Element Identifier where to perform the job

**Returns:**

a Vector of Result, one for each Job in the collection (with the same order as inserted)

**Exceptions:**

- GlobusProxyException* - Unable to get proxy certificate information
- JobCollectionException* - Some error occurred while executing the operation

### 1.3.3.16 void JobCollection::unsetCredPath () throws Exception

Set the Proxy certificate as default

**Exceptions:**

- GlobusProxyException* - Unable to get the default proxy certificate

## 1.4. JOBID CLASS REFERENCE

### MISCELLANEOUS

- boolean *equals* (JobId jid)
- void *clear* ()
- boolean *isSet* ()
- String *toString* ()
- void *fromString* (String jid) throws Exception
- synchronized void *setJobId* (Url lbServer) throws Exception

### PUBLIC METHODS

#### Constructors

- *JobId* ()
- *JobId* (Url lbAddress) throws Exception
- *JobId* (String jid) throws Exception

#### Get Methods

- Url *getLBAddress* () throws Exception
- String *getUniqueString* () throws Exception

### 1.4.1. CONSTRUCTOR & DESTRUCTOR DOCUMENTATION

#### 1.4.1.1 JobId::JobId ()

Instantiates an empty JobId object

#### 1.4.1.2 JobId::JobId (Url lbAddress) throws Exception

#### 1.4.1.3 JobId::JobId (String jid) throws Exception

Instantiates a JobId object from the passed dg\_jobId in String format.

**Parameters:**

*job\_id\_String* - a String representig a classAd expression

**Exceptions:**

- IllegalArgumentException* - When a String is passed in a wrong format

## 1.4.2. MEMBER FUNCTION DOCUMENTATION

### 1.4.2.1 void JobId::clear ()

Unsets the JobId instance. Clear all it's members

### 1.4.2.2 boolean JobId::equals (JobId *jid*)

Check if the job identifiers are equals

**Returns:**

true if the job identifiers are the same, false otherwise

### 1.4.2.3 void JobId::fromString (String *jid*) throws Exception

Set the JobId instance from the dg\_jobId in String format given as input.

**Parameters:**

*jid* = the job identifier string representation

**Exceptions:**

*IllegalArgumentException* - Wrong jobId format

### 1.4.2.4 Url JobId::getLBAddress () throws Exception

**Returns:**

the LB address into its String format

**Exceptions:**

*EmptyIdException* - If the jobId has not been initialised yet

### 1.4.2.5 String JobId::getUniqueString () throws Exception

**Returns:**

the unique String into its String format

**Exceptions:**

*EmptyIdException* - If the jobId has not been initialised yet

### 1.4.2.6 boolean JobId::isSet ()

Check whether the jobId has been already created (true) or not (false)

**Returns:**

true (jobId created) or false (jobId not yet created)

### 1.4.2.7 synchronized void JobId::setJobId (Url *lbServer*) throws Exception

Set the JobId instance from the dg\_jobId in String format given as input.

**Parameters:**

*lbServer* - An Url instance representing the LB address

**Exceptions:**

*IllegalArgumentException* - lbServer in a bad format

### 1.4.2..8 String JobId::toString ()

Converts the jobId into a String

**Returns:**

a string representing the job identifiers (if set), an empty string otherwise

## 1.5. JOBSTATE CLASS REFERENCE

### PUBLIC METHODS

- void `toFile` (java.io.File outputState)
- void `setId` (String jobId)
- void `check` () throws Exception
- void `setAttribute` (String attrName, Expr attrValue) throws Exception
- void `checkAttribute` (String attrName, Expr attrValue) throws Exception

### STATIC PUBLIC ATTRIBUTES

- String `JOBID` = "State\_Id"
- String `CURRENT` = "CurrentStep"
- String `JOBSTEPS` = "JobSteps"
- String `USERDATA` = "UserData"

### 1.5.1. DETAILED DESCRIPTION

This class stores the information related to a particular state of a checkpointable Job

**Version:**

0.1

**Author:**

Alessandro Maraschini <[alessandro.maraschini@datamat.it](mailto:alessandro.maraschini@datamat.it)>

### 1.5.2. MEMBER FUNCTION DOCUMENTATION

#### 1.5.2..1 void JobState::check () throws Exception

Check the validity of the JobState instance

#### 1.5.2..2 void JobState::checkAttribute (String attrName, Expr attrValue) throws Exception

#### 1.5.2..3 void JobState::setAttribute (String attrName, Expr attrValue) throws Exception

#### 1.5.2..4 void JobState::setId (String jobId)

#### 1.5.2..5 void JobState::toFile (java.io.File outputState)

### 1.5.3. MEMBER DATA DOCUMENTATION

#### 1.5.3..1 String JobState::CURRENT = "CurrentStep" [static]

#### 1.5.3..2 String JobState::JOBID = "State\_Id" [static]

#### 1.5.3..3 String JobState::JOBSTEPS = "JobSteps" [static]

#### 1.5.3..4 String JobState::USERDATA = "UserData" [static]

### 1.6. JOBSTATUS CLASS REFERENCE

#### PUBLIC METHODS

- String name ()
- int status ()
- String toString ()

#### STATIC PUBLIC ATTRIBUTES

- final String attNames []
- final int CANCEL\_REASON = 0
- final int CANCELLING = 1
- final int CE\_NODE = 2
- final int CHILDREN = 3
- final int CHILDREN\_HIST = 4
- final int CHILDREN\_NUM = 5
- final int CHILDREN\_STATES = 6
- final int CONDOR\_ID = 7
- final int CONDOR\_JDL = 8
- final int CPU\_TIME = 9
- final int DESTINATION = 10
- final int DONE\_CODE = 11
- final int EXIT\_CODE = 12
- final int EXPECT\_FROM = 13
- final int EXPECT\_UPDATE = 14
- final int GLOBUS\_ID = 15
- final int JDL = 16
- final int JOB\_ID = 17
- final int JOBTYP = 18
- final int LAST\_UPDATE\_TIME = 19
- final int LOCAL\_ID = 20
- final int LOCATION = 21
- final int MATCHED\_JDL = 22
- final int NETWORK\_SERVER = 23
- final int OWNER = 24
- final int PARENT\_JOB = 25
- final int REASON = 26
- final int RESUBMITTED = 27
- final int RSL = 28
- final int SEED = 29
- final int STATE\_ENTER\_TIME = 30
- final int STATE\_ENTER\_TIMES = 31
- final int SUBJOB\_FAILED = 32
- final int USER\_TAGS = 33
- final int USER\_VALUES = 34
- final int STATUS = 35
- final String code []
- final int UNDEF = 0
- final int SUBMITTED = 1
- final int WAITING = 2
- final int READY = 3
- final int SCHEDULED = 4
- final int RUNNING = 5
- final int DONE = 6
- final int CLEARED = 7
- final int ABORTED = 8
- final int CANCELLED = 9
- final int UNKNOWN = 10
- final int PURGED = 11

### 1.6.1. DETAILED DESCRIPTION

LB JobStatus Class Wrapped

**Version:**

0.1

**Author:**

Alessandro Maraschini <alessandro.maraschini@datamat.it>

### 1.6.2. MEMBER FUNCTION DOCUMENTATION

#### 1.6.2.1 String JobStatus::name ()

#### 1.6.2.2 int JobStatus::status ()

#### 1.6.2.3 String JobStatus::toString ()

### 1.6.3. MEMBER DATA DOCUMENTATION

#### 1.6.3.1 final int JobStatus::ABORTED = 8 [static]

Return Status Code: Aborted by system (at any stage)

#### 1.6.3.2 final String JobStatus::attNames[] [static]

#### 1.6.3.3 final int JobStatus::CANCEL\_REASON = 0 [static]

JobStatus Attribute: Cancel Reason

#### 1.6.3.4 final int JobStatus::CANCELLED = 9 [static]

Return Status Code: Cancelled by user

#### 1.6.3.5 final int JobStatus::CANCELLING = 1 [static]

JobStatus Attribute: Cancellation request

#### 1.6.3.6 final int JobStatus::CE\_NODE = 2 [static]

Worker node where the job is executed

#### 1.6.3.7 final int JobStatus::CHILDREN = 3 [static]

list of subjob IDs

#### 1.6.3.8 final int JobStatus::CHILDREN\_HIST = 4 [static]

summary (histogram) of children job states

#### 1.6.3.9 final int JobStatus::CHILDREN\_NUM = 5 [static]

number of subjobs

#### 1.6.3.10 final int JobStatus::CHILDREN\_STATES = 6 [static]

full status information of the children

**1.6.3..11 final int JobStatus::CLEARED = 7 [static]**

Return Status Code: Output transferred back to user and freed

**1.6.3..12 final String JobStatus::code[] [static]**

Initial value:

```
{  
    "undefined",  
    "Submitted",  
    "Waiting",  
    "Ready",  
    "Scheduled",  
    "Running",  
    "Done",  
    "Cleared",  
    "Aborted",  
    "Cancelled",  
    "Unknown",  
    "Purged" }
```

**1.6.3..13 final int JobStatus::CONDOR\_ID = 7 [static]**

Id within Condor-G

**1.6.3..14 final int JobStatus::CONDOR\_JDL = 8 [static]**

ClassAd passed to Condor-G for last job execution

**1.6.3..15 final int JobStatus::CPU\_TIME = 9 [static]**

Consumed CPU time

**1.6.3..16 final int JobStatus::DESTINATION = 10 [static]**

JobStatus Attribute: Ce Id

**1.6.3..17 final int JobStatus::DONE = 6 [static]**

Return Status Code: Execution finished, output not yet available

**1.6.3..18 final int JobStatus::DONE\_CODE = 11 [static]**

Return code

**1.6.3..19 final int JobStatus::EXIT\_CODE = 12 [static]**

Unix exit code

**1.6.3..20 final int JobStatus::EXPECT\_FROM = 13 [static]**

Sources of the missing information

**1.6.3..21 final int JobStatus::EXPECT\_UPDATE = 14 [static]**

JobStatus Attribute: Logged information not arrived

**1.6.3..22 final int JobStatus::GLOBUS\_ID = 15 [static]**

JobStatus Attribute: Globus allocated Id

**1.6.3..23 final int JobStatus::JDL = 16 [static]**

JobStatus Attribute: Job Description Language

**1.6.3..24 final int JobStatus::JOB\_ID = 17 [static]**

JobStatus Attribute: JobId

**1.6.3..25 final int JobStatus::JOBTYPE = 18 [static]**

Type of job

**1.6.3..26 final int JobStatus::LAST\_UPDATE\_TIME = 19 [static]**

JobStatus Attribute: Last known event of the job

**1.6.3..27 final int JobStatus::LOCAL\_ID = 20 [static]**

JobStatus Attribute: Id within LRMS

**1.6.3..28 final int JobStatus::LOCATION = 21 [static]**

JobStatus Attribute: Location

**1.6.3..29 final int JobStatus::MATCHED\_JDL = 22 [static]**

Full job description after matchmaking

**1.6.3..30 final int JobStatus::NETWORK\_SERVER = 23 [static]**

Network server handling the job

**1.6.3..31 final int JobStatus::OWNER = 24 [static]**

JobStatus Attribute: Job owner

**1.6.3..32 final int JobStatus::PARENT\_JOB = 25 [static]**

parent job of subjob

**1.6.3..33 final int JobStatus::PURGED = 11 [static]**

Return Status Code: Status cannot be determined

**1.6.3..34 final int JobStatus::READY = 3 [static]**

Return Status Code: Matching resources found

**1.6.3..35 final int JobStatus::REASON = 26 [static]**

JobStatus Attribute: Status Reason

**1.6.3..36 final int JobStatus::RESUBMITTED = 27 [static]**

JobStatus Attribute: The job was resubmitted

**1.6.3..37 final int JobStatus::RSL = 28 [static]**

JobStatus Attribute: RSL to Globus

**1.6.3..38 final int JobStatus::RUNNING = 5 [static]**

Return Status Code: Executable is running

**1.6.3..39 final int JobStatus::SCHEDULED = 4 [static]**

Return Status Code: Accepted by LRMS queue

**1.6.3..40 final int JobStatus::SEED = 29 [static]**

JobStatus Attribute: string used for generation of subjob IDs

**1.6.3..41 final int JobStatus::STATE\_ENTER\_TIME = 30 [static]**

JobStatus Attribute: Status enter time

**1.6.3..42 final int JobStatus::STATE\_ENTER\_TIMES = 31 [static]**

JobStatus Attribute: When all previous states were entered

**1.6.3..43 final int JobStatus::STATUS = 35 [static]**

JobStatus Attribute: The Status of the job

**1.6.3..44 final int JobStatus::SUBJOB\_FAILED = 32 [static]**

JobStatus Attribute: Subjob failed (the parent job will fail too)

**1.6.3..45 final int JobStatus::SUBMITTED = 1 [static]**

Return Status Code: Just submitted by user interface

**1.6.3..46 final int JobStatus::UNDEF = 0 [static]**

Return Status Code: indicates invalid, i.e. uninitialized instance

**1.6.3..47 final int JobStatus::UNKNOWN = 10 [static]**

Return Status Code: Status cannot be determined

**1.6.3..48 final int JobStatus::USER\_TAGS = 33 [static]**

JobStatus Attribute: List of user-specified information tags

**1.6.3..49 final int JobStatus::USER\_VALUES = 34 [static]**

JobStatus Attribute: Values assigned to user-specified information tags

**1.6.3..50 final int JobStatus::WAITING = 2 [static]**

Return Status Code: Accepted by WMS, waiting for resource allocation

## 1.7. USERCREDENTIAL CLASS REFERENCE

### PUBLIC METHODS

- `UserCredential ()` throws Exception
- `UserCredential (File credPath)` throws Exception
- `void setProxy (File credPath)` throws GlobusProxyException
- `void unsetProxy ()` throws GlobusProxyException
- `void checkProxy ()` throws GlobusProxyException
- `String getSubject ()` throws GlobusProxyException
- `String getIssuer ()` throws GlobusProxyException
- `boolean getCredType ()` throws Exception
- `int getStrenght ()` throws Exception
- `int getTimeLeft ()` throws GlobusProxyException

### STATIC PUBLIC METHODS

- `UserCredential createProxy (String passPhrase)` throws Exception
- `UserCredential createProxy (String passPhrase, String userProxy, String userCert, String userKey, int bits, int hours, boolean limited)` throws Exception

### PROTECTED METHODS

- `UserCredential (GlobusProxy proxy)`

#### 1.7.1. DETAILED DESCRIPTION

The `UserCredential` class provides methods that allow getting information about the user credentials. It does not allow the creation of proxy certificates that have to be generated by using the `grid-proxy-init Globus` command (the only way of handling credentials that is considered really safe). Namely this is needed since the pass-phrase (very sensitive information) should not be passed through any complex (hence likely to be insecure) software components like GUI. It is recalled that proxy existence and correct setting of the X509\* variables is required by all job monitoring and control methods.

#### Version:

0.1

#### Author:

Alessandro Maraschini <[alessandro.maraschini@datamat.it](mailto:alessandro.maraschini@datamat.it)>

#### 1.7.2. CONSTRUCTOR & DESTRUCTOR DOCUMENTATION

##### 1.7.2.1 `UserCredential::UserCredential (GlobusProxy proxy)` [protected]

Empty Constructor try to Load the default proxycertificate

#### Exceptions:

*GlobusProxyException* - Unable to create the proxy

*IOException* unable to find one or more specified files

*GeneralSecurityException* unable to decrypt the passphrase

##### 1.7.2.2 `UserCredential::UserCredential ()` throws Exception

Empty Constructor try to Load the default proxycertificate

#### Exceptions:

*GlobusProxyException* - Unable to get the default proxy certificate

##### 1.7.2.3 `UserCredential::UserCredential (File credPath)` throws Exception

### 1.7.3. MEMBER FUNCTION DOCUMENTATION

#### 1.7.3.1 void UserCredential::checkProxy () throws GlobusProxyException

Check if the Proxy Certificate is valid

**Exceptions:**

*GlobusProxyException* - Unable to get the proxy certificate

#### 1.7.3.2 UserCredential UserCredential::createProxy (String *passPhrase*, String *userProxy*, String *userCert*, String *userKey*, int *bits*, int *hours*, boolean *limited*) throws Exception [static]

#### 1.7.3.3 UserCredential UserCredential::createProxy (String *passPhrase*) throws Exception [static]

Create the default proxy

#### 1.7.3.4 boolean UserCredential::getCredType () throws Exception

Return whether the proxy is a full proxy (true) or a limited proxy (false)

**Exceptions:**

*GlobusProxyException* - Unable to get the proxy certificate

#### 1.7.3.5 String UserCredential::getIssuer () throws GlobusProxyException

Return the Issuer of the Proxy Certificate

**Exceptions:**

*GlobusProxyException* - Unable to get the proxy certificate

#### 1.7.3.6 int UserCredential::getStrenght () throws Exception

Return the Cred type of the Proxy Certificate

**Exceptions:**

*GlobusProxyException* - Unable to get the proxy certificate

#### 1.7.3.7 String UserCredential::getSubject () throws GlobusProxyException

Return the Subject of the Proxy Certificate

**Exceptions:**

*GlobusProxyException* - Unable to get the proxy certificate

#### 1.7.3.8 int UserCredential::getTimeLeft () throws GlobusProxyException

Return the Strenght of the Proxy Certificate

**Exceptions:**

*GlobusProxyException* - Unable to get the proxy certificate

#### 1.7.3.9 void UserCredential::setProxy (File *credPath*) throws GlobusProxyException

Set the proxy certificate to a non-default value

**Parameters:**

*credPath*- the path pointing to the proxy certificate file

**Exceptions:**

*GlobusProxyException* - Unable to get the specified proxy certificate

### 1.7.3..10 void `UserCredential::unsetProxy ()` throws `GlobusProxyException`

Unset a previous non-default proxy and look for the default one

**Exceptions:**

*GlobusProxyException* - Unable to get the default proxy certificate

## 1.8. USERJOBS CLASS REFERENCE

### PUBLIC METHODS

- void `setCredPath` (String cp) throws Exception
- void `unsetCredPath` () throws Exception
- Vector `getJobs` (Url lbAddress) throws Exception

### Constructors/Destructor

- `UserJobs` () throws Exception
- `UserJobs` (String credPath) throws Exception

### Action Methods

- Vector `getStates` (Url lbAddress) throws Exception
- Vector `cancel` (Url rbAddress, String userContact) throws Exception
- Vector `cancel` (Url rbAddress) throws Exception

### 1.8.1. DETAILED DESCRIPTION

Allow controlling all the jobs owned by the user The UserJobs class provides methods that allow controlling all the user's jobs during its lifetime. such as getting their status, logging information, or cancelling them

**Version:**

0.1

**Author:**

Alessandro Maraschini <alessandro.maraschini@datamat.it>

### 1.8.2. CONSTRUCTOR & DESTRUCTOR DOCUMENTATION

#### 1.8.2.1 `UserJobs::UserJobs ()` throws Exception

Instantiates an UserJobs object using default user credential

#### 1.8.2.2 `UserJobs::UserJobs (String credPath)` throws Exception

Instantiates an UserJobs object using non-default credential

**Parameters:**

*credPath* - The proxy certificate file where to retrieve credential from

### 1.8.3. MEMBER FUNCTION DOCUMENTATION

#### 1.8.3.1 `Vector UserJobs::cancel (Url rbAddress)` throws Exception

Remove all the user's jobs from the specified RB server and notify event by callback

**Parameters:**

*lb\_address* the full Resource Broker address given in the form <host>:<port>

**Returns:**

a Vector of Results

### 1.8.3..2 Vector UserJobs::cancel (Url *rbAddress*, String *userContact*) throws Exception

Remove all the user's jobs from the specified RB server and notify event by email

**Parameters:**

- lb\_address* the full Resource Broker address given in the form <host>:<port>
- user\_contact* - a valid e-mail address where to send the notification results

### 1.8.3..3 Vector UserJobs::getJobs (Url *lbAddress*) throws Exception

Retrieve the jobs owned by the user in a specific LB

**Parameters:**

- lb\_address* the full Logging and Bookkeeping address given in the form : [<protocol>://]<host>:<port>

### 1.8.3..4 Vector UserJobs::getStates (Url *lbAddress*) throws Exception

Retrieve the status of all the user's jobs

**Parameters:**

- lb\_address* - the full Logging and Bookkeeping address given in the form [<protocol>://]<host>:<port>
- jobsStatus* - the vector to be filled of all the jobs status information

### 1.8.3..5 void UserJobs::setCredPath (String *cp*) throws Exception

Set a different Proxy certificate from the default one and check it

**Parameters:**

- cp* - The full path of the proxy certificate file to be set

**Exceptions:**

- GlobusProxyException* - Unable to get the specified proxy certificate

### 1.8.3..6 void UserJobs::unsetCredPath () throws Exception

Set the Proxy certificate as default

**Exceptions:**

- GlobusProxyException* - Unable to get the default proxy certificate

## 2. BROKER INFORMATION

### 2.1. BROKERINFO CLASS REFERENCE

parses .BrokerInfo file containing job submission parameters.

#### PUBLIC METHODS

- ~BrokerInfo (void)
- std::string getBIFilename (void)
- BI.Result getCE (std::string &CE)
- BI.Result getDataAccessProtocol (std::vector< std::string > &DAPs)
- BI.Result getLFN2SFN (std::string LFN, std::vector< std::string > &SFNs)
- BI.Result getSEs (std::vector< std::string > &SEs)
- BI.Result getSEProtocols (std::string SE, std::vector< std::string > &SEProtos)
- BI.Result getSEPort (std::string SE, std::string SEProtocol, std::string &SEPort)
- BI.Result getCloseSEs (std::vector< std::string > &SEs)
- BI.Result getSEMoutPoint (std::string CloseSE, std::string &SEMout)
- BI.Result getInputData (std::vector< std::string > &LFNs)
- BI.Result getVirtualOrganization (std::string &VO)
- BI.Result getAccessCost (std::string &timeTotal, std::string &byteToTransf)
- int parser (std::string &outbuffer)

## STATIC PUBLIC METHODS

- BrokerInfo \* instance (void)

### 2.1.1. DETAILED DESCRIPTION

parses .BrokerInfo file containing job submission parameters.

#### Parameters:

*BrokerInfoFile*\_ holds the fullpath of the brokerinfo file

*fbrokerinfo*\_ handler to the brokerinfo file

*mbrokerinfo*\_ handler to the brokerinfo memory info

*instance*\_ it points to the current instance. this class is intended so to be instantiated only once (singleton).

### 2.1.2. CONSTRUCTOR & DESTRUCTOR DOCUMENTATION

#### 2.1.2.1 BrokerInfo::~~BrokerInfo (void)

\Default Destructor.

### 2.1.3. MEMBER FUNCTION DOCUMENTATION

#### 2.1.3.1 BI\_Result BrokerInfo::getAccessCost (std::string & timeTotal, std::string & byteToTransf)

#### Returns:

byte and time for SE data transfert @param timeTotal string. @param byteToTransfer string. @return BI.SUCCESS, or BI.ERROR if the operation failed

#### 2.1.3.2 std::string BrokerInfo::getBIFileName (void)

\Returns the BrokerInfo file to parse.

#### 2.1.3.3 BI\_Result BrokerInfo::getCE (std::string & CE)

#### Returns:

CE (Computing Element) Info. @param CE The ResourceID of the Computing Element to be returned. @return BI.SUCCESS, or BI.ERROR if the operation failed

#### 2.1.3.4 BI\_Result BrokerInfo::getCloseSEs (std::vector< std::string > & SEs)

#### Returns:

SE's protocol port number @param SE Vector of SEs close to the given CE. @return BI.SUCCESS, or BI.ERROR if the operation failed

#### 2.1.3.5 BI\_Result BrokerInfo::getDataAccessProtocol (std::vector< std::string > & DAPs)

#### Returns:

Data Access Protocol list @param DAPs Vector of Data Access Protocols specified by the user. @return BI.SUCCESS, or BI.ERROR if the operation failed

#### 2.1.3.6 BI\_Result BrokerInfo::getInputData (std::vector< std::string > & LFNs)

#### Returns:

InputData list @param LFN/GUID/LFC vector. @return BI.SUCCESS, or BI.ERROR if the operation failed

**2.1.3..7 BI\_Result BrokerInfo::getLFN2SFN (std::string LFN, std::vector< std::string > & SFNs)****Returns:**

LFN to SFN mapping @param LFN String of LFN specified. @param SFNs Vector of corresponding SFNs. @return BLSUCCESS, or BLERROR if the operation failed

**2.1.3..8 BI\_Result BrokerInfo::getSEMOUNTPOINT (std::string CloseSE, std::string & SEMOUNT)****Returns:**

SE MountPoint list following the order of SEs returned by \the getCloseSEs method. @param SEMOUNTS Vector of mount points for SEs. @return BLSUCCESS, or BLERROR if the operation failed

**2.1.3..9 BI\_Result BrokerInfo::getSEPort (std::string SE, std::string SEProtocol, std::string & SEPort)****Returns:**

SE Ports list ordered following the order of SEs \returned by getSEs @param SEPorts Vector of ports used by SEs. @return BLSUCCESS, or BLERROR if the operation failed

**2.1.3..10 BI\_Result BrokerInfo::getSEProtocols (std::string SE, std::vector< std::string > & SEProtos)****Returns:**

SE Protocols list ordered following the order of SEs \returned by getSEs @param SEProtocs Vector of protocols supported by SEs. @return BLSUCCESS, or BLERROR if the operation failed

**2.1.3..11 BI\_Result BrokerInfo::getSEs (std::vector< std::string > & SEs)****Returns:**

SE (Storage Element) list @param SEs Vector of SEs serving PFNs. @return BLSUCCESS, or BLERROR if the operation failed

**2.1.3..12 BI\_Result BrokerInfo::getVirtualOrganization (std::string & VO)****Returns:**

Virtual Organization string @param VO string. @return BLSUCCESS, or BLERROR if the operation failed

**2.1.3..13 BrokerInfo\* BrokerInfo::instance (void) [static]**

\Returns the handle to the instantiated object.

**2.1.3..14 int BrokerInfo::parser (std::string & outbuffer)**

\Parsing utilities

## 2.2. BROKERINFOEX CLASS REFERENCE

## 3. DATA MANAGEMENT

### 3.1. REPLICAMANAGER INTERFACE REFERENCE

#### 3.1.1. DETAILED DESCRIPTION

The ReplicaManager interface defines all methods that can be called by a client using the `edg-replica-manager` Reptor. This is the primary API that should be used by all client applications.

**Author:**

Peter Kunszt , Heinz Stockinger

**Version:**

**Id:**

ReplicaManager.java,v 1.13 2003/03/25 19:25:44 kurts Exp

### 3.2. EDGLocalREPLICACATALOG INTERFACE REFERENCE

#### PUBLIC METHODS

- void `addMapping` (final String guid, final String pfn)
- void `deleteMapping` (final String guid)
- void `removeMapping` (final String guid, final String pfn)
- boolean `guidExists` (final String guid)
- boolean `pfnExists` (final String pfn)
- String[] `getPfn` (final String guid)
- String `guidForPfn` (final String pfn)
- void `createAttributeDefinition` (final String attrName)
- String[] `getAttributeDefinitions` ()
- void `removeAttributeDefinition` (final String attrName)
- boolean `attributeDefinitionExists` (final String attrName)
- String `setPfnAttribute` (final String pfnName, final String attrDefnName, final String attrValue)
- String `getPfnAttribute` (final String pfnName, final String attrDefnName)
- void `removePfnAttribute` (final String pfnName, final String attrDefnName)
- String[] `getMappingsByGuid` (final String guidPattern)
- String[] `getMappingsByPfn` (final String pfnPattern)
- String[] `getMappingsByPfnAttribute` (final String attrName, final String attrPattern)
- String `ping` ()

#### 3.2.1. DETAILED DESCRIPTION

This is the interface to the Local Replica Catalog component of an RLS Server.

**Author:**

Peter Kunszt , James Casey

**Version:**

**Id:**

EdgLocalReplicaCatalog.java,v 1.15 2003/03/31 19:57:25 pkunszt Exp

### 3.2.2. MEMBER FUNCTION DOCUMENTATION

#### 3.2.2.1 void EdgLocalReplicaCatalog::addMapping (final String *guid*, final String *pfn*)

add a new mapping to the Catalog. If there is already a mapping, we add this mapping to the set of mappings. If there are no mappings already, we create a new mapping.

**Parameters:**

*guid* The guid  
*pfn* The physical filename

#### 3.2.2.2 boolean EdgLocalReplicaCatalog::attributeDefinitionExists (final String *attrName*)

do we have an attribute definition for a given attribute name?

**Parameters:**

*attrName* the name of attribute to check for

**Returns:**

**true** if the attribute name is there.

#### 3.2.2.3 void EdgLocalReplicaCatalog::createAttributeDefinition (final String *attrName*)

Create a new Attribute Definition in the Catalog.

**Parameters:**

*attrName* The name of the new attribute

**Exceptions:**

*AttributeDefinitionExistsException* if an attribute with the given name already exists

#### 3.2.2.4 void EdgLocalReplicaCatalog::deleteMapping (final String *guid*)

delete all parts of an guid to pfn mapping including attached attributes.

**Parameters:**

*guid* The guid for which to delete the mapping

#### 3.2.2.5 String [] EdgLocalReplicaCatalog::getAttributeDefinitions ()

Get the current list of attribute definition names

**Returns:**

the names, or null if there are no definitions

#### 3.2.2.6 String [] EdgLocalReplicaCatalog::getMappingsByGuid (final String *guidPattern*)

Return the guid to physical filename mappings which contain a guid that matches the given *guidPattern*. We assume unix regexp semantics. The result will be an empty set if no mappings are found.

