GFD-R-P.xxx
SAGA-RG

Andre Merzky[1]
Mark Santcroos
Steve Fisher
Ole Weidner

Version: 1.0

December 14, 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 a SAGA language binding, it depends upon the SAGA Core API Specification [1], and on the currently 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.

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 [1], 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 [1] 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 in appendix A.

2.1 Class Hierarchy Considerations

As for 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`.

The SAGA API defines an interface and class hierarchy which is normally followed by language bindings. For Python, a strict adherence to that hierarchy is neither required nor useful: Python's duck-typing paradigm [**?**] encourages to flatten inherited base classes into the actual object implementations. The paragraphs below discuss the cases where this is used in 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 MAY 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). Instead, all SAGA objects MUST (directly or indirectly inherit from Python's base `object` class, and are thus rendered as *'new style'* classes [**?**].

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. The namespace package thus functions as an interface package, and 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.

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 slightly 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 they:

- 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. We follow the recommendation in [**?**] to define constants on a module level, written in all capital letters with underscores separating words.

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 [**?**].

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 the content of the 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

The OGF takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the OGF Secretariat.

The OGF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights which may cover technology that may be required to practice this recommendation. Please address the information to the OGF Executive Director.

4.3 Disclaimer

This document and the information contained herein is provided on an "As Is" basis and the OGF disclaims all warranties, express or implied, including but not limited to any warranty that the use of the information herein will not infringe any rights or any implied warranties of merchantability or fitness for a particular purpose.

4.4 Full Copyright Notice

References

[1] T. Goodale, S. Jha, H. Kaiser, T. Kielmann, P. Kleijer, A. Merzky, J. Shalf, and C. Smith. GFD.90 – SAGA Core API Specification. OGF Proposed Recommendation, Open Grid Forum, 2007.

A Python Binding as Interface Code

This appendix contains the normative Python Bindings, as python source code. Any Python implementation of SAGA SHOULD define this and only this interface. Note that Python does not allow to specify method return types, nor does it enforce types in the first place. The return types for methods are given as comments, and SHOULD be respected by the implementation. **FIXME: ...**

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

class SagaException (Exception) :
  __init__      (self, message, object=None) : pass
  #   message:   string
  #   object:    <object type>
  #   ret:       obj

  get_message   (self)                       : pass
  #   ret:       string

  get_object    (self)                       : pass
  #   ret:       any

  get_traceback (self)                       : pass
  #   ret:       string

  get_all_exceptions (self)                  : pass
  #   ret:       list [Exception]

  get_all_messages (self)                    : pass
  #   ret:       list [string]


  message    = property (get_message)         # string
  object     = property (get_object)          # object type
  traceback  = property (get_traceback)       # string
  exceptions = property (get_all_exceptions)  # list [Exception]
  messages   = property (get_all_message)     # list [string]



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
  #   url:     string
  #   ret:     None

  def __str__        (self)           : pass
  #   ret:     string

  def translate      (self, scheme)   : pass
  #   scheme:  string
  #   ret:     saga.Url


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

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

# Context attributes:
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_HOST     = "RemoteID"
REMOTE_PORT     = "RemotePort"

class Context (attributes.Attributes) :
  def __init__     (self, name=None)  : pass
  #   name: string     # type is a reserved word, thus 'name'
  #   ret:  None

  def set_defaults (self)             : pass
  #   ret:  None
```

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

class Session (object) :
  def __init__       (self, default=True) : pass
  #   default: bool
  #   ret:     None

  def add_context    (self, ctx)            : pass
  #   ctx:     saga.Context
  #   ret:     None

  def remove_context (self, ctx)            : pass
  #   ctx:     saga.Context
  #   ret:     None

  def list_contexts  (self)               : pass
  #   ret:     list[saga.Context]


  contexts = property (...)  # mutable list [saga.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
  #   ugid :       string
  #   perm :       flags enum
  #   ttype:       saga.task.type enum
  #   ret:         None / saga.Task

  def permissions_deny  (self, ugid, perm, ttype=None) : pass
  #   ugid :       string
  #   perm :       flags enum
  #   ttype:       saga.task.type enum
  #   ret:         None / saga.Task

  def permissions_check (self, ugid, perm, ttype=None) : pass
  #   ugid :       string
  #   perm :       flags enum
  #   ttype:       saga.task.type enum
  #   ret:         bool / saga.Task

  def permissions_list  (self,             ttype=None) : pass
  #   ttype:       saga.task.type enum
  #   ret:         dict {string ugid : flags enum} / saga.Task

  def get_owner         (self,             ttype=None) : pass
  #   ttype:       saga.task.type enum
  #   ret:         string / saga.Task

  def get_group         (self,             ttype=None) : pass
  #   ttype:       saga.task.type enum
  #   ret:         string / saga.Task


  permissions = property (permissions_list)  # dict {string ugid : flags enum}
  owner       = property (get_owner)         # string
  group       = property (get_group)         # string
```

