GFD-R-P.xxx
SAGA-RG

Andre Merzky[1]
Mark Santcroos
Steve Fisher
Ole Weidner

Version: 1.0

October 24, 2012

SAGA API Bindings: Python

Status of This Document

This document provides information to the grid community, proposing a standard for a Python language binding to the Simple API for Grid Applications (SAGA). As SAGA language binding, it depends upon the SAGA Core API Specification [**?**], and on the so-far defined SAGA API extension packages [**?**]. This document is supposed to be used as reference for implementors of this language bindings. Distribution of this document is unlimited.

FIXME: run_job only supports interactive jobs.

Copyright Notice

Abstract

...

[1]editor

Contents

1 Introduction

1.1 Notational Conventions

In structure, notation and conventions, this documents follows those of the SAGA Core API specification [?], unless noted otherwise.

1.2 Security Considerations

As the SAGA API is to be implemented on different types of Grid (and non-Grid) middleware, it does not specify a single security model, but rather provides hooks to interface to various security models – see the documentation of the `saga::context` class in the SAGA Core API specification [?] for details.

A SAGA implementation is considered secure if and only if it fully supports (i.e. implements) the security models of the middleware layers it builds upon, and neither provides any (intentional or unintentional) means to by-pass these security models, nor weakens these security models' policies in any way.

2 SAGA Python Bindings

This section will motivate and discuss the general design principles for the SAGA Python bindings. That results in a set of rules which prescribe the translation of the SAGA API as specified in GFD.90 and in the SAGA API Extension specification documents. Those rules SHOULD also be applied to future SAGA API extensions.

The explicit python bindings are listed as Python module and class prototypes in appendix A.

2.1 Class Hierarchy Considerations

The SAGA API defines an interface and object hierarchy. While we expect that language bindings will, in general, follow that hierarchy for consistency and also for practical reasons, it is that case for Python that a strict insistence on that hierarchy is neither required nor useful.

As other language bindings (i.e. C++, Java), the package names will not be part of the module hierarchy for the SAGA Core Look & Feel classes. For functional API packages, the package name is part of the module path: i.e., `saga.Context` instead of `saga.context.Context`, but `saga.job.Service` instead of `saga.JobService`.

In particular, Python's prevalent duck-typing paradigm [?] encourages to ignore strict object and interface hierarchies, and instead supports to flatten those into the actual object implementations. The paragraphs below discuss the cases where this is used or supported by the SAGA Python bindings.

2.1.1 SAGA Object Interface

Most SAGA classes as specified in GFD.90 inherit from the base `saga.object` class. That class provides a unique object ID for class instances, deep copy semantics, object type inspection and access to the `saga.session` instance which manages that object.

Python provides most of these facilities natively: it has type inspection and unique object IDs, and the core python library comes with a generic deep copy call. The python bindings are thus not expected to implement the `saga.object` class, but CAN attach the remaining `get_session()` method directly to the respective object types (for reasons discussed later, the session will also be exposed as object property).

FIXME: add back-reference

2.1.2 SAGA Namespace Package

The GFD.90 'namespace' package defines a common interface for several downstream packages which, amongst others, interface to entities organized in namespaces, such as physical files, logical files (replicas), information services, etc. As the namespace package thus functions as an interface package, implementations MAY flatten it into the respective deriving class implementations. While that would not allow to directly instantiate namespace class entities, Python's duck typing and loose type system would still allow to interchangeably use derivatives interchangeably. Implementations thus MAY flatten the namespace classes into inheriting packages.

FIXME: But, why is it an *advantage* to flatten the ns package?

2.1.3 SAGA Buffer Class

The `saga.Buffer` class of GFD.90 is used for a variety of I/O operations, on streams, files, messages, RPC-calls etc. Its primary purpose (as opposed to using plain data arrays) is to support both implementation and user managed memory segments, and thus to support zero copy implementations for I/O operations.

Python applications traditionally tend not to interfere with Python level memory management, and zero copy implementations are not a first level concern. The Python bindings thus flatten the `buffer` class into plain data arrays (strings actually, which can contain encoded data), e.g. for file I/O, or flattens the buffer semantics into inheriting classes, e.g for the `rpc.Parameter` and `message.Message` classes.

2.2 SAGA Attributes and Python Properties

Python's native way to express class attributes is to expose them as class or object properties. The SAGA Python bindings follow that model. SAGA Attributes have, however, as lightly different semantic in most cases: they do not represent attributes of the local application class instance, but mostly properties of remote entities that these class instances represent. In that context, it must be noted that

- cannot be accessed via asynchronous operations,
- cannot be monitored via callbacks,

- cannot be inspected for vector / scalar types,
- cannot be listed (?),
- may not be extensible (unlike in python proper).