**Parameters:**

*guidPattern* The GUID pattern we are searching for

**Returns:**

The mappings, as a `String[]`. This has an even number of entries in the for {*guid*, *pfn*, *guid*, *pfn*, ...}

### 3.2.2..7 String [] EdgLocalReplicaCatalog::getMappingsByPfn (final String pfnPattern)

Return the guid to physical filename mappings which contain a physical filename that matches the given pfnPattern. We assume unix regex semantics. The result will be an empty set if no mappings are found.

**Parameters:**

*pfnPattern* The Physical FileName pattern we are searching for

**Returns:**

The mappings, as a String[]. This has an even number of entries in the for {guid, pfn, guid, pfn, ...}

### 3.2.2..8 String [] EdgLocalReplicaCatalog::getMappingsByPfnAttribute (final String attrName, final String attrPattern)

Return the guid to physical filename mappings which contain a attribute that matches the given pattern. We assume unix regex semantics. The result will be an empty set if no mappings are found.

**Parameters:**

*attrName* the attribute to check

*attrPattern* The attribute pattern we are searching for

**Exceptions:**

*NoSuchAttributeDefinitionException*

**Returns:**

The mappings, as a String[]. This has an even number of entries in the for {guid, pfn, guid, pfn, ...}

### 3.2.2..9 String EdgLocalReplicaCatalog::getPfnAttribute (final String pfnName, final String attrDefnName)

return the value of a given attribute for an PFN.

**Parameters:**

*pfnName* the PFN

*attrDefnName* the AttributeDefinition name to check for

**Returns:**

the attribute value

**Exceptions:**

*NoSuchPfnException* if the PFN does not exist in the Catalog

*NoSuchAttributeDefinitionException* if the AttributeDefintion does not exist in the catalog.

### 3.2.2..10 String [] EdgLocalReplicaCatalog::getPfn (final String guid)

For a given guid, return all the physical filenames we map to in this Catalog. This will always have at least one entry, and never be empty.

**Parameters:**

*guid* The GUID

**Exceptions:**

*NoSuchGuidException* If the guid does not exist in the catalog.

### 3.2.2..11 boolean EdgLocalReplicaCatalog::guidExists (final String guid)

Do we have a given guid in the catalog.

**Parameters:**

*guid* The guid we are checking for

**Returns:**

**true** if we have a mapping for the guid, **false** otherwise

### 3.2.2..12 String EdgLocalReplicaCatalog::guidForPfn (final String pfn)

For a given pfn, return the guid that points to it. The return value will never be **null**.

**Parameters:**

*pfn* The physical filename.

**Returns:**

the guid that maps to it

**Exceptions:**

*NoSuchPfnException* If the pfn does not exist in the catalog.

### 3.2.2..13 boolean EdgLocalReplicaCatalog::pfnExists (final String pfn)

Do we have a given physical filename in the catalog

**Parameters:**

*pfn* The physical filename we are checking for

**Returns:**

**true** if we have a mapping for the physical filename, **false** otherwise

### 3.2.2..14 String EdgLocalReplicaCatalog::ping ()

Is the server alive. It returns a stats string. If everything is ok, this will start with OK else it will start with FAIL.

**Returns:**

the stats string

### 3.2.2..15 void EdgLocalReplicaCatalog::removeAttributeDefinition (final String attrName)

Delete an Attribute Definition from the Catalog. This will also remove any attribute value with this definition from GUID and Physical FileName entries in the Catalog.

**Parameters:**

*attrName* the name of the attribute definition to remove

### 3.2.2..16 void EdgLocalReplicaCatalog::removeMapping (final String guid, final String pfn)

remove from the catalog a single guid to pfn mapping.

**Parameters:**

*guid* the guid whose mapping we look at

*pfn* the physical filename to remove

### 3.2.2..17 void EdgLocalReplicaCatalog::removePfnAttribute (final String pfnName, final String attrDefnName)

remove the value of a given attribute for an PFN.

**Parameters:**

*pfnName* the PFN

*attrDefnName* the AttributeDefinition name to check for

### 3.2.2..18 String EdgLocalReplicaCatalog::setPfnAttribute (final String pfnName, final String attrDefnName, final String attrValue)

Add an attribute value to a PFN.

#### Parameters:

- pfnName* The PFN
- attrDefnName* the name of attribute
- attrValue* The attribute value we'll add.

#### Exceptions:

- NoSuchPfnException* if the PFN does not exist in the Catalog
- NoSuchAttributeDefinitionException* if the definition of the attribute does not exist in the Catalog.

## 3.3. EDGREPLICAMETADATACATALOG INTERFACE REFERENCE

### PUBLIC METHODS

- void **addAlias** (final String guid, final String alias)
- void **removeAlias** (final String guid, final String alias)
- boolean **aliasExists** (final String alias)
- boolean **guidExists** (final String guid)
- String **guidForAlias** (final String alias)
- String[] **getAliases** (final String guid)
- void **createAttributeDefinition** (final String attrName)
- String[] **getAttributeDefinitions** ()
- boolean **attributeDefinitionExists** (final String attrName)
- void **removeAttributeDefinition** (final String attrName)
- String **setGuidAttribute** (final String guidName, final String attrDefnName, final String attrValue)
- String **getGuidAttribute** (final String guidName, final String attrDefnName)
- void **removeGuidAttribute** (final String guidName, final String attrDefnName)
- String **setAliasAttribute** (final String aliasName, final String attrDefnName, final String attrValue)
- String **getAliasAttribute** (final String aliasName, final String attrDefnName)
- void **removeAliasAttribute** (final String aliasName, final String attrDefnName)
- String[] **getMappingsByGuid** (final String guidPattern)
- String[] **getMappingsByAlias** (final String aliasPattern)
- String[] **getMappingsByGuidAttribute** (final String attrName, final String attrPattern)
- String[] **getMappingsByAliasAttribute** (final String attrName, final String attrPattern)
- String **ping** ()

### 3.3.1. DETAILED DESCRIPTION

This is the interface to the EDG Metadata Catalog

#### Author:

James Casey

#### Version:

#### Id:

EdgReplicaMetadataCatalog.java,v 1.1 2003/03/06 16:08:52 jamesc Exp

### 3.3.2. MEMBER FUNCTION DOCUMENTATION

**3.3.2..1 void EdgReplicaMetadataCatalog::addAlias (final String guid, final String alias)**

**3.3.2..2 boolean EdgReplicaMetadataCatalog::aliasExists (final String alias)**

**3.3.2..3 boolean EdgReplicaMetadataCatalog::attributeDefinitionExists (final String attrName)**

**3.3.2..4 void EdgReplicaMetadataCatalog::createAttributeDefinition (final String *attrName*)**

**3.3.2..5 String EdgReplicaMetadataCatalog::getAliasAttribute (final String *aliasName*, final String *attrDefnName*)**

returns the attribute value, or null if there is no attribute value set

**3.3.2..6 String [] EdgReplicaMetadataCatalog::getAliases (final String *guid*)**

**3.3.2..7 String [] EdgReplicaMetadataCatalog::getAttributeDefinitions ()**

**3.3.2..8 String EdgReplicaMetadataCatalog::getGuidAttribute (final String *guidName*, final String *attrDefnName*)**

returns the attribute value, or null if there is no attribute value set

**3.3.2..9 String [] EdgReplicaMetadataCatalog::getMappingsByAlias (final String *aliasPattern*)**

**3.3.2..10 String [] EdgReplicaMetadataCatalog::getMappingsByAliasAttribute (final String *attrName*, final String *attrPattern*)**

**3.3.2..11 String [] EdgReplicaMetadataCatalog::getMappingsByGuid (final String *guidPattern*)**

**3.3.2..12 String [] EdgReplicaMetadataCatalog::getMappingsByGuidAttribute (final String *attrName*, final String *attrPattern*)**

**3.3.2..13 boolean EdgReplicaMetadataCatalog::guidExists (final String *guid*)**

**3.3.2..14 String EdgReplicaMetadataCatalog::guidForAlias (final String *alias*)**

**3.3.2..15 String EdgReplicaMetadataCatalog::ping ()**

Is the server alive. It returns a stats string. If everything is ok, this will start with OK else it will start with FAIL.

**Returns:**

the stats string

**3.3.2..16 void EdgReplicaMetadataCatalog::removeAlias (final String *guid*, final String *alias*)**

**3.3.2..17 void EdgReplicaMetadataCatalog::removeAliasAttribute (final String *aliasName*, final String *attrDefnName*)**

**3.3.2..18 void EdgReplicaMetadataCatalog::removeAttributeDefinition (final String *attrName*)**

**3.3.2..19 void EdgReplicaMetadataCatalog::removeGuidAttribute (final String *guidName*, final String *attrDefnName*)**

**3.3.2..20 String EdgReplicaMetadataCatalog::setAliasAttribute (final String *aliasName*, final String *attrDefnName*, final String *attrValue*)**

Returns the old attribute, or null if no old attribute exists

### 3.3.2..21 String EdgReplicaMetadataCatalog::setGuidAttribute (final String *guidName*, final String *attrDefnName*, final String *attrValue*)

Returns the old attribute, or null if no old attribute exists

## 3.4. EDGREPLICAOPTIMIZATION INTERFACE REFERENCE

### PUBLIC METHODS

- AccessCost[] [getAccessCost](#) (ROSFile[] file, ROSComputingElement[] ce, String[] transport)
- BestNetworkCost [getBestNetworkCost](#) (ROSFile file, String destinationSE)
- String [ping](#) ()

#### 3.4.1. DETAILED DESCRIPTION

A SOAP API for the Replica Optimization part of the Replica Manager. This is only used internally.

Note: For TB 2.0 'best' files are replicated to the "close SE". All the parameter arrays should not contain null entries and should all be the same length.

**See also:**

[Design of a Replica Optimisation Framework](#)

**Author:**

[Kurt Stockinger](#), [Heinz Stockinger](#), [Peter Kunszt](#)

**Version:**

**Id:**

EdgReplicaOptimization.java,v 1.3 2003/03/24 10:57:22 kurts Exp

#### 3.4.2. MEMBER FUNCTION DOCUMENTATION

##### 3.4.2..1 AccessCost [] EdgReplicaOptimization::getAccessCost (ROSFile *file*[], ROSComputingElement *ce*[], String *transport*[])

Calculate the expected file access cost per-Logical-File-Name. It makes the assumptions that no replication occurs.

**Parameters:**

- files* the array of ROS File Names.
- ce* the array of ROSComputingElement.
- transport* FileHandler to be used.

**Returns:**

The AccessCost array in same order as *ce* array

**See also:**

[Design of a Replica Optimisation Framework](#)

##### 3.4.2..2 BestNetworkCost EdgReplicaOptimization::getBestNetworkCost (ROSFile *file*, String *destinationSE*)

Return the FileName (PFN) of the best file in terms of network latencies.

**Parameters:**

- file* ROS File.
- destinationSE* destination StorageElement.

**Returns:**

FileName

**See also:**

[Design of a Replica Optimisation Framework](#)

### 3.4.2..3 String EdgReplicaOptimization::ping ()

Send a contact message to the Optor server

**Returns:**

success if Optor is listening

## 4. R-GMA

### 4.1. APIBASE CLASS REFERENCE

Inherited by [Consumer](#), and [Declarable](#).

#### PUBLIC METHODS

- void [close \(\)](#)
- void [disconnect \(\)](#) throws [RGMAException](#)
- [ServletConnection](#) [getServletConnection \(\)](#) throws [RGMAException](#)
- [ResultSet](#) [getStatus \(\)](#) throws [RGMAException](#)
- [TimeInterval](#) [getTerminationInterval \(\)](#) throws [RGMAException](#)
- boolean [getTupleChecking \(\)](#) throws [RGMAException](#)
- void [setTupleChecking](#) (boolean tc) throws [RGMAException](#)
- void [setTerminationInterval](#) ([TimeInterval](#) interval) throws [RGMAException](#)
- void [showSignOfLife \(\)](#) throws [RGMAException](#)

#### PROTECTED METHODS

- [APIBase \(\)](#)
- long [calcTerminationInterval](#) (String intervalString)
- void [checkIfConnected \(\)](#) throws [RGMAException](#)
- void [getProperties](#) (String className, String propertyName) throws [RGMAException](#)
- void [reconnect](#) ([ServletConnection](#) connection) throws [RGMAException](#)
- [ResultSet](#) [sendCommand](#) (String command) throws [RGMAException](#)
- [ResultSet](#) [sendCommand](#) (String command, String parameterName, String parameterValue) throws [RGMAException](#)
- [ResultSet](#) [sendCommand](#) (String command, [ServletConnection](#) connection) throws [RGMAException](#)
- void [finalize \(\)](#) throws [Throwable](#)

#### PROTECTED ATTRIBUTES

- Category [cat](#)
- int [connectionId](#)
- String [exceptionType](#)
- String [servletURL](#)
- int [IS\\_CONNECTED](#) = 2
- int [IS\\_DISCONNECTED](#) = 1
- int [IS\\_CLOSED](#) = 0
- int [currentState](#)

#### STATIC PROTECTED ATTRIBUTES

- [NetLogger](#) [nl](#) = [NetLoggerFactory](#).[getInstance\(\)](#)
- final long [DEFAULT\\_TERMINATION\\_INTERVAL](#) = 1200 \* 1000

#### 4.1.1. DETAILED DESCRIPTION

An API is used to communicate with a servlet. The servlet can be instructed to carry on without the client.

## 4.1.2. CONSTRUCTOR & DESTRUCTOR DOCUMENTATION

### 4.1.2.1 APIBase::APIBase () [protected]

Constructs an default APIBase object.

## 4.1.3. MEMBER FUNCTION DOCUMENTATION

### 4.1.3.1 long APIBase::calcTerminationInterval (String intervalString) [protected]

**Time** Interval before which the api must have issued an action of the servlet, otherwise it is removed Format = DDDDDHHMMSS  
XXX: delete this when setTerminationInterval(String intervalString) goes

### 4.1.3.2 void APIBase::checkIfConnected () throws RGMAException [protected]

Checks if the APIBase is still connected. Throws an [RGMAException](#) with an appropriate message if `close()` or `disconnect()` have been called. Note, this method should be called at the start of every public method.

**Exceptions:**

[RGMAException](#) if the object is not connected.

### 4.1.3.3 void APIBase::close ()

Releases all resources held by the corresponding servlet. The `close()` method should be called from a `finally` block. This ensures that all necessary clean-up operations are carried out. The class has a `finalize` method; however as this is not guaranteed to be called you should always ensure that you call `close` yourself. As a precaution, when using R-GMA, it is good to invoke the garbage collector explicitly before program termination as indicated by the pattern below:

```
static void main(String[] args) {
    Program program = new Program(...);
    program = null;
    system.gc();
}
```

The above ensures that after the `program = null` statement, all objects may be garbage collected. The system garbage collector is then called.

Once `close()` has been called, [RGMAExceptions](#) will be raised if any other methods is called.

### 4.1.3.4 void APIBase::disconnect () throws RGMAException

Disconnects from the servlet in a way which allows a new API instance to reconnect. The normal action of the `finalize` method is clean up the servlet. The servlet will continue to exist until the specified time interval has passed. After this time the Constructor with an argument of the [ServletConnection](#) will give an error. Once `disconnect` has been called, [RGMAExceptions](#) will be raised if other methods of the API are called.

**Exceptions:**

[RGMAException](#) if the object has been closed or disconnected

### 4.1.3.5 void APIBase::finalize () throws Throwable [protected]

Called by the `GarbageCollector` whenever the object can't be reached.

**Exceptions:**

[Throwable](#)

**4.1.3..6 void APIBase::getProperties (String *className*, String *propertyName*) throws **RGMAException** [protected]**

get Properties from somewhere. XXX: this should be renamed? as at the moment it is only used for retrieving the servletURL (and names are hardwired in code)... could do better here!

**4.1.3..7 ServletConnection APIBase::getServletConnection () throws **RGMAException****

Returns a [ServletConnection](#) which may be used to reconnect to the server.

**Returns:**

a servlet connection

**Exceptions:**

**RGMAException** if the object has been closed or disconnected

**4.1.3..8 ResultSet APIBase::getStatus () throws **RGMAException****

Get status information.

**Returns:**

[ResultSet](#) containing status information for this producer

**Exceptions:**

**RGMAException** if the object has been closed or disconnected

**4.1.3..9 TimeInterval APIBase::getTerminationInterval () throws **RGMAException****

Returns the time interval before which the API must have issued an action of the servlet. Otherwise all knowledge of the Producer will be lost.

**Returns:**

interval time by which the API will be used again

**Exceptions:**

**RGMAException** if the object has been closed or disconnected

**See also:**

[setTerminationInterval](#)

**4.1.3..10 boolean APIBase::getTupleChecking () throws **RGMAException****

Returns value of TupleChecking.

**Returns:**

true if TupleChecking is on

**Exceptions:**

**RGMAException** if the object has been closed or disconnected

**See also:**

[setTupleChecking](#)

**4.1.3..11 void APIBase::reconnect (ServletConnection connection) throws RGMAException**  
[protected]

Reconnects to an existing *ServletConnection*. It is an error to have more than one API object making use of the same *ServletConnection*.

**Parameters:**

*connection* is the *ServletConnection* which will be reconnected

**Exceptions:**

*RGMAException* if the object has been closed

**4.1.3..12 ResultSet APIBase::sendCommand (String command, ServletConnection connection) throws RGMAException** [protected]

sends a command using a connection that already has command parameters added (including the connectionId).

**Parameters:**

*command* a servlet command

*connection* a connection to a servlet, with command parameters already added

**Returns:**

RGMA *ResultSet* response from the servlet

**Exceptions:**

*RGMAException* thrown if the servlet can't be reached, or if the APIBase instance at the servlet has been closed

**4.1.3..13 ResultSet APIBase::sendCommand (String command, String parameterName, String parameterValue) throws RGMAException** [protected]

sends a command with a parameter to the servlet with a parameter.

**Parameters:**

*command* a *String* setting the command to be sent

*parameterName* name of the parameter

*parameterValue* value of the parameter

**Returns:**

RGMA *ResultSet* response from the servlet

**Exceptions:**

*RGMAException* thrown if the servlet can't be reached, or if the APIBase instance at the servlet has been closed

#### 4.1.3..14 **ResultSet** `APIBase::sendCommand (String command)` throws **RGMAException** [protected]

sends a command to the servlet

##### Parameters:

*command* a `String` setting the command to be sent

##### Returns:

RGMA `ResultSet` response from the servlet

##### Exceptions:

**RGMAException** thrown if the servlet can't be reached, or if the `APIBase` instance at the servlet has been closed

#### 4.1.3..15 `void APIBase::setTerminationInterval (TimeInterval interval)` throws **RGMAException**

Sets the time interval before which the API must have issued an action to the servlet. Otherwise all knowledge of the API object will be lost.

##### Parameters:

*interval* time by which the API will be used again

##### Exceptions:

**RGMAException** if the object has been closed or if there are problems connecting to the servlet.

##### See also:

`getTerminationInterval`

#### 4.1.3..16 `void APIBase::setTupleChecking (boolean tc)` throws **RGMAException**

Set value of `TupleChecking`.

##### Parameters:

*tc* true to enable tuple checking

##### Exceptions:

**RGMAException**

##### See also:

`getTupleChecking`

#### 4.1.3..17 `void APIBase::showSignOfLife ()` throws **RGMAException**

This informs the servlet that the API is still alive.

##### Exceptions:

**RGMAException**

#### 4.1.4. MEMBER DATA DOCUMENTATION

##### 4.1.4.1 Category `APIBase::cat` [protected]

Category used by log4j for logging.

##### 4.1.4.2 `int APIBase::connectionId` [protected]

Uniquely identifies this APIBase with the servlet.

##### 4.1.4.3 `int APIBase::currentState` [protected]

The current state of the client. The client may be in one of three states:

See also:

`IS_CONNECTED` , `IS_DISCONNECTED` or `IS_CLOSED`.

##### 4.1.4.4 `final long APIBase::DEFAULT_TERMINATION_INTERVAL = 1200 * 1000` [static, protected]

The termination interval in milliseconds used in GRRP. By default this is set to 20 minutes.

##### 4.1.4.5 `String APIBase::exceptionType` [protected]

The type of the exception set when `RGMAExceptions` are thrown.

##### 4.1.4.6 `int APIBase::IS_CLOSED = 0` [protected]

A status flag that indicates the connection is closed between the client and server.

##### 4.1.4.7 `int APIBase::IS_CONNECTED = 2` [protected]

A status flag that indicates a valid connection with the Server.

##### 4.1.4.8 `int APIBase::IS_DISCONNECTED = 1` [protected]

A status flag that indicates the client has disconnected from the Server.

##### 4.1.4.9 `NetLogger APIBase::nl = NetLoggerFactory.getInstance()` [static, protected]

A Netlogger to write to.

##### 4.1.4.10 `String APIBase::servletURL` [protected]

URL of the servlet that will support this APIBase.

#### 4.2. ARCHIVER CLASS REFERENCE

Inherits [Declarable](#).

## PUBLIC METHODS

- **Archiver** (**ServletConnection** consumerProducerConnection) throws **RGMAException**
- **Archiver** (String rdbms, String user, String password) throws **RGMAException**
- **Archiver** (Insertable ins) throws **RGMAException**
- void **add** (String tableName, String predicate, int flags) throws **RGMAException**
- void **add** (String tableName, String predicate, int flags, Vector producerConnections) throws **RGMAException**
- void **declareTable** (Vector servletConnections, String tableName, String predicate, String tableDesc, String cleanUpPredicate, **TimeInterval** cleanUpInterval) throws **RGMAException**
- int **getBufferSize** (String tableName) throws **RGMAException**
- void **setBufferSize** (String tableName, int bufferSize) throws **RGMAException**

## PROTECTED METHODS

- **Archiver** () throws **RGMAException**

### 4.2.1. DETAILED DESCRIPTION

Archiver is able to consume, archive and produce **data**. For each table it is instructed to handle it instantiates a consumer, to "select \*" that table and stores it in an RDBMS (via JDBC) and announces itself as a producer of that information.

It is normally instantiated with the CleanUpable it should use to publish to. Tables are then "declared". By declaring a table, you indicate the table to be consumed and cause the **data** to be consumed and then inserted into the Producer.

When the Archiver is no longer needed, its `close()` method should be called. This ensures that the registry is updated accordingly, and that resources associated with the Archiver, such as database connections, are released. Ideally, `close()` should be used within a `finally` block:

```
try { ... } catch (RGMAException e) { ... } finally { archiver.close(); }
```

### 4.2.2. CONSTRUCTOR & DESTRUCTOR DOCUMENTATION

#### 4.2.2.1 **Archiver::Archiver ()** throws **RGMAException** [protected]

Default constructor for initialising variables

#### 4.2.2.2 **Archiver::Archiver (ServletConnection consumerProducerConnection)** throws **RGMAException**

Reconnect a ConsumerProducer.

#### Parameters:

**consumerProducerConnection** identifies the servlet to reconnect to.

#### Exceptions:

**RGMAException**

#### 4.2.2.3 Archiver::Archiver (String *rbms*, String *user*, String *password*) throws **RGMAException**

Create an archiver and connect it to the specified RDBMS via JDBC.

**Parameters:**

*rbms* JDBC RDBMS identification

*user* JDBC user name

*password* JDBC user password

**Exceptions:**

***RGMAException***

**Deprecated:**

use (`Insertable ins`) instead

#### 4.2.2.4 Archiver::Archiver (**Insertable** *ins*) throws **RGMAException**

Create an Archiver where the Producer (an `Insertable`) is passed in. The Archiver takes control of the insertable and will disconnect it if it's not already disconnected. The `Insertable` should *not* be used after it has been passed to this constructor.

**Parameters:**

*ins* An `Insertable` object

**Exceptions:**

***RGMAException***

### 4.2.3. MEMBER FUNCTION DOCUMENTATION

#### 4.2.3.1 void Archiver::add (String *tableName*, String *predicate*, int *flags*, Vector *producerConnections*) throws **RGMAException**

Specify table to consume, archive and, in turn, produce

**Parameters:**

*tableName* is the name of the SQL table

*predicate* an SQL WHERE clause expressing a predicate that is true for the registered table. Currently of the form WHERE (column\_1=value\_1 AND column\_2=value\_2 AND ...)

*flags* is a set of boolean flags encoded in an integer

*producerConnections* a Vector of `producerConnections` to identify the producers

**Exceptions:**

***RGMAException***

**Deprecated:**

#### 4.2.3..2 void Archiver::add (String *tableName*, String *predicate*, int *flags*) throws **RGMAException**

Specify table to consume, archive and, in turn, produce

##### Parameters:

*tableName* is the name of the SQL table

*predicate* an SQL WHERE clause expressing a predicate that is true for the registered table. Currently of the form WHERE (column\_1=value\_1 AND column\_2=value\_2 AND ...)

*flags* is a set of boolean flags encoded in an integer

##### Exceptions:

**RGMAException** .

##### Deprecated:

#### 4.2.3.3 void Archiver::declareTable (Vector *servletConnections*, String *tableName*, String *predicate*, String *tableDesc*, String *cleanUpPredicate*, TimeInterval *cleanUpInterval*) throws **RGMAException**

Declares a table - creating it if necessary. If the table is already known within the schema and is inconsistent with the *tableDesc* this will fail.

##### Parameters:

*producerConnections* a Vector of ServletConnections for the Consumers to connect to. The Query will be sent to each Producer and the results returned. This means that if you ask for max(...) with a Vector of size two you will get two results returned.

*tableName* is the name of the SQL table

*predicate* an SQL WHERE clause expressing a predicate that is true for the registered table. Currently of the form WHERE (column\_1=value\_1 AND column\_2=value\_2 AND ...)

*tableDesc* is the table description. This is the same as the SQL for CREATE TABLE. Some of the columns of the table must be marked PRIMARY KEY. This may be null.

*cleanUpPredicate* an SQL WHERE clause which is used to define the tuples to be eliminated from the Producer. This could be implemented as a cron job running at an interval defined by the *cleanUpInterval*.

*cleanUpInterval* interval between cleaning up information as defined by the *cleanUpPredicate*.

##### Exceptions:

**RGMAException** .

#### 4.2.3..4 int Archiver::getBufferSize (String *tableName*) throws RGMAException

Returns the buffer size for the consumer involved in archiving a particular table.

**Parameters:**

*tableName* the name of the table which the consumer is archiving

**Deprecated:**

#### 4.2.3..5 void Archiver::setBufferSize (String *tableName*, int *bufferSize*) throws RGMAException

Sets the buffer size for the consumer involved in archiving a particular table. Archiving will only begin for this table once a non-zero buffer size has been set. Streaming will stop if the buffer size is set back to 0.

**Parameters:**

*tableName* - the name of the table which will be archived.

*bufferSize* - desired buffer size measured as number of events

**Deprecated:**

### 4.3. CANONICALPRODUCER CLASS REFERENCE

Inherits [Declarable](#).

#### PUBLIC METHODS

- [CanonicalProducer](#) (int port, int flag) throws RGMAException
- [CanonicalProducer](#) (String address, int flag) throws RGMAException
- [CanonicalProducer](#) ([ServletConnection](#) connection) throws RGMAException

#### STATIC PUBLIC ATTRIBUTES

- final int [LATEST](#) = 2
- final int [HISTORY](#) = 4

#### PROTECTED METHODS

- [CanonicalProducer](#) () throws RGMAException

#### 4.3.1. DETAILED DESCRIPTION

CanonicalProducer is a Publisher which is able to answer Canonical queries. A Canonical query can only access the most recent tuples for a given primary key (except for the time stamp) Unlike a normal producer it can deal with multiple tables. When the CanonicalProducer is no longer needed, it's `disconnect` method should be called, ideally from a finally block. This ensures that the registry is updated accordingly, and that resources associated with the CanonicalProducer, such as database connections, are released. It may be desirable to have the constructor taking a Properties String.

#### 4.3.2. CONSTRUCTOR & DESTRUCTOR DOCUMENTATION

##### 4.3.2..1 CanonicalProducer::CanonicalProducer () throws RGMAException [protected]

default constructor for setting instance variables

##### Exceptions:

*RGMAException*

##### 4.3.2..2 CanonicalProducer::CanonicalProducer (int port, int flag) throws RGMAException

Construct a CanonicalProducer which will use a Socket connection

##### Parameters:

*port, the* port on which the CanonicalProducer will listen for communications from the Servlet.

*flag, a* flag which indicates the type of queries this CanonicalProducer can support.

**Latest** a query over the latest tuples, where latest is defined as the most recent time for each primary key (excluding the time fields which are present in all primary keys).

**History** a query over all tuples.

##### 4.3.2..3 CanonicalProducer::CanonicalProducer (String address, int flag) throws RGMAException

Construct a CanonicalProducer which will use the HTTP protocol

##### Parameters:

*address, the* URL of the code which will accept connections from the CanonicalProducerServlet.

*flag, a* flag which indicates what type of queries this CanonicalProducer can support.

**Latest** a query over the latest tuples, where latest is defined as the most recent time for each primary key (excluding the time fields which are present in all primary keys).

**History** a query over all tuples.

#### 4.3.2.4 CanonicalProducer::CanonicalProducer (ServletConnection connection) throws RGMAException

Reconnects to an existing `ServletConnection`. It is an error to have more than one API object making use of the same `ServletConnection`.

**Parameters:**

`sc` is the `CanonicalProducer` servlet connection which will be reconnected

### 4.3.3. MEMBER DATA DOCUMENTATION

#### 4.3.3.1 final int CanonicalProducer::HISTORY = 4 [static]

A history query

#### 4.3.3.2 final int CanonicalProducer::LATEST = 2 [static]

A latest query

## 4.4. CIRCULARBUFFERPRODUCER CLASS REFERENCE

Inherits `Insertable`.

