
COPASI API

COPASI Team

Jun 10, 2022

CONTENTS:

1	Quickstart	3
1.1	Building the library	3
1.2	Building the documentation	3
1.3	Running the tests	4
2	Examples	7
2.1	Loading a COPASI file	7
2.2	Saving a COPASI file	7
2.3	Using Visitors	7
2.4	Performing Transactions	7
3	API Documentation	9
4	Indices and tables	79
	Index	81



QUICKSTART

1.1 Building the library

1.2 Building the documentation

The documentation is automatically built on readthedocs with every commit. However, you can still generate the documentation locally along your normal build (see *Building the library*). For that you will need the following requirements installed:

- doxygen <<https://www.doxygen.nl>>
- python3

Next you need the following python packages `breathe` and `sphinx_rtd_theme`. So we start by creating a virtual environment, activating it and installing the packages into it.

```
~ > python3 -m venv venv
~ > . ./venv/bin/activate
(venv) ~ > pip install sphinx_rtd_theme breathe
....
Successfully installed Jinja2-2.11.3 MarkupSafe-1.1.1 Pygments-2.8.1
alabaster-0.7.12 babel-2.9.0 breathe-4.29.0 certifi-2020.12.5 chardet-4.0.0
docutils-0.16 idna-2.10 imagesize-1.2.0 packaging-20.9 pyparsing-2.4.7
pytz-2021.1 requests-2.25.1 six-1.15.0 snowballstemmer-2.1.0 sphinx-3.5.4
sphinx-rtd-theme-0.5.2 sphinxcontrib-applehelp-1.0.2 sphinxcontrib-devhelp-1.0.2
sphinxcontrib-htmlhelp-1.0.3 sphinxcontrib-jsmath-1.0.1 sphinxcontrib-qthelp-1.0.3
sphinxcontrib-serializinghtml-1.1.4 urllib3-1.26.4
(venv) ~ >
```

Since the documentation is not generated by default, you have to reconfigure your cmake project for the COPASI API next. So change into your build folder from before, and reconfigure with the option `-DWITH_DOCUMENTATION=ON`.

```
(venv) ~ > cd copasi-api/build
(venv) build > cmake -DWITH_DOCUMENTATION=ON ..

-- COPASI dependencies: /tmp/copasi-api/dependencies/lib/cmake
...

Options:

  Enable namespace = OFF
```

(continues on next page)

(continued from previous page)

```
Generate documentation = ON
Additional Defines      =

-- Configuring done
-- Generating done
-- Build files have been written to: /tmp/copasi-api/build
(venv) build >
```

Errors would have shown if doxygen or sphinx could not be found in the process. Now you are ready to build the documentation with:

```
(venv) build > make Sphinx
[ 50%] Generating documentation with Sphinx
Running Sphinx v3.5.4
loading pickled environment... done
building [mo]: targets for 0 po files that are out of date
building [html]: targets for 3 source files that are out of date
updating environment: 0 added, 3 changed, 0 removed
reading sources... [100%] quickstart/get-started
looking for now-outdated files... none found
pickling environment... done
checking consistency... done
preparing documents... done
writing output... [100%] quickstart/get-started
generating indices... genindex done
writing additional pages... search done
copying images... [100%] _static/COPASI_Conly_176x176.png
copying static files... done
copying extra files... done
dumping search index in English (code: en)... done
dumping object inventory... done
build succeeded.

The HTML pages are in sphinx.
[100%] Built target Sphinx

(venv) build >
```

And at this point you have the HTML pages generated in `./docs/sphinx/` with the main document being `./docs/sphinx/index.html`

1.3 Running the tests

We use the testing framework catch2 <<https://github.com/catchorg/Catch2>> and integrated it with the cmake build, so after building the library you can run the tests using ctest:

```
(venv) build > ctest -V
```

If you want to run tests on another build configuration, you can specify those using the `-C` option. So for example for the debug build:


```
(venv) build > ctest -C Debug -V
```

You can also run the test binary directly, but in that case test files provided in `./tests/test-data` will not be automatically found, as the source dir is not known.

```
(venv) build > ./tests/test_api
~~~~~
test_api.exe is a Catch v1.5.6 host application.
Run with -? for options

-----
load copasi file and access via regular COPASI api
-----
/copasi-api/tests/TestCore.cpp(32)
.....

/copasi-api/tests/TestCore.cpp(38): FAILED:
  REQUIRE( dm->loadModel(fileName, 0) == true )
with expansion:
    false == true

=====
test cases: 2 | 1 passed | 1 failed
assertions: 22 | 21 passed | 1 failed
```

In that case you can specify an environment variable `srcdir` pointing to it:

```
(venv) build > srcdir=/copasi-api/tests ./tests/test_api
=====
All tests passed (24 assertions in 2 test cases)
```

Additional options of the test runner:

```
(venv) build > ./tests/test_api -?
Catch v1.5.6
usage:
  test_api [<test name, pattern or tags> ...] [options]

where options are:
  -?, -h, --help           display usage information
  -l, --list-tests         list all/matching test cases
  -t, --list-tags         list all/matching tags
  -s, --success            include successful tests in output
  -b, --break              break into debugger on failure
  -e, --nothrow            skip exception tests
  -i, --invisibles         show invisibles (tabs, newlines)
  -o, --out <filename>    output filename
  -r, --reporter <name>   reporter to use (defaults to console)
  -n, --name <name>       suite name
  -a, --abort              abort at first failure
  -x, --abortx <no. failures> abort after x failures
  -w, --warn <warning name> enable warnings
  -d, --durations <yes|no> show test durations
```

(continues on next page)

(continued from previous page)

<code>-f, --input-file <filename></code>	load test names to run from a file
<code>-, --filenames-as-tags</code>	adds a tag for the filename
<code>--list-test-names-only</code>	list all/matching test cases names only
<code>--list-reporters</code>	list all reporters
<code>--order <decl lex rand></code>	test case order (defaults to decl)
<code>--rng-seed <'time' number></code>	set a specific seed for random numbers
<code>--force-colour</code>	force colourised output (deprecated)
<code>--use-colour <yes no></code>	should output be colourised

EXAMPLES

2.1 Loading a COPASI file

2.2 Saving a COPASI file

2.3 Using Visitors

2.4 Performing Transactions

API DOCUMENTATION

Here the full API documentation