For those reasons, a GFD.90-like attribute interface is also provided in Python. Following similar arguments, the property interface is also provided as complement to various `get_xyz()` methods (readonly), and to `get_xyz()/set_xyz()` pairs (read/write). Finally, the property interface is in some cases used to expose local object state in general. For example, a `saga.Session` object will expose a `'contexts'` list as properties, whose manipulation maps to the default `add_context()/remove_context()` methods.

Another way to expose attributes in Python is the dict(ionary) interface. Compared to the property interface, a dict additionally allows inspection of and iteration over attribute keys. Despite that additionally exposed semantics (which maps well to the GFD.90 attribute semantics), the SAGA Python bindings will not be expressed via the dict interface, to keep the binding focused and simple.

As in GFD.90, attribute and metric names are specified in 'CamelCase'. As per Python convention [?], property names are changed to `'under_score'` notation.

2.3 Attribute Value Types

GFD.90 defines the attribute value types, but explicitly maps those to strings. As Python provides flexible and Transparent means of type conversion, the Python bindings support natively typed attribute values.

The `saga.job.Description`'s `Environment` attribute is types as list of strings, where the strings are formatted as `"key=value"`. Additionally, the Python bindings allow to express that attribute's value as a python dictionary.

2.4 Enums and Defines

The SAGA API includes a number of enums, which are usually related to classes within a specific API package. Python does not have a native notion of enums – those are commonly [?] expressed as module variables. This is also reflected in the present Python bindings.

Further, GFD.90 recommends bindings to define constants expressions for predefined attribute and metric names. Those are also defined as module variables.

Note that module variables (enums and string defines) are in all `UPPER_CASE`, as suggested by [?].

2.5 Comparison to PySAGA a SAGA-Python

The Python bindings as defined in this document deviate slightly from the existing Python implementations of SAGA, namely PySAGA, SAGA-Python and Bliss – after all, the explicit purpose of this document is to reconcile the various exising SAGA Python APIs. As PySAGA is the API closest to the binding described here, and its binding is well documented [**?**] and motivated [**?**], we here only detail the differences to PySAGA. Compared to the PySAGA API, this document

FIXME: add refs, also in intro.

- contains updates to synchronize the API with GFD.90 errata;
- changes the attribute interface from `Attributes.attributes['key']` to `Attributes.key`;
- removes the Buffer and StdIO classes, replacing them with strings;
- allow implementations to flatten interfaces into inheriting classes;
- define enum constants on module level, in `UPPER_CASE` notation;
- remove naming redundancies, such as `job.JobDescription` vs. `job.Description` etc.

3 Example Code

4 Intellectual Property Issues

4.1 Contributors

This document is the result of the joint efforts of many contributors. The author listed here and on the title page is the one taking responsibility for the content of the document, and all errors. The editor (underlined) is committed to taking permanent stewardship for this document and can be contacted in the future for inquiries.

Your Name
you@mail.net
Some Institution or Company
or Whatever...
123 Some Street
45678 Some Place
Some Where