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

class Attributes (object, task.Async) :
  def set_attribute          (self, key, val, ttype=None)  : pass
  #   key:         string
  #   val:         string / dict / any
  #   ttype:       saga.task.type enum
  #   ret:         None / saga.Task

  def get_attribute          (self, key,      ttype=None)  : pass
  #   key:         string key
  #   ttype:       saga.task.type enum
  #   ret:         string / dict / any / saga.Task

  def set_vector_attribute   (self, key, val, ttype=None)  : pass
  #   key:         string
  #   val:         list [string]
  #   ttype:       saga.task.type enum
  #   ret:         None / saga.Task

  def get_vector_attribute   (self, key,      ttype=None)  : pass
  #   key:         string
  #   ttype:       saga.task.type enum
  #   ret:         list [string] / saga.Task

  def remove_attribute       (self, key,      ttype=None)  : pass
  #   key:         string
  #   ttype:       saga.task.type enum
  #   ret:         None / saga.Task

  def list_attributes        (self,           ttype=None)  : pass
  #   ttype:       saga.task.type enum
  #   ret:         [string] / saga.Task

  def find_attributes        (self, pat,      ttype=None)  : pass
  #   pat:         string
  #   ttype:       saga.task.type enum
  #   ret:         [string] / saga.Task

  def attribute_exists       (self, key,      ttype=None)  : pass
  #   key:         bool
  #   ttype:       saga.task.type enum
  #   ret:         bool / saga.Task

  def attribute_is_readonly  (self, key,      ttype=None)  : pass
  #   key:         bool
  #   ttype:       saga.task.type enum
  #   ret:         bool / saga.Task

  def attribute_is_writable  (self, key,      ttype=None)  : pass
  #   key:         bool
  #   ttype:       saga.task.type enum
  #   ret:         bool / saga.Task

  def attribute_is_removable (self, key,      ttype=None)  : pass
```

```
  #   key:         bool
  #   ttype:       saga.task.type enum
  #   ret:         bool / saga.Task

  def attribute_is_vector    (self, key,      ttype=None)  : pass
  #   key:         bool
  #   ttype:       saga.task.type enum
  #   ret:         bool / saga.Task

  def as_dict                (self,           ttype=None)  : pass
  #   ttype:       saga.task.type enum
  #   ret:         dict {key : val} / saga.Task


  # 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, ctx)                    : pass
  #   monitorable: any
  #   metric :     saga.Metric
  #   ctx:         saga.Context
  #   ret:         bool


class Metric (attributes.Attributes) :
  def __init__     (self, name, desc, mode, unit, type, val) : pass
  #   name :       string
  #   desc :       string
  #   mode :       string
  #   unit :       string
  #   type :       string
  #   val  :       any
  #   ret  :       None

  def add_callback (self, cb)                                : pass
  #   cb:          saga.Callback
  #   ret:         None

  def remove_callback (self, cookie)                         : pass
  #   cookie:      int
  #   ret:         None

  def fire         (self)                                    : pass
  #   ret:         None



class Monitorable (object) :

  def list_metrics (self)                                    : pass
  #   ret:         list [string]

  def get_metric   (self, name)                              : pass
  #   name:        string
  #   ret:         saga.Metric

  def add_callback (self, name, cb)                          : pass
  #   name:        string
  #   cb:          saga.Callback
  #   ret:         int
```

```
  def remove_callback (self, name, cookie)                  : pass
  #   name:        string
  #   cookie:      int
  #   ret:         None

  def list_callbacks (self)                                 : pass
  #   ret:         dict {name:string : list [saga.Callback]}


  callbacks = property (list_callbacks)  # dict {name:string : list [saga.Callback]}
  metrics   = property (list_metrics)    # list [string]


###################################################################
#
class Steerable (Monitorable) :

  def add_metric     (self, metric) : pass
  #   metric:        saga.Metric
  #   ret:           None

  def remove_metric  (self, name)   : pass
  #   name:          string
  #   ret:           None

  def fire_metric    (self, name)   : pass
  #   name:          string
  #   ret:           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"

class Async () :        # tagging interface
  pass


class Task (monitoring.Monitorable) :
  def run         (self)                          : pass
  #   ret:        None

  def cancel      (self, timeout=None)            : pass
  #   timeout:    float
  #   ret:        None

  def wait        (self, timeout=-1)              : pass
  #   timeout:    float
  #   ret:        None

  def get_state   (self)                          : pass
  #   ret:        Task state enum

  def get_result  (self)                          : pass
  #   ret:        <result type>

  def get_object  (self)                          : pass
  #   ret:        <object type>

  def raise       (self)                          : pass
  #   ret:        <exception type>


  state     = property (get_state)       # state enum
  result    = property (get_result)      # result type
  object    = property (get_object)      # object type
  exception = property (get_exception)   # exception type
```

```
class TaskContainer (monitoring.Monitorable) :

  def __init__    (self)                          : pass
  #   ret:        None

  def add         (self, task)                    : pass
  #   task:       saga.Task
  #   ret:        None

  def remove      (self, task)                    : pass
  #   task:       saga.Task
  #   ret:        None

  def run         (self)                          : pass
  #   ret:        None

  def wait        (self, waitmode=ALL, timeout=-1):pass
  #   waitmode:   Task waitmode enum
  #   timeout:    float
  #   ret:        list [saga.Task]

  def cancel      (self, timeout=None)            : pass
  #   timeout:    float
  #   ret:        None

  def size        (self)                          : pass
  #   ret:        int

  def get_tasks   (self)                          : pass
  #   ret:        list [saga.Task]

  def get_states  (self)                          : pass
  #   ret:        list [Task state enum]


  size   = property (get_size)                 # int
  tasks  = property (add, remove, get_tasks)   # mutable list [saga.Task]
  states = property (get_states)               # list [state enum]
```