### PUBLIC METHODS

- `CircularBufferProducer` (`ServletConnection` connection) throws `RGMAException`
- `CircularBufferProducer` (`String` tableName, `String` predicate, `int` flags) throws `RGMAException`
- `int` `getRemoteBufferSize` () throws `RGMAException`
- `void` `setRemoteBufferSize` (`int` remoteBufferSize) throws `RGMAException`
- `void` `setTerminationInterval` (`String` intervalString) throws `RGMAException`
- `ProducerConnection` `getProducerConnection` () throws `RGMAException`

### STATIC PUBLIC ATTRIBUTES

- `final int` `archive` = 1

### PROTECTED METHODS

- `CircularBufferProducer` () throws `RGMAException`

#### 4.4.1. DETAILED DESCRIPTION

`CircularBufferProducer` is able to register a table when it is created and subsequently to publish information.

*Note that this class should no longer be used. Instead consider the `StreamProducer`.*

**Deprecated:**

#### 4.4.2. CONSTRUCTOR & DESTRUCTOR DOCUMENTATION

**4.4.2.1 CircularBufferProducer::CircularBufferProducer () throws RGMAException**  
[protected]

**4.4.2.2 CircularBufferProducer::CircularBufferProducer (ServletConnection connection) throws RGMAException**

**4.4.2.3 CircularBufferProducer::CircularBufferProducer (String tableName, String predicate, int flags) throws RGMAException**

For this constructor the tableName must already be known within the schema

**Parameters:**

*tableName* is the name of the SQL table

*predicate* an SQL WHERE clause expressing a predicate that is true for the registered table. Currently of the form WHERE (column\_1=value\_1 AND column\_2=value\_2 AND ...)

*flags* is a set of boolean flags encoded in an integer

**Deprecated:**

#### 4.4.3. MEMBER FUNCTION DOCUMENTATION

**4.4.3.1 ProducerConnection CircularBufferProducer::getProducerConnection () throws RGMAException**

returns `ProducerConnection`

**Returns:**

<{`ProducerConnection`}>

**Deprecated:**

use `getServletConnection`

**4.4.3.2 int CircularBufferProducer::getRemoteBufferSize () throws RGMAException**

Get the CircularBufferproducer servlet's buffer size

**Deprecated:**

**4.4.3.3 void CircularBufferProducer::setRemoteBufferSize (int remoteBufferSize) throws RGMAException**

Set the CircularBufferproducer servlet's buffer size

**Deprecated:**

#### 4.4.3..4 void `CircularBufferProducer::setTerminationInterval (String intervalString)` throws `RGMAException`

Sets the time interval before which the API must have issued an action to the servlet. Otherwise all knowledge of the API object will be lost. Format = DDDDHHMMSS

**Deprecated:**

use `(TimeInterval interval)`

#### 4.4.4. MEMBER DATA DOCUMENTATION

##### 4.4.4.1 final int `CircularBufferProducer::archive = 1` [static]

Indicates that this is an archive.

**Deprecated:**

### 4.5. CLEANABLE CLASS REFERENCE

Inherits `Insertable`.

Inherited by `DataBaseProducer`, and `LatestProducer`.

#### PUBLIC METHODS

- void `declareTable (String tableName, String predicate, String tableDesc, String cleanUpPredicate, TimeInterval cleanUpInterval)` throws `RGMAException`

#### PROTECTED METHODS

- `Cleanable ()` throws `RGMAException`
- `Cleanable (ServletConnection sc)` throws `RGMAException`

#### 4.5.1. DETAILED DESCRIPTION

A `Cleanable` is a special kind of `Insertable` where a clean up policy can be defined. A clean up policy is defined by an SQL DELETE statement of the users choosing which is run at user selected intervals.

#### 4.5.2. CONSTRUCTOR & DESTRUCTOR DOCUMENTATION

##### 4.5.2.1 `Cleanable::Cleanable ()` throws `RGMAException` [protected]

Default constructor.

##### 4.5.2.2 `Cleanable::Cleanable (ServletConnection sc)` throws `RGMAException` [protected]

Reconnects to an existing `ServletConnection`. It is an error to have more than one API object making use of the same `ServletConnection`.

**Parameters:**

`sc` is the `ServletConnection` which will be reconnected.

### 4.5.3. MEMBER FUNCTION DOCUMENTATION

#### 4.5.3.1 void Cleanable::declareTable (String *tableName*, String *predicate*, String *tableDesc*, String *cleanUpPredicate*, TimeInterval *cleanUpInterval*) throws RGMAException

Declares a table (creating it if necessary) with a specific clean up policy. If the table is already known within the schema, and is inconsistent with *tableDesc*, this will fail.

**Parameters:**

*tableName* is the name of the SQL table

*predicate* an SQL WHERE clause expressing a predicate that is true for the registered table. Currently of the form WHERE (column\_1=value\_1 AND column\_2=value\_2 AND ...)

*tableDesc* is the table description. This may be null. This is the same as the SQL for CREATE TABLE. Some of the columns of the table must be marked PRIMARY KEY.

*cleanUpPredicate* an SQL WHERE clause which is used to define the tuples to be eliminated from the Producer. This could be implemented as a cron job running at an interval defined by the *cleanUpInterval* parameter.

*cleanUpInterval* interval between cleaning up information as defined by the *cleanUpPredicate* parameter..

### 4.6. CONSUMER CLASS REFERENCE

Inherits `APIBase`.

#### PUBLIC METHODS

- `Consumer` (String *selectStatement*, int *queryType*) throws `RGMAException`
- `Consumer` (String *selectStatement*, `ServletConnection` *pc*, int *queryType*) throws `RGMAException`
- `Consumer` (String *selectStatement*, Vector *pcs*, int *queryType*) throws `RGMAException`
- `Consumer` (`ServletConnection` *connection*) throws `RGMAException`
- void `start` () throws `RGMAException`
- void `start` (TimeInterval *interval*) throws `RGMAException`
- void `abort` () throws `RGMAException`
- boolean `hasAborted` () throws `RGMAException`
- boolean `isExecuting` () throws `RGMAException`
- `ResultSet` `blockingPop` () throws `RGMAException`
- `ResultSet` `pop` (int *maxCount*) throws `RGMAException`
- `ResultSet` `popIfPossible` () throws `RGMAException`
- `ResultSet` `popIfPossible` (int *maxCount*) throws `RGMAException`
- `ResultSet` `blockingPop` (int *maxCount*) throws `RGMAException`

- boolean `canPop ()` throws `RGMAException`
- void `setBufferSize (int bufferSize)` throws `RGMAException`
- int `getBufferSize ()` throws `RGMAException`
- int `count ()` throws `RGMAException`
- `ResultSet pop ()` throws `RGMAException`
- void `setTerminationInterval (String intervalString)` throws `RGMAException`
- `Consumer (String selectStatement, ProducerConnection producerConnection)` throws `RGMAException`
- `Consumer (String selectStatement)` throws `RGMAException`
- `Consumer (String selectStatement, Vector producerConnections)` throws `RGMAException`
- `ResultSet execute ()` throws `RGMAException`

#### STATIC PUBLIC ATTRIBUTES

- final int `CONTINUOUS` = 1
- final int `LATEST` = 2
- final int `HISTORY` = 4
- final int `OLD` = 16

#### PROTECTED METHODS

- `Consumer ()` throws `RGMAException`

#### 4.6.1. DETAILED DESCRIPTION

Executes an SQL query to return tuples to the user. Capable of finding one or more producers of information and executing one of the following query types:

**Latest** a query over the latest tuples, where latest is defined as the most recent time for each primary key (excluding the time fields which are present in all primary keys).

**History** a query over all tuples.

**Continuous** a query over a stream of tuples - essentially acting as a filter.

For a continuous query, you will get tuples dated approximately from the current time. It is possible to specify "CONTINUOUS+OLD" to start streaming from the oldest data possible from the selected producer.

Note: this API includes a number of deprecated methods. *These will be removed at the next release.* Instantiation of a `Consumer` with a deprecated constructor does not permit the use of the methods: `start`, `abort`, `isExecuting`, `blockingPop`, `canPop` and `popIfPossible`. An exception will be thrown if these methods are called from an object created with a deprecated constructor.

Objects constructed using the current (non-deprecated) constructors may not use deprecated methods.

## 4.6.2. CONSTRUCTOR & DESTRUCTOR DOCUMENTATION

### 4.6.2.1 Consumer::Consumer () throws **RGMAException** [protected]

Constructor, sets up logging and gets consumerServletLocation from init properties.

### 4.6.2.2 Consumer::Consumer (String *selectStatement*, int *queryType*) throws **RGMAException**

Constructs a Consumer using a String representing the SQL query.

#### Parameters:

*selectStatement* the desired SQL SELECT statement.

*queryType* identifying the kind of query (CONTINUOUS, CONTINUOUS + OLD, LATEST or HISTORY).

### 4.6.2.3 Consumer::Consumer (String *selectStatement*, **ServletConnection** *pc*, int *queryType*) throws **RGMAException**

Constructs a Consumer using a String representing the SQL query and a specified **ServletConnection**. Unlike the other constructors which rely on the **Registry** to find a Producer, this constructor takes a **ServletConnection** as the second argument.

#### Parameters:

*selectStatement* the desired SQL select statement.

*sc* identifies the Producer of the information.

*queryType* identifying the kind of query (CONTINUOUS, LATEST or HISTORY).

### 4.6.2.4 Consumer::Consumer (String *selectStatement*, **Vector** *pcs*, int *queryType*) throws **RGMAException**

Constructs a Consumer using a String representing the SQL query and a vector containing **ServletConnections**. The query will be sent to each Producer and the results returned.

#### Parameters:

*selectStatement* the desired SQL select statement.

*scs* a vector containing **ServletConnections**, which identify which Producers the Consumer should use when answering the query.

*queryType* identifying the kind of query (CONTINUOUS, CONTINUOUS + OLD, LATEST or HISTORY).

### 4.6.2.5 Consumer::Consumer (**ServletConnection** *connection*) throws **RGMAException**

Reconnects to an existing **ServletConnection**. It is an error to have more than one API object making use of the same **ServletConnection**.

#### Parameters:

*consumerConnection* is the consumer servlet connection which will be reconnected.

#### 4.6.2..6 Consumer::Consumer (String *selectStatement*, ProducerConnection *producerConnection*) throws RGMAException

Constructs a consumer using a String representing the SQL query and a specified `ProducerConnection`. This is unlike the other constructor which relies on the registry to find a producer, this constructor takes a `ProducerConnection` as the second argument.

**Parameters:**

*selectStatement* the desired SQL select statement.  
*pc* identifies the Producer of the information.

**Deprecated:**

use `(String selectStatement, ServletConnection sc, int queryType)`

#### 4.6.2..7 Consumer::Consumer (String *selectStatement*) throws RGMAException

Constructs a consumer using a String representing the SQL query.

**Parameters:**

*selectStatement* the desired SQL select statement.

**Deprecated:**

use `(String selectStatement, int queryType)` instead.

#### 4.6.2..8 Consumer::Consumer (String *selectStatement*, Vector *producerConnections*) throws RGMAException

Constructs a consumer using a String representing the SQL query and a vector containing `ProducerConnections`.

**Parameters:**

*selectStatement* the desired SQL select statement.  
*pcs* a vector containing `producerConnections`, which identify which producers the consumer should use when answering the query.

**Deprecated:**

use `(String selectStatement, Vector pcs, int queryType)`.

### 4.6.3. MEMBER FUNCTION DOCUMENTATION

#### 4.6.3..1 void Consumer::abort () throws RGMAException

Abort an executing query and any blocking pop requests

**See also:**

`blockingPop()` , `start()`

**Exceptions:**

**RGMAException** is thrown if `isExecuting()` returns false (an `abort()` can only occur after a `start()`).

#### 4.6.3..2 **ResultSet Consumer::blockingPop (int maxCount) throws RGMAException**

Return up to the next maxCount tuples of information. These tuples of information will no longer be available

**See also:**

`blockingPop()`

**Parameters:**

*maxCount* maximum number of tuples to return

**Exceptions:**

*RGMAException* if unable to contact the ConsumerServlet.

#### 4.6.3..3 **ResultSet Consumer::blockingPop () throws RGMAException**

Returns all the available information. If no data is available, the request will block until such data becomes available.

Once popped, these tuples of information will no longer be available. This operation will only return an empty set if the query has been aborted.

Note, this method should be run in a separate thread to allow the query to be aborted.

**Exceptions:**

*RGMAException* unable to contact the ConsumerServlet.

#### 4.6.3..4 **boolean Consumer::canPop () throws RGMAException**

Returns true if there is a tuple of information available.

**Exceptions:**

*RGMAException* if unable to contact the ConsumerServlet.

#### 4.6.3..5 **int Consumer::count () throws RGMAException**

New API:

Number of available events. Returns the number of pieces of information which can be popped from the ConsumerServlet buffer.

Old API:

Number of available events. Returns the number of pieces of information which can be popped from the consumer servlet buffer. This information should have been streamed in when bufferSize was set to a value > 0. This will lead to an exception if bufferSize =0.

#### 4.6.3..6 **ResultSet Consumer::execute () throws RGMAException**

Execute the Consumer's query to return a `ResultSet`. This method should be used to issue one SQL query to the producer and get the last tuple from the producer servlet buffer. See the introduction to the class.

Deprecated:

#### 4.6.3.7 int Consumer::getBufferSize () throws RGMAException

Get consumer servlet bufferSize.

**Returns:**

the buffer size in use.

**Exceptions:**

*RGMAException* if unable to contact the ConsumerServlet.

#### 4.6.3.8 boolean Consumer::hasAborted () throws RGMAException

Return true if an `abort()` has been issued

**See also:**

`abort()`. This may be due to a `start` which has timed out or a call to `abort`

**Exceptions:**

*RGMAException* if unable to contact the ConsumerServlet.

#### 4.6.3.9 boolean Consumer::isExecuting () throws RGMAException

Returns the execution status. A query is said to be executing after it has been started

**See also:**

`start`.

**Returns:**

true while a query is executing, false is returned once `start()` completes or when an `abort()` is issued

**See also:**

`abort()`, `start()`.

#### 4.6.3.10 ResultSet Consumer::pop () throws RGMAException

New API:

Return all the available pieces of information. These pieces of information will no longer be available.

*If you have instantiated with a deprecated constructor, then this will only return the oldest piece of information from the consumer servlet queue (buffer) and remove it.*

**Exceptions:**

*RGMAException* there must be some tuples to return.

Old API:

Return the oldest piece of information from the consumer servlet queue (buffer) and remove it.

This information should have been streamed in when bufferSize was set to a value > 0. This will lead to an exception if bufferSize = 0. **The schema of the XML will almost certainly be changed.**

#### 4.6.3.11 **ResultSet Consumer::pop (int maxCount) throws RGMAException**

Return up to the next maxCount tuples of information. These tuples of information will no longer be available.

**Parameters:**

*maxCount* maximum number of tuples to return.

**Exceptions:**

*RGMAException* if there are no tuples to pop.

#### 4.6.3.12 **ResultSet Consumer::popIfPossible (int maxCount) throws RGMAException**

Return up to the next maxCount tuples of information. These tuples of information will no longer be available. If there is no data null is returned.

**Parameters:**

*maxCount* maximum number of tuples to return.

**Exceptions:**

*RGMAException* if unable to contact the ConsumerServlet.

#### 4.6.3.13 **ResultSet Consumer::popIfPossible () throws RGMAException**

Return all the available tuples of information. These tuples of information will no longer be available. If there no tuples exist, null is returned.

**Exceptions:**

*RGMAException* if unable to contact the ConsumerServlet

#### 4.6.3.14 **void Consumer::setBufferSize (int bufferSize) throws RGMAException**

Set consumer servlet buffersize.

New API:

Sets the buffer size used to hold ResultSets of information. The buffer is maintained by the ConsumerServlet. By default, the buffer size is 0 so no ResultSets will be stored. Make sure this is called prior to `start()` to ensure ResultSets are stored as expected.

Note, the `pop()` methods extract tuples, however, the buffer contains ResultSets (which encapsulate tuples). The buffer size is set in ResultSets so may conceivably contain many tuples but few ResultSets.

Old API:

If this is greater than zero it allows information to be streamed from the producer servlet buffer to the consumer servlet buffer as it becomes available. In order to stop this streaming, `bufferSize` should be reset to zero.

**Parameters:**

*bufferSize* - desired buffer size measured as number of events.

**Exceptions:**

*RGMAException* if unable to contact the ConsumerServlet.

**4.6.3..15 void Consumer::setTerminationInterval (String intervalString) throws RGMAException**

Sets the time interval before which the API must have issued an action to the servlet. Otherwise all knowledge of the API object will be lost. Format = DDDDHHMMSS

**Deprecated:**

use `(TimeInterval interval)`

**4.6.3..16 void Consumer::start (TimeInterval interval) throws RGMAException**

Start executing the Consumer's query using a time limit. The operation will be aborted after the specified time interval. After aborting `hasAborted` will return true.

This method is particularly suited for streaming (ie Consumer's with a CONTINUOUS or CONTINUOUS + OLD queryType).

**Parameters:**

*interval* maximum time to allow for the operation before it aborts.

**Exceptions:**

*RGMAException* is thrown if `start(interval)` is called when `isExecuting()` is true (cannot duplicate requests). The `TimeInterval` must also be  $> 0$ .

**4.6.3..17 void Consumer::start () throws RGMAException**

Start executing the Consumer's query. If using CONTINUOUS or CONTINUOUS + OLD queryTypes, the `start()` will initiate streaming with each Producer; to stop streaming, use `abort()`. If using LATEST or HISTORY queryTypes, `start()` will send the Consumer query to each Producer for executing, to stop executing,

**See also:**

`abort()`.

#### Exceptions:

*RGMAException* is thrown if `start()` is called when `isExecuting()` is true (cannot duplicate requests).

#### 4.6.4. MEMBER DATA DOCUMENTATION

##### 4.6.4.1 `final int Consumer::CONTINUOUS = 1` [static]

A continuous query.

##### 4.6.4.2 `final int Consumer::HISTORY = 4` [static]

A history query.

##### 4.6.4.3 `final int Consumer::LATEST = 2` [static]

A latest query.

##### 4.6.4.4 `final int Consumer::OLD = 16` [static]

A query type modifier for old `data` retained by a `StreamProducer`. To use it as an argument specify: `CONTINUOUS+OLD`.

#### 4.7. DATABASEPRODUCER CLASS REFERENCE

Inherits `Cleanable`.

#### PUBLIC METHODS

- `DataBaseProducer` (`String` rdbms, `String` user, `String` password) throws `RGMAException`
- `DataBaseProducer` (`ServletConnection` connection) throws `RGMAException`
- `void add` (`String` tableName, `String` predicate, `int` flags, `String` tableDesc) throws `RGMAException`
- `void add` (`String` tableName, `String` predicate, `int` flags) throws `RGMAException`
- `void setTerminationInterval` (`String` intervalString) throws `RGMAException`
- `ProducerConnection` `getProducerConnection` () throws `RGMAException`

#### STATIC PUBLIC ATTRIBUTES

- `final int archive = 1`

#### PROTECTED METHODS

- `DataBaseProducer` () throws `RGMAException`

#### 4.7.1. DETAILED DESCRIPTION

DataBaseProducer is able to answer historical queries. Information published via a DataBaseProducer is persistent. A clean up policy may be implemented for individual tables: for example to remove old information. It is possible to connect to an existing DataBase and thereby make that data available to a Consumer. If no DataBase is specified, a private RDBMS is created for the lifetime of the DataBaseProducer. .

#### 4.7.2. CONSTRUCTOR & DESTRUCTOR DOCUMENTATION

##### 4.7.2.1 DataBaseProducer::DataBaseProducer () throws **RGMAException** [protected]

##### 4.7.2.2 DataBaseProducer::DataBaseProducer (String *rdbms*, String *user*, String *password*) throws **RGMAException**

Construct a DataBaseProducer.publishing to the users Virtual Organisation (VO) and using a specified RDBMS.

**Parameters:**

*rdbms* JDBC RDBMS identification

*user* RDBMS user name

*password* RDBMS user password

**Exceptions:**

***RGMAException***

##### 4.7.2.3 DataBaseProducer::DataBaseProducer (**ServletConnection** *connection*) throws **RGMAException**

Reconnects to an existing **ServletConnection**. It is an error to have more than one API object making use of the same **ServletConnection**.

**Parameters:**

*sc* is the DataBaseProducer servlet connection which will be reconnected

#### 4.7.3. MEMBER FUNCTION DOCUMENTATION

##### 4.7.3.1 void DataBaseProducer::add (String *tableName*, String *predicate*, int *flags*) throws **RGMAException**

For this method the *tableName* must already be known within the schema

**Parameters:**

*tableName* is the name of the SQL table

*predicate* an SQL WHERE clause expressing a predicate that is true for the registered table. Currently of the form WHERE (column\_1=value\_1 AND column\_2=value\_2 AND ...)

*flags* is a set of boolean flags encoded in an integer

#### 4.7.3.2 void DataBaseProducer::add (String *tableName*, String *predicate*, int *flags*, String *tableDesc*) throws RGMAException

If the table is already known within the schema and is inconsistent with the *tableDesc* this will fail.

**Parameters:**

*tableName* is the name of the SQL table

*predicate* an SQL WHERE clause expressing a predicate that is true for the registered table. Currently of the form WHERE (column\_1=value\_1 AND column\_2=value\_2 AND ...)

*flags* RGMA ignores this.

*tableDesc* is the table description. This is the same as the SQL for CREATE TABLE. Some of the columns of the table must be marked PRIMARY KEY

**Exceptions:**

**RGMAException** if the object has been disconnected or is closed, or if there are problems declaring the table.

**Deprecated:**

use (String, String, String)

#### 4.7.3.3 ProducerConnection DataBaseProducer::getProducerConnection () throws RGMAException

returns `ProducerConnection`

**Returns:**

<{`ProducerConnection`}>

**Deprecated:**

use `getServletConnection`

#### 4.7.3.4 void DataBaseProducer::setTerminationInterval (String *intervalString*) throws RGMAException

Sets the time interval before which the API must have issued an action to the servlet. Otherwise all knowledge of the API object will be lost. Format = DDDDHHMMSS

**Deprecated:**

use `APIBase::setTerminationInterval`

### 4.7.4. MEMBER DATA DOCUMENTATION

#### 4.7.4.1 final int DataBaseProducer::archive = 1 [static]

Indicates that this is an archive.

**Deprecated:**

## 4.8. DECLARABLE CLASS REFERENCE

Inherits `APIBase`.

Inherited by `Archiver`, `CanonicalProducer`, and `Insertable`.

### PUBLIC METHODS

- void `declareTable` (`String tableName`, `String predicate`, `String tableDesc`) throws `RGMAException`
- void `declareTable` (`String tableName`, `String predicate`) throws `RGMAException`
- void `undeclareTable` (`String tableName`) throws `RGMAException`

### PROTECTED METHODS

- `Declarable` () throws `RGMAException`

#### 4.8.1. DETAILED DESCRIPTION

A `Declarable` is the R-GMA component which makes information available to a `Consumer`. The `Declarable` is in two parts, a client and a server. The server can be instructed to carry on without the client.

#### 4.8.2. CONSTRUCTOR & DESTRUCTOR DOCUMENTATION

**4.8.2..1 `Declarable::Declarable` () throws `RGMAException` [protected]**

#### 4.8.3. MEMBER FUNCTION DOCUMENTATION

**4.8.3..1 void `Declarable::declareTable` (`String tableName`, `String predicate`) throws `RGMAException`**

declares a table. For this method the `tableName` must already be known within the schema. This method will normally only be used for tables which are well known.

#### Parameters:

*tableName* is the name of the SQL table

*predicate* an SQL WHERE clause expressing a predicate that is true for the registered table. Currently of the form WHERE (column\_1=value\_1 AND column\_2=value\_2 AND ...)

#### Exceptions:

**`RGMAException`** if the object has been disconnected or is closed, or if there are problems declaring the table.

#### 4.8.3.2 void Declarable::declareTable (String *tableName*, String *predicate*, String *tableDesc*) throws **RGMAException**

declares a table - creating it if necessary. If the table is already known within the schema and is inconsistent with the *tableDesc* this will fail.

##### Parameters:

*tableName* is the name of the SQL table

*predicate* an SQL WHERE clause expressing a predicate that is true for the registered table. Currently of the form WHERE (column\_1=value\_1 AND column\_2=value\_2 AND ...)

*tableDesc* is the table description. This is the same as the SQL for CREATE TABLE. Some of the columns of the table must be marked PRIMARY KEY

##### Exceptions:

**RGMAException** if the object has been disconnected or is closed, or if there are problems declaring the table.

#### 4.8.3.3 void Declarable::undeclareTable (String *tableName*) throws **RGMAException**

Undeclares table of given name.

##### Parameters:

*tableName* is the name of the SQL table.

##### Exceptions:

**RGMAException** if the object has been disconnected or is closed, or if there are problems undeclaring the table.

## 4.9. FIXEDPOINT CLASS REFERENCE

### PUBLIC METHODS

- `FixedPoint` (long units, long fraction)
- `FixedPoint` (double value)
- `FixedPoint` (String s)
- `FixedPoint` ()
- `FixedPoint` (FixedPoint other)
- `FixedPoint add` (FixedPoint other)
- `FixedPoint add` (long value)
- `FixedPoint subtract` (FixedPoint other)
- `FixedPoint subtract` (long value)
- `FixedPoint multiply` (int value)
- `FixedPoint divide` (int value)
- `int compareTo` (FixedPoint other)
- `int compareTo` (long value)
- `boolean equals` (Object other)
- `boolean equals` (Object other, FixedPoint delta)
- `double doubleValue` ()

- float floatValue ()
- long longValue ()
- int intValue ()
- short shortValue ()
- byte byteValue ()
- long getUnits ()
- long getFraction ()
- long getFraction (FixedPoint precision)
- double getFractionAsDouble ()
- String toString ()
- String toString (int minUnitsDigits)

#### 4.9.1. DETAILED DESCRIPTION

An unsigned fixed point number with resolution of  $10^{-18}$ .  
This is intended for use as a representation of `Time`.

#### 4.9.2. CONSTRUCTOR & DESTRUCTOR DOCUMENTATION

##### 4.9.2.1 FixedPoint::FixedPoint (long units, long fraction)

Construct a FixedPoint number.

**Parameters:**

*units* Integer part of the number.

*fraction* Fractional part of the number expressed as units of  $10^{-18}$ .

##### 4.9.2.2 FixedPoint::FixedPoint (double value)

Construct a FixedPoint number from a floating point number. Since the floating point number is only an approximation of a decimal fraction this constructor should not be used where accuracy is important.

##### 4.9.2.3 FixedPoint::FixedPoint (String s)

Construct a FixedPoint number from a decimal string representation.

**Parameters:**

*s* A string of the form "d\*(\d\*)?".

##### 4.9.2.4 FixedPoint::FixedPoint ()

Construct a FixedPoint number with value 0.0.

##### 4.9.2.5 FixedPoint::FixedPoint (FixedPoint other)

Construct a FixedPoint number with the value of another.

### 4.9.3. MEMBER FUNCTION DOCUMENTATION

#### 4.9.3.1 FixedPoint FixedPoint::add (long value)

Add a value to this FixedPoint number.

**Returns:**

The result of the addition as a new FixedPoint number.

#### 4.9.3.2 FixedPoint FixedPoint::add (FixedPoint other)

Add this FixedPoint number to another.

**Returns:**

The result of the addition as a new FixedPoint number.

#### 4.9.3.3 byte FixedPoint::byteValue ()

Return the value of this FixedPoint number as an byte.

#### 4.9.3.4 int FixedPoint::compareTo (long value)

Compare this FixedPoint number to a long integer value.

**Returns:**

a value less than 0, if this is less than value; a value greater than 0, if this is greater than value; 0 if this is equal to value.

#### 4.9.3.5 int FixedPoint::compareTo (FixedPoint other)

Compare this FixedPoint number to another.

**Returns:**

a value less than 0, if this is less than other; a value greater than 0, if this is greater than other; 0 if this is equal to other.

#### 4.9.3.6 FixedPoint FixedPoint::divide (int value)

Divide this FixedPoint number by a scalar value.

**Returns:**

The result of the division as a new FixedPoint number.

#### 4.9.3.7 double FixedPoint::doubleValue ()

Approximate the value of this FixedPoint number with a double.

#### 4.9.3.8 boolean FixedPoint::equals (Object other, FixedPoint delta)

Test equality of this FixedPoint number and another object, given a margin of error.

#### 4.9.3.9 boolean FixedPoint::equals (Object *other*)

Test equality of this FixedPoint number and another object.

#### 4.9.3.10 float FixedPoint::floatValue ()

Approximate the value of this FixedPoint number with a float.

#### 4.9.3.11 long FixedPoint::getFraction (FixedPoint *precision*)

Return the fractional part of this FixedPoint number as an integral number of units of  $10^{-18}$ , specifying the precision desired.

#### 4.9.3.12 long FixedPoint::getFraction ()

Return the fractional part of this FixedPoint number as an integral number of units of  $10^{-18}$ .

#### 4.9.3.13 double FixedPoint::getFractionAsDouble ()

#### 4.9.3.14 long FixedPoint::getUnits ()

Return the units part of this FixedPoint number.

#### 4.9.3.15 int FixedPoint::intValue ()

Return the value of this FixedPoint number as an int.

#### 4.9.3.16 long FixedPoint::longValue ()

Truncate the value of this FixedPoint number to a long.

#### 4.9.3.17 FixedPoint FixedPoint::multiply (int *value*)

Multiply this FixedPoint number by a scalar value.

**Returns:**

The result of the multiplication as a new FixedPoint number.

#### 4.9.3.18 short FixedPoint::shortValue ()

Return the value of this FixedPoint number as an short.

#### 4.9.3.19 FixedPoint FixedPoint::subtract (long *value*)

Subtract a value from this FixedPoint number. If this number is less than the other, the result is 0.0.

**Returns:**

The result of the subtraction as a new FixedPoint number.