4.2 Intellectual Property Statement

4.3 Disclaimer

4.4 Full Copyright Notice

A Python Binding as Interface Code

```
# Core API: saga/exception.py
# ===========================

class SagaException (Exception) :
  __init__      (self, message, object) : pass # check GFD.90
  get_message   (self) : pass # string
  get_object    (self) : pass # <Object>
  get_traceback (self) : pass # string

  message   = property (get_message)
  object    = property (get_object)
  traceback = property (get_traceback)

class NotImplemented       (SagaException) : pass
class IncorrectURL         (SagaException) : pass
class BadParameter         (SagaException) : pass
class AlreadyExists        (SagaException) : pass
class DoesNotExist         (SagaException) : pass
class IncorrectState       (SagaException) : pass
class PermissionDenied     (SagaException) : pass
class AuthorizationFailed  (SagaException) : pass
class AuthenticationFailed (SagaException) : pass
class Timeout              (SagaException) : pass
class NoSuccess            (SagaException) : pass
```

```
# Core API: saga/object.py
# ========================

# The saga.Object class in python would be almost empty, as get_type()
# is not needed (python has type inspection); get_id() is not needed
# (python object instances have IDs); and clone() is not needed
# (python core library provides deep copy) -- the last call,
# get_session(), can easily be added to all session managed saga
# objects.
# Core API: saga/url.py
# =====================

class Url (object) :
  def __init__       (self, url)      : pass # None
# def set_string     (self, name)     : pass # None
# def get_string     (self)           : pass # string
# def set_scheme     (self, scheme)   : pass # None
# def get_scheme     (self)           : pass # string
# def set_host       (self, host)     : pass # None
# def get_host       (self)           : pass # string
# def set_port       (self, port)     : pass # None
# def get_port       (self)           : pass # int
# def set_fragment   (self, fragment) : pass # None
# def get_fragment   (self)           : pass # string
# def set_path       (self, path)     : pass # None
# def get_path       (self)           : pass # string
# def set_query      (self, query)    : pass # None
# def get_query      (self)           : pass # string
# def set_userinfo   (self, userinfo) : pass # None
# def get_userinfo   (self)           : pass # string
  def translate      (self, scheme)   : pass # URL
  def __str__        (self)           : pass # string

  scheme    = property (get_scheme,   set_scheme  )
  host      = property (get_host,     set_host    )
  port      = property (get_port,     set_port    )
  fragment  = property (get_fragment, set_fragment)
  path      = property (get_path,     set_path    )
  query     = property (get_query,    set_query   )
  userinfo  = property (get_userinfo, set_userinfo)
```

```
# Core API: saga/context.py
# ========================

# Context attributes:
# FIXME: check against GFD.90
TYPE            = "Type"
SERVER          = "Server"
CERT_REPOSITORY = "CertRepository"
USER_PROXY      = "UserProxy"
USER_CERT       = "UserCert"
USER_KEY        = "UserKey"
USER_ID         = "UserID"
USER_PASS       = "UserPass"
USER_VO         = "UserVO"
LIFETIME        = "LifeTime"
REMOTE_ID       = "RemoteID"
REMOTE_PORT     = "RemotePort"

class Context (attributes.Attributes) :
  def __init__     (self, name=None)  : pass # None
  def set_defaults (self)             : pass # None
```

```
# Core API: saga/session.py
# ========================

class Session (object) :
  def __init__       (self, default=True) : pass # None
  def add_context    (self, c)            : pass # None
  def remove_context (self, c)            : pass # None
  def list_contexts  (self)               : pass # None
  def close          (self, timeout=None) : pass # None

  contexts = property (...) # mutable list of contexts
```

```
# Core API: saga/permissions.py
# ============================

# permission flags enum:
QUERY =  1
READ  =  2
WRITE =  4
EXEC  =  8
OWNER = 16
ALL   = 31

class Permissions (task.Async) :
  def permissions_allow (self, ugid, perm, ttype=None) : pass # None
  def permissions_deny  (self, ugid, perm, ttype=None) : pass # None
  def permissions_check (self, ugid, perm, ttype=None) : pass # bool
  def permissions_list  (self,             ttype=None) : pass # dict
  def get_owner         (self,             ttype=None) : pass # string
  def get_group         (self,             ttype=None) : pass # string

  permissions = property (permissions_list)
  owner       = property (get_owner)
  group       = property (get_group)
```

```
# Core API: saga/attributes.py
# ============================

class Attributes (object) :
  def set_attribute          (self, key, val) : pass   # None
  def get_attribute          (self, key)      : pass   # string