```
# 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 {string:string} / list [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, session=None)            : pass
  #   rm:        saga.Url
  #   session:   saga.Session
  #   ret:       obj

  def create     (self, rm=None, session=None, ttype=None) : pass
  #   rm:        saga.Url
  #   session:   saga.Session
  #   ttype:     saga.task.type enum
  #   ret:       saga.Task

  def create_job (self, jd,              ttype=None)       : pass
  #   jd:        saga.job.Description
  #   ttype:     saga.task.type enum
  #   ret:       saga.job.Job / saga.Task

  def run_job    (self, cmd, host="",    ttype=None)       : pass
  #   cmd:       string
  #   host:      string
  #   ttype:     saga.task.type enum
  #   ret:       saga.job.Job / saga.Task

  def list       (self,                  ttype=None)       : pass
  #   ttype:     saga.task.type enum
  #   ret:       list [string] / saga.Task

  def get_job    (self, job_id,          ttype=None)       : pass
  #   job_id:    string
  #   ttype:     saga.task.type enum
  #   ret:       saga.job.Job / saga.Task

  def get_self   (self,                  ttype=None)       : pass
  #   ttype:     saga.task.type enum
  #   ret:       saga.job.Self / saga.Task


  jobs = property (list)      # list [saga.job.Job]    # FIXME: dict {string id : saga.job.Job} ?
  self = property (get_self)  # saga.job.Self



class Job (task.Task,  attributes.Attributes,
           task.Async, permissions.Permissions) :
  def get_description (self,         ttype=None)          : pass
  #   ttype:     saga.task.type enum
  #   ret:       saga.job.Description / saga.Task

  def get_stdin  (self,              ttype=None)          : pass
  #   ttype:     saga.task.type enum
  #   ret:       os.File / saga.Task

  def get_stdout (self,              ttype=None)          : pass
```

```
  #   ttype:     saga.task.type enum
  #   ret:       os.File / saga.Task

  def get_stderr (self,              ttype=None)          : pass
  #   ttype:     saga.task.type enum
  #   ret:       os.File / saga.Task

  def suspend    (self,              ttype=None)          : pass
  #   ttype:     saga.task.type enum
  #   ret:       None / saga.Task

  def resume     (self,              ttype=None)          : pass
  #   ttype:     saga.task.type enum
  #   ret:       None / saga.Task

  def checkpoint (self,              ttype=None)          : pass
  #   ttype:     saga.task.type enum
  #   ret:       None / saga.Task

  def migrate    (self, jd,          ttype=None)          : pass
  #   jd:        saga.job.Description
  #   ttype:     saga.task.type enum
  #   ret:       None / saga.Task

  def signal     (self, signum,      ttype=None)          : pass
  #   signum:    int
  #   ttype:     saga.task.type enum
  #   ret:       None / saga.Task


  description = property (get_description)  # Description
  stdin       = property (get_stdin)        # os.File
  stdout      = property (get_stdout)       # os.File
  stderr      = property (get_stderr)       # os.File



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, session=None,flags=None)            : pass
  #   name:          saga.Url
  #   session:       saga.Session
  #   flags:         flags enum
  #   ret:           None

  def create         (self, name, session=None,flags=None, ttype=None) : pass
  #   name:          saga.Url
  #   session:       saga.Session
  #   flags:         flags enum
  #   ttype:         saga.task.type enum
  #   ret:           saga.Task

  def get_url        (self,                           ttype=None)       : pass
  #   ttype:         saga.task.type enum
  #   ret:           saga.Url / saga.Task

  def get_cwd        (self,                           ttype=None)       : pass
  #   ttype:         saga.task.type enum
  #   ret:           string / saga.Task

  def get_name       (self,                           ttype=None)       : pass
  #   ttype:         saga.task.type enum
  #   ret:           string / saga.Task

  def is_dir_self    (self,                           ttype=None)       : pass
  #   ttype:         saga.task.type enum
  #   ret:           bool / saga.Task

  def is_entry_self  (self,                           ttype=None)       : pass
  #   ttype:         saga.task.type enum
  #   ret:           bool / saga.Task

  def is_link_self   (self,                           ttype=None)       : pass
  #   ttype:         saga.task.type enum
  #   ret:           bool / saga.Task

  def read_link_self (self,                           ttype=None)       : pass
  #   ttype:         saga.task.type enum
  #   ret:           saga.Url / saga.Task

  def copy_self      (self, tgt,         flags=None, ttype=None)       : pass
  #   tgt:           saga.Url
```

```
  #   flags:         enum flags
  #   ttype:         saga.task.type enum
  #   ret:           None / saga.Task

  def link_self      (self, tgt,         flags=None, ttype=None)       : pass
  #   tgt:           saga.Url
  #   flags:         enum flags
  #   ttype:         saga.task.type enum
  #   ret:           None / saga.Task

  def move_self      (self, tgt,         flags=None, ttype=None)       : pass
  #   tgt:           saga.Url
  #   flags:         flags enum
  #   ttype:         saga.task.type enum
  #   ret:           None / saga.Task

  def remove_self    (self,              flags=None, ttype=None)       : pass
  #   flags:         flags enum
  #   ttype:         saga.task.type enum
  #   ret:           None / saga.Task

  def close          (self, timeout=None,            ttype=None)       : pass
  #   timeout:       float
  #   ttype:         saga.task.type enum
  #   ret:           None / saga.Task

  def permissions_allow_self (self, id, perms, flags=None, ttype=None) : pass
  #   id:            string
  #   perms:         saga.permissions.flags enum
  #   flags:         flags enum
  #   ttype:         saga.task.type enum
  #   ret:           None / saga.Task

  def permissions_deny_self  (self, id, perms, flags=None, ttype=None) : pass
  #   id:            string
  #   perms:         saga.permissions.flags enum
  #   flags:         flags enum
  #   ttype:         saga.task.type enum
  #   ret:           None / saga.Task


  url  = property (get_url)   # saga.Url
  cwd  = property (get_cwd)   # string
  name = property (get_name)  # string



class Directory (Entry, task.Async) :
  def change_dir     (self, url,                   ttype=None)        : pass
  #   url:           saga.Url
  #   ttype:         saga.task.type enum
  #   ret:           None / saga.Task

  def list           (self, npat=".", flags=None,   ttype=None)       : pass
  #   npat:          string
  #   flags:         flags enum
  #   ttype:         saga.task.type enum
```