```
class cpsapi : public cpsapiContainer
```

Public Types

```
typedef cpsapi self
```

The class

```
typedef cpsapiContainer base
```

The base class

Public Functions

```
cpsapi() = delete
```

```
cpsapi(const cpsapi &src) = delete
```

```
virtual ~cpsapi()
```

Public Static Functions

```
static void init()
```

```
static void release()
```

```
static cpsapiDataModel &addDataModel(const std::string &name = "")
```

```
static bool deleteDataModel(const std::string &name = "")
```

```
static cpsapiDataModel &dataModel(const std::string &name = "")
```

```
static cpsapiVector<cpsapiDataModel> getDataModels()
```

```
static const std::set<std::string> listModelNames()
```

```
static bool loadFromFile(const std::string &fileName, const std::string &modelName = "")
```

```
static bool loadFromString(const std::string &content, const std::string &referenceDir = "", const std::string
                           &modelName = "")

static cpsapiModel &model(const std::string &name = "")

static void beginTransaction(const std::string &name = "")

static void endTransaction(const std::string &name = "")

static bool addCompartment(const std::string &name, const std::string &modelName = "")

static bool deleteCompartment(const std::string &name, const std::string &modelName = "")

static cpsapiCompartment compartment(const std::string &name = "", const std::string &modelName = "")

static cpsapiVector<cpsapiCompartment> getCompartments(const std::string &modelName = "")

static bool addSpecies(const std::string &name, const std::string &compartmentName = "", const std::string
                           &modelName = "")

static bool deleteSpecies(const std::string &name, const std::string &compartmentName = "", const
                           std::string &modelName = "")

static cpsapiSpecies species(const std::string &name, const std::string &compartmentName = "", const
                              std::string &modelName = "")

static cpsapiVector<cpsapiSpecies> getSpecies(const std::string &modelName = "")

static bool addGlobalQuantity(const std::string &name, const std::string &modelName = "")

static bool deleteGlobalQuantity(const std::string &name, const std::string &modelName = "")

static cpsapiGlobalQuantity globalQuantity(const std::string &name = "", const std::string &modelName =
                                             "")

static cpsapiVector<cpsapiGlobalQuantity> getGlobalQuantities(const std::string &modelName = "")

static bool addReaction(const std::string &name, const std::string &modelName = "")

static bool deleteReaction(const std::string &name, const std::string &modelName = "")

static cpsapiReaction reaction(const std::string &name = "", const std::string &modelName = "")

static cpsapiVector<cpsapiReaction> getReactions(const std::string &modelName = "")

static bool addEvent(const std::string &name, const std::string &modelName = "")

static bool deleteEvent(const std::string &name, const std::string &modelName = "")

static cpsapiEvent event(const std::string &name = "", const std::string &modelName = "")

static cpsapiVector<cpsapiEvent> getEvents(const std::string &modelName = "")
```

Private Static Attributes

static std::map<std::string, CDataModel*> **DataModels**

static *cpsapiDataModel* **DefaultDataModel**

static CFunction ***pDefaultFunction**

static CUnitDefinition ***pDefaultUnitDefinition**

class **cpsapiCompartment** : public *cpsapiModelEntity*

Public Types

enum **Property**

Enumeration of the exposed properties

Values:

enumerator **DIMENSIONALITY**

enumerator **EXPRESSION**

enumerator **INITIAL_EXPRESSION**

enumerator **INITIAL_VALUE**

enumerator **SIMULATION_TYPE**

enumerator **ADD_NOISE**

enumerator **NOISE_EXPRESSION**

enumerator **NAME**

enumerator **OBJECT_UNIQUE_NAME**

enumerator **CN**

enumerator **UNIT**

typedef *cpsapiCompartment* **self**

The class

typedef *cpsapiModelEntity* **base**

The base class

typedef CCompartment **wrapped**

The wrapped COPASI class

Public Functions

cpsapiCompartment(*wrapped* *pWrapped = nullptr)

Specific constructor

Parameters **wrapped** – * pWrapped

virtual ~**cpsapiCompartment**()

Destroy the cpsapi Compartment object.

template<typename **Visitor**>

void **accept**(*Visitor* &visitor)

Accept the given visitor

Template Parameters **Visitor** –

Parameters **Visitor** – & visitor

cpsapiSpecies **addSpecies**(const std::string &name)

bool **deleteSpecies**(const std::string &name = "")

cpsapiSpecies **species**(const std::string &name = "")

cpsapiVector<*cpsapiSpecies*> **getSpecies**() const

bool **setProperty**(const *Property* &property, const *cpsapiData* &value, const CCore::Framework &framework = CCore::Framework::__SIZE)

cpsapiData **getProperty**(const *Property* &property, const CCore::Framework &framework = CCore::Framework::__SIZE) const

Public Static Attributes

static const Properties **SupportedProperties**

Static set of supported properties

Protected Functions

virtual bool **setProperty**(const *cpsapiProperty::Type* &property, const *cpsapiData* &value, const CCore::Framework &framework) override

Set the property

Parameters

- **const** – cpsapiProperty::Type & property
- **const** – *cpsapiData* & value

- **const** – CCore::Framework &framework

Returns bool success

virtual *cpsapiData* **getProperty**(const *cpsapiProperty::Type* &property, const CCore::Framework &framework) const override

Retrieve the property

Parameters

- **const** – cpsapiProperty::Type & property
- **const** – CCore::Framework &framework

Returns *cpsapiData* property

Private Functions

cpsapiSpecies **__species**(const std::string &name) const

void **updateDefaultSpecies**(const *cpsapiSpecies* &species)

class **cpsapiContainer** : public *cpsapiObject*

Subclassed by *cpsapi*, *cpsapiDataModel*, *cpsapiEvent*, *cpsapiEventAssignment*, *cpsapiModelEntity*, *cpsapiParameter*, *cpsapiReaction*, *cpsapiVector< Object >*

Public Types

typedef *cpsapiContainer* **self**

The class

typedef *cpsapiObject* **base**

The base class

typedef CDataContainer **wrapped**

The wrapped COPASI class

Public Functions

virtual **~cpsapiContainer**()

Destructor

template<typename **Visitor**>

void **accept**(*Visitor* &visitor)

Accept the given visitor

Template Parameters **Visitor** –

Parameters **Visitor** – & visitor

Protected Functions

cpsapiContainer(*wrapped* *pWrapped = nullptr, const *cpsapiObjectData::Type* &type = *cpsapiObjectData::Type::Container*)

Specific constructor

Parameters

- **wrapped** – * pWrapped (default: nullptr)
- **const** – Type & type (default: Type::Container)

class **cpsapiData** : protected std::variant<C_FLOAT64, C_INT32, C_UINT32, size_t, bool, std::string, CRegisteredCommonName, std::shared_ptr<*cpsapiObject*>, std::vector<*cpsapiData*>>

Public Types

enum **Type**

Values:

enumerator **Double**

enumerator **Int32**

enumerator **UnsignedInt32**

enumerator **SizeType**

enumerator **Bool**

enumerator **String**

enumerator **CommonName**

enumerator **Object**

enumerator **Vector**

enumerator **__SIZE**

typedef std::vector<*cpsapiData*> **Vector**

Public Functions