#### 4.9.3..20 FixedPoint FixedPoint::subtract (FixedPoint *other*)

Subtract another FixedPoint number from this one. If this number is less than the other, the result is 0.0.

**Returns:**

The result of the subtraction as a new FixedPoint number.

#### 4.9.3..21 String FixedPoint::toString (int *minUnitsDigits*)

Produce a string representation, specifying the minimum number of digits before the decimal point.

**Parameters:**

*minUnitsDigits* Minimum number of digits before the decimal point.

**Returns:**

A string representation in the form "d\*\d+".

#### 4.9.3..22 String FixedPoint::toString ()

Produce a string representation.

**Returns:**

A string representation in the form "d+\d+".

### 4.10. INSERTABLE CLASS REFERENCE

Inherits [Declarable](#).

Inherited by [CircularBufferProducer](#), [Cleanable](#), and [StreamProducer](#).

#### PUBLIC METHODS

- [Insertable](#) ([ServletConnection](#) pc) throws [RGMAException](#)
- boolean [getAutoInsertTimestamp](#) () throws [RGMAException](#)
- void [insert](#) (String sqlInsert) throws [RGMAException](#)
- void [insert](#) (Vector sqlInserts) throws [RGMAException](#)
- void [setAutoInsertTimestamp](#) (boolean ait) throws [RGMAException](#)

#### PROTECTED METHODS

- [Insertable](#) () throws [RGMAException](#)

#### 4.10.1. CONSTRUCTOR & DESTRUCTOR DOCUMENTATION

4.10.1..1 [Insertable::Insertable](#) () throws [RGMAException](#) [protected]

#### 4.10.1.2 Insertable::Insertable (ServletConnection *pc*) throws RGMAException

Reconnects to an existing `ServletConnection`. It is an error to have more than one API object making use of the same `ServletConnection`.

**Parameters:**

*pc* is the consumer servlet connection which will be reconnected

#### 4.10.2. MEMBER FUNCTION DOCUMENTATION

##### 4.10.2.1 boolean Insertable::getAutoInsertTimestamp () throws RGMAException

Returns value of `AutoInsertTimestamp`. By default, this flag is true, and so `MeasurementDate` and `MeasurementTime` fields will be added to tuples if they are not already set.

**Returns:**

true if `AutoInsertTimestamp` is on

**Exceptions:**

*RGMAException* if the object has been closed or disconnected

**See also:**

`setTupleChecking`

##### 4.10.2.2 void Insertable::insert (Vector *sqlInserts*) throws RGMAException

If the timestamp field is null, the Producer will generate a time stamp for you.

**Parameters:**

*rows* Vector of Strings of SQL insert statement

##### 4.10.2.3 void Insertable::insert (String *sqlInsert*) throws RGMAException

If the timestamp field is null, the Producer will generate a time stamp for you.

**Parameters:**

*sqlInsert* an SQL insert statement

##### 4.10.2.4 void Insertable::setAutoInsertTimestamp (boolean *ait*) throws RGMAException

Set value of `AutoInsertTimestamp`. By default this is set to true.

**Parameters:**

*tc* true to enable automatic insertion of timestamp (`MeasurementDate` & `Time`)

**Exceptions:**

*RGMAException*

**See also:**

`getTupleChecking`

#### 4.11. LATESTPRODUCER CLASS REFERENCE

Inherits `Cleanable`.

##### PUBLIC METHODS

- `LatestProducer (ServletConnection connection)` throws `RGMAException`
- `LatestProducer ()` throws `RGMAException`

##### 4.11.1. DETAILED DESCRIPTION

A `LatestProducer` is a `Publisher` which is able to answer "latest snapshot" queries. A latest snapshot query can only access the most recent tuples for a given primary key (except for the time stamp) When the `SnapshotProducer` is no longer needed, it's `disconnect` method should be called, ideally from a finally block. This ensures that the registry is updated accordingly, and that resources associated with the `SnapshotProducer`, such as database connections, are released.

##### 4.11.2. CONSTRUCTOR & DESTRUCTOR DOCUMENTATION

###### 4.11.2.1 `LatestProducer::LatestProducer (ServletConnection connection)` throws `RGMAException`

Reconnects to an existing `ServletConnection`. It is an error to have more than one API object making use of the same `ServletConnection`.

##### Parameters:

*pc* is the `SnapshotProducer` servlet connection which will be reconnected

###### 4.11.2.2 `LatestProducer::LatestProducer ()` throws `RGMAException`

#### 4.12. NETLOGGERFACTORY CLASS REFERENCE

#### 4.13. PRODUCERCONNECTION CLASS REFERENCE

##### PUBLIC METHODS

- `ProducerConnection (String producerServlet, int connectionId)`
- `String getProducerServlet ()`
- `int getConnectionId ()`
- `String toString ()`

##### 4.13.1. DETAILED DESCRIPTION

Identifies a connection to a producer servlet by a producer.

##### Deprecated:

use `ServletConnection`

#### 4.13.2. CONSTRUCTOR & DESTRUCTOR DOCUMENTATION

##### 4.13.2..1 `ProducerConnection::ProducerConnection (String producerServlet, int connectionId)`

Construct a `ProducerConnection` using a string representing the producer URL and an integer representing the producer's unique connection identifier.

**Parameters:**

- producerServlet* the URL of the producer servlet.
- connectionId* identifies the specific producer for the desired connection.

#### 4.13.3. MEMBER FUNCTION DOCUMENTATION

##### 4.13.3..1 `int ProducerConnection::getConnectionId ()`

get producer connection identifier.

##### 4.13.3..2 `String ProducerConnection::getProducerServlet ()`

get producer servlet URL as a String.

##### 4.13.3..3 `String ProducerConnection::toString ()`

#### 4.14. PROPERTYGETTER CLASS REFERENCE

##### PUBLIC METHODS

- `PropertyGetter (String name)` throws `RGMAException`
- `Properties` `getProperties ()`

##### 4.14.1. DETAILED DESCRIPTION

Obtains properties from a set of locations. Currently the only location considered is \$HOME

#### 4.14.2. CONSTRUCTOR & DESTRUCTOR DOCUMENTATION

##### 4.14.2..1 `PropertyGetter::PropertyGetter (String name)` throws `RGMAException`

#### 4.14.3. MEMBER FUNCTION DOCUMENTATION

##### 4.14.3..1 `Properties` `PropertyGetter::getProperties ()`

#### 4.15. REGISTRY CLASS REFERENCE

##### PUBLIC METHODS

- `Registry (String servletURL)` throws `RGMAException`

- Vector `getProducerConnections` (String `tableName`, String `predicate`, int `flags`) throws `RGMAException`
- String `getProducerInfo` (`ProducerConnection pc`) throws `RGMAException`
- String `getRegistryServletLocation` ()
- String `getStatus` () throws `RGMAException`

#### 4.15.1. DETAILED DESCRIPTION

Registry of producers. Currently registry has no state, but when we address multiples VOs this is expected to change.

#### 4.15.2. CONSTRUCTOR & DESTRUCTOR DOCUMENTATION

##### 4.15.2..1 Registry::Registry (String *servletURL*) throws **RGMAException**

Creates a registry object that is used to register producers.

**Parameters:**

*location* the URL of the registry servlet that deals with request of this registry object.

XXX: Once replication has been implemented, we'll probably want to avoid using this constructor as all registry URL's will be defined in a `registryconfig.xml` file.

**Exceptions:**

**RGMAException** if the URL specified in the location is bad.

#### 4.15.3. MEMBER FUNCTION DOCUMENTATION

##### 4.15.3..1 Vector Registry::getProducerConnections (String *tableName*, String *predicate*, int *flags*) throws **RGMAException**

Discover producers from registry based on the table name, the predicate and flags

**Parameters:**

*tableName* is the name of the SQL table

*predicate* an SQL WHERE clause expressing a predicate that is true for the registered table. Currently of the form WHERE (column\_1=value\_1 AND column\_2=value\_2 AND ...)

*flags* is a set of boolean flags encoded in an integer

**Returns:**

Vector of `ProducerConnections`

#### 4.15.3.2 String Registry::getProducerInfo (ProducerConnection pc) throws RGMAException

Look up ProducerInfo of a producer identified by a ProducerConnections pc. It will return null if the ProducerConnection is not known.

**Parameters:**

*pc* ProducerConnection

**Returns:**

ProducerInfo CURRENTLY STILL A STRING !!!

#### 4.15.3.3 String Registry::getRegistryServletLocation ()

Returns the URL of the RegistryServlet.

**Returns:**

the URL of the RegistryServlet.

#### 4.15.3.4 String Registry::getStatus () throws RGMAException

Get status information of the registry

**Returns:**

status information as a String

### 4.16. REPLICACONFIGREADER CLASS REFERENCE

#### PUBLIC METHODS

- ReplicaConfigReader (int replicaType)
- Vector getURLs ()
- long getReplicationPeriod ()
- boolean isResourceAvailable ()

#### STATIC PUBLIC ATTRIBUTES

- final int REGISTRY\_CONFIG = 0
- final int SCHEMA\_CONFIG = 1

#### PROTECTED ATTRIBUTES

- Category cat

#### 4.16.1. DETAILED DESCRIPTION

The ReplicaConfigReader will read-in configuration parameters from a resource (in this case an XML file). A list of back-up Servlet URL's are read-in from the resource. Clients to this API can use this list to obtain a valid URL. This is necessary when Registry and Schema replication is enabled as new clients will require a working URL if one or more URL's are found to be dud (i.e the Service is no longer active).

#### 4.16.2. CONSTRUCTOR & DESTRUCTOR DOCUMENTATION

##### 4.16.2.1 ReplicaConfigReader::ReplicaConfigReader (int replicaType)

Method ReplicaConfigReader() will setup various params for the config reader. Things to note, the registryconfig.xml file and schemaconfig.xml are found relative to the \$RGMA\_PROPS location. Unless this property is set, the config reader will fail to find this file.

**Parameters:**

*replicaType* For Registry related use, set to ReplicatConfigReader.REGISTRY\_CONFIG. For Schema related use, set to ReplicatConfigReader.SCHEMA\_CONFIG.

XXX Constructor should throw an exception if config file cannot be found. Until replication is fully tested ignore for the time being.

**Exceptions:**

*RGMAException*

#### 4.16.3. MEMBER FUNCTION DOCUMENTATION

##### 4.16.3.1 long ReplicaConfigReader::getReplicationPeriod ()

Method getReplicationPeriod()

**Returns:**

long

##### 4.16.3.2 Vector ReplicaConfigReader::getURLs ()

Method getURLs()

**Returns:**

Vector

##### 4.16.3.3 boolean ReplicaConfigReader::isResourceAvailable ()

Method isResourceAvailable() will check if the resource is available (i.e can be located without any i/o problems).

**Returns:**

boolean

#### 4.16.4. MEMBER DATA DOCUMENTATION

##### 4.16.4.1 `Category ReplicaConfigReader::cat` [protected]

Initialise a Category for log4j logging

##### 4.16.4.2 `final int ReplicaConfigReader::REGISTRY_CONFIG = 0` [static]

Constant to indicate reading from the RegistryConfig file

##### 4.16.4.3 `final int ReplicaConfigReader::SCHEMA_CONFIG = 1` [static]

Constant to indicate reading from the SchemaConfig file

#### 4.17. RESILIENTSTREAMPRODUCER CLASS REFERENCE

Inherits `StreamProducer`.

##### PUBLIC METHODS

- `ResilientStreamProducer ()` throws `RGMAException`
- `ResilientStreamProducer (ServletConnection connection)` throws `RGMAException`

##### PROTECTED METHODS

- `void init ()` throws `RGMAException`

##### 4.17.1. DETAILED DESCRIPTION

`StreamProducers` is able to register a table when it is created and subsequently to publish information.

##### 4.17.2. CONSTRUCTOR & DESTRUCTOR DOCUMENTATION

###### 4.17.2.1 `ResilientStreamProducer::ResilientStreamProducer ()` throws `RGMAException`

default constructor for setting instance variables

##### Exceptions:

*`RGMAException`*

###### 4.17.2.2 `ResilientStreamProducer::ResilientStreamProducer (ServletConnection connection)` throws `RGMAException`

Reconnects to an existing `ServletConnection`. It is an error to have more than one API object making use of the same `ServletConnection`.

##### Parameters:

`sc` is the servlet connection which will be reconnected

**Exceptions:**

*RGMAException* if another API object is already connected;

**4.17.3. MEMBER FUNCTION DOCUMENTATION**

**4.17.3.1 void ResilientStreamProducer::init () throws RGMAException [protected]**

Create a ResilientStreamProducer publishing to a Vector of Virtual Organisations (VOs) which can have the NON\_PRIMARY flag set.

**Parameters:**

*vos* is a vector of Virtual Organisations to which the Producer should be registered.

*flags* the flag NON\_PRIMARY can be set to indicate that this is not a primary producer.

XXX: to do!

```
public ResilientStreamProducer(Vector vos, int flags) throws RGMAException { }  
/** Construct a StreamProducer.publishing to a Vector of VOs.
```

**Parameters:**

*vos* is a vector of virtual organisations to which the Producer should be registered.

XXX-AC: not yet implemented VO functionality!

```
public ResilientStreamProducer(Vector vos) throws RGMAException { this();  
cat.info("entering constructor"); ServletConnection connection = new Servlet-  
Connection(servletURL); connection.addParameter("terminationInterval", DEFAULT_-  
TERMINATION_INTERVAL); ResultSet result = sendCommand("createInstance",  
connection); result.next(); connectionId = result.getInt("connectionId");  
cat.info("exiting constructor"); }
```

Reimplemented from [StreamProducer](#).

**4.18. RESULTSET CLASS REFERENCE**

**PUBLIC METHODS**

- String [getString](#) (int columnNumber) throws RGMAException
- String [getString](#) (String columnName) throws RGMAException
- boolean [getBoolean](#) (int columnNumber) throws RGMAException
- boolean [getBoolean](#) (String columnName) throws RGMAException
- int [getInt](#) (int columnNumber) throws RGMAException
- int [getInt](#) (String columnName) throws RGMAException
- boolean [next](#) ()
- boolean [previous](#) ()
- void [beforeFirst](#) ()
- void [afterLast](#) ()
- [ResultSetMetaData](#) [getMetaData](#) ()
- int [size](#) ()
- String [toString](#) ()

#### 4.18.1. DETAILED DESCRIPTION

R-GMA implementation of a ResultSet. The Class implements a subset of the methods of the java.sql.ResultSet class.

#### 4.18.2. MEMBER FUNCTION DOCUMENTATION

##### 4.18.2..1 void ResultSet::afterLast ()

Positions row cursor after last row.

##### 4.18.2..2 void ResultSet::beforeFirst ()

Positions row cursor before first row.

##### 4.18.2..3 boolean ResultSet::getBoolean (String *columnName*) throws **RGMAException**

##### 4.18.2..4 boolean ResultSet::getBoolean (int *columnNumber*) throws **RGMAException**

##### 4.18.2..5 int ResultSet::getInt (String *columnName*) throws **RGMAException**

##### 4.18.2..6 int ResultSet::getInt (int *columnNumber*) throws **RGMAException**

##### 4.18.2..7 **ResultSetMetaData** ResultSet::getMetaData ()

##### 4.18.2..8 String ResultSet::getString (String *columnName*) throws **RGMAException**

##### 4.18.2..9 String ResultSet::getString (int *columnNumber*) throws **RGMAException**

##### 4.18.2..10 boolean ResultSet::next ()

Increments row pointer. If pointer is beyond last element, pointer is reset to one place beyond last element.

**Returns:**

True if a new row of `data` is available.

##### 4.18.2..11 boolean ResultSet::previous ()

Decrements row pointer. If pointer is before first element, pointer is reset to one place before first element.

**Returns:**

True if a new row of `data` is available.



#### 4.20.1. MEMBER FUNCTION DOCUMENTATION

4.20.1.1 void `RGMAErrorHandler::error` (`SAXParseException e`) throws `SAXException`

4.20.1.2 void `RGMAErrorHandler::fatalError` (`SAXParseException e`) throws `SAXException`

4.20.1.3 void `RGMAErrorHandler::warning` (`SAXParseException w`)

#### 4.21. RGMAEXCEPTION CLASS REFERENCE

##### PUBLIC METHODS

- `RGMAException` (`String msg`)
- `RGMAException` (`String msg`, `String type`, `String source`)
- `RGMAException` (`String msg`, `String type`, `String source`, `boolean recoverable`)
- `RGMAException` (`String msg`, `Exception e`, `String type`, `String source`)
- `RGMAException` (`String msg`, `Exception e`, `String type`, `String source`, `boolean recoverable`)
- `String getType` ()
- `String getSource` ()
- `boolean isRecoverable` ()

##### 4.21.1. CONSTRUCTOR & DESTRUCTOR DOCUMENTATION

###### 4.21.1.1 `RGMAException::RGMAException` (`String msg`)

class to handle Exceptions within R-GMA.

**Parameters:**

*msg* is the error message of the Exception

###### 4.21.1.2 `RGMAException::RGMAException` (`String msg`, `String type`, `String source`)

class to handle Exceptions within R-GMA.

**Parameters:**

*msg* is the error message of the Exception

*type* is the R-GMA specific type of the Exception

*source* is either "api" or "servlet"

###### 4.21.1.3 `RGMAException::RGMAException` (`String msg`, `String type`, `String source`, `boolean recoverable`)

**Parameters:**

*msg* is the error message of the Exception

*type* is the R-GMA specific type of the Exception

*source* is either "api" or "servlet"

*recoverable* recoverability flag, default is false if not given

#### 4.21.1.4 RGMAException::RGMAException (String *msg*, Exception *e*, String *type*, String *source*)

**Parameters:**

- msg* is the error message of the Exception
- type* is the R-GMA specific type of the Exception
- source* is either "api" or "servlet"
- e* An underlying RGMAException that caused this new Exception

#### 4.21.1.5 RGMAException::RGMAException (String *msg*, Exception *e*, String *type*, String *source*, boolean *recoverable*)

**Parameters:**

- msg* is the error message of the Exception
- e* An underlying Exception that caused this new Exception
- recoverable* recoverability flag, default is false if not given

### 4.21.2. MEMBER FUNCTION DOCUMENTATION

#### 4.21.2.1 String RGMAException::getSource ()

The source at which the error occurred. This is API/Servlet or an other R-GMA component.

#### 4.21.2.2 String RGMAException::getType ()

Getter for Exception type.

#### 4.21.2.3 boolean RGMAException::isRecoverable ()

Returns true if this exception is recoverable. (e.g. only wrong user data)

## 4.22. SCHEMA CLASS REFERENCE

### PUBLIC METHODS

- Schema (String *servletURL*) throws RGMAException
- void *createTable* (String *createStatement*) throws RGMAException
- ResultSet *getAllSchemaTables* () throws RGMAException
- String[] *getColumnNames* (int[] *columnIds*) throws RGMAException
- String[] *getColumnTypes* (int[] *columnIds*) throws RGMAException
- Vector *getPrimaryKey* (String *tableName*) throws RGMAException
- String *getStatus* () throws RGMAException
- String *getTableDesc* (String *tableName*) throws RGMAException
- ResultSet *getTableInfo* (String *tableId*) throws RGMAException
- int[] *translateColumnNames* (String *tableId*, String[] *fixedColumns*) throws RGMAException
- String *translateTableName* (String *tableName*) throws RGMAException

## PROTECTED METHODS

- `Schema ()`
- `ServletConnection getServletConnection ()` throws `RGMAException`
- `ResultSet sendCommand (String command, ServletConnection connection)` throws `RGMAException`

## PROTECTED ATTRIBUTES

- String `servletURL`

### 4.22.1. DETAILED DESCRIPTION

Schema of Tables. This object is used to access a Schema Servlet that holds the schemas for tables that can be registered with a registry. A `Registry` is closely associated with a Schema as the former uses the latter. Having a `Registry` and Schema separates the registration and description of tables.

### 4.22.2. CONSTRUCTOR & DESTRUCTOR DOCUMENTATION

#### 4.22.2.1 `Schema::Schema ()` [protected]

#### 4.22.2.2 `Schema::Schema (String servletURL)` throws `RGMAException`

Creates a schema object that is used to hold descriptions of tables.

#### Parameters:

*servletURL* is the URL of the schema servlet that deals with requests for this schema.

#### Exceptions:

`MalformedURLException` if the URL specified in the location is bad.

### 4.22.3. MEMBER FUNCTION DOCUMENTATION

#### 4.22.3.1 `void Schema::createTable (String createStatement)` throws `RGMAException`

Adds a table to the schema based on the provided CREATE TABLE statement.

#### Parameters:

*createStatement* SQL CREATE TABLE statement to define the table.

#### 4.22.3.2 `ResultSet Schema::getAllSchemaTables ()` throws `RGMAException`

Returns the names and identifiers of all tables in the global schema.

#### Returns:

A `ResultSet` containing the columns (int tableId, String name)

#### 4.22.3.3 String [] Schema::getColumnNames (int columnIds[]) throws RGMAException

Method will retrieve the column name for each column ID specified within the columnIds[]. A list of column names are returned which map one-to-one with the columnIds.

**Parameters:**

*columnIds* the column IDs to lookup.

**Returns:**

A list of column names. Each column name matches the corresponding columnId in the same order.

#### 4.22.3.4 String [] Schema::getColumnTypes (int columnIds[]) throws RGMAException

Gets the column types for each column ID in an array

**Parameters:**

*an* int array of column IDs.

**Returns:**

a String array of corresponding column types.

#### 4.22.3.5 Vector Schema::getPrimaryKey (String tableName) throws RGMAException

Returns the primary key for the given table.

**Parameters:**

*tableName* the name of the table

**Returns:**

columnNames the names of the columns that make up the primary key

#### 4.22.3.6 ServletConnection Schema::getServletConnection () throws RGMAException [protected]

Returns a [ServletConnection](#) which may be used to reconnect to the server. Note this method was added to aid testing

**Returns:**

a servlet connection

**Exceptions:**

[RGMAException](#) if the object has been closed or disconnected

#### 4.22.3.7 String Schema::getStatus () throws RGMAException

Gets status information.

**Returns:**

status information

#### 4.22.3..8 String Schema::getTableDesc (String *tableName*) throws **RGMAException**

Returns a CREATE TABLE statement for the given table.

#### 4.22.3..9 ResultSet Schema::getTableInfo (String *tableId*) throws **RGMAException**

Returns the names of columns associated with a particular schema table

**Parameters:**

*tableId* the Identifier of the table

**Returns:**

ResultSet containing Strings identified by "columnName"

#### 4.22.3..10 ResultSet Schema::sendCommand (String *command*, ServletConnection *connection*) throws **RGMAException** [protected]

sends a command using a connection that already has command parameters added (including the connectionId).

**Parameters:**

*command* a servlet command

*connection* a connection to a servlet, with command parameters already added

**Returns:**

RGMA ResultSet response from the servlet

**Exceptions:**

**RGMAException** thrown if the servlet can't be reached.

#### 4.22.3..11 int [] Schema::translateColumnNames (String *tableId*, String *fixedColumns*[]) throws **RGMAException**

Translate an array of column names into an array of columnId strings

**Parameters:**

*a* tableId

*a* String array of column names to be translated.

**Returns:**

an int array of columnId values

#### 4.22.3..12 String Schema::translateTableName (String *tableName*) throws **RGMAException**

Translates a tableName into a tableId.

**Parameters:**

*a* tableName

**Returns:**

tableId

#### 4.22.4. MEMBER DATA DOCUMENTATION

##### 4.22.4..1 String Schema::servletURL [protected]

#### 4.23. SERVLETCONNECTION CLASS REFERENCE

##### PUBLIC METHODS

- `ServletConnection` (String servletURL, int connectionId) throws `RGMAException`
- `int` `getConnectionId` ()
- `String` `getServletURL` ()
- `String` `toString` ()

##### STATIC PUBLIC ATTRIBUTES

- `final String` `GET` = "GET"
- `final String` `POST` = "POST"
- `final String` `SSLINIT` = "SSLINIT"

##### PROTECTED METHODS

- `ServletConnection` ()
- `ServletConnection` (String servletURL) throws `RGMAException`
- `void` `addParameter` (String name, String value) throws `RGMAException`
- `void` `addParameter` (String name, int value) throws `RGMAException`
- `void` `addParameter` (String name, double value) throws `RGMAException`
- `void` `addParameter` (String name, boolean value) throws `RGMAException`
- `void` `addParameter` (String name, long value) throws `RGMAException`
- `String` `connect` (String operation) throws `RGMAException`
- `String` `getTrustFile` ()
- `void` `setRequestMethod` (String method)
- `void` `setBody` (String message)
- `BufferedReader` `streamingConnect` (String operation) throws `RGMAException`

##### STATIC PROTECTED METHODS

- `void` `setupSSLConnections` () throws `RGMAException`
- `void` `setupRgmaSSLFactory` () throws `RGMAException`

##### STATIC PROTECTED ATTRIBUTES

- `String` `sslInit`
- `SSLSocketFactory` `RgmaSSLFactory`

#### 4.23.1. DETAILED DESCRIPTION

A `ServletConnection` Object is used to connect to a particular Http based Servlet. The URL is defined by the client and Http GET or Http POST can be used. By default, GET is the preferred transport method although this can be changed by using the `setRequestMethod()`. Provision for HTTPS over SSL is also supported.

#### 4.23.2. CONSTRUCTOR & DESTRUCTOR DOCUMENTATION

##### 4.23.2..1 `ServletConnection::ServletConnection (String servletURL, int connectionId)` throws **RGMAException**

##### 4.23.2..2 `ServletConnection::ServletConnection ()` [protected]

Needed for creating a `MockServletConnection`.

##### 4.23.2..3 `ServletConnection::ServletConnection (String servletURL)` throws **RGMAException** [protected]

Method `ServletConnection()`

**Parameters:**

*servletURL*

**Exceptions:**

***RGMAException***

#### 4.23.3. MEMBER FUNCTION DOCUMENTATION

##### 4.23.3..1 `void ServletConnection::addParameter (String name, long value)` throws **RGMAException** [protected]

Add a long parameter (name value pair).

**Parameters:**

*name* name of parameter

*value* value of parameter

##### 4.23.3..2 `void ServletConnection::addParameter (String name, boolean value)` throws **RGMAException** [protected]

Add a boolean parameter (name value pair).

**Parameters:**

*name* name of parameter

*value* value of parameter

**4.23.3.3 void ServletConnection::addParameter (String name, double value) throws RGMAException** [protected]

Add a double parameter (name value pair).

**Parameters:**

*name* name of parameter

*value* value of parameter

**4.23.3.4 void ServletConnection::addParameter (String name, int value) throws RGMAException** [protected]

Add an integer parameter (name value pair).

**Parameters:**

*name* name of parameter

*value* value of parameter

**4.23.3.5 void ServletConnection::addParameter (String name, String value) throws RGMAException** [protected]

Add a string parameter (name value pair).

**Parameters:**

*name* name of parameter

*value* value of parameter

**4.23.3.6 String ServletConnection::connect (String operation) throws RGMAException** [protected]

Connect using the specified operation.

**Parameters:**

*operation* name of operation the servlet is to perform. The operation name is appended to the base servlet URI to which are appended any parameters to form the complete URI.

**Returns:**

XML string

**4.23.3.7 int ServletConnection::getConnectionId ()**

Method `getConnectionId()`

**Returns:**

int

#### 4.23.3..8 String ServletConnection::getServletURL ()

Method `getServletURL()`

**Returns:**

String

#### 4.23.3..9 String ServletConnection::getTrustFile () [protected]

Method `getTrustFile()`

**Returns:**

String

#### 4.23.3..10 void ServletConnection::setBody (String message) [protected]

Method `setBody()` will set the contents of Http body (used when POST has been configured)

**See also:**

`setRequestMethod()`.

**Parameters:**

*message*

#### 4.23.3..11 void ServletConnection::setRequestMethod (String method) [protected]

Method `setRequestMethod()`

**Parameters:**

*method*

#### 4.23.3..12 void ServletConnection::setupRgmaSSLFactory () throws RGMAException [static, protected]

Setup the SSLSocketFactory

#### 4.23.3..13 void ServletConnection::setupSSLConnections () throws RGMAException [static, protected]

Setup the SSL Connections by setting up the SSL factory in `HttpsURLConnection` can then simply use the URL connection for everything!

#### 4.23.3..14 BufferedReader ServletConnection::streamingConnect (String operation) throws RGMAException [protected]

Connect with specified operation.

**Parameters:**

*operation* name of operation the servlet is to perform. The operation name is appended to the base servlet URI to which are appended any parameters to form the complete URI.

**Returns:**

BufferedReader and the reading is handled by the caller

#### 4.23.3..15 String ServletConnection::toString ()

**See also:**

`java.lang.ObjecttoString()`

#### 4.23.4. MEMBER DATA DOCUMENTATION

##### 4.23.4..1 final String ServletConnection::GET = "GET" [static]

Constant represents Http Get

##### 4.23.4..2 final String ServletConnection::POST = "POST" [static]

Constant represents Http Post

##### 4.23.4..3 SSLSocketFactory ServletConnection::RgmaSSLFactory [static, protected]

Socket factory

##### 4.23.4..4 String ServletConnection::sslInit [static, protected]

Static variable indicating SSL socket has been initialized

##### 4.23.4..5 final String ServletConnection::SSLINIT = "SSLINIT" [static]

Constant representing initialization of SSL connection

#### 4.24. STREAMPRODUCER CLASS REFERENCE

Inherits [Insertable](#).

Inherited by [ResilientStreamProducer](#).

#### PUBLIC METHODS

- `StreamProducer ()` throws `RGMAException`
- `StreamProducer (ServletConnection connection)` throws `RGMAException`
- `int getMaxBufferSize ()` throws `RGMAException`
- `void setMaxBufferSize (int maxBufferSize)` throws `RGMAException`
- `void setMinRetentionPeriod (TimeInterval interval)` throws `RGMAException`
- `TimeInterval getMinRetentionPeriod ()` throws `RGMAException`

## PROTECTED METHODS

- void `init ()` throws `RGMAException`

### 4.24.1. DETAILED DESCRIPTION

`StreamProducers` is able to register a table when it is created and subsequently to publish information.