```
#   ret:           list [saga.Url] / saga.Task

def find           (self, npat, flags=RECURSIVE,  ttype=None)        : pass
#   npat:          string
#   flags:         flags enum
#   ttype:         saga.task.type enum
#   ret:           list [saga.Url] / saga.Task

def exists         (self, name,                   ttype=None)        : pass
#   name:          saga.Url
#   ttype:         saga.task.type enum
#   ret:           bool / saga.Task

def is_dir         (self, name,                   ttype=None)        : pass
#   name:          saga.Url
#   ttype:         saga.task.type enum
#   ret:           bool / saga.Task

def is_entry       (self, name,                   ttype=None)        : pass
#   name:          saga.Url
#   ttype:         saga.task.type enum
#   ret:           bool / saga.Task

def is_link        (self, name,                   ttype=None)        : pass
#   name:          saga.Url
#   ttype:         saga.task.type enum
#   ret:           bool / saga.Task

def read_link      (self, name,                   ttype=None)        : pass
#   name:          saga.Url
#   ttype:         saga.task.type enum
#   ret:           saga.Url / saga.Task

def get_num_entries (self,                         ttype=None)       : pass
#   ttype:         saga.task.type enum
#   ret:           int / saga.Task

def get_entry      (self, num,                    ttype=None)        : pass
#   num:           int
#   ttype:         saga.task.type enum
#   ret:           saga.Url / saga.Task

def copy           (self, src, tgt, flags=None,   ttype=None)        : pass
#   src:           saga.Url
#   tgt:           saga.Url
#   flags:         flags enum
#   ttype:         saga.task.type enum
#   ret:           None / saga.Task

def link           (self, src, tgt, flags=None,   ttype=None)        : pass
#   src:           saga.Url
#   tgt:           saga.Url
#   flags:         flags enum
#   ttype:         saga.task.type enum
#   ret:           None / saga.Task

def move           (self, src, tgt, flags=None,   ttype=None)        : pass
```

```
#   src:           saga.Url
#   tgt:           saga.Url
#   flags:         flags enum
#   ttype:         saga.task.type enum
#   ret:           None / saga.Task

def remove         (self, tgt,      flags=None,   ttype=None)        : pass
#   tgt:           saga.Url
#   flags:         flags enum
#   ttype:         saga.task.type enum
#   ret:           None / saga.Task

def make_dir       (self, tgt,      flags=None,   ttype=None)        : pass
#   tgt:           saga.Url
#   flags:         flags enum
#   ttype:         saga.task.type enum
#   ret:           None / saga.Task

def open           (self, name,     flags=None,   ttype=None)        : pass
#   name:          saga.Url
#   flags:         flags enum
#   ttype:         saga.task.type enum
#   ret:           Entry / saga.Task

def open_dir       (self, name,     flags=None,   ttype=None)        : pass
#   name:          saga.Url
#   flags:         flags enum
#   ttype:         saga.task.type enum
#   ret:           Direct. / saga.Task

def permissions_deny  (self, tgt, id, perms, flags=None, ttype=None) : pass
#   tgt:           saga.Url
#   id:            string
#   perms:         saga.permission.flags enum
#   flags:         flags enum
#   ttype:         saga.task.type enum
#   ret:           None / saga.Task

def permissions_allow (self, tgt, id, perms, flags=None, ttype=None) : pass
#   tgt:           saga.Url
#   id:            string
#   perms:         saga.permissions.flags enum
#   flags:         flags enum
#   ttype:         saga.task.type enum
#   ret:           None / saga.Task


num_entries = property (get_num_entries)  # int
```

```
# 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
  #   ttype:    saga.task.type enum
  #   ret:      bool / saga.Task

  def get_size_self (self,                ttype=None) : pass
  #   ttype:    saga.task.type enum
  #   ret:      int / saga.Task

  def read      (self,       size=-1,     ttype=None) : pass
  #   size :    int
  #   ttype:    saga.task.type enum
  #   ret:      string / bytearray / saga.Task

  def write     (self, data,              ttype=None) : pass
  #   data :    string / bytearray
  #   ttype:    saga.task.type enum
  #   ret:      int / saga.Task

  def seek      (self, off, whence=START, ttype=None) : pass
  #   off :     int
  #   whence:   seek_mode enum
  #   ttype:    saga.task.type enum
  #   ret:      int / saga.Task

  def read_v    (self, iovecs             ttype=None) : pass
  #   iovecs:   list [tuple (int, int)]
  #   ttype:    saga.task.type enum
  #   ret:      list [bytearray] / saga.Task

  def write_v   (self, data,              ttype=None) : pass
  #   data:     list [tuple (int, string / bytearray)]
  #   ttype:    saga.task.type enum
```

```
  #   ret:      list [int] / saga.Task

  def size_p    (self, pattern,           ttype=None) : pass
  #   pattern:  string
  #   ttype:    saga.task.type enum
  #   ret:      int / saga.Task

  def read_p    (self, pattern,           ttype=None) : pass
  #   pattern:  string
  #   ttype:    saga.task.type enum
  #   ret:      string / bytearray / saga.Task

  def write_p   (self, pattern, data,     ttype=None) : pass
  #   pattern:  string
  #   data:     string / bytearray
  #   ttype:    saga.task.type enum
  #   ret:      int / saga.Task

  def modes_e   (self,                    ttype=None) : pass
  #   ttype:    saga.task.type enum
  #   ret:      list [string] / saga.Task

  def size_e    (self, emode, spec,       ttype=None) : pass
  #   emode:    string
  #   spec:     string
  #   ttype:    saga.task.type enum
  #   ret:      int / saga.Task

  def read_e    (self, emode, spec,       ttype=None) : pass
  #   emode:    string
  #   spec:     string
  #   ttype:    saga.task.type enum
  #   ret:      bytearray / saga.Task

  def write_e   (self, emode, spec, data, ttype=None) : pass
  #   emode:    string
  #   spec:     string
  #   data:     string / bytearray
  #   ttype:    saga.task.type enum
  #   ret:      int / saga.Task

  size = property (get_size_self)  # int


class Directory (namespace.Directory, task.Async) :

  def get_size  (self, name, flags=None,  ttype=None) : pass
  #   name:     saga.Url
  #   flags:    saga.namespace.flags enum
  #   ttype:    saga.task.type enum
  #   ret:      int / saga.Task

  def is_file   (self, name,              ttype=None) : pass
  #   name:     saga.Url
  #   ttype:    saga.task.type enum
  #   ret:      bool / saga.Task
```