  def set_vector_attribute   (self, key, val) : pass   # None
  def get_vector_attribute   (self, key)      : pass   # [string]
  def remove_attribute       (self, key)      : pass   # None
  def list_attributes        (self)           : pass   # [string]
  def find_attributes        (self, pattern)  : pass   # [string]
  def attribute_exists       (self, key)      : pass   # bool
  def attribute_is_readonly  (self, key)      : pass   # bool
  def attribute_is_writable  (self, key)      : pass   # bool
  def attribute_is_removable (self, key)      : pass   # bool
  def attribute_is_vector    (self, key)      : pass   # bool
  def attributes_as_dict     (self)           : pass   # dict{key:val}

  # Attributes are also exposed as Python properties.
```

```
# Core API: saga/metric.py
# ========================

# Metric attributes:
NAME        = "Name"
DESCRIPTION = "Description"
MODE        = "Mode"
UNIT        = "Unit"
TYPE        = "Type"
VALUE       = "Value"

class Callback (object) :
  def cb  (self, monitorable, metric, cctx)   : pass # bool

class Metric (attributes.Attributes) :
  def __init__        (self, name, desc, mode,
                             unit, type, val) : pass # None
  def add_callback    (self, cb)              : pass # None
  def remove_callback (self, cookie)          : pass # None
  def fire            (self)                  : pass # None

class Monitorable (object) :
  def list_metrics    (self)                  : pass # [string]
  def get_metric      (self, name)            : pass # None
  def add_callback    (self, name, cb)        : pass # int
  def remove_callback (self, name, id)        : pass # None
  def list_callbacks  (self)                  : pass # dict{id:cp}

  callbacks = property (list_callbacks)
  metrics   = property (list_metrics)

class Steerable (Monitorable) :
  def add_metric      (self, metric)          : pass # None
  def remove_metric   (self, name)            : pass # None
  def fire_metric     (self, name)            : pass # None
```

```
# Core API: saga/task.py
# =====================

# task state enum:
NEW      = "New"
RUNNING  = "Running"
DONE     = "Done"
CANCELED = "Canceled"
FAILED   = "Failed"

# TaskContainer wait_mode enum:
ALL      = "All"
ANY      = "Any"

# Task type enum:
SYNC     = "Sync"
ASYNC    = "Async"
TASK     = "Task"

# Task and TaskContainer metrics:
STATE    = "State"      # FIXME: why no state_detail?

class Async () :        # tagging class
  pass

class Task (monitoring.Monitorable) :
  def run         (self)                         : pass # None
  def cancel      (self, timeout=None)           : pass # None
  def wait        (self, timeout=-1)             : pass # None
  def get_state   (self)                         : pass # state
  def get_result  (self)                         : pass # <restype>
  def get_object  (self)                         : pass # <objtype>
  def raise       (self)                         : pass # <Exception>

  state     = property (get_state)
  result    = property (get_result)
  object    = property (get_object)
  exception = property (get_exception)

class TaskContainer (monitoring.Monitorable) :
  # check GFD.90 for cookies
  def __init__    (self)                         : pass # None
  def add         (self, task)                   : pass # None
  def remove      (self, cookie)                 : pass # None
  def run         (self)                         : pass # None
  def wait        (self, waitmode=ALL, timeout=-1):pass # [task]
  def cancel      (self, timeout=None)           : pass # None
  def size        (self)                         : pass # int
  def list_tasks  (self)                         : pass # [cookies]
  def get_task    (self, cookie)                 : pass # Task
  def get_tasks   (self)                         : pass # [Task]
  def get_states  (self)                         : pass # [state]

  size   = property (get_size)
  tasks  = property (...)
  states = property (get_states)
```

```
# Job API Package : saga/job/job.py
# =================================

# job states enum:
NEW                    = task.NEW
RUNNING                = task.RUNNING
DONE                   = task.DONE
CANCELED               = task.CANCELED
FAILED                 = task.FAILED
SUSPENDED              = "Suspended"

# JobDescription attributes:
EXECUTABLE             = "Executable"
ARGUMENTS              = "Arguments"
SPMD_VARIATION         = "SPMDVariation"
TOTAL_CPU_COUNT        = "TotalCPUCount"
NUMBER_OF_PROCESSES    = "NumberOfProcesses"
PROCESSES_PER_HOST     = "ProcessesPerHost"
THREADS_PER_PROCESS    = "ThreadsPerProcess"
ENVIRONMENT            = "Environment"          # dict or [string]
WORKING_DIRECTORY      = "WorkingDirectory"
INTERACTIVE            = "Interactive"
INPUT                  = "Input"
OUTPUT                 = "Output"