### 4.24.2. CONSTRUCTOR & DESTRUCTOR DOCUMENTATION

#### 4.24.2..1 `StreamProducer::StreamProducer ()` throws `RGMAException`

default constructor for setting instance variables

#### Exceptions:

*`RGMAException`*

#### 4.24.2..2 `StreamProducer::StreamProducer (ServletConnection connection)` throws `RGMAException`

Reconnects to an existing `ServletConnection`. It is an error to have more than one API object making use of the same `ServletConnection`.

#### Parameters:

*`sc`* is the servlet connection which will be reconnected

### 4.24.3. MEMBER FUNCTION DOCUMENTATION

#### 4.24.3..1 `int StreamProducer::getMaxBufferSize ()` throws `RGMAException`

Create a `StreamProducer` publishing to a Vector of Virtual Organisations (VOs) which can have the `NON_PRIMARY` flag set.

#### Parameters:

*`vos`* is a vector of Virtual Organisations to which the Producer should be registered.

*`flags`* the flag `NON_PRIMARY` can be set to indicate that this is not a primary producer.

XXX: to do!

```
public StreamProducer(Vector vos, int flags) throws RGMAException { }
```

```
/** Construct a StreamProducer.publishing to a Vector of VOs.
```

#### Parameters:

*`vos`* is a vector of virtual organisations to which the Producer should be registered.

XXX: not yet implemented VO functionality!

```
public StreamProducer(Vector vos) throws RGMAException { this();  
cat.info("entering constructor"); ServletConnection connection = new Servlet-  
Connection(servletURL); connection.addParameter("terminationInterval", DEFAULT_-  
TERMINATION_INTERVAL); ResultSet result = sendCommand("createInstance",  
connection); result.next(); connectionId = result.getInt("connectionId");  
cat.info("exiting constructor"); }  
  
/** get the StreamProducer servlet's maximum buffer size
```

**Exceptions:**

*RGMAException*

**Returns:**

the maximum size of the producer's buffer.

**4.24.3.2 TimeInterval StreamProducer::getMinRetentionPeriod () throws RGMAException**

gets the minimum period of time to keep the `data` in the buffer.

**4.24.3.3 void StreamProducer::init () throws RGMAException** [protected]

Reimplemented in `ResilientStreamProducer`.

**4.24.3.4 void StreamProducer::setMaxBufferSize (int *maxBufferSize*) throws RGMAException**

set the StreamProducer servlet's maximum buffer size

**Parameters:**

*maxBufferSize*

**Exceptions:**

*RGMAException*

**4.24.3.5 void StreamProducer::setMinRetentionPeriod (TimeInterval *interval*) throws RGMAException**

Set a minimum time period to hold an inserted tuple. This applies even if all current consumers have taken the tuple. (Default period is zero).

**Parameters:**

*interval* minimum time period to hold an inserted tuple, even if all current consumers have taken the tuple.

**Exceptions:**

*RGMAException* if a negative interval is set.

## 4.25. STREAMREQUEST CLASS REFERENCE

### PUBLIC METHODS

- String `toString ()`
- boolean `equals (Object o)`
- int `hashCode ()`

### 4.25.1. MEMBER FUNCTION DOCUMENTATION

#### 4.25.1.1 boolean `StreamRequest::equals (Object o)`

#### 4.25.1.2 int `StreamRequest::hashCode ()`

#### 4.25.1.3 String `StreamRequest::toString ()`

## 4.26. TIME CLASS REFERENCE

### PUBLIC METHODS

- `Time ()` throws `RGMAException`
- `Time (long utc)` throws `RGMAException`
- `Time (int utc32, int eo)` throws `RGMAException`
- `Time (String iso8601)` throws `RGMAException`
- `Time (String date, String time)` throws `RGMAException`
- void `setTai (long value)`
- void `setUtc (long value)`
- void `setUtcAsInt (int value)`
- void `setIso8601 (String iso8601)` throws `RGMAException`
- void `setDateTime (String datetime)` throws `RGMAException`
- void `setDateTime (String date, String time)` throws `RGMAException`
- void `setSubseconds (double value)`
- void `setSubseconds (long value)`
- void `setLeapseconds (int value)`
- void `setPrecision (double value)`
- void `setAccuracy (double value)`
- void `setEpochOffset (int value)`
- void `setDetails (byte[] detailStructure)` throws `RGMAException`
- void `setDetailsUnpacked (byte[] detailStructure)` throws `RGMAException`
- long `getUtc ()`
- int `getUtcAsInt ()`
- long `getTai ()` throws `RGMAException`
- String `getDate ()` throws `RGMAException`
- String `getTime ()` throws `RGMAException`
- String `getIso8601 ()` throws `RGMAException`
- String `getDateTime ()` throws `RGMAException`

- double `getSubsecondsAsDouble` ()
- long `getSubseconds` ()
- int `getLeapseconds` ()
- double `getPrecision` ()
- double `getAccuracy` ()
- int `getEpochOffset` ()
- byte[] `getDetails` () throws `RGMAException`
- byte[] `getDetailsUnpacked` () throws `RGMAException`
- String `toString` ()
- int `compareTo` (Time other)
- boolean `before` (Time other)
- boolean `after` (Time other)
- boolean `equals` (Time other)
- double `subtract` (Time other)
- `FixedPoint` `getUtcSeconds` ()
- void `setUtcSeconds` (`FixedPoint` us)

#### 4.26.1. DETAILED DESCRIPTION

Time represents a date/time value.

This class provides a simple mechanism for handling time in several formats. The intention is to provide a standard Time format for use within R-GMA development, as well as for users of R-GMA.

Used in the simplest way, Time objects provide a resolution of one second, as that is adequate for many R-GMA applications. A UTC count of seconds is used as the underlying representation, as this is the most common representation in use. Other formats are converted to UTC before storage.

Further information about the time is stored separately. Leapseconds, subsecond values, precision and accuracy are catered for.

Time objects are intended to be used within R-GMA and applications running on R-GMA. When time information is stored in a database, the time will need to be in a suitable format. The getter methods return values suitable for use in the standard R-GMA MeasurementTime column types.

Default values for the fields are:

- UTC time: 0.0 seconds
- Leapseconds: 0 leapseconds
- Precision: Not a Number (Double.NaN)
- Accuracy: Not a Number (Double.NaN)

#### 4.26.2. CONSTRUCTOR & DESTRUCTOR DOCUMENTATION

##### 4.26.2.1 Time::Time () throws `RGMAException`

Empty constructor to allow initialisation with various setters. Creates the Time object with default values.

#### 4.26.2..2 Time::Time (long *utc*) throws RGMAException

Create a Time object using a UTC count of seconds. Initialises other fields with default values.

**Parameters:**

*utc* Count in seconds since the UTC Epoch.

#### 4.26.2..3 Time::Time (int *utc32*, int *eo*) throws RGMAException

Create a Time object using a UTC count of seconds and an epoch offset.

**Parameters:**

*utc32* Count in seconds since the UTC Epoch.

*eo* epoch offset.

#### 4.26.2..4 Time::Time (String *iso8601*) throws RGMAException

Create a Time object using an ISO8601 string representation. Initialises other fields with default values.

ISO8601 has the form: 'YYYY-MM-DDThh:mm:ss'

NOTE: This modifies the leapseconds value to an appropriate value for the given time.

**Parameters:**

*iso8601* ISO8601 formatted date/time.

#### 4.26.2..5 Time::Time (String *date*, String *time*) throws RGMAException

Set the date/time using DATE and TIME standard formats.

**Parameters:**

*date* a String representation of a date, formatted as 'YYYY-MM-DD'

*time* a String representation of a time, formatted as 'hh:mm:ss'

### 4.26.3. MEMBER FUNCTION DOCUMENTATION

#### 4.26.3..1 boolean Time::after (Time *other*)

Test order of this Time and another.

**Returns:**

true if this is after other.

#### 4.26.3..2 boolean Time::before (Time *other*)

Test order of this Time and another.

**Returns:**

true if this is before other.

#### 4.26.3.3 int Time::compareTo (Time *other*)

Compare this Time to another.

**Returns:**

a value less than 0, if this is less than other; a value greater than 0, if this is greater than other; 0 if this is equal to other.

#### 4.26.3.4 boolean Time::equals (Time *other*)

Test equality of this Time and another. This compares seconds and leapseconds. Precision and accuracy are not currently taken into account when making the comparison.

**Parameters:**

*other* Another Time object.

**Returns:**

Whether or not this object is equal in value to the other.

#### 4.26.3.5 double Time::getAccuracy ()

Return the accuracy value.

**Returns:**

Accuracy value.

#### 4.26.3.6 String Time::getDate () throws RGMAException

return time as an ISO8601 string

#### 4.26.3.7 String Time::getDateTime () throws RGMAException

Return the date/time as a DATETIME string.

**Returns:**

DATETIME formatted date/time.

#### 4.26.3.8 byte [] Time::getDetails () throws RGMAException

Get a binary structure representing the time details (Subseconds, Leapseconds, Accuracy, Precision, Epoch Offset).

**PACKED**

This structure packs the details using appropriately sized fields. Typically it produces a smaller structure than `getDetailsUnpacked`.

Each field is preceded by a pack code byte which specifies the type and size of the field.

```

7          0
+-----+-----+
|type|size|
+-----+-----+

```

Type values between 1 and 5 indicate subseconds, leapseconds, epoch offset, precision and accuracy respectively. 0 and F are reserved. Values between 6 and E are available for future use.

The size specifies the size of the field in  $\log_2(\text{number of bits})$ . 1 byte = 8 bits =  $2^3$ , so this field has a size value of three. Valid sizes are 3 (byte), 4 (short), 5 (int), 6(long).

The size field is overloaded, so a value of 0, 1 or 2 in the size field should be interpreted as a literal value. F should be interpreted as -1. This takes advantage of the fact that we can't address fields of sizes  $2^0$ ,  $2^1$  and  $2^2$ . Fields of size  $2^F$  bits would be addressable, but a short representation of -1 (or 0xFFFFFFFF as an int) is useful.

Following the pack code byte is the field itself, with the correct number of bytes as specified in the first byte. For example, the following diagram represents a leapsecond value of 32 (or 0x20).

```

+-----+-----+-----+-----+
| 2 | 3 | 2 | 0 |
+-----+-----+-----+-----+
*
```

**Returns:**

Binary structure representing all time details.

**4.26.3.9 byte [] Time::getDetailsUnpacked () throws RGMAException**

Get a binary structure representing the time details (Subseconds, Leapseconds, Accuracy, Precision, Epoch Offset).

**UNPACKED**

This structure stores the details with the same sizes used internally by the Time class: double (64-bit) subseconds, precision and accuracy fields, int (32-bit) leapseconds and epoch offset fields.

```

63          0
+-----+-----+-----+-----+-----+-----+-----+-----+
*0 |                subseconds                |
+-----+-----+-----+-----+-----+-----+-----+-----+
*1 |  leapseconds  |  epochOffset  |
+-----+-----+-----+-----+-----+-----+-----+-----+
*2 |                precision                |
+-----+-----+-----+-----+-----+-----+-----+-----+
*3 |                accuracy                |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

**Returns:**

Binary structure representing all time details.

**4.26.3..10 int Time::getEpochOffset ()**

Return the epoch offset value.

**Returns:**

Offset to indicate date/time before 1970 or after early-2038

**4.26.3..11 String Time::getIso8601 () throws RGMAException**

Return the date/time as an ISO8601 string.

**Returns:**

ISO8601 formatted date/time.

**4.26.3..12 int Time::getLeapseconds ()**

Return the leapseconds value.

**Returns:**

Current leapsecond TAI-UTC offset

**4.26.3..13 double Time::getPrecision ()**

Return the precision value.

**Returns:**

Precision value.

**4.26.3..14 long Time::getSubseconds ()**

Return the subseconds value.

**Returns:**

Subseconds.

**4.26.3..15 double Time::getSubsecondsAsDouble ()**

Return the subseconds value approximated as a double.

**4.26.3..16 long Time::getTai () throws RGMAException**

Return the date/time as a TAI count in seconds.

**Returns:**

Count in seconds since TAI Epoch.

#### 4.26.3..17 String Time::getTime () throws RGMAException

return time as an ISO8601 string

#### 4.26.3..18 long Time::getUtc ()

Return the date/time as a UTC count in seconds.

This does not take leap seconds into account.

**Returns:**

Count in seconds since UTC Epoch.

#### 4.26.3..19 int Time::getUtcAsInt ()

Return the date/time as a UTC count in seconds.

This returns an int value which must be interpreted along with the value returned by `getEpochOffset`. This does not take leap seconds into account.

**Returns:**

Count in seconds since UTC Epoch.

#### 4.26.3..20 FixedPoint Time::getUtcSeconds ()

#### 4.26.3..21 void Time::setAccuracy (double value)

Set the accuracy value.

**Parameters:**

*value* Accuracy.

#### 4.26.3..22 void Time::setDateTime (String date, String time) throws RGMAException

Set the date/time using a DATETIME formatted string:

'YYYY-MM-DD hh:mm:ss'.

NOTE: This modifies the leapseconds value to an appropriate value for the given time.

**Parameters:**

*datetime* DATETIME formatted date/time.

#### 4.26.3..23 void Time::setDateTime (String datetime) throws RGMAException

Set the date/time using a DATETIME formatted string:

'YYYY-MM-DD hh:mm:ss'.

NOTE: This modifies the leapseconds value to an appropriate value for the given time.

**Parameters:**

*datetime* DATETIME formatted date/time.

**4.26.3..24 void Time::setDetails (byte *detailStructure*[]) throws **RGMAException****

Set details (Subseconds, Leapseconds, Accuracy, Precision, Epoch Offset) from a packed binary structure. See the `getDetails` method for more information on this structure.

**Parameters:**

*detailStructure* Binary structure representing all time details.

**4.26.3..25 void Time::setDetailsUnpacked (byte *detailStructure*[]) throws **RGMAException****

Set details (Subseconds, Leapseconds, Accuracy, Precision, Epoch Offset) from an unpacked binary structure. See the `getDetailsUnpacked` method for more information on this structure.

**Parameters:**

*detailStructure* Binary structure representing all time details.

**4.26.3..26 void Time::setEpochOffset (int *value*)**

Set the epoch offset.

This allows 32-bit date/time representations to refer to past and future dates beyond the current 68-year (0x80000000L second) era.

NOTE: This modifies the leapseconds value to an appropriate value for the given time.

**Parameters:**

*value* Offset to indicate date/time before 1970 or after early-2038.

**4.26.3..27 void Time::setIso8601 (String *iso8601*) throws **RGMAException****

Set the date/time using an ISO8601 formatted string.

NOTE: This modifies the leapseconds value to an appropriate value for the given time.

**Parameters:**

*iso8601* ISO8601 formatted date/time.

**4.26.3..28 void Time::setLeapseconds (int *value*)**

Set the leapseconds value.

This value should be altered for times created during a leap second.

**Parameters:**

*value* Current leapsecond TAI-UTC offset.

**4.26.3..29 void Time::setPrecision (double *value*)**

Set the precision value.

**Parameters:**

*value* Precision.

#### 4.26.3..30 void Time::setSubseconds (long value)

Set the subseconds value.

**Parameters:**

*value* Subseconds in units of  $10^{-18}$ .

#### 4.26.3..31 void Time::setSubseconds (double value)

Set the subseconds value.

**Parameters:**

*value* Subsecond time value.

#### 4.26.3..32 void Time::setTai (long value)

Set the date/time using a TAI value.

NOTE: This modifies the leapseconds value to an appropriate value for the given time.

**Parameters:**

*value* Count in seconds since TAI Epoch.

#### 4.26.3..33 void Time::setUtc (long value)

Set the date/time using a UTC value.

NOTE: This modifies the leapseconds value to an appropriate value for the given time.

**Parameters:**

*value* Count in seconds since UTC Epoch.

#### 4.26.3..34 void Time::setUtcAsInt (int value)

Set the date/time using a UTC value.

NOTE: This modifies the leapseconds value to an appropriate value for the given time.

**Parameters:**

*value* Count in seconds since UTC Epoch.

#### 4.26.3..35 void Time::setUtcSeconds (FixedPoint us)

#### 4.26.3..36 double Time::subtract (Time other)

Substract another Time from this one. Precision and accuracy are not taken into account.

If this number is less than the other, the result is a negative number. (i.e. behaviour is different from FixedPoint)

**Returns:**

Fractional number of seconds, as a double.

#### 4.26.3.37 String Time::toString ()

String representation of date/time with details. This method is intended for use when testing, and not as a useful string representation of the value.

### 4.27. TIMECONVERTER CLASS REFERENCE

#### PUBLIC METHODS

- `TimeConverter ()`
- `TimeConverter (String path)`
- `String utcToIso8601 (Time t)` throws `RGMAException`
- `String utcToDateTime (Time t)` throws `RGMAException`
- `void setTimeWithIso8601 (Time t, String iso8601)` throws `RGMAException`
- `void setTimeWithDateTime (Time t, String datetime)` throws `RGMAException`
- `long utcToTai (long seconds)` throws `RGMAException`
- `long taiToUtc (long seconds)`

#### PUBLIC ATTRIBUTES

- LeapsecondData `ld`

#### 4.27.1. DETAILED DESCRIPTION

A class to convert between different time formats.

Numerical UTC offsets are catered for, symbolic timezones (EST, etc) are not.

This class is used by the `Time` class which generates the time value.

#### CONVERSIONS

ISO8601 <--> UTC Binary

ISO8601 <--> TAI Binary

UTC Binary <--> TAI Binary

#### 4.27.2. CONSTRUCTOR & DESTRUCTOR DOCUMENTATION

##### 4.27.2..1 TimeConverter::TimeConverter ()

Constructor using the default location for the leapsecond information file, `~/etc/leapsec.dat`.

#### 4.27.2.2 TimeConverter::TimeConverter (String path)

Constructor specifying the location of the leapsecond information file.

**Parameters:**

*path* the string representation of the path to the file

#### 4.27.3. MEMBER FUNCTION DOCUMENTATION

##### 4.27.3.1 void TimeConverter::setTimeWithDateTime (Time t, String datetime) throws RGMAException

Function to convert to a UTC binary timestamp from an ascii timestamp.

**Parameters:**

*t* The Time object to be modified.

*datetime* A DATETIME formatted timestamp ("dddd-dd-dd dd:dd:dd").

##### 4.27.3.2 void TimeConverter::setTimeWithIso8601 (Time t, String iso8601) throws RGMAException

Function to convert to a UTC binary timestamp from an ascii timestamp.

**Parameters:**

*t* The Time object to be modified.

*iso8601* An ISO8601 formatted timestamp.

##### 4.27.3.3 long TimeConverter::taiToUtc (long seconds)

Convert from TAI seconds to UTC seconds.

**Parameters:**

*seconds* TAI Count in seconds

**Returns:**

Count in seconds since UTC Epoch

##### 4.27.3.4 String TimeConverter::utcToDateTime (Time t) throws RGMAException

Convert a UTC binary timestamp to a DATETIME timestamp.

**Parameters:**

*t* time to convert.

**Returns:**

An ISO8601 formatted timestamp.

#### 4.27.3.5 String TimeConverter::utcToIso8601 (Time *t*) throws RGMAException

Convert a UTC binary timestamp to an ISO8601 timestamp.

**Parameters:**

*t* time to convert.

**Returns:**

An ISO8601 formatted timestamp.

#### 4.27.3.6 long TimeConverter::utcToTai (long *seconds*) throws RGMAException

Convert from UTC seconds to TAI seconds.

**Parameters:**

*seconds* Count in seconds since UTC Epoch

**Returns:**

TAI Count in seconds

### 4.27.4. MEMBER DATA DOCUMENTATION

#### 4.27.4.1 LeapsecondData TimeConverter::ld

## 4.28. TIMEDETAILS CLASS REFERENCE

### STATIC PUBLIC METHODS

- void `setDetailsLarge` (Time *mt*, byte[] *detailStructure*)
- void `unpackDetails` (Time *mt*, byte[] *detailStructure*) throws RGMAException
- byte[] `getDetailsLarge` (Time *mt*)
- byte[] `packDetails` (Time *mt*) throws RGMAException

#### 4.28.1. DETAILED DESCRIPTION

TimeDetails provides utilities for manipulating binary structures representing date/time details (Subseconds, Leapseconds, Accuracy, Precision, Epoch Offset).

### 4.28.2. MEMBER FUNCTION DOCUMENTATION

4.28.2.1 byte [] TimeDetails::getDetailsLarge (Time *mt*) [static]

4.28.2.2 byte [] TimeDetails::packDetails (Time *mt*) throws RGMAException [static]

4.28.2.3 void TimeDetails::setDetailsLarge (Time *mt*, byte *detailStructure*[]) [static]

4.28.2.4 void TimeDetails::unpackDetails (Time *mt*, byte *detailStructure*[]) throws RGMAException [static]

## 4.29. TIMEINTERVAL CLASS REFERENCE

### PUBLIC METHODS

- `TimeInterval ()` throws `RGMAException`
- `TimeInterval (Time start, Time finish)` throws `RGMAException`
- `TimeInterval (double seconds)` throws `RGMAException`
- `void setBySeconds (double seconds)` throws `RGMAException`
- `double getBySeconds ()` throws `RGMAException`

### 4.29.1. DETAILED DESCRIPTION

A time interval.

### 4.29.2. CONSTRUCTOR & DESTRUCTOR DOCUMENTATION

#### 4.29.2..1 `TimeInterval::TimeInterval ()` throws `RGMAException`

Construct a time interval of zero length.

#### Exceptions:

*`RGMAException`*

#### 4.29.2..2 `TimeInterval::TimeInterval (Time start, Time finish)` throws `RGMAException`

Construct a time interval using the difference between two times.

#### Parameters:

*start* Start time of interval

*finish* Finish time of interval

#### Exceptions:

*`RGMAException`*

#### 4.29.2..3 `TimeInterval::TimeInterval (double seconds)` throws `RGMAException`

Construct a time interval using a count of seconds.

#### Parameters:

*seconds* count of seconds

#### Exceptions:

*`RGMAException`*

### 4.29.3. MEMBER FUNCTION DOCUMENTATION

#### 4.29.3..1 `double TimeInterval::getBySeconds ()` throws `RGMAException`

returns the time interval as fractional seconds.

**Returns:**

seconds length of interval expressed in seconds

**Exceptions:**

*RGMAException*

**4.29.3..2 void TimeInterval::setBySeconds (double *seconds*) throws *RGMAException***

Set a time interval by its length in seconds.

**Parameters:**

*seconds* length of interval expressed in seconds

**Exceptions:**

*RGMAException*

## 4.30. XMLCONVERTER CLASS REFERENCE

### PUBLIC METHODS

- `XMLConverter ()` throws *RGMAException*
- `ResultSet convertXMLResponse (String xml)` throws *RGMAException*
- `void warning (SAXParseException ex)`
- `void error (SAXParseException ex)`
- `void fatalError (SAXParseException ex)` throws *SAXException*

#### 4.30.1. DETAILED DESCRIPTION

Parses XML and constructs `ResultSet`

#### 4.30.2. CONSTRUCTOR & DESTRUCTOR DOCUMENTATION

##### 4.30.2..1 XMLConverter::XMLConverter () throws *RGMAException*

Constructor

#### 4.30.3. MEMBER FUNCTION DOCUMENTATION

##### 4.30.3..1 `ResultSet XMLConverter::convertXMLResponse (String xml)` throws *RGMAException*

Generates `resultSet` from `xml` string.

**Exceptions:**

*RGMAException* Thrown if input string is empty or null.

##### 4.30.3..2 void XMLConverter::error (SAXParseException *ex*)

Error.

### 4.30.3.3 void XMLConverter::fatalError (SAXParseException *ex*) throws SAXException

Fatal error.

### 4.30.3.4 void XMLConverter::warning (SAXParseException *ex*)

Warning.

## 5. FABRIC

### 5.1. METRIC STRUCT REFERENCE

#### PUBLIC ATTRIBUTES

- int `metricId`
- char `metricName` [MAX\_METRICNAME\_LENGTH+1]
- Metric \* `next`

#### 5.1.1. DETAILED DESCRIPTION

Metric identifier type.

The structure can hold a numeric and a textual identifier of the metric. If only one is defined ( `metricId == -1` or `metricName == ""` ) the other can be obtained with mapping functions.

An item of this type may be part of a linked list (field `next`). Most API calls work on linked lists of metrics.

#### 5.1.2. MEMBER DATA DOCUMENTATION

##### 5.1.2.1 int Metric::metricId

Metric instance numeric identifier. -1 if undefined. This numeric identifier of a metric instance allows space saving in protocols and database.

##### 5.1.2.2 char Metric::metricName[MAX\_METRICNAME\_LENGTH+1]

Metric instance textual identifier. This is an easily readable identifier of a metric instance. (NULL terminated)

##### 5.1.2.3 struct Metric\* Metric::next

To be used to create linked lists of Metric\_t items.

### 5.2. METRICBASE CLASS REFERENCE

#### PUBLIC METHODS

- `metricBase` ()

- `metricBase (metricParams *p)`
- `virtual ~metricBase ()`
- `virtual char * info ()`
- `virtual int sample ()`
- `virtual int periodicCheck ()`
- `int storeSample01 (const char *value)`
- `int log (int error_level, char *message, ...)`

## PUBLIC ATTRIBUTES

- `int numericId`

## PROTECTED ATTRIBUTES

- `metricParams * parameters`

### 5.2.1. DETAILED DESCRIPTION

This is the metric base definition class. All metric classes are derived from this one.

### 5.2.2. CONSTRUCTOR & DESTRUCTOR DOCUMENTATION

#### 5.2.2.1 `metricBase::metricBase ()`

Default Constructor.

#### 5.2.2.2 `metricBase::metricBase (metricParams *p)`

Constructor with configuration parameters provided. Overloaded constructor which initializes parameters with the argument This one is always used so that parameters are available to user init code When defining a new metric class constructor, call this constructor or make sure you delete parameters p.

#### Parameters:

*p* : a pointer to a `metricParams` instance containing some key/value pairs.

#### 5.2.2.3 `virtual metricBase::~~metricBase ()` [virtual]

Destructor. Declared virtual so that derived class instances can be completely destroyed when deleting it with a `metricBase` pointer.

### 5.2.3. MEMBER FUNCTION DOCUMENTATION

#### 5.2.3.1 `virtual char* metricBase::info ()` [virtual]

Info method. Information to be returned when queried about metric instance status. Write all status information you want to be displayed here.

**Returns:**

A pointer to a NULL terminated string, or NULL.

**5.2.3..2 int metricBase::log (int *error\_level*, char \* *message*, ...)**

Log method. Log a message related to a metric.

**Parameters:**

*error\_level* : type of log message. See definitions below.

*message* : message is possibly a 'printf' command with parameters. But it MUST NOT include new line characters.

**Returns:**

0 if successfull, -1 otherwise.

**5.2.3..3 virtual int metricBase::periodicCheck () [virtual]**

Periodic processing method. Function called periodically in the sensor loop. Implement periodic checking code here if any.

**Returns:**

0 if successfull, -1 otherwise.

**5.2.3..4 virtual int metricBase::sample () [virtual]**

Sampling method. Function called at each sampling interval to trigger a measurement. Implement measurement code here if any.

**Returns:**

0 if successfull, -1 otherwise.

**5.2.3..5 int metricBase::storeSample01 (const char \* *value*)**

Function to store samples type 01 - limited length (max. 2000 bytes) string value.

**See also:**

sensor interface documentation

**5.2.4. MEMBER DATA DOCUMENTATION**

**5.2.4..1 int metricBase::numericId**

Numeric identifier of the metric instance.

**5.2.4..2 metricParams\* metricBase::parameters [protected]**

Parameters of the metric instance. It consists of key/value pairs of configuration items.

### 5.3. METRICPARAMS CLASS REFERENCE

#### PUBLIC METHODS

- `metricParams ()`
- `~metricParams ()`
- `int addParam (char *key, char *value)`
- `const char * getParam (char *key)`
- `void print ()`

#### 5.3.1. DETAILED DESCRIPTION

This class is used to store list of parameters.

#### 5.3.2. CONSTRUCTOR & DESTRUCTOR DOCUMENTATION

##### 5.3.2..1 `metricParams::metricParams ()`

Constructor. List empty when created.

##### 5.3.2..2 `metricParams::~~metricParams ()`

Destructor. Destroy whole list.

#### 5.3.3. MEMBER FUNCTION DOCUMENTATION

##### 5.3.3..1 `int metricParams::addParam (char * key, char * value)`

Add a new key/value pair.

- both must be non null.
- key must not already be in the list.

#### Parameters:

*key* : New key to be added. (NULL-terminated)

*value* : Value to be associated with the key. (NULL-terminated)

#### Returns:

0 if success. -1 if an error occurred.

##### 5.3.3..2 `const char* metricParams::getParam (char * key)`

Search method. Try to find the value for a given key.

#### Parameters:

*key* : Key to be searched. (NULL-terminated)

#### Returns:

If not found, 0 is returned. If found, a pointer to the NULL-terminated value is returned.

### 5.3.3.3 void metricParams::print ()

Print method. Print key / value pairs on stdout.

## 5.4. METRICVALUE STRUCT REFERENCE

### PUBLIC ATTRIBUTES

- int freeValue
- int valueLength
- char \* value

#### 5.4.1. DETAILED DESCRIPTION

Metric value type.

Definition of the metric value type.

The freeValue element is a boolean to indicate (e.g. to the MR\_freeSamples() interface) that the value should be freed when structure is destroyed.

#### 5.4.2. MEMBER DATA DOCUMENTATION

##### 5.4.2.1 int MetricValue::freeValue

1 if 'value' has to be freed with the structure, 0 otherwise.

##### 5.4.2.2 char\* MetricValue::value

NULL terminated string containing the value.

##### 5.4.2.3 int MetricValue::valueLength

Number of bytes in value, including NULL terminating character.

## 5.5. NODE STRUCT REFERENCE

### PUBLIC ATTRIBUTES

- int nodeId
- char nodeName [MAX\_NODENAME\_LENGTH+1]
- Node \* next

#### 5.5.1. DETAILED DESCRIPTION

Node identifier type.

The structure can hold a numeric and a textual identifier of the node. If only one is defined ( nodeId == -1 or nodeName == "" ) the other can be obtained with mapping functions.

An item of this type may be part of a linked list (field next). Most API calls work on linked lists of metrics.

## 5.5.2. MEMBER DATA DOCUMENTATION

### 5.5.2.1 struct Node\* Node::next

To be used to create linked lists of Node\_t items.

### 5.5.2.2 int Node::nodeId

Node numeric identifier -1 if undefined.

### 5.5.2.3 char Node::nodeName[MAX\_NODENAME\_LENGTH+1]

Node name. (NULL terminated)

## 5.6. SAMPLE STRUCT REFERENCE

### PUBLIC ATTRIBUTES

- Node\_t node
- Metric\_t metric
- MetricTimestamp timestamp
- MetricValue\_t value
- MetricTimestamp recvTime
- Sample \* next

### 5.6.1. DETAILED DESCRIPTION

Sample definition.

A sample is a metric measurement. Basically, it includes a node identifier, a metric identifier, a timestamp, and a value.

Additional features:

- Time of sample reception can be recorded using the recvTime field.
- An item of this type may be part of a linked list (field next).

## 5.6.2. MEMBER DATA DOCUMENTATION

### 5.6.2.1 Metric\_t Sample::metric

Metric measured.

### 5.6.2.2 struct Sample\* Sample::next

To be used to create linked lists of Sample\_t items.

### 5.6.2.3 Node.t Sample::node

Target of the measurement: the node corresponding to the measurement. That is not necessarily the node which made the measurement.

### 5.6.2.4 MetricTimestamp Sample::recvTime

Time at which the sample was received by the repository. -1 if undefined.

### 5.6.2.5 MetricTimestamp Sample::timestamp

Time at which the sample was taken.

### 5.6.2.6 MetricValue.t Sample::value

Value measured.

## 6. STORAGE ELEMENT

### 6.1. FILECONTROL INTERFACE REFERENCE

#### 6.1.1. DETAILED DESCRIPTION

This Interface defines the SE Control API, derived from the reservations API.

#### <SPAN STYLE="TEXT-DECORATION: UNDERLINE;">DEFINITIONS</SPAN>

<span style="font-style: italic;">File:</span>

A file is identified by a SFN which contains (1) hostname, and (2) a virtual file path.

Note that SFNs do not contain port numbers since that would associate the SFN with a protocol.

Example: gppse01.gridpp.rl.ac.uk/directory/file

Here "/directory/file" is the virtual file path.

<span style="font-style: italic;">Client:</span>

A client in this context is anyone with access to an SE (and sufficient access rights to the files in it). For example,

<div style="margin-left: 40px;">a user with command line tools;

a Replica Manager; or

a job or Job Scheduling Service.

</div>

<span style="font-style: italic;">Pinning:</span>

Put a file in disk cache and ensure that it stays there for a specific length of time. It is an error if the file does not exist.

`<span style="font-style: italic;">Id:</span>`

Data necessary and sufficient to identify a single reservation to the SE. The SE assigns a unique pin id. [Possibly not unique for TB2.0]

`<span style="font-style: italic;">Release:</span>`

Releasing a pin, or unpinning a file, means to remove a pin on a file. The file can be removed from disk cache only when all pins on it are released.

`<span style="font-style: italic;">Reservation:</span>`

In this context, "reservation" refers to reserving an existing file for reading or writing (using the `cache()` command), creating a new file (using the `create()` command), and allocating space for uploading files (see the `reserve()` command).

`<span style="font-style: italic;">StFN:</span>`

The Storage File Name. This is a name used internally in the SE.

An example is