```
def open_dir  (self, name, flags=READ,  ttype=None) : pass
#   name:     saga.Url
#   flags:    saga.namespace.flags enum
#   ttype:    saga.task.type enum
#   ret:      saga.filesystem.Directory / saga.Task

def open      (self, name, flags=READ,  ttype=None) : pass
#   name:     saga.Url
#   flags:    saga.namespace.flags enum
#   ttype:    saga.task.type enum
#   ret:      saga.filesystem.File / saga.Task
```

```
# 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
  #   ttype:          saga.task.type enum
  #   ret:            bool / saga.Task

  def add_location    (self, name,             ttype=None) : pass
  #   name:           saga.Url
  #   ttype:          saga.task.type enum
  #   ret:            None / saga.Task

  def remove_location (self, name,             ttype=None) : pass
  #   name:           saga.Url
  #   ttype:          saga.task.type enum
  #   ret:            None / saga.Task

  def update_location (self, old, new,         ttype=None) : pass
  #   old:            saga.Url
  #   new:            saga.Url
  #   ttype:          saga.task.type enum
  #   ret:            None / saga.Task

  def list_locations  (self,                   ttype=None) : pass
  #   ttype:          saga.task.type enum
  #   ret:            list [saga.Url] / saga.Task

  def replicate       (self, name, flags=None, ttype=None) : pass
  #   name:           saga.Url
  #   flags:          flags enum
  #   ttype:          saga.task.type enum
  #   ret:            None / saga.Task


class LogicalDirectory (namespace.Directory,
                        attributes.Attributes, task.Async) :
  def is_file         (self, name,             ttype=None) : pass
  #   name:           saga.url
  #   ttype:          saga.task.type enum
  #   ret:            bool / saga.Task
```

```
def open_dir        (self, name, flags=READ, ttype=None) : pass
#   name:           saga.url
#   flags:          flags enum
#   ttype:          saga.task.type enum
#   ret:            Directory / saga.Task

def open            (self, name, flags=READ, ttype=None) : pass
#   name:           saga.Url
#   flags:          flags enum
#   ttype:          saga.task.type enum
#   ret:            LogicalFile / saga.Task

def find            (self, name_pattern, attr_pattern,
                     flags=RECURSIVE,       ttype=None) : pass
#   name_pattern:   string
#   attr_pattern:   string
#   flags:          flags enum
#   ttype:          saga.task.type enum
#   ret:            list [saga.Url] / saga.Task
```

```
# 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, session=None)              : pass
  #   name:       saga.Url
  #   session:    saga.Session
  #   ret:        None

  def create      (self, name=None, session=None,  ttype=None) : pass
  #   name:       saga.Url
  #   session:    saga.Session
  #   ttype:      saga.task.type enum
  #   ret:        saga.Task

  def get_url     (self,                      ttype=None)       : pass
  #   ttype:      saga.task.type enum
  #   ret:        saga.Url / saga.Task

  def serve       (self, timeout=-1,          ttype=None)       : pass
  #   timeout:    float
  #   ttype:      saga.task.type enum
  #   ret:        Stream / saga.Task

  def close       (self, timeout=None,        ttype=None)       : pass
  #   timeout:    float
```

```
  #   ttype:      saga.task.type enum
  #   ret:        None / saga.Task


  url = property (get_url)  # saga.Url



class Stream (attributes.Attributes, monitoring.Monitorable, task.Async) :

  def __init__    (self, name=None, session=None)              : pass
  #   name:       saga.Url
  #   session:    saga.Session
  #   ret:        None

  def create      (self, name=None, session=None,  ttype=None) : pass
  #   name:       saga.Url
  #   session:    saga.Session
  #   ttype:      saga.task.type enum
  #   ret:        saga.Task

  def get_url     (self,                     ttype=None)       : pass
  #   ttype:      saga.task.type enum
  #   ret:        saga.Url / saga.Task

  def get_context (self,                     ttype=None)       : pass
  #   ttype:      saga.task.type enum
  #   ret:        saga.Context / saga.Task

  def connect     (self,                     ttype=None)       : pass
  #   ttype:      saga.task.type enum
  #   ret:        None / saga.Task

  def wait        (self, what, timeout=-1,   ttype=None)       : pass
  #   what:       stream_activity enum
  #   timeout:    float
  #   ttype:      saga.task.type enum
  #   ret:        None / saga.Task

  def close       (self,       timeout=None, ttype=None)       : pass
  #   timeout:    float
  #   ttype:      saga.task.type enum
  #   ret:        None / saga.Task

  def read        (self,       size=-1,      ttype=None)       : pass
  #   size:       int
  #   ttype:      saga.task.type enum
  #   ret:        bytearray / saga.Task

  def write       (self, data, size=-1,      ttype=None)       : pass
  #   data:       string / bytearray
  #   size:       int
  #   ttype:      saga.task.type enum
  #   ret:        None / saga.Task


  url     = property (get_url)      # saga.Url
```

```
  context = property (get_context)  # saga.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
  #   data:          string / bytearray
  #   size:          int
  #   mode:          mode enum
  #   ret:           None

  def set_io_mode    (self, mode)                               : pass
  #   mode:          mode enum
  #   ret:           None

  def get_io_mode    (self)                                     : pass
  #   ret:           mode enum