ERROR                  = "Error"
FILE_TRANSFER          = "FileTransfer"
CLEANUP                = "Cleanup"
JOB_START_TIME         = "JobStartTime"
TOTAL_CPU_TIME         = "TotalCPUTime"
TOTAL_PHYSICAL_MEMORY  = "TotalPhysicalMemory"
CPU_ARCHITECTURE       = "CPUArchitecture"
OPERATING_SYSTEM_TYPE  = "OperatingSystemType"
CANDIDATE_HOSTS        = "CandidateHosts"
QUEUE                  = "Queue"
JOB_CONTACT            = "JobContact"

# Job attributes:
JOB_ID                 = "JobID"
EXECUTION_HOSTS        = "ExecutionHosts"
CREATED                = "Created"
STARTED                = "Started"
FINISHED               = "Finished"
EXIT_CODE              = "ExitCode"
TERMSIG                = "Termsig"
# WORKING_DIRECTORY    = "WorkingDirectory"  # collision

# Job metrics:
STATE                  = "State"
STATE_DETAIL           = "StateDetail"
SIGNAL                 = "Signal"
CPU_TIME               = "CPUTime"
MEMORY_USE             = "MemoryUse"
VMEMORY_USE            = "VmemoryUse"
PERFORMANCE            = "Performance"
```

```
class Description (attributes.Attributes) :
  pass

class Service (task.Async) :
  def __init__   (self, rm=None, s=None)             : pass # None
  def create     (self, rm=None, s=None, ttype=None) : pass # Task
  def create_job (self, jd,              ttype=None) : pass # Job
  def run_job    (self, cmd, host="",    ttype=None) : pass # Job
  def list       (self,                  ttype=None) : pass # [string]
  def get_job    (self, job_id,          ttype=None) : pass # Job
  def get_self   (self,                  ttype=None) : pass # Self

  jobs = property (list)

class Job (task.Task,  attributes.Attributes,
           task.Async, permissions.Permissions) :
  def get_job_description (self,         ttype=None) : pass # Descr.
  def get_stdin           (self,         ttype=None) : pass # os.File
  def get_stdout          (self,         ttype=None) : pass # os.File
  def get_stderr          (self,         ttype=None) : pass # os.File
  def suspend             (self,         ttype=None) : pass # None
  def resume              (self,         ttype=None) : pass # None
  def checkpoint          (self,         ttype=None) : pass # None
  def migrate             (self, jd,     ttype=None) : pass # None
  def signal              (self, signum, ttype=None) : pass # None

  stdin  = property (get_stdin )
  stdout = property (get_stdout)
  stderr = property (get_stderr)

class Self (Job, monitoring.Steerable) :
  pass
```

```
# Namespace API Package: saga/namespace/namespace.py
# ===================================================

# namespace flags enum:
OVERWRITE        =  1
RECURSIVE        =  2
DEREFERENCE      =  4
CREATE           =  8
EXCLUSIVE        = 16
LOCK             = 32
CREATE_PARENTS   = 64

class Entry (permissions.Permissions, task.Async) :
  def __init__       (self, name, s=None,flags=None)              : pass # None
  def create         (self, name, s=None,flags=None, ttype=None) : pass # Task
  def get_url        (self,                          ttype=None) : pass # Url
  def get_cwd        (self,                          ttype=None) : pass # string
  def get_name       (self,                          ttype=None) : pass # string
  def is_dir_self    (self,                          ttype=None) : pass # bool
  def is_entry_self  (self,                          ttype=None) : pass # bool
  def is_link_self   (self,                          ttype=None) : pass # bool
  def read_link_self (self,                          ttype=None) : pass # Url
  def copy_self      (self, tgt,         flags=None, ttype=None) : pass # None
  def link_self      (self, tgt,         flags=None, ttype=None) : pass # None
  def move_self      (self, tgt,         flags=None, ttype=None) : pass # None
  def remove_self    (self,              flags=None, ttype=None) : pass # None
  def close          (self, timeout=None,            ttype=None) : pass # None
  def permissions_allow_self (self, id, p, flags=None, ttype=None) : pass # None
  def permissions_deny_self  (self, id, p, flags=None, ttype=None) : pass # None

  url  = property (...)
  cwd  = property (...)
  name = property (...)

class Directory (Entry, task.Async) :
  def change_dir      (self, url,                   ttype=None) : pass # None
  def list            (self, npat=".", flags=None, ttype=None) : pass # [Url]
  def find            (self, npat,     flags=RECURSIVE,
                                                    ttype=None) : pass # [Url]
  def exists          (self, name,                  ttype=None) : pass # bool
  def is_dir          (self, name,                  ttype=None) : pass # bool
  def is_entry        (self, name,                  ttype=None) : pass # bool
  def is_link         (self, name,                  ttype=None) : pass # bool
  def read_link       (self, name,                  ttype=None) : pass # Url