```
cpsapiData()  
cpsapiData(const C_FLOAT64 &value)  
cpsapiData(const C_INT32 &value)  
cpsapiData(const C_UINT32 &value)  
cpsapiData(const size_t &value)  
cpsapiData(const bool &value)  
cpsapiData(const std::string &value)  
cpsapiData(const char *value)  
cpsapiData(const CCommonName &value)  
cpsapiData(const Vector &value)  
template<class Object>  
cpsapiData(const Object &value)  
cpsapiData(const Type &type, void *pValue)  
cpsapiData(const cpsapiData &src)  
cpsapiData &operator=(const cpsapiData &rhs)  
~cpsapiData()  
C_FLOAT64 toDouble() const  
C_INT32 toInt32() const  
C_UINT32 toUnsignedInt32() const  
size_t toSizeType() const  
bool toBool() const  
std::string toString() const  
CCommonName toCommonName() const  
const Vector &toData() const  
template<class Object>  
Object toObject() const  
Type getType() const  
template<>  
inline cpsapiObject toObject() const
```

Private Types

```
typedef std::variant<C_FLOAT64, C_INT32, C_UINT32, size_t, bool, std::string, CRegisteredCommonName,  
std::shared_ptr<cpsapiObject>, std::vector<cpsapiData>> base
```

```
class cpsapiDataCollector : public COutputInterface
```

Public Functions

```
virtual ~cpsapiDataCollector()
```

Destructor

```
virtual bool compile(CObjectInterface::ContainerList listOfContainer) override  
compile the object list from name vector
```

Parameters **CObjectInterface::ContainerList** – listOfContainer

Returns bool success

```
virtual void output(const Activity &activity) override  
Perform an output event for the current activity
```

Parameters **const** – Activity & activity

```
virtual void separate(const Activity &activity) override  
Introduce an additional separator into the output
```

Parameters **const** – Activity & activity

```
virtual void finish() override  
Finish the output
```

```
void addDataReferenceBefore(const CCommonName &cn)
```

```
void addDataReferenceDuring(const CCommonName &cn)
```

```
void addDataRefenceAfter(const CCommonName &cn)
```

```
const std::vector<CRegisteredCommonName> &getDataReferencesBefore() const
```

```
const std::vector<CRegisteredCommonName> &getDataReferencesDuring() const
```

```
const std::vector<CRegisteredCommonName> &getDataReferencesAfter() const
```

```
void clearReferences()
```

```
const cpsapiData::Vector &getDataBefore() const
```

```
const cpsapiData::Vector &getDataDuring() const
```

```
const cpsapiData::Vector &getDataAfter() const
```

```
std::vector<cpsapiData::Type> getDataTypesBefore() const
```

```
std::vector<cpsapiData::Type> getDataTypesDuring() const
```

```
std::vector<cpsapiData::Type> getDataTypesAfter() const
```

```
void clearData()
```

Private Functions

```
bool generateValues(const CObjectInterface::ContainerList &listOfContainer, const
                    std::vector<CRegisteredCommonName> &cns,
                    std::vector<std::pair<cpsapiData::Type, void*>> &values)
```

Private Members

```
cpsapiData::Vector mDataBefore
```

```
cpsapiData::Vector mDataDuring
```

```
cpsapiData::Vector mDataAfter
```

```
std::vector<CRegisteredCommonName> mCnsBefore
```

```
std::vector<CRegisteredCommonName> mCnsDuring
```

```
std::vector<CRegisteredCommonName> mCnsAfter
```

```
std::vector<std::pair<cpsapiData::Type, void*>> mValuesBefore
```

```
std::vector<std::pair<cpsapiData::Type, void*>> mValuesDuring
```

```
std::vector<std::pair<cpsapiData::Type, void*>> mValuesAfter
```

Private Static Functions

```
static void collect(cpsapiData::Vector &data, const std::vector<std::pair<cpsapiData::Type, void*>>
                    &values)
```

```
class cpsapiDataModel : public cpsapiContainer
```

Public Types

```
enum Property
```

Enumeration of the exposed properties

Values:

```
typedef cpsapiDataModel self
```

Static set of supported properties The class

typedef *cpsapiContainer* **base**

The base class

typedef CDataModel **wrapped**

The wrapped COPASI class

Public Functions

cpsapiDataModel(*wrapped* *pDataModel = nullptr)

Specific constructor

Parameters *wrapped* – * pDataModel (default: nullptr)

virtual ~**cpsapiDataModel**()

Destructor

template<typename **Visitor**>

void **accept**(*Visitor* &visitor)

Accept the given visitor

Template Parameters **Visitor** –

Parameters **Visitor** – & visitor

bool **loadFromFile**(const std::string &fileName)

Load a model from a file (any supported format)

Parameters **const** – std::string & fileName

Returns bool success

bool **loadFromString**(const std::string &content, const std::string &referenceDir = "")

Load a model from a file (any supported format)

Parameters **const** – std::string & fileName

Returns bool success

cpsapiModel &**model**()

void **beginTransaction**()

void **endTransaction**()

cpsapiCompartment **addCompartment**(const std::string &name)

bool **deleteCompartment**(const std::string &name)

cpsapiCompartment **compartment**(const std::string &name = "")

cpsapiVector<*cpsapiCompartment*> **getCompartments**()

cpsapiSpecies **addSpecies**(const std::string &name, const std::string &compartmentName = "")

bool **deleteSpecies**(const std::string &name, const std::string &compartmentName = "")

cpsapiSpecies **species**(const std::string &name = "", const std::string &compartmentName = "")