  def set_size       (self, size)                               : pass
  #   size:          int
  #   ret:           None

  def get_size       (self)                                     : pass
  #   ret:           int

  def set_data       (self, data)                               : pass
  #   data:          string / bytearray
  #   ret:           None

  def get_data       (self)                                     : pass
  #   ret:           bytearray

  def close          (self)                                     : pass
  #   ret:           None


  io_mode = property (get_io_mode, set_io_mode) # io_mode enum
  size    = property (get_size, set_size)       # int
  data    = property (get_data, set_data)       # bytearray



class RPC (permissions.Permissions, task.Async) :
  def __init__       (self, funcname, session=None)             : pass
  #   funcname:      string
  #   session:       saga.Session
  #   ret:           None

  def create         (self, funcname, session=None, ttype=None) : pass
  #   funcname:      string
  #   session:       saga.Session
  #   ttype:         saga.task.type enum
```

```
#   ret:           saga.Task

def call           (self, parameters,       ttype=None)       : pass
#   parameters:    list [Parameter]
#   ttype:         saga.task.type enum
#   ret:           None / saga.Task

def close          (self, timeout=None,     ttype=None)       : pass
#   timeout:       float
#   ttype:         saga.task.type enum
#   ret:           mode enum / saga.Task
```

```
# 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"  # duplication from flag
DELETE         = "Delete"
TTL            = "TTL"

class Advert (namespace.Entry, attributes.Attributes, task.Async) :
  def set_ttl_self    (self, ttl,              ttype=None) : pass
  #   ttl :           int
  #   ttype:          saga.task.type enum
  #   ret:            None / saga.Task

  def get_ttl_self    (self,                   ttype=None) : pass
  #   ttype:          saga.task.type enum
  #   ret:            int / saga.Task

  def store_object    (self, object,           ttype=None) : pass
  #   object :        <object type>
  #   ttype:          saga.task.type enum
  #   ret:            None / saga.Task

  def retrieve_object (self,                   ttype=None) : pass
  #   ttype:          saga.task.type enum
  #   ret:            any / saga.Task

  def delete_object   (self,                   ttype=None) : pass
  #   ttype:          saga.task.type enum
  #   ret:            None / saga.Task


class AdvertDirectory (namespace.Directory, attributes.Attributes, task.Async) :
  def set_ttl_self    (self,      ttl,         ttype=None) : pass
```

```
#   ttl :           int
#   ttype:          saga.task.type enum
#   ret:            None / saga.Task

def get_ttl_self    (self,                  ttype=None) : pass
#   ttype:          saga.task.type enum
#   ret:            int / saga.Task

def set_ttl         (self, tgt, ttl,        ttype=None) : pass
#   tgt :           saga.Url
#   ttl :           int
#   ttype:          saga.task.type enum
#   ret:            None / saga.Task

def get_ttl         (self, tgt,             ttype=None) : pass
#   tgt :           saga.Url
#   ttype:          saga.task.type enum
#   ret:            int / saga.Task

def find            (self, name_pattern, attr_pattern, obj_type,
                     flags=RECURSIVE,       ttype=None) : pass
#   name_pattern:   string
#   attr_pattern:   string
#   obj_type:       string
#   flags:          flags enum
#   ret:            list [saga.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 __init__      (self, topology    = POINT_TO_POINT,
                           reliability = RELIABLE,
                           atomicity   = EXACTLY_ONCE,
                           ordering    = ORDERED,
                           correctness = VERIFIED
                           session     = None)              : pass
  #   topology:     topology    enum
  #   reliability:  reliability enum
  #   atomicity:    atomicity   enum
  #   ordering:     ordering    enum
  #   correctness:  correctness enum
  #   session:      saga.Session
  #   ret:          None

  def create        (self, topology    = POINT_TO_POINT,
                           reliability = RELIABLE,
                           atomicity   = EXACTLY_ONCE,
                           ordering    = ORDERED,
                           correctness = VERIFIED
                           session     = None,
                           ttype       = None)              : pass
  #   topology:     topology    enum
  #   reliability:  reliability enum
  #   atomicity:    atomicity   enum
  #   ordering:     ordering    enum
  #   correctness:  correctness enum
  #   session:      saga.Session
  #   ttype:        saga.task.type enum
  #   ret:          saga.Task

  def get_url       (self,                       ttype=None) : pass
  #   ttype:        saga.task.type enum
  #   ret:          saga.Url / saga.Task

  def get_receivers (self,                       ttype=None) : pass
  #   ttype:        saga.task.type enum
  #   ret:          list [saga.Url] / saga.Task

  def serve         (self, n=-1, timeout=-1,     ttype=None) : pass
  #   n:            int
  #   timeout:      float
  #   ttype:        saga.task.type enum
  #   ret:          None / saga.Task

  def serve_once    (self,       timeout=-1,     ttype=None) : pass
  #   timeout:      float
  #   ttype:        saga.task.type enum
  #   ret:          Endpoint / saga.Task

  def connect       (self, url=None, timeout=-1,ttype=None) : pass
  #   url:          saga.Url
  #   timeout:      float
  #   ttype:        saga.task.type enum
  #   ret:          None / saga.Task
```