  def get_num_entries (self,                        ttype=None) : pass # int
  def get_entry       (self, enum,                  ttype=None) : pass # Url
  def copy            (self, src, tgt, flags=None, ttype=None) : pass # None
  def link            (self, src, tgt, flags=None, ttype=None) : pass # None
  def move            (self, src, tgt, flags=None, ttype=None) : pass # None
  def remove          (self, tgt,      flags=None, ttype=None) : pass # None
  def make_dir        (self, tgt,      flags=None, ttype=None) : pass # None
  def open            (self, name,     flags=None, ttype=None) : pass # Entry
  def open_dir        (self, name,     flags=None, ttype=None) : pass # Direct.


  def permissions_deny  (self, tgt, id, p, flags=None, ttype=None) : pass # None
  def permissions_allow (self, tgt, id, p, flags=None, ttype=None) : pass # None

  num_entries = property (get_num_entries)
```

```
# Filesystem API Package: saga/filesystem/filesystem.py
# ====================================================

# filesystem flags enum:
OVERWRITE      =    1
RECURSIVE      =    2
DEREFERENCE    =    4
CREATE         =    8
EXCLUSIVE      =   16
LOCK           =   32
CREATE_PARENTS =   64
TRUNCATE       =  128
APPEND         =  256
READ           =  512
WRITE          = 1024
READ_WRITE     = 1536
BINARY         = 2048

# filesystem seek_mode enum:
START          = "Start"
CURRENT        = "Current"
END            = "End"

class File (namespace.Entry, task.Async) :
  def is_file_self  (self,                    ttype=None) : pass # bool
  def get_size_self (self,                    ttype=None) : pass # int
  def read          (self,       size=-1,     ttype=None) : pass # string
  def write         (self, data, size=-1,     ttype=None) : pass # int
  def seek          (self, off, whence=START, ttype=None) : pass # int
  def read_v        (self, off,               ttype=None) : pass # string
  def write_v       (self, data,              ttype=None) : pass # int
  def size_p        (self, pattern,           ttype=None) : pass # int
  def read_p        (self, pattern,           ttype=None) : pass # string
  def write_p       (self, pattern, data,     ttype=None) : pass # int
  def modes_e       (self,                    ttype=None) : pass # [string]
  def size_e        (self, emode, spec,       ttype=None) : pass # int
  def read_e        (self, emode, spec,       ttype=None) : pass # string
  def write_e       (self, emode, spec, data, ttype=None) : pass # int

  size = property (get_size_self)

class Directory (namespace.Directory, task.Async) :
  def get_size      (self, name, flags=None,  ttype=None) : pass # int
  def is_file       (self, name,              ttype=None) : pass # Bool
  def open_dir      (self, name, flags=READ,  ttype=None) : pass # Directory
  def open          (self, name, flags=READ,  ttype=None) : pass # File
```

```
# Replica API Package: saga/replica/replica.py
# ==============================================

# replica flags enum:
OVERWRITE      =    1
RECURSIVE      =    2
DEREFERENCE    =    4
CREATE         =    8
EXCLUSIVE      =   16
LOCK           =   32
CREATE_PARENTS =   64
#                 128 # reserved for TRUNCATE
#                 256 # reserved for APPEND
READ           =  512
WRITE          = 1024
READ_WRITE     = 1536
#                2048 # reserved for BINARY

class LogicalFile (namespace.Entry, attributes.Attributes, task.Async) :
  def is_file_self    (self,                   ttype=None) : pass # Bool
  def add_location    (self, name,             ttype=None) : pass # None
  def remove_location (self, name,             ttype=None) : pass # None
  def update_location (self, old, new,         ttype=None) : pass # None
  def list_locations  (self,                   ttype=None) : pass # [Url]
  def replicate       (self, name, flags=None, ttype=None) : pass # None

class LogicalDirectory (namespace.Directory,
                        attributes.Attributes, task.Async) :
  def is_file         (self, name,             ttype=None) : pass # Bool
  def open_dir        (self, name, flags=READ, ttype=None) : pass # Directory
  def open            (self, name, flags=READ, ttype=None) : pass # LogicalFile
  def find            (self, name_pattern, attr_pattern,
                       flags=RECURSIVE,        ttype=None) : pass # [Url]
```

```
# Stream API Package: saga/stream/stream.py
# ========================================

# stream state enum:
NEW            = "New"
OPEN           = "Open"
CLOSED         = "Closed"
# DROPPED      = "Dropped" # see metric
ERROR          = "Error"

# stream activity enum: # see Metrics
# READ         = "Read"
# WRITE        = "Write"
# EXCEPTION    = "Exception"

# StreamService metric
CLIENT_CONNECT = "client_connect"

# Stream attributes
TIMEOUT        = "Timeout"
BLOCKING       = "Blocking"
COMPRESSION    = "Compression"