```
<div style="margin-left: 40px;">/castor/cern.ch/grid/test/se/eu_demo/demotest03</div>
```

`<span style="font-style: italic;">Timeout:</span>`

The client can specify a timeout to the SE for the commands `cache()`, `create()` and `reserve()`. Timeout can also be specified for the status versions of these commands to extend the lifetime of the request.

The SE can honour the request or may choose give the user a shorter or longer time (depending on local policies, etc). The timeout is specified as a relative time. This corresponds to the current (v2.1) SRM model where relative time is passed to avoid having to synchronise clocks on machines.

`<span style="font-style: italic;">TURL:</span>`

The TURL is the Transfer URL. Currently the SE will offer only passive transfer, i.e. it makes files available for transfer but will not actively transfer the file. This corresponds to `srmGet` (`srmPrepareToGet` in v2.1) in PULL mode and `srmPut` (`srmPrepareToPut`) in PUSH mode.

`<span style="font-style: italic;">Notes:</span>`

Commands below all act on a single file. We can easily have commands acting on several SFNs, but the pinning semantics becomes more complicated. Also, the client side will have to sort requests according to which SE they go to.

Currently, commands are all synchronous.

Asynchronous commands will be implemented in the following way: For "slow" operations (operations that may take a long time because they may involve transferring files from or to a mass storage system), i.e. `create()`, `cache()`, and `reserve()`, the client polls using a request id - the SE will provide an estimate of the time when the client should check again.

This interface is the control interface; it provides neither file information (at least not dynamic file information such as the function `getSECosts()`) nor file transfer.

## 6.2. FILEHANDLE INTERFACE REFERENCE

### 6.2.1. DETAILED DESCRIPTION

FileHandle is the Interface that defines basic actions on files.

**Author:**

Erwin Laure , Heinz Stockinger , Peter Kunszt

**Author:**

Steve Hicks

**Version:**

**Id:**

FileHandle.java,v 1.1 2003/03/17 11:33:24 rtam Exp

## 6.3. FILEINFO CLASS REFERENCE

### PUBLIC METHODS

- `FileInfo ()`
- `void setSEInfo (final StorageElementInfo info)`
- `void setName (final URI name)`
- `void setSize (final long size)`
- `void setOwner (final String owner) throws NullPointerException`
- `StorageElementInfo getSEInfo ()`
- `URI getName ()`
- `long getSize ()`
- `String getOwner ()`
- `String toString ()`
- `boolean equals (Object o)`
- `int hashCode ()`

### 6.3.1. DETAILED DESCRIPTION

The File Information object has information about a file stored in an SE.

**Author:**

Peter Kunszt

**Version:**

\$Id \$

### 6.3.2. CONSTRUCTOR & DESTRUCTOR DOCUMENTATION

#### 6.3.2.1 FileInfo::FileInfo ()

Constructor.

### 6.3.3. MEMBER FUNCTION DOCUMENTATION

#### 6.3.3.1 boolean FileInfo::equals (Object *o*)

#### 6.3.3.2 URI FileInfo::getName ()

Get the file name.

#### 6.3.3.3 String FileInfo::getOwner ()

Get the file owner.

**Returns:**

the owner of the file

#### 6.3.3.4 StorageElementInfo FileInfo::getSEInfo ()

Get the storage element information.

#### 6.3.3.5 long FileInfo::getSize ()

Get the file size.

#### 6.3.3.6 int FileInfo::hashCode ()

#### 6.3.3.7 void FileInfo::setName (final URI *name*)

Set the file name

**Parameters:**

*name* the file name

#### 6.3.3.8 void FileInfo::setOwner (final String *owner*) throws NullPointerException

Set the file owner.

**Parameters:**

*owner* string

#### 6.3.3.9 void FileInfo::setSEInfo (final StorageElementInfo *info*)

Set the storage element information

**Parameters:**

*info* the StorageElementInfo object

#### 6.3.3.10 void FileInfo::setSize (final long *size*)

Set the file size.

**Parameters:**

*size* The size in bytes

### 6.3.3..11 String FileInfo::toString ()

Print contents into a string.

## 6.4. MODE CLASS REFERENCE

### PUBLIC METHODS

- `Mode` (`int p_mode`)
- `Mode` (`Mode p_mode`)
- `boolean equals` (`Mode p_mode`)
- `int hashCode` ()
- `int getMode` ()
- `String toString` ()

### STATIC PUBLIC ATTRIBUTES

- `final int READ_MODE` = 0
- `final int WRITE_MODE` = 1
- `final Mode READ` = `new Mode(READ_MODE)`
- `final Mode WRITE` = `new Mode(WRITE_MODE)`

#### 6.4.1. DETAILED DESCRIPTION

Represents a transfer mode.

Usage: `Mode.READ` or `Mode.WRITE`

#### 6.4.2. CONSTRUCTOR & DESTRUCTOR DOCUMENTATION

##### 6.4.2..1 `Mode::Mode (int p_mode)`

##### 6.4.2..2 `Mode::Mode (Mode p_mode)`

#### 6.4.3. MEMBER FUNCTION DOCUMENTATION

##### 6.4.3..1 `boolean Mode::equals (Mode p_mode)`

##### 6.4.3..2 `int Mode::getMode ()`

Returns the mode.

**Returns:**

`int`

##### 6.4.3..3 `int Mode::hashCode ()`

#### 6.4.3.4 String Mode::toString ()

### 6.4.4. MEMBER DATA DOCUMENTATION

6.4.4.1 final Mode Mode::READ = new Mode(READ\_MODE) [static]

6.4.4.2 final int Mode::READ\_MODE = 0 [static]

6.4.4.3 final Mode Mode::WRITE = new Mode(WRITE\_MODE) [static]

6.4.4.4 final int Mode::WRITE\_MODE = 1 [static]

## 6.5. STORAGEELEMENT INTERFACE REFERENCE

### 6.5.1. DETAILED DESCRIPTION

Interface to a generic Storage Element.

**Author:**

Peter Kunszt

**Author:**

modifications by Steve Hicks

**Version:**

**Id:**

StorageElement.java,v 1.2 2003/04/02 10:31:44 shicks Exp

## 6.6. TRANSFERFILEHANDLE INTERFACE REFERENCE

### 6.6.1. DETAILED DESCRIPTION

TransferFileHandle defines basic actions on a transfer file (TURI).

**Author:**

Steve Hicks

## 7. NETWORK

### 7.1. NETWORKCOST CLASS REFERENCE

**PUBLIC METHODS**

- float `getCost ()`
- float `getError ()`

- void `setCost` (float *cost*)
- void `setError` (float *error*)
- `NetworkCost` ()

### 7.1.1. DETAILED DESCRIPTION

#### Author :

Robert Harakaly Container class to store results of network cost calculation

### 7.1.2. CONSTRUCTOR & DESTRUCTOR DOCUMENTATION

#### 7.1.2..1 `NetworkCost::NetworkCost` ()

Creates a new instance of `networkCost`

### 7.1.3. MEMBER FUNCTION DOCUMENTATION

#### 7.1.3..1 float `NetworkCost::getCost` ()

#### 7.1.3..2 float `NetworkCost::getError` ()

#### 7.1.3..3 void `NetworkCost::setCost` (float *cost*)

#### 7.1.3..4 void `NetworkCost::setError` (float *error*)

## 7.2. NETWORKCOSTCONTEXTIF INTERFACE REFERENCE

### PUBLIC METHODS

- void `setDuplex` (boolean *duplex*)
- boolean `getDuplex` ()

### PUBLIC ATTRIBUTES

- boolean `checkDuplex` = true
- final int `TP_GRIDFTP` = 1
- final int `TP_RFIO` = 2

### STATIC PUBLIC ATTRIBUTES

- final String `VERSION` = "V1.5.0"

### 7.2.1. DETAILED DESCRIPTION

Network Cost Suite (NCS) is a JAVA library implementing evaluation of the `data` transfer cost over the network. It consists of public interface class `NetworkCostContext` the cost function implementation and the set of the "backend" classes (`<Storage>NetworkCostContext`). Public interface of the NCS is the interface class `NetworkCostContext`. This interface class is implemented in each backend class thus `getNetworkCosts` function can be called from any `<Storage>NetworkCostContext` class.

Available backend classes:

- `MdsNetworkCostContext`
- `RGMAArchiverNetworkCostContext`
- `WgetNetworkCostContext`
- `DummyNetworkCostContext` (only for testing purposes)

Interface class defining interface and constants used for building `NetCostFunction` "backends". `NetCostFunction` suite consists of `NetworkCostFunction` class calculating the cost of transfer and estimation error and "backend". Backend defines the access "protocol" to the `data` storage, retrieves `data` and returns the result in desired format. Backend class must **extend** `NetworkCostFunction` class and **implement** `NetworkCostContext` interface class

### 7.2.2. MEMBER FUNCTION DOCUMENTATION

**7.2.2.1** `boolean NetworkCostContextIF::getDuplex ()`

**7.2.2.2** `void NetworkCostContextIF::setDuplex (boolean duplex)`

### 7.2.3. MEMBER DATA DOCUMENTATION

**7.2.3.1** `boolean NetworkCostContextIF::checkDuplex = true`

**7.2.3.2** `final int NetworkCostContextIF::TP_GRIDFTP = 1`

**7.2.3.3** `final int NetworkCostContextIF::TP_RFIO = 2`

**7.2.3.4** `final String NetworkCostContextIF::VERSION = "V1.5.0" [static]`

## 8. JAVA SECURITY

### 8.1. SECURITYINFO INTERFACE REFERENCE

#### PUBLIC METHODS

- `String getAuthorizationPolicy ()`

- List `getAuthorizedAttributes ()`
- List `getRequestedAttributes ()`
- `X509Certificate` `getClientCert ()`
- `X509Certificate[]` `getClientCertChain ()`
- String `getClientName ()`
- `VOMSEExtension` `getVOMSEExtension ()`

### 8.1.1. DETAILED DESCRIPTION

An interface from which an external application can get information from underlying authentication and authorization processes.

**See also:**

`SecurityInfoContainer`

**Author:**

mulmo

### 8.1.2. MEMBER FUNCTION DOCUMENTATION

#### 8.1.2.1 String `SecurityInfo::getAuthorizationPolicy ()`

**Returns:**

String The name of the policy used in the authorization process

#### 8.1.2.2 List `SecurityInfo::getAuthorizedAttributes ()`

**Returns:**

List of String (the approved authorization attributes)

**See also:**

`getRequestedAttributes ()`

#### 8.1.2.3 `X509Certificate` `SecurityInfo::getClientCert ()`

**Returns:**

`X509Certificate` The identity certificate of the authenticated client

#### 8.1.2.4 `X509Certificate []` `SecurityInfo::getClientCertChain ()`

**Returns:**

`X509Certificate[]` The client's certificate chain

### 8.1.2..5 String SecurityInfo::getClientName ()

Returns the name of the authenticated client. Typically, this is the Subject Distinguished Name of the client certificate.

**Returns:**

String The name of the authenticated client.

### 8.1.2..6 List SecurityInfo::getRequestedAttributes ()

**Returns:**

List of String (the requested authorization attributes)

**See also:**

[getAuthorizedAttributes\(\)](#)

### 8.1.2..7 VOMSEExtension SecurityInfo::getVOMSEExtension ()

**Returns:**

[VOMSEExtension](#) a VOMS extension associated with the client

**See also:**

[org.edg.security.voms.VOMSEExtension](#)

## 8.2. SECURITYINFOCONTAINER CLASS REFERENCE

### STATIC PUBLIC METHODS

- [SecurityInfo](#) [getSecurityInfo](#) ()

#### 8.2.1. DETAILED DESCRIPTION

Container class from which the current [SecurityInfo](#) can be retrieved. [Security-Info](#) instances are created, one per thread, by special Servlet and/or SOAP plugins. See the installation guide for more information about these plugins and how they are configured

**Author:**

mulmo

**See also:**

[SecurityInfo](#)

## 8.2.2. MEMBER FUNCTION DOCUMENTATION

### 8.2.2.1 **SecurityInfo** SecurityInfoContainer::getSecurityInfo () [static]

**Returns:**

The **SecurityInfo** associated with the currently running thread, or null if no such object have been previously assigned by the special security plugins.

**See also:**

**SecurityInfo**

## 8.3. VOMSEXTENSION CLASS REFERENCE

### PUBLIC METHODS

- **VOMSExtension** (X509Certificate cert) throws Exception
- **VOMSExtension** (byte[] extensionValue) throws Exception
- List **getVOMSInfos** ()
- String **toString** ()

### STATIC PUBLIC METHODS

- **VOMSExtension** **fromCert** (X509Certificate cert)

### STATIC PUBLIC ATTRIBUTES

- final String **VOMS\_OID** = "1.3.6.1.4.1.8005.100.100.1"

### PROTECTED METHODS

- int **parse** (byte[] blob, int index, VOMSInfo info) throws Exception

### 8.3.1. DETAILED DESCRIPTION

Parses the information from and verifies the signature of a VOMS extension in a certificate.

One **VOMSExtension** may contain several **VOMSInfo** instances.

**Author:**

mulmo

**See also:**

**VOMSInfo**

## 8.3.2. CONSTRUCTOR & DESTRUCTOR DOCUMENTATION

### 8.3.2.1 VOMSEExtension::VOMSEExtension (X509Certificate *cert*) throws Exception

**Parameters:**

*cert* the Certificate to parse out the extension from

**Exceptions:**

*Exception* in case of error

**See also:**

VOMSEExtension(byte[])

### 8.3.2.2 VOMSEExtension::VOMSEExtension (byte *extensionValue*[]) throws Exception

**Parameters:**

*extensionValue* the content of the VOMS extension, extracted from a proxy certificate

**Exceptions:**

*Exception* in case of error

## 8.3.3. MEMBER FUNCTION DOCUMENTATION

### 8.3.3.1 VOMSEExtension VOMSEExtension::fromCert (X509Certificate *cert*) [static]

Returns a VOMS extension object from the `data` contained in the certificate, or null</null> if no such extension was present.

**Parameters:**

*cert* the Certificate to parse out the extension from

**Returns:**

VOMSEExtension if the cert contained such an extension, or null

**See also:**

VOMSEExtension(X509Certificate)

### 8.3.3.2 List VOMSEExtension::getVOMSInfos ()

**Returns:**

List of VOMSInfo contained in this extension

**See also:**

VOMSInfo

**8.3.3.3 int VOMSExtension::parse (byte *blob*[], int *index*, VOMSInfo *info*) throws Exception**  
[protected]

**Parameters:**

- blob* the binary object to dissect
- index* the start index in the binary object
- info* the VOMSInfo object to populate with *data*

**Returns:**

int the index of the next VOMSInfo, or -1

**8.3.3.4 String VOMSExtension::toString ()**

**8.3.4. MEMBER DATA DOCUMENTATION**

**8.3.4.1 final String VOMSExtension::VOMS\_OID = "1.3.6.1.4.1.8005.100.100.1"** [static]

The VOMS extension OID

## 9. VOMS

### 9.1. VOMS STRUCT REFERENCE

#### PUBLIC ATTRIBUTES

- int *siglen*
- string *signature*
- string *user*
- string *userca*
- string *server*
- string *serverca*
- string *voname*
- string *uri*
- string *date1*
- string *date2*
- *data\_type* type
- vector< data > *std*
- string *custom*

#### 9.1.1. MEMBER DATA DOCUMENTATION

**9.1.1.1 string voms::custom**

**9.1.1.2 string voms::date1**

**9.1.1.3 string voms::date2**

9.1.1.4 string voms::server

9.1.1.5 string voms::serverca

9.1.1.6 int voms::siglen

9.1.1.7 string voms::signature

9.1.1.8 vector<data> voms::std

9.1.1.9 data\_type voms::type

9.1.1.10 string voms::uri

9.1.1.11 string voms::user

9.1.1.12 string voms::userca

9.1.1.13 string voms::voname

## 9.2. DATA STRUCT REFERENCE

### PUBLIC ATTRIBUTES

- string group
- string role
- string cap

### 9.2.1. MEMBER DATA DOCUMENTATION

9.2.1.1 string data::cap

9.2.1.2 string data::group

9.2.1.3 string data::role

## 9.3. EXT\_VOMS STRUCT REFERENCE

### PUBLIC ATTRIBUTES

- voms voms
- int datalen
- string real\_data
- string signed\_data

### 9.3.1. MEMBER DATA DOCUMENTATION

9.3.1.1 `int ext_voms::datalen`

9.3.1.2 `string ext_voms::real_data`

9.3.1.3 `string ext_voms::signed_data`

9.3.1.4 `struct voms ext_voms::voms`

### 9.4. VOMS STRUCT REFERENCE

#### PUBLIC ATTRIBUTES

- `int siglen`
- `string signature`
- `string user`
- `string userca`
- `string server`
- `string serverca`
- `string voname`
- `string uri`
- `string date1`
- `string date2`
- `data_type type`
- `vector< data > std`
- `string custom`

### 9.4.1. MEMBER DATA DOCUMENTATION

9.4.1.1 `string voms::custom`

9.4.1.2 `string voms::date1`

9.4.1.3 `string voms::date2`

9.4.1.4 `string voms::server`

9.4.1.5 `string voms::serverca`

9.4.1.6 `int voms::siglen`

9.4.1.7 `string voms::signature`

9.4.1..8 `vector<data> voms::std`

9.4.1..9 `data_type voms::type`

9.4.1..10 `string voms::uri`

9.4.1..11 `string voms::user`

9.4.1..12 `string voms::userca`

9.4.1..13 `string voms::voname`

## 9.5. VOMSDATA STRUCT REFERENCE

### PUBLIC ATTRIBUTES

- `vector< voms > data`
- `string workvo`
- `string extra_data`

### 9.5.1. MEMBER DATA DOCUMENTATION

9.5.1..1 `vector<voms> vomsdata::data`

9.5.1..2 `string vomsdata::extra_data`

9.5.1..3 `string vomsdata::workvo`

## 9.6. VOMSADMIN INTERFACE REFERENCE

### 9.6.1. DETAILED DESCRIPTION

Virtual Organisation Membership Service Administration interface.

**Version:**

**Name:**

**Author:**

Akos Frohner , Karoly Lorentey

## 9.7. VOMSCOMPATIBILITY INTERFACE REFERENCE

### 9.7.1. DETAILED DESCRIPTION

Virtual Organisation Membership Service Compatibility service for the mkgridmap utility.

**Version:**

**Name:**

## 9.8. VOMSCORE INTERFACE REFERENCE

### 9.8.1. DETAILED DESCRIPTION

Virtual Organisation Membership Service Core interface.

**Version:**

**Name:**

**Author:**

Akos Frohner , Karoly Lorentey

## 9.9. VOMSHISTORY INTERFACE REFERENCE

### 9.9.1. DETAILED DESCRIPTION

Virtual Organisation Membership Service History interface. This service provides methods for "back-in-time" queries, which could describe the context of a situation based on the state of the database at a given transaction.

**Version:**

**Name:**

**Author:**

Akos Frohner , Karoly Lorentey

## 9.10. VOMSREQUEST INTERFACE REFERENCE

### 9.10.1. DETAILED DESCRIPTION

Virtual Organisation Membership Service user request interface.

**Version:**

**Name:**

**Author:**

Akos Frohner , Karoly Lorentey

## 9.11. ACLENTY CLASS REFERENCE

### PUBLIC METHODS

- `ACLEntry ()`
- `String getAdminDN ()`
- `String getAdminCA ()`
- `void setAdminDN (String dn)`
- `void setAdminCA (String ca)`
- `String getOperationName ()`
- `void setOperationName (String operation) throws RemoteException`
- `boolean isAllow ()`
- `void setAllow (boolean allow)`

#### 9.11.1. DETAILED DESCRIPTION

Represents **ACL entries** within the VOMS database.

**Version:**

**Name:**

**Author:**

Akos Frohner

#### 9.11.2. CONSTRUCTOR & DESTRUCTOR DOCUMENTATION

##### 9.11.2..1 ACLEntry::ACLEntry ()

#### 9.11.3. MEMBER FUNCTION DOCUMENTATION

##### 9.11.3..1 String ACLEntry::getAdminCA ()

##### 9.11.3..2 String ACLEntry::getAdminDN ()

Returns the principal of this ACL entry.

### 9.11.3.3 String ACLEntry::getOperationName ()

Returns the operation field of this ACL entry.

### 9.11.3.4 boolean ACLEntry::isAllow ()

Returns the allow field of this ACL entry.

### 9.11.3.5 void ACLEntry::setAdminCA (String ca)

### 9.11.3.6 void ACLEntry::setAdminDN (String dn)

Sets the principal of this ACL entry.

### 9.11.3.7 void ACLEntry::setAllow (boolean allow)

Sets the allow field of this ACL entry.

### 9.11.3.8 void ACLEntry::setOperationName (String operation) throws RemoteException

Sets the operation field of this ACL entry.

## 9.12. QUALIFIEDROLE CLASS REFERENCE

### PUBLIC METHODS

- void `setGroup` (String groupname)
- String `getGroup` ()
- void `setRole` (String rolename)
- String `getRole` ()

### 9.12.1. DETAILED DESCRIPTION

A class describing a role instance, i.e. a role qualified by a group name.

### 9.12.2. MEMBER FUNCTION DOCUMENTATION

#### 9.12.2.1 String QualifiedRole::getGroup ()

Get the name of the group associated with this role instance.

#### 9.12.2.2 String QualifiedRole::getRole ()

Get the name of the role.

#### 9.12.2.3 void QualifiedRole::setGroup (String groupname)

Set the name of the group associated with this role instance.

#### 9.12.2..4 void QualifiedRole::setRole (String rolename)

Set the name of the role.

### 9.13. CREATEUSERREQUEST CLASS REFERENCE

Inherits [Request](#).

#### PUBLIC METHODS

- void [setUser](#) (User user)
- User [getUser](#) ()

#### 9.13.1. DETAILED DESCRIPTION

A class describing requests for user creation.

#### Author:

Karoly Lorentey

#### 9.13.2. MEMBER FUNCTION DOCUMENTATION

##### 9.13.2..1 User CreateUserRequest::getUser ()

Get the user to be added.

##### 9.13.2..2 void CreateUserRequest::setUser (User user)

Set the user to be added.

### 9.14. REQUEST CLASS REFERENCE

Inherited by [CreateUserRequest](#).

#### PUBLIC METHODS

- void [setId](#) (String id)
- String [getId](#) ()
- void [setClientEmail](#) (String email)
- String [getClientEmail](#) ()
- void [setClientComment](#) (String comment)
- String [getClientComment](#) ()
- void [setStatus](#) (String status)
- String [getStatus](#) ()
- void [setAdminComment](#) (String comment)
- String [getAdminComment](#) ()

### 9.14.1. DETAILED DESCRIPTION

An abstract class containing the common attributes of all user requests.

**Author :**

Karoly Lorentey

### 9.14.2. MEMBER FUNCTION DOCUMENTATION

#### 9.14.2.1 String Request::getAdminComment ()

Get the comment text provided by the administrator who processed this request.

#### 9.14.2.2 String Request::getClientComment ()

Get the comment text provided by the user who created this request.

#### 9.14.2.3 String Request::getClientEmail ()

Get the email address of the user who created this request.

#### 9.14.2.4 String Request::getId ()

Get the unique id of this request.

#### 9.14.2.5 String Request::getStatus ()

Get the status of this request.

#### 9.14.2.6 void Request::setAdminComment (String *comment*)

Set the comment text provided by the administrator who processed this request.

#### 9.14.2.7 void Request::setClientComment (String *comment*)

Set the comment text provided by the user who created this request.

#### 9.14.2.8 void Request::setClientEmail (String *email*)

Set the email address of the user who created this request.

#### 9.14.2.9 void Request::setId (String *id*)

Set the unique id of the request.

#### 9.14.2.10 void Request::setStatus (String *status*)

Set the status of this request.

## 9.15. USER CLASS REFERENCE

### PUBLIC METHODS

- `User ()`
- `String getDN ()`
- `void setDN (String dn)`
- `String getCA ()`
- `void setCA (String ca)`
- `String getCN ()`
- `void setCN (String cn)`
- `String getCertUri ()`
- `void setCertUri (String certUri)`
- `String getMail ()`
- `void setMail (String mail)`

### 9.15.1. DETAILED DESCRIPTION

Identifies a user in the VOMS database.

**Version:**

**Name:**

**Author:**

Akos Frohner

### 9.15.2. CONSTRUCTOR & DESTRUCTOR DOCUMENTATION

#### 9.15.2.1 `User::User ()`

### 9.15.3. MEMBER FUNCTION DOCUMENTATION

#### 9.15.3.1 `String User::getCA ()`

Gets the distinguished name of the CA that issued the certificate of this user.

#### 9.15.3.2 `String User::getCertUri ()`

Gets the URL of the user's certificate.

#### 9.15.3.3 `String User::getCN ()`

Gets the common name.

#### 9.15.3.4 String User::getDN ()

Gets the distinguished name of this user.

#### 9.15.3.5 String User::getMail ()

Gets the email address of the user.

#### 9.15.3.6 void User::setCA (String *ca*)

#### 9.15.3.7 void User::setCertUri (String *certUri*)

#### 9.15.3.8 void User::setCN (String *cn*)

#### 9.15.3.9 void User::setDN (String *dn*)

#### 9.15.3.10 void User::setMail (String *mail*)

## 10. LCAS

### 10.1. LCAS\_CRED\_ID\_S STRUCT REFERENCE

structure representing an LCAS credential.