```
  def close         (self, receiver=None,       ttype=None) : pass
  #   receiver:     saga.Url
  #   ttype:        saga.task.type enum
  #   ret:          None / saga.Task

  def send          (self, msg, receivers=None, ttype=None) : pass
  #   msg:          Message
  #   receivers:    list [saga.Url]
  #   ttype:        saga.task.type enum
  #   ret:          None / saga.Task

  def test          (self, sender=None, receiver=None,
                           timeout=-1,          ttype=None) : pass
  #   sender:       saga.Url
  #   receiver:     saga.Url
  #   timeout:      float
  #   ttype:        saga.task.type enum
  #   ret:          int / saga.Task

  def recv          (self, sender=None, receiver=None,
                           timeout=-1,          ttype=None) : pass
  #   sender:       saga.Url
  #   receiver:     saga.Url
  #   timeout:      float
  #   ttype:        saga.task.type enum
  #   ret:          Message / saga.Task

  url       = property (get_url)        # saga.Url
  receivers = property (get_receivers)  # list [saga.Url]


class Message (saga.Attributes) :

  def __init__      (self, data=None, size=-1)              : pass
  #   data:         string / bytearray
  #   size:         int
  #   ret:          None

  def get_sender    (self)                                  : pass
  #   ret:          saga.Url

  def set_id        (self, id)                              : pass
  #   ret:          None

  def get_id        (self)                                  : pass
  #   ret:          string

  def set_size      (self, size)                            : pass
  #   size:         int
  #   ret:          None

  def get_size      (self)                                  : pass
  #   ret:          int

  def set_data      (self, data)                            : pass
  #   data:         string / bytearray
```

```
#   ret:          None

def get_data      (self)                                  : pass
#   ret:          bytearray

def close         (self)                                  : pass
#   ret:          None


sender = property (get_sender)          # saga.Url
id     = property (get_id, set_id)      # string
size   = property (get_size, set_size)  # int
data   = property (get_data, set_data)  # bytearray
```

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

# ServiceDescription attributes
ATTRIBUTE           = "Url"
OBJECT              = "Type"
UID                 = "UID"
SITE                = "Site"
NAME                = "Name"
IMPLEMENTOR         = "Implementor"
RELATED_SERVICE_IDS = "RelatedServiceIDs" # differs from get_related_services()

class Discoverer (object) :

  def __init__       (self, url=None, session=None) : pass
  #   url:           saga.Url
  #   session:       saga.Session
  #   ret: None

  def list_services  (self,             service_filter=None,
                      data_filter=None, authz_filter=None) : pass
  #   service_filter: string
  #   data_filter:    string
  #   authz_filter:   string
  #   ret:            list [ServiceDescription]


class ServiceDescription (attributes.Attributes) :

  def get_url        (self)                         : pass
  #   ret:           saga.Url

  def get_data       (self)                         : pass
  #   ret:           ServiceData

  def get_related_services (self)                   : pass
  #   ret:           list [ServiceDescription]


  url              = property (get_url)              # saga.Url
  data             = property (get_data)             # ServiceData
  related_services = property (get_related_services) # list [ServiceDescription]



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

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

class EntityDataSet (object) :
  def __init__             (self, model, name, filter=None,
                            url=None, session=None)     : pass
  #   model:               string
  #   name:                string
  #   filter:              string
  #   url:                 saga.Url
  #   session:             saga.Session
  #   ret:                 None

  def get_data             (self)                       : pass
  #   ret:                 list [EntityData]

  def get_related_entities (self, name, filter=None)    : pass
  #   name:                string
  #   filter:              string
  #   ret:                 EntityDataSet

  def list_related_entity_names (self)                  : pass
  #   ret:                 list [string]


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

```
# Resource API Package: saga/resource/resource.py
# ================================================

# resource type enum
COMPUTE      =  "Compute"
NETWORK      =  "Network"
STORAGE      =  "Storage"
POOL         =  "Pool"

# resource state enum
UNKNOWN      =  0
PENDING      =  1
ACTIVE       =  2
DRAINING     =  4
RUNNING      =  7
CLOSED       =  8
EXPIRED      = 16
FAILED       = 32
FINAL        = 56

# resource description attributes
TYPE         = "Type"
TEMPLATE     = "Template"
DYNAMIC      = "Dynamic"
START        = "Start"
END          = "End"
DURATION     = "Duration"

# compute/network/storage resource description attributes
MACHINE_OS   = "MachineOS"
MACHINE_ARCH = "MachineArch"
HOSTNAMES    = "Hostnames"
MEMORY       = "Memory"
SIZE         = "Size"
SLOTS        = "Slots"
ACCESS       = "Access"



class Description        (attributes.Attributes)  : pass
class ComputeDescription (Description)            : pass
class NetworkDescription (Description)            : pass
class StorageDescription (Description)            : pass



class  Manager (saga.Async) :
  def __init__          (self, url, session=None) : pass
  #   url:              saga.Url
  #   session:          saga.Session
  #   ret:              None

  def create            (self, url, session=None, ttype=None) : pass
  #   url:              saga.Url
  #   session:          saga.Session
  #   ttype:            saga.task.type enum
```

```
#   ret:              saga.Task

def get_resources    (self, type=Any, ttype=None)          : pass
#   type:             type enum
#   ttype:            saga.task.type enum
#   ret:              list [Resource] / saga.Task

def list_resources    (self, type=Any, ttype=None)          : pass
#   type:             type enum
#   ttype:            saga.task.type enum
#   ret:              list [string] / saga.Task

def get_resource_description (self, id, ttype=None)         : pass
#   id:               string
#   ttype:            saga.task.type enum
#   ret:              Description / saga.Task

def list_templates    (self, type=Any, ttype=None)          : pass
#   type:             type enum
#   ttype:            saga.task.type enum
#   ret:              list [string] / saga.Task

def get_template     (self, tmpl, ttype=None)              : pass
#   tmpl:             string
#   ttype:            saga.task.type enum
#   ret:              Description / saga.Task


def aquire_compute    (self, cd, ttype=None)                : pass
#   cd:               ComputeDescription / string / saga.job.Service
#   ttype:            saga.task.type enum
#   ret:              Compute / saga.Task

def release_compute   (self, id, ttype=None)                : pass
#   id:               string
#   ttype:            saga.task.type enum
#   ret:              None / saga.Task


def aquire_storage    (self, sd, ttype=None)                : pass
#   sd:               StorageDescription / string / saga.filesystem.Directory
#   ttype:            saga.task.type enum
#   ret:              Storage / saga.Task

def release_storage   (self, id, ttype=None)                : pass
#   id:               string
#   ttype:            saga.task.type enum
#   ret:              None / saga.Task


def aquire_network    (self, nd, ttype=None)                : pass
#   nd:               NetworkDescription / Network ID
#   ttype:            saga.task.type enum
#   ret:              Network / saga.Task

def release_network   (self, id, ttype=None)                : pass
#   id:               string
```