```

cpsapiVector<cpsapiSpecies> getSpecies()

cpsapiReaction addReaction(const std::string &name)

bool deleteReaction(const std::string &name)

cpsapiReaction reaction(const std::string &name = "")

cpsapiVector<cpsapiReaction> getReactions()

cpsapiEvent addEvent(const std::string &name)

bool deleteEvent(const std::string &name)

cpsapiEvent event(const std::string &name = "")

cpsapiVector<cpsapiEvent> getEvents()

cpsapiGlobalQuantity addGlobalQuantity(const std::string &name)

bool deleteGlobalQuantity(const std::string &name)

cpsapiGlobalQuantity globalQuantity(const std::string &name = "")

cpsapiVector<cpsapiGlobalQuantity> getGlobalQuantities()

cpsapiVector<cpsapiTask> getTasks()

cpsapiTask task(const std::string &name = "")

cpsapiMethod method()

cpsapiProblem problem()

```

Private Functions

```

cpsapiTask __task(const std::string &name) const

```

```

class cpsapiEvent : public cpsapiContainer

```

Public Types

enum **Property**

Enumeration of the exposed properties

Values:

enumerator **DELAY_ASSIGNMENT**

enumerator **FIRE_AT_INITIALTIME**

enumerator **PERSISTENT_TRIGGER**

enumerator **TRIGGER_EXPRESSION**

enumerator **DELAY_EXPRESSION**

enumerator **PRIORITY_EXPRESSION**

enumerator **NAME**

enumerator **OBJECT_UNIQUE_NAME**

enumerator **CN**

enum **Reference**

Enumeration of the exposed references

Values:

enumerator **NAME**

enumerator **OBJECT_UNIQUE_NAME**

typedef *cpsapiEvent* **self**

The class

typedef *cpsapiContainer* **base**

The base class

typedef CEvent **wrapped**

The wrapped COPASI class

Public Functions

cpsapiEvent(*wrapped* *pWrapped = nullptr, const *cpsapiObjectData::Type* &type = *cpsapiObjectData::Type::Event*)

Specific constructor

Parameters

- **wrapped** – * pWrapped (default: nullptr)
- **const** – Type & type (default: Type::cpsapiEvent)

virtual ~**cpsapiEvent**()

Destructor

template<typename **Visitor**>

void **accept**(*Visitor* &visitor)

Accept the given visitor

Template Parameters *Visitor* –

Parameters *Visitor* – & visitor

cpsapiEventAssignment **addEventAssignment**(const std::string &name)

Add a an assignment to the event

Parameters **const** – std::string & name

Returns *cpsapiEventAssignment*

bool **deleteEventAssignment**(const std::string &name = "")

Delete the Assignment with the given name

Parameters **const** – std::string & name = (default: current assignment)

Returns bool success

cpsapiEventAssignment **eventAssignment**(const std::string &name = "")

Retrieve an assignment

Parameters **const** – std::string & name = (default: current assignment)

Returns *cpsapiEventAssignment*

cpsapiVector<*cpsapiEventAssignment*> **getEventAssignments**() const

Retrieve the vector af assignments

Returns *cpsapiVector*< *cpsapiEventAssignment* >

bool **setProperty**(const *Property* &property, const *cpsapiData* &value, const CCore::Framework &framework = CCore::Framework::__SIZE)

Set a property of the object to the provided value under the given framework. The value must match the underlying value of the property. The default framework is unspecified

Parameters

- **const** – Property & property
- **const** – *cpsapiData* & value
- **const** – CCore::Framework & framework (default: CCore::Framework::__SIZE)

Returns bool success

cpsapiData **getProperty**(const *Property* &property, const CCore::Framework &framework = CCore::Framework::__SIZE) const

Set a property of the object to the provided value under the given framework. The value must match the underlying value of the property. The default framework is unspecified

Parameters

- **const** – Property & property
- **const** – CCore::Framework & framework (default: CCore::Framework::__SIZE)

Returns *cpsapiData* property

CCommonName **getDataCN**(const *Reference* &reference, const CCore::Framework &framework = CCore::Framework::__SIZE) const

Retrieve a property of the object to the provided value under the given framework. The default framework is unspecified

Parameters

- **const** – Reference & reference
- **const** – CCore::Framework & framework (default: CCore::Framework::__SIZE)

Returns CCommonName

Public Static Attributes

static const Properties **SupportedProperties**

Static set of supported properties

static const Properties **HiddenProperties**

Static set of hidden properties

static const References **SupportedReferences**

Static set of supported references

static const References **HiddenReferences**

Static set of hidden references

Protected Functions

virtual bool **setProperty**(const *cpsapiProperty::Type* &property, const *cpsapiData* &value, const CCore::Framework &framework) override

Set the property

Parameters

- **const** – cpsapiProperty::Type & property
- **const** – *cpsapiData* & value
- **const** – CCore::Framework &framework

Returns bool success

virtual *cpsapiData* **getProperty**(const *cpsapiProperty::Type* &property, const CCore::Framework &framework) const override

Retrieve the property

Parameters

- **const** – cpsapiProperty::Type & property
- **const** – CCore::Framework &framework

Returns *cpsapiData* property

virtual CCommonName **getDataCN**(const *cpsapiReference*::Type &reference, const CCore::Framework &framework) const override

Retrieve the data reference

Parameters

- **const** – *cpsapiReference*::Type & reference
- **const** – CCore::Framework &framework

Returns CCommonName