#### PUBLIC ATTRIBUTES

- gss\_cred\_id\_t *cred*
- char \* *dn*

#### 10.1.1. DETAILED DESCRIPTION

structure representing an LCAS credential.

#### 10.1.2. MEMBER DATA DOCUMENTATION

##### 10.1.2.1 gss\_cred\_id\_t lcas\_cred\_id\_s::cred

the original gss (globus) credential

##### 10.1.2.2 char\* lcas\_cred\_id\_s::dn

the user distinguished name (DN)

## 11. LCMAPS

### 11.1. CRED\_DATA\_S STRUCT REFERENCE

#### PUBLIC ATTRIBUTES

- char \* dn
- uid\_t \* uid
- int cntUid
- gid\_t \* priGid
- int cntPriGid
- gid\_t \* secGid
- int cntSecGid

#### 11.1.1. MEMBER DATA DOCUMENTATION

11.1.1.1 int cred\_data\_s::cntPriGid

11.1.1.2 int cred\_data\_s::cntSecGid

11.1.1.3 int cred\_data\_s::cntUid

11.1.1.4 char\* cred\_data\_s::dn

11.1.1.5 gid\_t\* cred\_data\_s::priGid

11.1.1.6 gid\_t\* cred\_data\_s::secGid

11.1.1.7 uid\_t\* cred\_data\_s::uid

### 11.2. LCMAPS\_ARGUMENT\_S STRUCT REFERENCE

#### PUBLIC ATTRIBUTES

- char \* argName
- char \* argType
- int argInOut
- void \* value

#### 11.2.1. MEMBER DATA DOCUMENTATION

11.2.1.1 int lcmaps\_argument\_s::argInOut

11.2.1.2 char\* lcmaps\_argument\_s::argName

### 11.2.1.3 char\* lcmaps\_argument\_s::argType

### 11.2.1.4 void\* lcmaps\_argument\_s::value

## 11.3. LCMAPS\_CRED\_ID\_S STRUCT REFERENCE

structure representing an LCMAPS credential.

### PUBLIC ATTRIBUTES

- gss\_cred\_id\_t cred
- char \* dn

### 11.3.1. DETAILED DESCRIPTION

structure representing an LCMAPS credential.

### 11.3.2. MEMBER DATA DOCUMENTATION

#### 11.3.2.1 gss\_cred\_id\_t lcmaps\_cred\_id\_s::cred

the original gss (globus) credential

#### 11.3.2.2 char\* lcmaps\_cred\_id\_s::dn

the user distinguished name (DN)

## 12. SLASHGRID

### 12.1. DIRENTRY STRUCT REFERENCE

### PUBLIC ATTRIBUTES

- unsigned long d\_fileno
- unsigned short d\_reclen
- unsigned char d\_type
- unsigned char d\_namlen
- char d\_name [SLGR\_MAXNAMLEN+1]
- off\_t d\_size
- time\_t d\_ctime
- time\_t d\_mtime
- time\_t d\_atime

### 12.1.1. MEMBER DATA DOCUMENTATION

#### 12.1.1.1 time\_t DirEntry::d\_atime

12.1.1..2 `time_t DirEntry::d_ctime`

12.1.1..3 `unsigned long DirEntry::d_fileno`

12.1.1..4 `time_t DirEntry::d_mtime`

12.1.1..5 `char DirEntry::d_name[SLGR_MAXNAMLEN + 1]`

12.1.1..6 `unsigned char DirEntry::d_namlen`

12.1.1..7 `unsigned short DirEntry::d_reclen`

12.1.1..8 `off_t DirEntry::d_size`

12.1.1..9 `unsigned char DirEntry::d_type`

## 12.2. FSPLUGIN STRUCT REFERENCE

### PUBLIC ATTRIBUTES

- `mntent fstabline`
- `char * sofile`
- `char * version`
- `void(* PluginInit )(struct plugparam *, struct fsplugin *)`
- `void(* PluginStat )(struct plugparam *, struct fsplugin *)`
- `void(* PluginUserPerm )(struct plugparam *, struct fsplugin *)`
- `void(* PluginExclPerm )(struct plugparam *, struct fsplugin *)`
- `void(* PluginFileType )(struct plugparam *, struct fsplugin *)`
- `void(* PluginOpen )(struct plugparam *, struct fsplugin *)`
- `void(* PluginOpenFd )(struct plugparam *, struct fsplugin *)`
- `void(* PluginClose )(struct plugparam *, struct fsplugin *)`
- `void(* PluginRemove )(struct plugparam *, struct fsplugin *)`
- `void(* PluginRename )(struct plugparam *, struct fsplugin *)`
- `void(* PluginSymlink )(struct plugparam *, struct fsplugin *)`
- `void(* PluginReadlink )(struct plugparam *, struct fsplugin *)`
- `void(* PluginRmdir )(struct plugparam *, struct fsplugin *)`
- `void(* PluginMkdir )(struct plugparam *, struct fsplugin *)`
- `void(* PluginCreate )(struct plugparam *, struct fsplugin *)`
- `void(* PluginSetExec )(struct plugparam *, struct fsplugin *)`
- `void * handle`
- `void * next`
- `void * instance`

## 12.2.1. MEMBER DATA DOCUMENTATION

12.2.1.1 struct `mntent fsplugin::fstabline`

12.2.1.2 void\* `fsplugin::handle`

12.2.1.3 void\* `fsplugin::instance`

12.2.1.4 void\* `fsplugin::next`

12.2.1.5 void(\* `fsplugin::PluginClose`)(struct `plugparam *`, struct `fsplugin *`)

12.2.1.6 void(\* `fsplugin::PluginCreate`)(struct `plugparam *`, struct `fsplugin *`)

12.2.1.7 void(\* `fsplugin::PluginExclPerm`)(struct `plugparam *`, struct `fsplugin *`)

12.2.1.8 void(\* `fsplugin::PluginFileType`)(struct `plugparam *`, struct `fsplugin *`)

12.2.1.9 void(\* `fsplugin::PluginInit`)(struct `plugparam *`, struct `fsplugin *`)

12.2.1.10 void(\* `fsplugin::PluginMkdir`)(struct `plugparam *`, struct `fsplugin *`)

12.2.1.11 void(\* `fsplugin::PluginOpen`)(struct `plugparam *`, struct `fsplugin *`)

12.2.1.12 void(\* `fsplugin::PluginOpenFd`)(struct `plugparam *`, struct `fsplugin *`)

12.2.1.13 void(\* `fsplugin::PluginReadlink`)(struct `plugparam *`, struct `fsplugin *`)

12.2.1.14 void(\* `fsplugin::PluginRemove`)(struct `plugparam *`, struct `fsplugin *`)

12.2.1.15 void(\* `fsplugin::PluginRename`)(struct `plugparam *`, struct `fsplugin *`)

12.2.1.16 void(\* `fsplugin::PluginRmdir`)(struct `plugparam *`, struct `fsplugin *`)

12.2.1.17 void(\* `fsplugin::PluginSetExec`)(struct `plugparam *`, struct `fsplugin *`)

12.2.1.18 void(\* `fsplugin::PluginStat`)(struct `plugparam *`, struct `fsplugin *`)

12.2.1.19 void(\* `fsplugin::PluginSymlink`)(struct `plugparam *`, struct `fsplugin *`)

12.2.1.20 void(\* `fsplugin::PluginUserPerm`)(struct `plugparam *`, struct `fsplugin *`)

12.2.1..21 char\* fsplugin::sofile

12.2.1..22 char\* fsplugin::version

### 12.3. PLUGPARAM STRUCT REFERENCE

#### PUBLIC ATTRIBUTES

- unsigned long unique
- pid\_t pid
- uidentry \* uidentry
- stat stat
- GACLperm perm
- int type
- int flags
- int fd
- char source [SLGR\_MAXPATHLEN+SLGR\_MAXNAMLEN+2]
- char target [SLGR\_MAXPATHLEN+SLGR\_MAXNAMLEN+2]
- int result

#### 12.3.1. MEMBER DATA DOCUMENTATION

12.3.1..1 int plugparam::fd

12.3.1..2 int plugparam::flags

12.3.1..3 GACLperm plugparam::perm

12.3.1..4 pid\_t plugparam::pid

12.3.1..5 int plugparam::result

12.3.1..6 char plugparam::source[SLGR\_MAXPATHLEN+SLGR\_MAXNAMLEN+2]

12.3.1..7 struct stat plugparam::stat

12.3.1..8 char plugparam::target[SLGR\_MAXPATHLEN+SLGR\_MAXNAMLEN+2]

12.3.1..9 int plugparam::type

12.3.1..10 struct uidentry\* plugparam::uidentry

12.3.1..11 unsigned long plugparam::unique

## 12.4. UIDENTRY STRUCT REFERENCE

### PUBLIC ATTRIBUTES

- `uid_t uid`
- `GACLuser * user`
- `char * certfile`
- `time_t certfiletime`
- `time_t certexpires`
- `char * keypass`
- `uidentry * next`

### 12.4.1. MEMBER DATA DOCUMENTATION

12.4.1.1 `time_t uidentry::certexpires`

12.4.1.2 `char* uidentry::certfile`

12.4.1.3 `time_t uidentry::certfiletime`

12.4.1.4 `char* uidentry::keypass`

12.4.1.5 `struct uidentry* uidentry::next`

12.4.1.6 `uid_t uidentry::uid`

12.4.1.7 `GACLuser* uidentry::user`

## 13. GRID ACCESS CONTROL

### 13.1. \_GACLAcl STRUCT REFERENCE

#### PUBLIC ATTRIBUTES

- `GACLentry * firstentry`

### 13.1.1. MEMBER DATA DOCUMENTATION

13.1.1.1 `GACLentry* _GACLacl::firstentry`

## 13.2. \_GACLcred STRUCT REFERENCE

### PUBLIC ATTRIBUTES

- char \* type
- \_GACLnamevalue \* firstname
- \_GACLcred \* next

### 13.2.1. MEMBER DATA DOCUMENTATION

13.2.1.1 struct [\\_GACLnamevalue](#)\* \_GACLcred::firstname

13.2.1.2 struct [\\_GACLcred](#)\* \_GACLcred::next

13.2.1.3 char\* \_GACLcred::type

## 13.3. \_GACLEntry STRUCT REFERENCE

### PUBLIC ATTRIBUTES

- [GACLcred](#) \* firstcred
- [GACLperm](#) allowed
- [GACLperm](#) denied
- void \* next

### 13.3.1. MEMBER DATA DOCUMENTATION

13.3.1.1 [GACLperm](#) \_GACLEntry::allowed

13.3.1.2 [GACLperm](#) \_GACLEntry::denied

13.3.1.3 [GACLcred](#)\* \_GACLEntry::firstcred

13.3.1.4 void\* \_GACLEntry::next

## 13.4. \_GACLNAMEVALUE STRUCT REFERENCE

### PUBLIC ATTRIBUTES

- char \* name
- char \* value
- \_GACLnamevalue \* next

### 13.4.1. MEMBER DATA DOCUMENTATION

13.4.1.1 char\* \_GACLnamevalue::name

13.4.1..2 struct `_GACLnamevalue*` `_GACLnamevalue::next`

13.4.1..3 char\* `_GACLnamevalue::value`

## 13.5. `_GACLUSER` STRUCT REFERENCE

### PUBLIC ATTRIBUTES

- `GACLcred * firstcred`

### 13.5.1. MEMBER DATA DOCUMENTATION

13.5.1..1 `GACLcred*` `_GACLuser::firstcred`

## 14. GRIDSITE

### 14.1. BODYPTR STRUCT REFERENCE

#### PUBLIC ATTRIBUTES

- `size_t size`
- `listchar * first`
- `listchar * last`

### 14.1.1. MEMBER DATA DOCUMENTATION

14.1.1..1 struct `listchar*` `bodyptr::first`

14.1.1..2 struct `listchar*` `bodyptr::last`

14.1.1..3 `size_t bodyptr::size`

### 14.2. LISTCHAR STRUCT REFERENCE

#### PUBLIC ATTRIBUTES

- `char * text`
- `listchar * next`

### 14.2.1. MEMBER DATA DOCUMENTATION

14.2.1..1 struct `listchar*` `listchar::next`

14.2.1..2 char\* `listchar::text`

## INDEX

- ~BrokerInfo
  - BrokerInfo, 31
- ~metricBase
  - metricBase, 108
- ~metricParams
  - metricParams, 110
- .\_GACLacl, 141
  - firstentry, 141
- .\_GACLcred, 142
  - firstname, 142
  - next, 142
  - type, 142
- .\_GACLentry, 142
  - allowed, 142
  - denied, 142
  - firstcred, 142
  - next, 142
- .\_GACLnamevalue, 142
  - name, 142
  - next, 142
  - value, 143
- .\_GACLuser, 143
  - firstcred, 143
- abort
  - Consumer, 57
- ABORTED
  - JobStatus, 23
- ACLEntry, 130
  - ACLEntry, 130
  - getAdminCA, 130
  - getAdminDN, 130
  - getOperationName, 130
  - isAllow, 131
  - setAdminCA, 131
  - setAdminDN, 131
  - setAllow, 131
  - setOperationName, 131
- add
  - Archiver, 47
  - DataBaseProducer, 63
  - FixedPoint, 68
- addAlias
  - EdgReplicaMetadataCatalog, 37
- addAttribute
  - JobAd, 9, 10
- addMapping
  - EdgLocalReplicaCatalog, 34
- addParam
  - metricParams, 110
- addParameter
  - ServletConnection, 87, 88
- after
  - Time, 95
- afterLast
  - ResultSet, 79
- aliasExists
  - EdgReplicaMetadataCatalog, 37
- allowed
  - .\_GACLentry, 142
- APIBase, 40
  - APIBase, 41
  - calcTerminationInterval, 41
  - cat, 45
  - checkIfConnected, 41
  - close, 41
  - connectionId, 45
  - currentState, 45
  - DEFAULT\_TERMINATION\_INTERVAL, 45
  - disconnect, 41
  - exceptionType, 45
  - finalize, 41
  - getProperties, 41
  - getServletConnection, 42
  - getStatus, 42
  - getTerminationInterval, 42
  - getTupleChecking, 42
  - IS\_CLOSED, 45
  - IS\_CONNECTED, 45
  - IS\_DISCONNECTED, 45
  - nl, 45
  - reconnect, 42
  - sendCommand, 43
  - servletURL, 45
  - setTerminationInterval, 44
  - setTupleChecking, 44
  - showSignOfLife, 44
- archive
  - CircularBufferProducer, 53
  - DataBaseProducer, 64
- Archiver, 45
  - add, 47
  - Archiver, 46, 47
  - declareTable, 48
  - getBufferSize, 48

setBufferSize, 49  
argInOut  
    lcmaps\_argument\_s, 136  
argName  
    lcmaps\_argument\_s, 136  
argType  
    lcmaps\_argument\_s, 136  
attNames  
    JobStatus, 23  
attributeDefinitionExists  
    EdgLocalReplicaCatalog, 34  
    EdgReplicaMetadataCatalog, 37  
attrNames  
    Event, 4  
  
before  
    Time, 95  
beforeFirst  
    ResultSet, 79  
blockingPop  
    Consumer, 57, 58  
bodyptr, 143  
    first, 143  
    last, 143  
    size, 143  
BrokerInfo, 30  
    ~BrokerInfo, 31  
    getAccessCost, 31  
    getBIFFileName, 31  
    getCE, 31  
    getCloseSEs, 31  
    getDataAccessProtocol, 31  
    getInputData, 31  
    getLFN2SFN, 31  
    getSEMountPoint, 32  
    getSEPort, 32  
    getSEProtocols, 32  
    getSEs, 32  
    getVirtualOrganization, 32  
    instance, 32  
    parser, 32  
BrokerInfoEx, 33  
byteValue  
    FixedPoint, 68  
  
calcTerminationInterval  
    APIBase, 41  
cancel  
    JobCollection, 17  
    UserJobs, 29  
CANCEL\_REASON  
    JobStatus, 23  
CANCELLED  
    JobStatus, 23  
CANCELLING  
    JobStatus, 23  
CanonicalProducer  
    CanonicalProducer, 50  
CanonicalProducer, 49  
    CanonicalProducer, 50  
HISTORY, 51  
LATEST, 51  
canPop  
    Consumer, 58  
cap  
    data, 126  
cat  
    APIBase, 45  
    ReplicaConfigReader, 77  
CE\_NODE  
    JobStatus, 23  
certexpires  
    uidentry, 141  
certfile  
    uidentry, 141  
certfiletime  
    uidentry, 141  
check  
    JobAd, 10  
    JobState, 21  
checkAll  
    JobAd, 10  
checkAttribute  
    JobState, 21  
checkDuplex  
    NetworkCostContextIF, 120  
checkIfConnected  
    APIBase, 41  
checkProxy  
    UserCredential, 28  
CHILDREN  
    JobStatus, 23  
CHILDREN\_HIST  
    JobStatus, 23  
CHILDREN\_NUM  
    JobStatus, 23  
CHILDREN\_STATES  
    JobStatus, 23  
CircularBufferProducer  
    CircularBufferProducer, 52  
CircularBufferProducer, 51

---

archive, 53  
CircularBufferProducer, 52  
getProducerConnection, 52  
getRemoteBufferSize, 52  
setRemoteBufferSize, 52  
setTerminationInterval, 52  
CLASSAD  
  Event, 4  
Cleanable, 53  
  Cleanable, 53  
  declareTable, 54  
clear  
  JobAd, 10  
  JobCollection, 17  
  JobId, 20  
CLEARED  
  JobStatus, 23  
close  
  APIBase, 41  
cntPriGid  
  cred\_data\_s, 136  
cntSecGid  
  cred\_data\_s, 136  
cntUid  
  cred\_data\_s, 136  
code  
  Event, 4  
  JobStatus, 24  
compareTo  
  FixedPoint, 68  
  Time, 95  
CONDOR\_ID  
  JobStatus, 24  
CONDOR\_JDL  
  JobStatus, 24  
connect  
  ServletConnection, 88  
connectionId  
  APIBase, 45  
Consumer, 54  
  abort, 57  
  blockingPop, 57, 58  
  canPop, 58  
  Consumer, 56, 57  
  CONTINUOUS, 62  
  count, 58  
  execute, 58  
  getBufferSize, 59  
  hasAborted, 59  
  HISTORY, 62  
  isExecuting, 59  
  LATEST, 62  
  OLD, 62  
  pop, 59, 60  
  popIfPossible, 60  
  setBufferSize, 60  
  setTerminationInterval, 61  
  start, 61  
CONTINUOUS  
  Consumer, 62  
convertXMLResponse  
  XMLConverter, 106  
copy  
  JobAd, 10, 11  
copyAd  
  JobAd, 11  
count  
  Consumer, 58  
CPU\_TIME  
  JobStatus, 24  
createAttributeDefinition  
  EdgLocalReplicaCatalog, 34  
  EdgReplicaMetadataCatalog, 37  
createProxy  
  UserCredential, 28  
createTable  
  Schema, 83  
CreateUserRequest, 132  
  getUser, 132  
  setUser, 132  
cred  
  lcas\_cred\_ids, 135  
  lcmaps\_cred\_ids, 137  
cred\_data\_s, 136  
  cntPriGid, 136  
  cntSecGid, 136  
  cntUid, 136  
  dn, 136  
  priGid, 136  
  secGid, 136  
  uid, 136  
CURRENT  
  JobState, 21  
currentState  
  APIBase, 45  
custom  
  voms, 125, 127  
d\_atime  
  DirEntry, 137  
d\_ctime

---

DirEntry, 137

d\_fileno  
DirEntry, 138

d\_mtime  
DirEntry, 138

d\_name  
DirEntry, 138

d\_namlen  
DirEntry, 138

d\_reclen  
DirEntry, 138

d\_size  
DirEntry, 138

d\_type  
DirEntry, 138

data, 126  
cap, 126  
group, 126  
role, 126  
vomldata, 128

DataBaseProducer  
DataBaseProducer, 63

DataBaseProducer, 62  
add, 63  
archive, 64  
DataBaseProducer, 63  
getProducerConnection, 64  
setTerminationInterval, 64

datalen  
ext\_voms, 127

datel  
voms, 125, 127

date2  
voms, 125, 127

Declarable, 65  
Declarable, 65  
declareTable, 65  
undeclareTable, 66

declareTable  
Archiver, 48  
Cleanable, 54  
Declarable, 65

DEFAULT\_TERMINATION\_INTERVAL  
APIBase, 45

delAttribute  
JobAd, 11

deleteMapping  
EdgLocalReplicaCatalog, 34

denied  
\_GACLentry, 142

DESCR  
Event, 5

DEST\_HOST  
Event, 5

DEST\_ID  
Event, 5

DEST\_INSTANCE  
Event, 5

DEST\_JOBID  
Event, 5

DESTINATION  
Event, 5  
JobStatus, 24

DirEntry, 137  
d\_atime, 137  
d\_ctime, 137  
d\_fileno, 138  
d\_mtime, 138  
d\_name, 138  
d\_namlen, 138  
d\_reclen, 138  
d\_size, 138  
d\_type, 138

disconnect  
APIBase, 41

divide  
FixedPoint, 68

dn  
cred\_data\_s, 136  
lcas\_cred\_id\_s, 135  
lcmaps\_cred\_id\_s, 137

DONE  
JobStatus, 24

DONE\_CODE  
JobStatus, 24

doubleValue  
FixedPoint, 68

EdgLocalReplicaCatalog, 33  
addMapping, 34  
attributeDefinitionExists, 34  
createAttributeDefinition, 34  
deleteMapping, 34  
getAttributeDefinitions, 34  
getMappingsByGuid, 34  
getMappingsByPfn, 34  
getMappingsByPfnAttribute, 35  
getPfnAttribute, 35  
getPfns, 35  
guidExists, 35  
guidForPfn, 35



---

pfnExists, 36  
ping, 36  
removeAttributeDefinition, 36  
removeMapping, 36  
removePfnAttribute, 36  
setPfnAttribute, 36  
EdgReplicaMetadataCatalog, 37  
  addAlias, 37  
  aliasExists, 37  
  attributeDefinitionExists, 37  
  createAttributeDefinition, 37  
  getAliasAttribute, 38  
  getAliases, 38  
  getAttributeDefinitions, 38  
  getGuidAttribute, 38  
  getMappingsByAlias, 38  
  getMappingsByAliasAttribute, 38  
  getMappingsByGuid, 38  
  getMappingsByGuidAttribute, 38  
  guidExists, 38  
  guidForAlias, 38  
  ping, 38  
  removeAlias, 38  
  removeAliasAttribute, 38  
  removeAttributeDefinition, 38  
  removeGuidAttribute, 38  
  setAliasAttribute, 38  
  setGuidAttribute, 38  
EdgReplicaOptimization, 39  
  getAccessCost, 39  
  getBestNetworkCost, 39  
  ping, 40  
empty  
  JobCollection, 17  
equals  
  FileInfo, 116  
  FixedPoint, 68  
  JobId, 20  
  Mode, 117  
  StreamRequest, 93  
  Time, 96  
error  
  RGMAErrorHandler, 81  
  XMLConverter, 106  
EVENT  
  Event, 5  
Event, 3  
  attrNames, 4  
  CLASSAD, 4  
  code, 4  
  DESCR, 5  
  DEST\_HOST, 5  
  DEST\_ID, 5  
  DEST\_INSTANCE, 5  
  DEST\_JOBID, 5  
  DESTINATION, 5  
  EVENT, 5  
  EXIT\_CODE, 5  
  FROM, 5  
  FROM\_HOST, 5  
  FROM\_INSTANCE, 5  
  HELPER\_NAME, 5  
  HELPER\_PARAMS, 6  
  HOST, 6  
  JDL, 6  
  JOB, 6  
  JOBID, 6  
  LEVEL, 6  
  LOCAL\_JOBID, 6  
  NAME, 6  
  name, 4  
  NODE, 6  
  NS, 6  
  NSUBJOBS, 6  
  PARENT, 6  
  PRIORITY, 6  
  QUEUE, 6  
  REASON, 6  
  RESULT, 7  
  RETVAL, 7  
  SEED, 7  
  SEQCODE, 7  
  SOURCE, 7  
  SRC\_INSTANCE, 7  
  SRC\_ROLE, 7  
  STATUS\_CODE, 7  
  SVC\_HOST, 7  
  SVC\_NAME, 7  
  SVC\_PORT, 7  
  TAG, 7  
  TIMESTAMP, 7  
  toString, 4  
  USER, 7  
  VALUE, 8  
exceptionType  
  APIBase, 45  
execute  
  Consumer, 58  
EXIT\_CODE  
  Event, 5



---

JobStatus, 24

EXPECT\_FROM

JobStatus, 24

EXPECT\_UPDATE

JobStatus, 24

ext\_voms, 126

datalen, 127

real\_data, 127

signed\_data, 127

voms, 127

extra\_data

vomsdata, 128

fatalError

RGMAErrorHandler, 81

XMLConverter, 106

fd

plugparam, 140

FileControl, 113

FileHandle, 115

FileInfo

FileInfo, 115

FileInfo, 115

equals, 116

FileInfo, 115

getName, 116

getOwner, 116

getSEInfo, 116

getSize, 116

hashCode, 116

setName, 116

setOwner, 116

setSEInfo, 116

setSize, 116

toString, 116

finalize

APIBase, 41

first

bodyptr, 143

firstcred

\_GACLentry, 142

\_GACLuser, 143

firstentry

\_GACLacl, 141

firstname

\_GACLcred, 142

FixedPoint

FixedPoint, 67

FixedPoint, 66

add, 68

byteValue, 68

compareTo, 68

divide, 68

doubleValue, 68

equals, 68

FixedPoint, 67

floatValue, 69

getFraction, 69

getFractionAsDouble, 69

getUnits, 69

intValue, 69

longValue, 69

multiply, 69

shortValue, 69

subtract, 69

toString, 70

flags

plugparam, 140

floatValue

FixedPoint, 69

freeValue

MetricValue, 111

FROM

Event, 5

FROM\_HOST

Event, 5

FROM\_INSTANCE

Event, 5

fromCert

VOMSExtension, 124

fromString

JobId, 20

fsplugin, 138

fstabline, 139

handle, 139

instance, 139

next, 139

PluginClose, 139

PluginCreate, 139

PluginExclPerm, 139

PluginFileType, 139

PluginInit, 139

PluginMkdir, 139

PluginOpen, 139

PluginOpenFd, 139

PluginReadlink, 139

PluginRemove, 139

PluginRename, 139

PluginRmdir, 139

PluginSetExec, 139

PluginStat, 139

PluginSymlink, 139  
PluginUserPerm, 139  
sofile, 139  
version, 140  
fstabline  
  fsplugin, 139  
GET  
  ServletConnection, 90  
getAccessCost  
  BrokerInfo, 31  
  EdgReplicaOptimization, 39  
getAccuracy  
  Time, 96  
getAdminCA  
  ACLEntry, 130  
getAdminComment  
  Request, 133  
getAdminDN  
  ACLEntry, 130  
getAliasAttribute  
  EdgReplicaMetadataCatalog, 38  
getAliases  
  EdgReplicaMetadataCatalog, 38  
getAllSchemaTables  
  Schema, 83  
getAttributeDefinitions  
  EdgLocalReplicaCatalog, 34  
  EdgReplicaMetadataCatalog, 38  
getAttributeExpr  
  JobAd, 11  
getAuthorizationPolicy  
  SecurityInfo, 121  
getAuthorizedAttributes  
  SecurityInfo, 121  
getAutoInsertTimestamp  
  Insertable, 71  
getBestNetworkCost  
  EdgReplicaOptimization, 39  
getBIFileName  
  BrokerInfo, 31  
getBoolean  
  JobAd, 11  
  ResultSet, 79  
getBufferSize  
  Archiver, 48  
  Consumer, 59  
getBySeconds  
  TimeInterval, 105  
getCA  
  User, 134  
getCE  
  BrokerInfo, 31  
getCertUri  
  User, 134  
getClientCert  
  SecurityInfo, 121  
getClientCertChain  
  SecurityInfo, 121  
getClientComment  
  Request, 133  
getClientEmail  
  Request, 133  
getClientName  
  SecurityInfo, 121  
getCloseSEs  
  BrokerInfo, 31  
getCN  
  User, 134  
getColumnCount  
  ResultSetMetaData, 80  
getColumnName  
  ResultSetMetaData, 80  
getColumnNames  
  ResultSetMetaData, 80  
  Schema, 83  
getColumnNumber  
  ResultSetMetaData, 80  
getColumnTypes  
  Schema, 84  
getConnectionId  
  ProducerConnection, 73  
  ServletConnection, 88  
getCost  
  NetworkCost, 119  
getCredType  
  UserCredential, 28  
getDataAccessProtocol  
  BrokerInfo, 31  
getDate  
  Time, 96  
getDateTime  
  Time, 96  
getDetails  
  Time, 96  
getDetailsLarge  
  TimeDetails, 104  
getDetailsUnpacked  
  Time, 97  
getDN  
  User, 134



---

getDouble  
    JobAd, 11

getDuplex  
    NetworkCostContextIF, 120

getEpochOffset  
    Time, 98

getError  
    NetworkCost, 119

getFraction  
    FixedPoint, 69

getFractionAsDouble  
    FixedPoint, 69

getGroup  
    QualifiedRole, 131

getGuidAttribute  
    EdgReplicaMetadataCatalog, 38

getId  
    Request, 133

getInputData  
    BrokerInfo, 31

getInt  
    JobAd, 12  
    ResultSet, 79

getIso8601  
    Time, 98

getIssuer  
    UserCredential, 28

getJobAdValue  
    JobAd, 12

getJobs  
    UserJobs, 30

getLBAddress  
    JobId, 20

getLeapseconds  
    Time, 98

getLFN2SFN  
    BrokerInfo, 31

getMail  
    User, 135

getMappingsByAlias  
    EdgReplicaMetadataCatalog, 38

getMappingsByAliasAttribute  
    EdgReplicaMetadataCatalog, 38

getMappingsByGuid  
    EdgLocalReplicaCatalog, 34  
    EdgReplicaMetadataCatalog, 38

getMappingsByGuidAttribute  
    EdgReplicaMetadataCatalog, 38

getMappingsByPfn  
    EdgLocalReplicaCatalog, 34

getMappingsByPfnAttribute  
    EdgLocalReplicaCatalog, 35

getMaxBufferSize  
    StreamProducer, 91

getMetaData  
    ResultSet, 79

getMinRetentionPeriod  
    StreamProducer, 92

getMode  
    Mode, 117

getName  
    FileInfo, 116

getOperationName  
    ACLEntry, 130

getOutput  
    JobCollection, 17

getOwner  
    FileInfo, 116

getParam  
    metricParams, 110

getPfnAttribute  
    EdgLocalReplicaCatalog, 35

getPfn  
    EdgLocalReplicaCatalog, 35