```
  #   ttype:            saga.task.type enum
  #   ret:              None / saga.Task


  templates = property (list_templates, get_template)  # dict {string : Description}
  resources = property (get_resources)                 # list [Resource]
  # FIXME: reconsider performance of keeping list of resource instances



class Resource (saga.Monitorable, saga.task.Async) :

  def __init__          (self, id)                : pass
  #   id:               string
  #   ret:              None

  def create            (self, id, ttype=None)    : pass
  #   id:               string
  #   ttype:            saga.task.type enum
  #   ret:              saga.Task

  def get_id            (self, ttype=None)                    : pass
  #   ttype:            saga.task.type enum
  #   ret:              string  / saga.Task

  def get_type          (self, ttype=None)                    : pass
  #   ttype:            saga.task.type enum
  #   ret:              type enum / saga.Task

  def get_state         (self, ttype=None)                    : pass
  #   ttype:            saga.task.type enum
  #   ret:              state enum / saga.Task

  def get_state_detail  (self, ttype=None)                    : pass
  #   ttype:            saga.task.type enum
  #   ret:              string  / saga.Task

  def get_manager       (self, ttype=None)                    : pass
  #   ttype:            saga.task.type enum
  #   ret:              Manager      / saga.Task

  def get_description   (self, ttype=None)                    : pass
  #   ttype:            saga.task.type enum
  #   ret:              Description  / saga.Task

  def reconfig          (self, rd, ttype=None)                : pass
  #   rd:               Description
  #   ttype:            saga.task.type enum
  #   ret:              None / saga.Task

  def release           (self, drain=False, ttype=None)       : pass
  #   drain:            bool
  #   ttype:            saga.task.type enum
  #   ret:              None / saga.Task

  def wait              (self, timeout=-1,
                               state=Final, ttype=None)       : pass
```

```
  #   timeout:          float
  #   state:            state enum
  #   ttype:            saga.task.type enum
  #   ret:              None / saga.Task

  id           = property (get_id)             # string
  type         = property (get_type)           # type enum
  state        = property (get_state)          # state enum
  state_detail = property (get_state_detail)   # string
  manager      = property (get_manager)        # Manager
  description  = property (get_description)    # Description



class Compute (Resource, saga.job.Service) :
  def submit            (selfjsdl, ttype=None)               : pass
  #   jsdl:             string
  #   ttype:            saga.task.type enum
  #   ret :             saga.job.Job / saga.Task

  def get_jobs          (self, ttype=None)                   : pass
  #   ttype:            saga.task.type enum
  #   ret:              saga.task.Container / saga.Task

  jobs = property (get_jobs)  # saga.TaskContainer
  # FIXME: reconsider performance of keeping list of job instances



class Storage (Resource, saga.filesystem.Directory) :
  pass


class Network (Resource) :
  pass


class Pool (Resource) :

  def close             (self, ttype=None)                   : pass
  #   ttype:            saga.task.type enum
  #   ret:              None / saga.Task

  def add               (self, res, ttype=None)              : pass
  #   res:              resource
  #   ttype:            saga.task.type enum
  #   ret:              None / saga.Task

  def add               (self, id, ttype=None)               : pass
  #   id:               string
  #   ttype:            saga.task.type enum
  #   ret:              None / saga.Task

  def remove            (self, id, ttype=None)               : pass
  #   id:               resource
  #   ttype:            saga.task.type enum
  #   ret:              None / saga.Task
```

```
def remove            (self, id, ttype=None)               : pass
#   id:               string
#   ttype:            saga.task.type enum
#   ret:              None / saga.Task

def list              (self, type=Any, ttype=None)         : pass
#   type              type enum
#   ttype:            saga.task.type enum
#   ret:              list [string] / saga.Task

def get               (self, type=Any, ttype=None)         : pass
#   type              type enum
#   ttype:            saga.task.type enum
#   ret:              list [resource] / saga.Task

def set_policy        (self, policy=None, ttype=None)      : pass
#   policy:           string
#   ttype:            saga.task.type enum
#   ret:              None / saga.Task

def get_policy        (self, ttype=None)                   : pass
#   ttype:            saga.task.type enum
#   ret:              string / saga.Task

resource_ids = property (list, add, remove)       # list [string]
resources    = property (get, add, remove)        # list [Resource]
policy       = property (set_policy, get_policy)  # string
# FIXME: reconsider performance of keeping list of resource instances
```

B Comparison to PySAGA and SAGA-Python

The Python bindings as defined in this document deviate in several places from the existing Python implementations of SAGA, namely PySAGA and SAGA-Python (old and new implementation) – after all, the explicit purpose of this document is to reconcile the various exising SAGA Python APIs. PySAGA is the implementation closest to the binding defined here, and its binding is well documented [**?**] and motivated [**?**]. This appendix thus lists the API differences to PySAGA, but not to the other Python implementations.

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 a dictionary member to the property interface: `Attributes.attributes['key']` → `Attributes.key`;
- removes the `Buffer` and `StdIO` classes, replacing them with Python `strings`;
- allows implementations to flatten interfaces into inheriting classes;
- defines enum constants on module level, in `UPPER_CASE` notation;
- removes naming redundancies, e.g. `job.JobDescription` → `job.Description`.