Private Functions

cpsapiEventAssignment **__eventAssignment**(const std::string &name) const

Resolve and retrieve the event assignment by name

Parameters **const** – std::string & name

Returns *cpsapiEventAssignment*

void **updateDefaultEventAssignment**(const *cpsapiEventAssignment* &assignment)

Update the default event assignment

Parameters **assignment** –

class **cpsapiEventAssignment** : public *cpsapiContainer*

Public Types

enum **Property**

Enumeration of the exposed properties

Values:

enumerator **EXPRESSION**

enumerator **OBJECT_REFERENCE_CN**

enumerator **NAME**

enumerator **OBJECT_UNIQUE_NAME**

enumerator **CN**

enum **Reference**

Enumeration of the exposed references

Values:

enumerator **NAME**

enumerator **OBJECT_UNIQUE_NAME**

typedef *cpsapiEventAssignment* **self**

The class

typedef *cpsapiContainer* **base**

The base class

typedef CEventAssignment **wrapped**

The wrapped COPASI class

Public Functions

cpsapiEventAssignment(*wrapped* *pWrapped = nullptr, const *cpsapiObjectData::Type* &type = *cpsapiObjectData::Type::EventAssignment*)

Specific constructor

Parameters

- **wrapped** – * pWrapped (default: nullptr)
- **const** – Type & type (default: Type::cpsapiEventAssignment)

virtual ~**cpsapiEventAssignment**()

Destructor

template<typename **Visitor**>
void **accept**(*Visitor* &visitor)

Accept the given visitor

Template Parameters Visitor –

Parameters Visitor – & visitor

bool **setProperty**(const *Property* &property, const *cpsapiData* &value, const CCore::Framework &framework = CCore::Framework::__SIZE)

Set a property of the object to the provided value under the given framework. The value must match the underlying value of the property. The default framework is unspecified

Parameters

- **const** – Property & property
- **const** – *cpsapiData* & value
- **const** – CCore::Framework & framework (default: CCore::Framework::__SIZE)

Returns bool success

cpsapiData **getProperty**(const *Property* &property, const CCore::Framework &framework = CCore::Framework::__SIZE) const

Set a property of the object to the provided value under the given framework. The value must match the underlying value of the property. The default framework is unspecified

Parameters

- **const** – Property & property

- **const** – CCore::Framework & framework (default: CCore::Framework::__SIZE)

Returns *cpsapiData* property

CCommonName **getDataCN**(const *Reference* &reference, const CCore::Framework &framework = CCore::Framework::__SIZE) const

Retrieve a property of the object to the provided value under the given framework. The default framework is unspecified

Parameters

- **const** – Reference & reference
- **const** – CCore::Framework & framework (default: CCore::Framework::__SIZE)

Returns CCommonName

Public Static Attributes

static const Properties **SupportedProperties**

Static set of supported properties

static const Properties **HiddenProperties**

Static set of hidden properties

static const References **SupportedReferences**

Static set of supported references

static const References **HiddenReferences**

Static set of hidden references

Protected Functions

virtual bool **setProperty**(const *cpsapiProperty::Type* &property, const *cpsapiData* &value, const CCore::Framework &framework) override

Set the property

Parameters

- **const** – *cpsapiProperty::Type* & property
- **const** – *cpsapiData* & value
- **const** – CCore::Framework &framework

Returns bool success

virtual *cpsapiData* **getProperty**(const *cpsapiProperty::Type* &property, const CCore::Framework &framework) const override

Retrieve the property

Parameters

- **const** – *cpsapiProperty::Type* & property
- **const** – CCore::Framework &framework

Returns *cpsapiData* property

virtual CCommonName **getDataCN**(const *cpsapiReference*::Type &reference, const CCore::Framework &framework) const override

Retrieve the data reference

Parameters

- **const** – cpsapiReference::Type & reference
- **const** – CCore::Framework &framework

Returns CCommonName

class **cpsapiFactory**

Public Types

enum **PartType**

Values:

enumerator **vectorCompartment**

enumerator **vectorSpecies**

enumerator **vectorGlobalQuantity**

enumerator **vectorReaction**

enumerator **vectorReactionParameter**

enumerator **vectorEvent**

enumerator **vectorEventAssignment**

enumerator **vectorDataModel**

enumerator **vectorTask**

enumerator **value**

enumerator **model**

enumerator **compartment**

enumerator **species**

enumerator **globalQuantity**

enumerator **reaction**

enumerator **reactionParameter**

enumerator **event**

enumerator **eventAssignment**

enumerator **dataModel**

enumerator **parameter**

enumerator **group**

enumerator **method**

enumerator **problem**

enumerator **task**

enumerator **__SIZE**

typedef void (***free_unique_t**)(void*)

Public Functions

template<>

inline std::shared_ptr<*cpsapiObject*> **make_shared**(CDataObject *pFrom)

template<>

inline std::shared_ptr<*cpsapiObject*> **make_shared**(const *cpsapiObject* &from)

template<>

inline std::unique_ptr<*cpsapiObject*, *cpsapiFactory::free_unique_t*> **make_unique**(CDataObject *pFrom)

template<>

inline std::unique_ptr<*cpsapiObject*, *cpsapiFactory::free_unique_t*> **make_unique**(const *cpsapiObject* &from)

template<>

inline void **free_unique**(void*)

Public Static Functions