NODELAY        = "Nodelay"
RELIABLE       = "Reliable"

# Stream metrics
STATE          = "State"
READ           = "Read"
WRITE          = "Write"
EXCEPTION      = "Exception"
DROPPED        = "Dropped"

class Service (monitoring.Monitorable, permissions.Permissions, task.Async) :
  def __init__    (self, name=None, s=None,  ttype=None) : pass # None
  def create      (self, name=None, s=None,  ttype=None) : pass # None
  def get_url     (self,                     ttype=None) : pass # Url
  def serve       (self, timeout=-1,         ttype=None) : pass # Stream
  def close       (self, timeout=None,       ttype=None) : pass # None

  url = property (get_url)

class Stream (attributes.Attributes, monitoring.Monitorable, task.Async) :
  def __init__    (self, name=None, s=None)              : pass # None
  def create      (self, name=None, s=None,  ttype=None) : pass # None
  def get_url     (self,                     ttype=None) : pass # Url
  def get_context (self,                     ttype=None) : pass # Context
  def connect     (self,                     ttype=None) : pass # None
  def wait        (self, what, timeout=-1,   ttype=None) : pass # None
  def close       (self,       timeout=None, ttype=None) : pass # None
  def read        (self,       size=-1,      ttype=None) : pass # string
  def write       (self, data, size=-1,      ttype=None) : pass # None

  url     = property (get_url)
  context = property (get_context)
```

```
# Remote Procedure Calls API Package: saga/rpc/rpc.py
# =====================================================

# rpc io_mode enum:
IN    = "In"
OUT   = "Out"
INOUT = "InOut"

class Parameter (object) :
  def __init__       (self, data=None, size=-1, mode=IN)  : pass # None
  def set_io_mode    (self, mode)                         : pass # None
  def get_io_mode    (self)                               : pass # mode
  def get_size       (self)                               : pass # int
  def set_size       (self, size)                         : pass # None
  def get_data       (self)                               : pass # string
  def set_data       (self, data)                         : pass # None
  def close          (self)                               : pass # None

  io_mode = property (get_io_mode, set_io_mode)
  size    = property (get_size, set_size)
  data    = property (get_data, set_data)

class RPC (permissions.Permissions, task.Async) :
  def __init__       (self, funcname, s=None)             : pass # None
  def create         (self, funcname, s=None, ttype=None) : pass # Task
  def call           (self, parameters,       ttype=None) : pass # None
  def close          (self, timeout=None,     ttype=None) : pass # mode
```

```
# Advert API Package: saga/advert/advert.py
# ========================================

# advert flags
OVERWRITE      =    1
RECURSIVE      =    2
DEREFERENCE    =    4
CREATE         =    8
EXCLUSIVE      =   16
LOCK           =   32
CREATE_PARENTS =   64
TRUNCATE       =  128
#                 256 # reserved for APPEND
READ           =  512
WRITE          = 1024
READ_WRITE     = 1536
#                2048 # reserved for BINARY

# Advert metrics
ATTRIBUTE      = "attribute"
OBJECT         = "object"
# TTL          = "ttl"  # collision

# AdvertDirectory metrics
ATTRIBUTE      = "Attribute"
CHANGE         = "Change"
CREATE         = "Create"  # FIXME: conflict with flag
DELETE         = "Delete"
TTL            = "TTL"

class Advert (namespace.Entry, attributes.Attributes, task.Async) :
  def set_ttl_self    (self, ttl,             ttype=None) : pass # None
  def get_ttl_self    (self, ttl,             ttype=None) : pass # int
  def store_object    (self, object,          ttype=None) : pass # None
  def retrieve_object (self,                  ttype=None) : pass # <object>
  def delete_object   (self,                  ttype=None) : pass # None

class AdvertDirectory (namespace.Directory,
                       attributes.Attributes, task.Async) :
  def set_ttl_self    (self,      ttl,        ttype=None) : pass # None
  def get_ttl_self    (self,      ttl,        ttype=None) : pass # int
  def set_ttl         (self, tgt, ttl,        ttype=None) : pass # None
  def get_ttl         (self, tgt, ttl,        ttype=None) : pass # int
  def find            (self, name_pattern, attr_pattern, obj_type,
                       flags=RECURSIVE,       ttype=None) : pass # [Url]
```

```
# Message API Package: saga/message/message.py
# ==============================================

# message state enum:
OPEN               = "Open"
CLOSED             = "Closed"

# default for message property enums:
ANY                = "Any"

# message topology enum:
POINT_TO_POINT     = "PointToPoint"
MULTICAST          = "Multicast"
PUBLISH_SUBSCRIBER = "PublishSubscriber"
PEER_TO_PEER       = "PeerToPeer"