getPrecision  
    Time, 98

getPrimaryKey  
    Schema, 84

getProducerConnection  
    CircularBufferProducer, 52  
    DataBaseProducer, 64

getProducerConnections  
    Registry, 74

getProducerInfo  
    Registry, 74

getProducerServlet  
    ProducerConnection, 73

getProperties  
    APIBase, 41  
    PropertyGetter, 73

getRegistryServletLocation  
    Registry, 75

getRemoteBufferSize  
    CircularBufferProducer, 52

getReplicationPeriod  
    ReplicaConfigReader, 76

getRequestedAttributes  
    SecurityInfo, 122

getRole  
    QualifiedRole, 131

getSecurityInfo  
    SecurityInfoContainer, 123

getSEInfo  
    FileInfo, 116

getSEMOUNTPOINT  
    BrokerInfo, 32

getSEPort  
    BrokerInfo, 32

getSEProtocols  
    BrokerInfo, 32

getServletConnection  
    APIBase, 42  
    Schema, 84

getServletURL  
    ServletConnection, 88

getSEs  
    BrokerInfo, 32

getSize  
    FileInfo, 116

getSource  
    RGMAException, 82

getStates  
    UserJobs, 30

getStatus  
    APIBase, 42  
    JobCollection, 17  
    Registry, 75  
    Request, 133  
    Schema, 84

getStrenght  
    UserCredential, 28

getString  
    JobAd, 12  
    ResultSet, 79

getSubject  
    UserCredential, 28

getSubseconds  
    Time, 98

getSubsecondsAsDouble  
    Time, 98

getTableDesc  
    Schema, 84

getTableInfo  
    Schema, 85

getTai  
    Time, 98

getTerminationInterval  
    APIBase, 42

getTime  
    Time, 98

getTimeLeft  
    UserCredential, 28

getTrustFile  
    ServletConnection, 89

getTupleChecking  
    APIBase, 42

getType  
    RGMAException, 82

getUniqueString  
    JobId, 20

getUnits  
    FixedPoint, 69

getURLs  
    ReplicaConfigReader, 76

getUser  
    CreateUserRequest, 132

getUtc  
    Time, 99

getUtcAsInt  
    Time, 99

getUtcSeconds  
    Time, 99

getVirtualOrganization  
    BrokerInfo, 32

getVOMSExtension  
    SecurityInfo, 122

getVOMSInfos  
    VOMSExtension, 124

GLOBUS\_ID  
    JobStatus, 24

group  
    data, 126

guidExists  
    EdgLocalReplicaCatalog, 35  
    EdgReplicaMetadataCatalog, 38

guidForAlias  
    EdgReplicaMetadataCatalog, 38

guidForPfn  
    EdgLocalReplicaCatalog, 35

handle  
    fsplugin, 139

hasAborted  
    Consumer, 59

hasAttribute  
    JobAd, 12

hashCode  
    FileInfo, 116  
    Mode, 117  
    StreamRequest, 93

HELPER\_NAME

Event, 5  
HELPER\_PARAMS  
  Event, 6  
HISTORY  
  CanonicalProducer, 51  
  Consumer, 62  
HOST  
  Event, 6  
info  
  metricBase, 108  
init  
  ResilientStreamProducer, 78  
  StreamProducer, 92  
insert  
  Insertable, 71  
  JobCollection, 17  
Insertable, 70  
  getAutoInsertTimestamp, 71  
  insert, 71  
  Insertable, 70  
  setAutoInsertTimestamp, 71  
insertAd  
  JobCollection, 17  
insertId  
  JobCollection, 17  
instance  
  BrokerInfo, 32  
  fsplugin, 139  
intValue  
  FixedPoint, 69  
IS\_CLOSED  
  APIBase, 45  
IS\_CONNECTED  
  APIBase, 45  
IS\_DISCONNECTED  
  APIBase, 45  
isAllow  
  ACLEntry, 131  
isExecuting  
  Consumer, 59  
isRecoverable  
  RGMAException, 82  
isResourceAvailable  
  ReplicaConfigReader, 76  
isSet  
  JobId, 20  
JDL  
  Event, 6  
  JobStatus, 25  
JOB  
  Event, 6  
JOB\_ID  
  JobStatus, 25  
JobAd  
  JobAd, 9  
JobAd, 8  
  addAttribute, 9, 10  
  check, 10  
  checkAll, 10  
  clear, 10  
  copy, 10, 11  
  copyAd, 11  
  delAttribute, 11  
  getAttributeExpr, 11  
  getBoolean, 11  
  getDouble, 11  
  getInt, 12  
  getJobAdValue, 12  
  getString, 12  
  hasAttribute, 12  
  JobAd, 9  
  JOBTYPE\_MPICH\_RANK\_FREE, 15  
  JOBTYPE\_MPICH\_REQ\_CPU, 15  
  JOBTYPE\_MPICH\_REQ\_RTE, 15  
  RANK\_DEFAULT, 15  
  REQ\_DEFAULT, 15  
  setAttribute, 13  
  setAttributeAd, 13  
  setAttributeDefault, 14  
  setAttributeExpr, 14  
  setLocalAccess, 14  
  toFile, 14  
  toLines, 14  
  toString, 14  
  toSubmissionString, 15  
JobCollection  
  JobCollection, 16  
JobCollection, 15  
  cancel, 17  
  clear, 17  
  empty, 17  
  getOutput, 17  
  getStatus, 17  
  insert, 17  
  insertAd, 17  
  insertId, 17  
  JobCollection, 16  
  jobs, 18  
  remove, 18

run, 18  
setCredPath, 18  
setMaxThreadNumber, 18  
size, 18  
submit, 18  
unsetCredPath, 19

JOBID  
Event, 6  
JobState, 21

JobId  
JobId, 19

JobId, 19  
clear, 20  
equals, 20  
fromString, 20  
getLBAddress, 20  
getUniqueString, 20  
isSet, 20  
JobId, 19  
setJobId, 20  
toString, 20

jobs  
JobCollection, 18

JobState, 21  
check, 21  
checkAttribute, 21  
CURRENT, 21  
JOBID, 21  
JOBSTEPS, 21  
setAttribute, 21  
setId, 21  
toFile, 21  
USERDATA, 21

JobStatus, 22  
ABORTED, 23  
attNames, 23  
CANCEL\_REASON, 23  
CANCELLED, 23  
CANCELLING, 23  
CE\_NODE, 23  
CHILDREN, 23  
CHILDREN\_HIST, 23  
CHILDREN\_NUM, 23  
CHILDREN\_STATES, 23  
CLEARED, 23  
code, 24  
CONDOR\_ID, 24  
CONDOR\_JDL, 24  
CPU\_TIME, 24  
DESTINATION, 24  
DONE, 24  
DONE\_CODE, 24  
EXIT\_CODE, 24  
EXPECT\_FROM, 24  
EXPECT\_UPDATE, 24  
GLOBUS\_ID, 24  
JDL, 25  
JOB\_ID, 25  
JOBTYPE, 25  
LAST\_UPDATE\_TIME, 25  
LOCAL\_ID, 25  
LOCATION, 25  
MATCHED\_JDL, 25  
name, 23  
NETWORK\_SERVER, 25  
OWNER, 25  
PARENT\_JOB, 25  
PURGED, 25  
READY, 25  
REASON, 25  
RESUBMITTED, 25  
RSL, 25  
RUNNING, 26  
SCHEDULED, 26  
SEED, 26  
STATE\_ENTER\_TIME, 26  
STATE\_ENTER\_TIMES, 26  
STATUS, 26  
status, 23  
SUBJOB\_FAILED, 26  
SUBMITTED, 26  
toString, 23  
UNDEF, 26  
UNKNOWN, 26  
USER\_TAGS, 26  
USER\_VALUES, 26  
WAITING, 26

JOBSTEPS  
JobState, 21

JOBTYPE  
JobStatus, 25

JOBTYPE\_MPICH\_RANK\_FREE  
JobAd, 15

JOBTYPE\_MPICH\_REQ\_CPU  
JobAd, 15

JOBTYPE\_MPICH\_REQ\_RTE  
JobAd, 15

keypass  
uidentry, 141



---

last  
    bodyptr, 143  
LAST\_UPDATE\_TIME  
    JobStatus, 25  
LATEST  
    CanonicalProducer, 51  
    Consumer, 62  
LatestProducer  
    LatestProducer, 72  
LatestProducer, 72  
    LatestProducer, 72  
lcas\_cred\_ids, 135  
    cred, 135  
    dn, 135  
lcmaps\_argument\_s, 136  
    argInOut, 136  
    argName, 136  
    argType, 136  
    value, 137  
lcmaps\_cred\_ids, 137  
    cred, 137  
    dn, 137  
ld  
    TimeConverter, 104  
LEVEL  
    Event, 6  
listchar, 143  
    next, 143  
    text, 143  
LOCAL\_ID  
    JobStatus, 25  
LOCAL\_JOBID  
    Event, 6  
LOCATION  
    JobStatus, 25  
log  
    metricBase, 109  
longValue  
    FixedPoint, 69  
MATCHED\_JDL  
    JobStatus, 25  
Metric, 107  
    metricId, 107  
    metricName, 107  
    next, 107  
metric  
    Sample, 112  
metricBase  
    metricBase, 108  
metricBase, 107  
    ~metricBase, 108  
    info, 108  
    log, 109  
    metricBase, 108  
    numericId, 109  
    parameters, 109  
    periodicCheck, 109  
    sample, 109  
    storeSample01, 109  
metricId  
    Metric, 107  
metricName  
    Metric, 107  
metricParams  
    metricParams, 110  
metricParams, 110  
    ~metricParams, 110  
    addParam, 110  
    getParam, 110  
    metricParams, 110  
    print, 110  
MetricValue, 111  
    freeValue, 111  
    value, 111  
    valueLength, 111  
Mode, 117  
    equals, 117  
    getMode, 117  
    hashCode, 117  
    Mode, 117  
    READ, 118  
    READ\_MODE, 118  
    toString, 117  
    WRITE, 118  
    WRITE\_MODE, 118  
multiply  
    FixedPoint, 69  
NAME  
    Event, 6  
name  
    \_GACLnamevalue, 142  
    Event, 4  
    JobStatus, 23  
NetLoggerFactory, 72  
NETWORK\_SERVER  
    JobStatus, 25  
NetworkCost  
    NetworkCost, 119  
NetworkCost, 118  
    getCost, 119

- getError, 119
- NetworkCost, 119
  - setCost, 119
  - setError, 119
- NetworkCostContextIF, 119
  - checkDuplex, 120
  - getDuplex, 120
  - setDuplex, 120
  - TP\_GRIDFTP, 120
  - TP\_RFIO, 120
  - VERSION, 120
- next
  - \_GACLcred, 142
  - \_GACLentry, 142
  - \_GACLnamevalue, 142
  - fsplugin, 139
  - listchar, 143
  - Metric, 107
  - Node, 112
  - ResultSet, 79
  - Sample, 112
  - uidentry, 141
- nl
  - APIBase, 45
- NODE
  - Event, 6
- Node, 111
  - next, 112
  - nodeId, 112
  - nodeName, 112
- node
  - Sample, 112
- nodeId
  - Node, 112
- nodeName
  - Node, 112
- NS
  - Event, 6
- NSUBJOBS
  - Event, 6
- numericId
  - metricBase, 109
- OLD
  - Consumer, 62
- OWNER
  - JobStatus, 25
- packDetails
  - TimeDetails, 104
- parameters
  - metricBase, 109
- PARENT
  - Event, 6
- PARENT\_JOB
  - JobStatus, 25
- parse
  - VOMSExtension, 124
- parser
  - BrokerInfo, 32
- periodicCheck
  - metricBase, 109
- perm
  - plugparam, 140
- pfnExists
  - EdgLocalReplicaCatalog, 36
- pid
  - plugparam, 140
- ping
  - EdgLocalReplicaCatalog, 36
  - EdgReplicaMetadataCatalog, 38
  - EdgReplicaOptimization, 40
- PluginClose
  - fsplugin, 139
- PluginCreate
  - fsplugin, 139
- PluginExclPerm
  - fsplugin, 139
- PluginFileType
  - fsplugin, 139
- PluginInit
  - fsplugin, 139
- PluginMkdir
  - fsplugin, 139
- PluginOpen
  - fsplugin, 139
- PluginOpenFd
  - fsplugin, 139
- PluginReadlink
  - fsplugin, 139
- PluginRemove
  - fsplugin, 139
- PluginRename
  - fsplugin, 139
- PluginRmdir
  - fsplugin, 139
- PluginSetExec
  - fsplugin, 139
- PluginStat
  - fsplugin, 139
- PluginSymlink

fsplugin, 139  
PluginUserPerm  
  fsplugin, 139  
plugparam, 140  
  fd, 140  
  flags, 140  
  perm, 140  
  pid, 140  
  result, 140  
  source, 140  
  stat, 140  
  target, 140  
  type, 140  
  uidentry, 140  
  unique, 140  
pop  
  Consumer, 59, 60  
popIfPossible  
  Consumer, 60  
POST  
  ServletConnection, 90  
previous  
  ResultSet, 79  
priGid  
  cred\_data\_s, 136  
print  
  metricParams, 110  
PRIORITY  
  Event, 6  
ProducerConnection  
  ProducerConnection, 73  
ProducerConnection, 72  
  getConnectionId, 73  
  getProducerServlet, 73  
  ProducerConnection, 73  
  toString, 73  
PropertyGetter  
  PropertyGetter, 73  
PropertyGetter, 73  
  getProperties, 73  
  PropertyGetter, 73  
PURGED  
  JobStatus, 25  
QualifiedRole, 131  
  getGroup, 131  
  getRole, 131  
  setGroup, 131  
  setRole, 131  
QUEUE  
  Event, 6  
RANK\_DEFAULT  
  JobAd, 15  
READ  
  Mode, 118  
READ\_MODE  
  Mode, 118  
READY  
  JobStatus, 25  
real\_data  
  ext\_voms, 127  
REASON  
  Event, 6  
  JobStatus, 25  
reconnect  
  APIBase, 42  
recvTime  
  Sample, 113  
Registry, 73  
  getProducerConnections, 74  
  getProducerInfo, 74  
  getRegistryServletLocation, 75  
  getStatus, 75  
  Registry, 74  
REGISTRY\_CONFIG  
  ReplicaConfigReader, 77  
remove  
  JobCollection, 18  
removeAlias  
  EdgReplicaMetadataCatalog, 38  
removeAliasAttribute  
  EdgReplicaMetadataCatalog, 38  
removeAttributeDefinition  
  EdgLocalReplicaCatalog, 36  
  EdgReplicaMetadataCatalog, 38  
removeGuidAttribute  
  EdgReplicaMetadataCatalog, 38  
removeMapping  
  EdgLocalReplicaCatalog, 36  
removePfnAttribute  
  EdgLocalReplicaCatalog, 36  
ReplicaConfigReader  
  ReplicaConfigReader, 76  
ReplicaConfigReader, 75  
  cat, 77  
  getReplicationPeriod, 76  
  getURLs, 76  
  isResourceAvailable, 76  
REGISTRY\_CONFIG, 77  
ReplicaConfigReader, 76  
SCHEMA\_CONFIG, 77



ReplicaManager, 33  
REQ\_DEFAULT  
    JobAd, 15  
Request, 132  
    getAdminComment, 133  
    getClientComment, 133  
    getClientEmail, 133  
    getId, 133  
    getStatus, 133  
    setAdminComment, 133  
    setClientComment, 133  
    setClientEmail, 133  
    setId, 133  
    setStatus, 133  
ResilientStreamProducer  
    ResilientStreamProducer, 77  
ResilientStreamProducer, 77  
    init, 78  
    ResilientStreamProducer, 77  
RESUBMITTED  
    JobStatus, 25  
RESULT  
    Event, 7  
result  
    plugparam, 140  
ResultSet, 78  
    afterLast, 79  
    beforeFirst, 79  
    getBoolean, 79  
    getInt, 79  
    getMetaData, 79  
    getString, 79  
    next, 79  
    previous, 79  
    size, 79  
    toString, 80  
ResultSetMetaData, 80  
    getColumnCount, 80  
    洗getColumnName, 80  
    getColumnNames, 80  
    getColumnNumber, 80  
RETVAL  
    Event, 7  
RGMAErrorHandler, 80  
    error, 81  
    fatalError, 81  
    warning, 81  
RGMAException, 81  
    getSource, 82  
    getType, 82  
    isRecoverable, 82  
    RGMAException, 81, 82  
RgmaSSLFactory  
    ServletConnection, 90  
role  
    data, 126  
RSL  
    JobStatus, 25  
run  
    JobCollection, 18  
RUNNING  
    JobStatus, 26  
Sample, 112  
    metric, 112  
    next, 112  
    node, 112  
    recvTime, 113  
    timestamp, 113  
    value, 113  
sample  
    metricBase, 109  
SCHEDULED  
    JobStatus, 26  
Schema, 82  
    createTable, 83  
    getAllSchemaTables, 83  
    getColumnNames, 83  
    getColumnTypes, 84  
    getPrimaryKey, 84  
    getServletConnection, 84  
    getStatus, 84  
    getTableDesc, 84  
    getTableInfo, 85  
    Schema, 83  
    sendCommand, 85  
    servletURL, 86  
    translateColumnNames, 85  
    translateTableName, 85  
SCHEMA\_CONFIG  
    ReplicaConfigReader, 77  
secGid  
    cred\_data\_s, 136  
SecurityInfo, 120  
    getAuthorizationPolicy, 121  
    getAuthorizedAttributes, 121  
    getClientCert, 121  
    getClientCertChain, 121  
    getClientName, 121  
    getRequestedAttributes, 122  
    getVOMSExtension, 122

SecurityInfoContainer, 122  
    getSecurityInfo, 123

SEED  
    Event, 7  
    JobStatus, 26

sendCommand  
    APIBase, 43  
    Schema, 85

SEQCODE  
    Event, 7

server  
    voms, 125, 127

serverca  
    voms, 126, 127

ServletConnection  
    ServletConnection, 87

ServletConnection, 86  
    addParameter, 87, 88  
    connect, 88  
    GET, 90  
    getConnectionId, 88  
    getServletURL, 88  
    getTrustFile, 89  
    POST, 90  
    RgmaSSLFactory, 90  
    ServletConnection, 87  
    setBody, 89  
    setRequestMethod, 89  
    setupRgmaSSLFactory, 89  
    setupSSLConnections, 89  
    SSLINIT, 90  
    sslInit, 90  
    streamingConnect, 89  
    toString, 90

servletURL  
    APIBase, 45  
    Schema, 86

setAccuracy  
    Time, 99

setAdminCA  
    ACLEntry, 131

setAdminComment  
    Request, 133

setAdminDN  
    ACLEntry, 131

setAliasAttribute  
    EdgReplicaMetadataCatalog, 38

setAllow  
    ACLEntry, 131

setAttribute  
    JobAd, 13  
    JobState, 21

setAttributeAd  
    JobAd, 13

setAttributeDefault  
    JobAd, 14

setAttributeExpr  
    JobAd, 14

setAutoInsertTimestamp  
    Insertable, 71

setBody  
    ServletConnection, 89

setBufferSize  
    Archiver, 49  
    Consumer, 60

setBySeconds  
    TimeInterval, 106

setCA  
    User, 135

setCertUri  
    User, 135

setClientComment  
    Request, 133

setClientEmail  
    Request, 133

setCN  
    User, 135

setCost  
    NetworkCost, 119

setCredPath  
    JobCollection, 18  
    UserJobs, 30

setDateTime  
    Time, 99

setDetails  
    Time, 99

setDetailsLarge  
    TimeDetails, 104

setDetailsUnpacked  
    Time, 100

setDN  
    User, 135

setDuplex  
    NetworkCostContextIF, 120

setEpochOffset  
    Time, 100

setError  
    NetworkCost, 119

setGroup  
    QualifiedRole, 131



---

setGuidAttribute  
    EdgReplicaMetadataCatalog, 38

setId  
    JobState, 21  
    Request, 133

setIso8601  
    Time, 100

setJobId  
    JobId, 20

setLeapseconds  
    Time, 100

setLocalAccess  
    JobAd, 14

setMail  
    User, 135

setMaxBufferSize  
    StreamProducer, 92

setMaxThreadNumber  
    JobCollection, 18

setMinRetentionPeriod  
    StreamProducer, 92

setName  
    FileInfo, 116

setOperationName  
    ACLEntry, 131

setOwner  
    FileInfo, 116

setPfnAttribute  
    EdgLocalReplicaCatalog, 36

setPrecision  
    Time, 100

setProxy  
    UserCredential, 28

setRemoteBufferSize  
    CircularBufferProducer, 52

setRequestMethod  
    ServletConnection, 89

setRole  
    QualifiedRole, 131

setSEInfo  
    FileInfo, 116

setSize  
    FileInfo, 116

setStatus  
    Request, 133

setSubseconds  
    Time, 100, 101

setTai  
    Time, 101

setTerminationInterval  
    APIBase, 44  
    CircularBufferProducer, 52  
    Consumer, 61  
    DataBaseProducer, 64  
    setTimeWithDateTime  
        TimeConverter, 103  
    setTimeWithIso8601  
        TimeConverter, 103  
    setTupleChecking  
        APIBase, 44  
    setUpRgmaSSLFactory  
        ServletConnection, 89  
    setupSSLConnections  
        ServletConnection, 89  
    setUser  
        CreateUserRequest, 132  
    setUtc  
        Time, 101  
    setUtcAsInt  
        Time, 101  
    setUtcSeconds  
        Time, 101  
    shortValue  
        FixedPoint, 69  
    showSignOfLife  
        APIBase, 44  
    siglen  
        voms, 126, 127  
    signature  
        voms, 126, 127  
    signed.data  
        ext\_voms, 127  
    size  
        bodyptr, 143  
        JobCollection, 18  
        ResultSet, 79  
    sofile  
        fsplugin, 139  
    SOURCE  
        Event, 7  
    source  
        plugparam, 140  
    SRC\_INSTANCE  
        Event, 7  
    SRC\_ROLE  
        Event, 7  
    SSLINIT  
        ServletConnection, 90  
    sslInit  
        ServletConnection, 90



---

start  
  Consumer, 61

stat  
  plugparam, 140

STATE\_ENTER\_TIME  
  JobStatus, 26

STATE\_ENTER\_TIMES  
  JobStatus, 26

STATUS  
  JobStatus, 26

status  
  JobStatus, 23

STATUS\_CODE  
  Event, 7

std  
  voms, 126, 127

StorageElement, 118

storeSample01  
  metricBase, 109

streamingConnect  
  ServletConnection, 89

StreamProducer  
  StreamProducer, 91

StreamProducer, 90  
  getMaxBufferSize, 91  
  getMinRetentionPeriod, 92  
  init, 92  
  setMaxBufferSize, 92  
  setMinRetentionPeriod, 92  
  StreamProducer, 91

StreamRequest, 93  
  equals, 93  
  hashCode, 93  
  toString, 93

SUBJOB\_FAILED  
  JobStatus, 26

submit  
  JobCollection, 18

SUBMITTED  
  JobStatus, 26

subtract  
  FixedPoint, 69  
  Time, 101

SVC\_HOST  
  Event, 7

SVC\_NAME  
  Event, 7

SVC\_PORT  
  Event, 7

TAG  
  Event, 7

  taiToUtc  
    TimeConverter, 103

target  
  plugparam, 140

text  
  listchar, 143

Time, 93  
  after, 95  
  before, 95  
  compareTo, 95  
  equals, 96  
  getAccuracy, 96  
  getDate, 96  
  getDateTime, 96  
  getDetails, 96  
  getDetailsUnpacked, 97  
  getEpochOffset, 98  
  getIso8601, 98  
  getLeapseconds, 98  
  getPrecision, 98  
  getSubseconds, 98  
  getSubsecondsAsDouble, 98  
  getTai, 98  
  getTime, 98  
  getUtc, 99  
  getUtcAsInt, 99  
  getUtcSeconds, 99  
  setAccuracy, 99  
  setDateTime, 99  
  setDetails, 99  
  setDetailsUnpacked, 100  
  setEpochOffset, 100  
  setIso8601, 100  
  setLeapseconds, 100  
  setPrecision, 100  
  setSubseconds, 100, 101  
  setTai, 101  
  setUtc, 101  
  setUtcAsInt, 101  
  setUtcSeconds, 101  
  subtract, 101  
  Time, 94, 95  
  toString, 102

TimeConverter  
  TimeConverter, 102

TimeConverter, 102  
  ld, 104  
  setTimeWithDateTime, 103  
  setTimeWithIso8601, 103

taiToUtc, 103  
TimeConverter, 102  
utcToDateTime, 103  
utcToIso8601, 103  
utcToTai, 104  
TimeDetails, 104  
  getDetailsLarge, 104  
  packDetails, 104  
  setDetailsLarge, 104  
  unpackDetails, 104  
TimeInterval  
  TimeInterval, 105  
TimeInterval, 105  
  getBySeconds, 105  
  setBySeconds, 106  
  TimeInterval, 105  
TIMESTAMP  
  Event, 7  
timestamp  
  Sample, 113  
toFile  
  JobAd, 14  
  JobState, 21  
toLines  
  JobAd, 14  
toString  
  Event, 4  
  FileInfo, 116  
  FixedPoint, 70  
  JobAd, 14  
  JobId, 20  
  JobStatus, 23  
  Mode, 117  
  ProducerConnection, 73  
  ResultSet, 80  
  ServletConnection, 90  
  StreamRequest, 93  
  Time, 102  
  VOMSExtension, 125  
toSubmissionString  
  JobAd, 15  
TP\_GRIDFTP  
  NetworkCostContextIF, 120  
TP\_RFIO  
  NetworkCostContextIF, 120  
TransferFileHandle, 118  
translateColumnNames  
  Schema, 85  
translateTableName  
  Schema, 85  
type  
  \_GACLcred, 142  
  plugparam, 140  
  voms, 126, 128  
uid  
  cred\_data\_s, 136  
  uidentry, 141  
uidentry, 141  
  certexpires, 141  
  certfile, 141  
  certfiletime, 141  
  keypass, 141  
  next, 141  
  plugparam, 140  
  uid, 141  
  user, 141  
undeclareTable  
  Declarable, 66  
UNDEF  
  JobStatus, 26  
unique  
  plugparam, 140  
UNKNOWN  
  JobStatus, 26  
unpackDetails  
  TimeDetails, 104  
unsetCredPath  
  JobCollection, 19  
  UserJobs, 30  
unsetProxy  
  UserCredential, 28  
uri  
  voms, 126, 128  
USER  
  Event, 7  
User, 134  
  getCA, 134  
  getCertUri, 134  
  getCN, 134  
  getDN, 134  
  getMail, 135  
  setCA, 135  
  setCertUri, 135  
  setCN, 135  
  setDN, 135  
  setMail, 135  
  User, 134  
user  
  uidentry, 141  
  voms, 126, 128



---

USER\_TAGS  
    JobStatus, 26

USER\_VALUES  
    JobStatus, 26

userca  
    voms, 126, 128

UserCredential  
    UserCredential, 27

UserCredential, 27  
    checkProxy, 28  
    createProxy, 28  
    getCredType, 28  
    getIssuer, 28  
    getStrenght, 28  
    getSubject, 28  
    getTimeLeft, 28  
    setProxy, 28  
    unsetProxy, 28  
    UserCredential, 27

USERDATA  
    JobState, 21

UserJobs  
    UserJobs, 29

UserJobs, 29  
    cancel, 29  
    getJobs, 30  
    getStates, 30  
    setCredPath, 30  
    unsetCredPath, 30  
    UserJobs, 29

utcToDateTime  
    TimeConverter, 103

utcToIso8601  
    TimeConverter, 103

utcToTai  
    TimeConverter, 104

VALUE  
    Event, 8

value  
    \_GACLnamevalue, 143  
    lcmaps\_argument\_s, 137  
    MetricValue, 111  
    Sample, 113

valueLength  
    MetricValue, 111

VERSION  
    NetworkCostContextIF, 120

version  
    fsplugin, 140

voms, 125, 127  
    custom, 125, 127  
    date1, 125, 127  
    date2, 125, 127  
    ext\_voms, 127  
    server, 125, 127  
    serverca, 126, 127  
    siglen, 126, 127  
    signature, 126, 127  
    std, 126, 127  
    type, 126, 128  
    uri, 126, 128  
    user, 126, 128  
    userca, 126, 128  
    voname, 126, 128

VOMS\_OID  
    VOMSExtension, 125

VOMSAdmin, 128

VOMSCompatibility, 129

VOMSCore, 129

vomsdata, 128  
    data, 128  
    extra\_data, 128  
    workvo, 128

VOMSExtension, 123  
    fromCert, 124  
    getVOMSInfos, 124  
    parse, 124  
    toString, 125  
    VOMS\_OID, 125  
    VOMSExtension, 124

VOMSHistory, 129

VOMSRequest, 129

voname  
    voms, 126, 128

WAITING  
    JobStatus, 26

warning  
    RGMAErrorHandler, 81  
    XMLConverter, 107

workvo  
    vomsdata, 128

WRITE  
    Mode, 118

WRITE\_MODE  
    Mode, 118

XMLConverter, 106  
    convertXMLResponse, 106  
    error, 106  
    fatalError, 106



warning, 107

XMLConverter, 106