```
template<class Type>
static void free_unique(void*)

static void init()

static const PartInterface &info(const std::type_index &typeIndex)

static const PartInterface &info(CDataObject *pObject)

static const PartInterface &info(const cpsapiObject &object)

template<class CType>
static inline std::shared_ptr<CType> make_shared(CDataObject *pFrom)

template<class CType>
static inline std::shared_ptr<CType> make_shared(const cpsapiObject &from)

template<class CType>
static std::unique_ptr<CType, free_unique_t> make_unique(CDataObject *pFrom)

template<class CType>
static std::unique_ptr<CType, free_unique_t> make_unique(const cpsapiObject &from)

template<class Visitor>
static void accept(Visitor &visitor, CDataObject *pObject)

static cpsapiObject *copy(const cpsapiObject &object)

static cpsapiObject *create(CDataObject *pFrom)

static CDataValue::Type getDataType(const CObjectInterface *pObject)

template<class Function>
static void callDerived(CDataObject *pObject)
```

Private Types

```
typedef std::map<std::type_index, std::shared_ptr<PartInterface>> CopasiMap
```

Private Static Functions

```
template<class cpsapi, class copasi>
static void insert(const PartType &partType)

static cpsapiObject *make(CDataObject *pObject, const PartInterface *pInfo = nullptr)
```


Private Static Attributes

static *CopasiMap* **copasiMap**

class **cpsapiGlobalQuantity** : public *cpsapiModelEntity*

Public Types

enum **Property**

Enumeration of the exposed properties

Values:

enumerator **EXPRESSION**

enumerator **INITIAL_EXPRESSION**

enumerator **INITIAL_VALUE**

enumerator **SIMULATION_TYPE**

enumerator **ADD_NOISE**

enumerator **NOISE_EXPRESSION**

enumerator **NAME**

enumerator **OBJECT_UNIQUE_NAME**

enumerator **CN**

enumerator **UNIT**

typedef *cpsapiGlobalQuantity* **self**

The class

typedef *cpsapiModelEntity* **base**

The base class

typedef CModelValue **wrapped**

The wrapped COPASI class

Public Functions

cpsapiGlobalQuantity(*wrapped* *pWrapped = nullptr)

Specific constructor

Parameters *wrapped* – * pWrapped

virtual ~**cpsapiGlobalQuantity**()

template<typename **Visitor**>

void **accept**(*Visitor* &visitor)

Accept the given visitor

Template Parameters **Visitor** –

Parameters **Visitor** – & visitor

bool **setProperty**(const *Property* &property, const *cpsapiData* &value, const CCore::Framework &framework = CCore::Framework::__SIZE)

cpsapiData **getProperty**(const *Property* &property, const CCore::Framework &framework = CCore::Framework::__SIZE) const

Public Static Attributes

static const Properties **SupportedProperties**

Static set of supported properties

Protected Functions

virtual bool **setProperty**(const *cpsapiProperty::Type* &property, const *cpsapiData* &value, const CCore::Framework &framework) override

Set the property

Parameters

- **const** – cpsapiProperty::Type & property
- **const** – *cpsapiData* & value
- **const** – CCore::Framework &framework

Returns bool success

virtual *cpsapiData* **getProperty**(const *cpsapiProperty::Type* &property, const CCore::Framework &framework) const override

Retrieve the property

Parameters

- **const** – cpsapiProperty::Type & property
- **const** – CCore::Framework &framework

Returns *cpsapiData* property

class **cpsapiGroup** : public *cpsapiParameter*

Public Types

enum **Property**

Enumeration of the exposed properties

Values:

enumerator **PARAMETER_VALUE**

enumerator **NAME**

enumerator **OBJECT_UNIQUE_NAME**

enumerator **CN**

enum **Reference**

Enumeration of the exposed references

Values:

enumerator **NAME**

enumerator **OBJECT_UNIQUE_NAME**

typedef *cpsapiGroup* **self**

The class

typedef *cpsapiParameter* **base**

The base class

typedef CCopasiParameterGroup **wrapped**

The wrapped COPASI class

Public Functions

cpsapiGroup(*wrapped* *pGroup, const *cpsapiObjectData::Type* &type = *cpsapiObjectData::Type::Group*)

Specific constructor

Parameters **wrapped** – * pGroup

virtual **~cpsapiGroup**()

Destructor

template<typename **Visitor**>

void **accept**(*Visitor* &visitor)

Accept the given visitor

Template Parameters **Visitor** –

Parameters **Visitor** – & visitor

cpsapiParameter **addParameter**(const std::string &name, const CDataValue &value,
CCopasiParameter::Type type = CCopasiParameter::Type::__SIZE)

Add parameter

Parameters

- **const** – std::string & name
- **const** – CDataValue & value
- **CCopasiParameter::Type** – type (default: CCopasiParameter::Type::__SIZE)

Returns *cpsapiParameter* parameter

cpsapiGroup **addGroup**(const std::string &name)

Add a group

Parameters **const** – std::string & name

Returns *cpsapiGroup* group

bool **deleteParameter**(const std::string &name = "")

Delete a parameter

Parameters **const** – std::string & name (default: name of current parameter)

Returns bool success

cpsapiParameter **parameter**(const std::string &name = "")

Retrieve a parameter

Parameters **const** – std::string & name (default: name of current parameter)

Returns *cpsapiParameter* parameter

std::vector<*cpsapiParameter*> **getParameters**() const

Retrieve all parameters

Returns std::vector< cpsapiParameter > parameters

bool **setProperty**(const *Property* &property, const *cpsapiData* &value, const CCore::Framework
&framework = CCore::Framework::__SIZE)

Set a property of the object to the provided value under the given framework. The value must match the underlying value of the property. The default framework is unspecified

Parameters

- **const** – Property & property
- **const** – *cpsapiData* & value
- **const** – CCore::Framework & framework (default: CCore::Framework::__SIZE)

Returns bool success

cpsapiData **getProperty**(const *Property* &property, const CCore::Framework &framework =
CCore::Framework::__SIZE) const

Set a property of the object to the provided value under the given framework. The value must match the underlying value of the property. The default framework is unspecified

Parameters

- **const** – Property & property