# message reliability enum:
UNRELIABLE         = "Unreliable"
CONSISTENT         = "Consistent"
SEMI_RELIABLE      = "SemiReliable"
RELIABLE           = "Reliable"

# message atomicity enum:
AT_MOST_ONCE       = "AtMostOnce"
AT_LEAST_ONCE      = "AtLeastOnce"
EXACTLY_ONCE       = "ExactlyOnce"

# message correctness enum:
UNVERIFIED         = "Unverified"
VERIFIED           = "Verified"

# message ordering enum:
UNORDERED          = "Unordered"
ORDERED            = "Ordered"
GLOBALLY_ORDERED   = "GloballyOrdered"

# endpoint attributes:
TOPOLOGY           = "Topology"
RELIABILITY        = "Reliability"
ATOMICITY          = "Atomicity"
CORRECTNESS        = "Correctness"
ORDERING           = "Ordering"

# endpoint metrics:
STATE              = "State"
CONNECT            = "Connect"
CLOSED             = "Closed"
MESSAGE            = "Message"

# message attributes:
ID                 = "ID"
SENDER             = "Sender"
```

```
class Endpoint (monitoring.Monitorable, task.Async) :
  def create        (self, topology    = POINT_TO_POINT,
                           reliability = RELIABLE,
                           atomicity   = EXACTLY_ONCE,
                           ordering    = ORDERED,
                           correctness = VERIFIED
                           session     = None)              : pass # None
  def create        (self, topology    = POINT_TO_POINT,
                           reliability = RELIABLE,
                           atomicity   = EXACTLY_ONCE,
                           ordering    = ORDERED,
                           correctness = VERIFIED
                           session     = None,  ttype=None) : pass # None
  def get_url       (self,                      ttype=None) : pass # Url
  def get_receivers (self,                      ttype=None) : pass # [Url]
  def serve         (self, n=-1, timeout=-1,    ttype=None) : pass # None
  def serve_once    (self,       timeout=-1,    ttype=None) : pass # Endpoint
  def close         (self, receiver=None,       ttype=None) : pass # None

  def send          (self, msg, receivers=None, ttype=None) : pass # None
  def test          (self, sender=None, receiver=None,
                           timeout=-1,          ttype=None) : pass # int
  def recv          (self, sender=None, receiver=None,
                           timeout=-1,          ttype=None) : pass # Message

  url       = property (get_url)
  receivers = property (get_receivers)


class Message (saga.Attributes) :
  def __init__      (self, data=None, size=-1)              : pass # None
  def get_sender    (self)                                  : pass # Url
  def get_id        (self)                                  : pass # string
  def get_size      (self)                                  : pass # int
  def set_size      (self, size)                            : pass # None
  def get_data      (self)                                  : pass # string
  def set_data      (self, data)                            : pass # None
  def close         (self)                                  : pass # None

  sender = property (get_sender)
  id     = property (get_id)
  size   = property (get_size, set_size)
  data   = property (get_data, set_data)
```

```
# Service Discovery API Package: saga/sd/sd.py
# ==============================================

# ServiceDescription attributes
ATTRIBUTE        = "Url"
OBJECT           = "Type"
UID              = "UID"
SITE             = "Site"
NAME             = "Name"
IMPLEMENTOR      = "Implementor"
RELATED_SERVICES = "RelatedServices" # FIXME: coll. w/ get_related_services ()

class Discoverer (object) :          # FIXME: no async??
  def __init__       (self, url, session=None) : pass # None
  def list_services  (self, service_filter, data_filter,
                      authz_filter=None)       : pass # [ServiceDescription]

class ServiceDescription (attributes.Attributes)     :
  def get_url        (self)                    : pass # Url
  def get_data       (self)                    : pass # ServiceData
  def get_related_services (self)              : pass # [ServiceDescription]

  url              = property (get_url)
  data             = property (get_data)
  related_services = property (get_related_services)

class ServiceData (attributes.Attributes) :
  pass
```

```
# Information Service Navigator API Package: saga/isn/isn.py
# ===========================================================

class EntityDataSet (object) : # FIXME: no async??
  def __init__             (self, model, name, filter,
                            url=None, session=s)  : pass # None
  def get_data             (self)                 : pass # [EntityData]
  def get_related_entities (name, filter=None)    : pass # EntityDataSet
  def list_related_entity_names
                           (self)                 : pass # [string]

class EntityData (attributes.Attributes) :
  pass
```