- **const** – CCore::Framework & framework (default: CCore::Framework::__SIZE)

Returns *cpsapiData* property

CCommonName **getDataCN**(const *Reference* &reference, const CCore::Framework &framework = CCore::Framework::__SIZE) const

Retrieve a property of the object to the provided value under the given framework. The default framework is unspecified

Parameters

- **const** – Reference & reference
- **const** – CCore::Framework & framework (default: CCore::Framework::__SIZE)

Returns CCommonName

Public Static Attributes

static const Properties **SupportedProperties**

Static set of supported properties

static const Properties **HiddenProperties**

Static set of hidden properties

static const References **SupportedReferences**

Static set of supported references

static const References **HiddenReferences**

Static set of hidden references

Protected Functions

cpsapiParameter **__parameter**(const std::string &name) const

void **updateDefaultParameter**(const *cpsapiParameter* ¶meter)

virtual bool **setProperty**(const *cpsapiProperty::Type* &property, const *cpsapiData* &value, const CCore::Framework &framework) override

Set the property

Parameters

- **const** – *cpsapiProperty::Type* & property
- **const** – *cpsapiData* & value
- **const** – CCore::Framework &framework

Returns bool success

virtual *cpsapiData* **getProperty**(const *cpsapiProperty::Type* &property, const CCore::Framework &framework) const override

Retrieve the property

Parameters

- **const** – *cpsapiProperty::Type* & property

- **const** – CCore::Framework &framework

Returns *cpsapiData* property

virtual CCommonName **getDataCN**(const *cpsapiReference*::Type &reference, const CCore::Framework &framework) const override

Retrieve the data reference

Parameters

- **const** – cpsapiReference::Type & reference
- **const** – CCore::Framework &framework

Returns CCommonName

class **cpsapiKineticLawVariable** : public *cpsapiObject*

Public Types

enum **Property**

Enumeration of the exposed properties

Values:

enumerator **NAME**

enumerator **OBJECT_UNIQUE_NAME**

enumerator **CN**

enumerator **ROLE**

enumerator **VALUE**

enumerator **MAPPING**

enum **Reference**

Enumeration of the exposed references

Values:

enumerator **NAME**

enumerator **OBJECT_UNIQUE_NAME**

enumerator **VALUE**

typedef *cpsapiKineticLawVariable* **self**

The class

typedef *cpsapiObject* **base**

The base class

typedef *KineticLawVariable* **wrapped**

The wrapped COPASI class

Public Functions

cpsapiKineticLawVariable(*wrapped* *pWrapped = nullptr)

Specific constructor

Parameters *wrapped* – * pWrapped

virtual ~**cpsapiKineticLawVariable**()

Destructor

virtual bool **isValid**() const override

Check whether the object is valid

Returns bool valid

template<typename **Visitor**>

void **accept**(*Visitor* &visitor)

Accept the given visitor

Template Parameters **Visitor** –

Parameters **Visitor** – & visitor

bool **setProperty**(const *Property* &property, const *cpsapiData* &value, const CCore::Framework &framework = CCore::Framework::__SIZE)

Set a property of the object to the provided value under the given framework. The value must match the underlying value of the property. The default framework is unspecified

Parameters

- **const** – Property & property
- **const** – *cpsapiData* & value
- **const** – CCore::Framework & framework (default: CCore::Framework::__SIZE)

Returns bool success

cpsapiData **getProperty**(const *Property* &property, const CCore::Framework &framework = CCore::Framework::__SIZE) const

Set a property of the object to the provided value under the given framework. The value must match the underlying value of the property. The default framework is unspecified

Parameters

- **const** – Property & property
- **const** – CCore::Framework & framework (default: CCore::Framework::__SIZE)

Returns *cpsapiData* property

CCommonName **getDataCN**(const *Reference* &reference, const CCore::Framework &framework = CCore::Framework::__SIZE) const

Retrieve a property of the object to the provided value under the given framework. The default framework is unspecified

Parameters

- **const** – Reference & reference
- **const** – CCore::Framework & framework (default: CCore::Framework::__SIZE)

Returns CCommonName

Public Static Attributes

static const Properties **SupportedProperties**

Static set of supported properties

static const Properties **HiddenProperties**

Static set of hidden properties

static const References **SupportedReferences**

Static set of supported references

static const References **HiddenReferences**

Static set of hidden references

Protected Functions

virtual bool **setProperty**(const *cpsapiProperty::Type* &property, const *cpsapiData* &value, const CCore::Framework &framework) override

Set the property

Parameters

- **const** – cpsapiProperty::Type & property
- **const** – *cpsapiData* & value
- **const** – CCore::Framework &framework

Returns bool success

virtual *cpsapiData* **getProperty**(const *cpsapiProperty::Type* &property, const CCore::Framework &framework) const override

Retrieve the property

Parameters

- **const** – cpsapiProperty::Type & property
- **const** – CCore::Framework &framework

Returns cpsapiData

virtual CCommonName **getDataCN**(const *cpsapiReference*::Type &reference, const CCore::Framework &framework) const override

Retrieve the data reference

Parameters

- **const** – *cpsapiReference*::Type & reference
- **const** – CCore::Framework &framework

Returns CCommonName

class **cpsapiModel** : public *cpsapiModelEntity*

Public Types

enum **Property**

Enumeration of the exposed properties

Values:

enumerator **INITIAL_VALUE**

enumerator **NAME**

enumerator **OBJECT_UNIQUE_NAME**

enumerator **CN**

enumerator **UNIT**

enumerator **VOLUME_UNIT**

enumerator **AREA_UNIT**

enumerator **LENGTH_UNIT**

enumerator **TIME_UNIT**

enumerator **QUANTITY_UNIT**

enumerator **MODEL_TYPE**

enumerator **AVOGADRO_NUMBER**

typedef *cpsapiModel* **self**

The class

```
typedef cpsapiModelEntity base
```

The base class

```
typedef CModel wrapped
```

The wrapped COPASI class

Public Functions

```
cpsapiModel(wrapped *pWrapped = nullptr)
```

Specific constructor

Parameters *wrapped* – * pWrapped

```
virtual ~cpsapiModel()
```

```
template<typename Visitor>
```

```
void accept(Visitor &visitor)
```

Accept the given visitor

Template Parameters *Visitor* –

Parameters *Visitor* – & visitor

```
void beginTransaction() const
```

```
void endTransaction() const
```

```
bool synchronize(std::set<const CDataObject*> &changedObjects, const CCore::Framework &framework)
```

```
bool compile()
```

```
cpsapiCompartment addCompartment(const std::string &name)
```

```
bool deleteCompartment(const std::string &name = "")
```

```
cpsapiCompartment compartment(const std::string &name = "")
```

```
cpsapiVector<cpsapiCompartment> getCompartments() const
```

```
cpsapiSpecies addSpecies(const std::string &name, const std::string &compartment = "")
```

```
bool deleteSpecies(const std::string &name = "", const std::string &compartment = "")
```

```
cpsapiSpecies species(const std::string &name = "", const std::string &compartment = "")
```

```
cpsapiVector<cpsapiSpecies> getSpecies() const
```

```
cpsapiGlobalQuantity addGlobalQuantity(const std::string &name)
```

```
bool deleteGlobalQuantity(const std::string &name = "")
```

```
cpsapiGlobalQuantity globalQuantity(const std::string &name = "")
```

```
cpsapiVector<cpsapiGlobalQuantity> getGlobalQuantities() const
```

```
cpsapiReaction addReaction(const std::string &name)
```

```
bool deleteReaction(const std::string &name = "")
```

```

cpsapiReaction reaction(const std::string &name = "")
cpsapiVector<cpsapiReaction> getReactions() const
cpsapiEvent addEvent(const std::string &name)
bool deleteEvent(const std::string &name = "")
cpsapiEvent event(const std::string &name = "")
cpsapiVector<cpsapiEvent> getEvents() const
cpsapiEventAssignment addEventAssignment(const std::string &name, const std::string &event = "")
bool deleteEventAssignment(const std::string &name = "", const std::string &event = "")
cpsapiEventAssignment eventAssignment(const std::string &name = "", const std::string &event = "")
cpsapiVector<cpsapiEventAssignment> getEventAssignments(const std::string &event = "") const
void deleteAllDependents(CDataContainer *pContainer)
bool setProperty(const Property &property, const cpsapiData &value, const CCore::Framework
                &framework = CCore::Framework::__SIZE)
cpsapiData getProperty(const Property &property, const CCore::Framework &framework =
                CCore::Framework::__SIZE) const

```

Public Static Attributes

static const Properties **SupportedProperties**

Static set of supported properties

static const Properties **HiddenProperties**

Static set of hidden properties

Protected Functions

virtual bool **setProperty**(const *cpsapiProperty::Type* &property, const *cpsapiData* &value, const CCore::Framework &framework) override

Set the property

Parameters

- **const** – *cpsapiProperty::Type* & property
- **const** – *cpsapiData* & value
- **const** – CCore::Framework &framework

Returns bool success

virtual *cpsapiData* **getProperty**(const *cpsapiProperty::Type* &property, const CCore::Framework &framework) const override

Retrieve the property

Parameters

- **const** – *cpsapiProperty::Type* & property
- **const** – CCore::Framework &framework

Returns *cpsapiData* property

Private Functions

cpsapiCompartment **__compartment**(const std::string &name) const

void **updateDefaultCompartment**(const *cpsapiCompartment* &compartment)

cpsapiSpecies **__species**(const std::string &name, const std::string &compartment) const

cpsapiGlobalQuantity **__globalQuantity**(const std::string &name) const

void **updateDefaultGlobalQuantity**(const *cpsapiGlobalQuantity* &globalQuantity)

cpsapiReaction **__reaction**(const std::string &name) const

void **updateDefaultReaction**(const *cpsapiReaction* &reaction)

cpsapiEvent **__EVENT**(const std::string &name) const

Retrieve the event with the given name Upper case name due __event being a Visual C keyword

Parameters **const** – std::string & name

Returns *cpsapiEvent*

void **updateDefaultEvent**(const *cpsapiEvent* &event)

void **deleteDependents**(const CDataObject::DataObjectSet &set)

class **cpsapiModelEntity** : public *cpsapiContainer*

Subclassed by *cpsapiCompartment*, *cpsapiGlobalQuantity*, *cpsapiModel*, *cpsapiSpecies*

Public Types

enum **Property**

Values:

enumerator **EXPRESSION**

enumerator **INITIAL_EXPRESSION**

enumerator **INITIAL_VALUE**

enumerator **SIMULATION_TYPE**

enumerator **ADD_NOISE**

enumerator **NOISE_EXPRESSION**

enumerator **NAME**

enumerator **OBJECT_UNIQUE_NAME**

enumerator **CN**

typedef *cpsapiModelEntity* **self**

The class

typedef *cpsapiContainer* **base**

The base class

Public Functions

cpsapiModelEntity() = delete

virtual ~**cpsapiModelEntity**()

template<typename **Visitor**>

void **accept**(*Visitor* &visitor)

Accept the given visitor

Template Parameters **Visitor** –

Parameters **Visitor** – & visitor

bool **setProperty**(const *Property* &property, const *cpsapiData* &value, const CCore::Framework
&framework = CCore::Framework::__SIZE)

cpsapiData **getProperty**(const *Property* &property, const CCore::Framework &framework =
CCore::Framework::__SIZE) const

Public Static Attributes

static const Properties **SupportedProperties**

Protected Functions

cpsapiModelEntity(CModelEntity *pModelEntity, const *cpsapiObjectData::Type* &type)

Specific constructor

Parameters

- **CModelEntity** – * pContainer
- **const** – Type & type

virtual bool **setProperty**(const *cpsapiProperty::Type* &property, const *cpsapiData* &value, const CCore::Framework &framework) override

Set the property

Parameters

- **const** – cpsapiProperty::Type & property
- **const** – *cpsapiData* & value
- **const** – CCore::Framework &framework

Returns bool success

virtual *cpsapiData* **getProperty**(const *cpsapiProperty::Type* &property, const CCore::Framework &framework) const override

Retrieve the property

Parameters

- **const** – cpsapiProperty::Type & property
- **const** – CCore::Framework &framework

Returns *cpsapiData* property

class **cpsapiObject**

#include <cpsapiObject.h> The *cpsapiObject* class is the base class for all COPASI CDataObjects exposed in the cpsapi.

Subclassed by *cpsapiContainer*, *cpsapiKineticLawVariable*, *cpsapiValue*

Public Types

enum **Property**

Enumeration of the exposed properties

Values:

enumerator **NAME**

enumerator **OBJECT_UNIQUE_NAME**

enumerator **CN**

enum **Reference**

Enumeration of the exposed references

Values:

enumerator **NAME**

enumerator **OBJECT_UNIQUE_NAME**

typedef std::set<*cpsapiProperty::Type*> **Properties**

typedef std::set<*cpsapiReference::Type*> **References**

typedef *cpsapiObjectData* **Data**

typedef *cpsapiObject* **self**

The base class

Public Functions

cpsapiObject(const *cpsapiObject* &src)

Copy constructor

Parameters **const** – *cpsapiObject* & src

virtual ~**cpsapiObject**()

Destructor

cpsapiObject &**operator**=(const *cpsapiObject* &rhs)

Assignment operator =

Parameters **const** – *cpsapiObject* & rhs

Returns *cpsapiObject* &

cpsapiObject &**operator**=(CDataObject *pObject)

Assignment operator =

Parameters **CDataObject** – * pObject

Returns *cpsapiObject* &

CDataObject ***operator**->() const

Dereferencing operator ->

Returns CDataObject*

CDataObject ***operator***() const

Dereferencing operator *

Returns CDataObject*

bool **operator==(const *cpsapiObject* &rhs) const**

Comparison operator for equality

Parameters **const** – *cpsapiObject* & rhs

Returns bool equal

bool **operator!=(const *cpsapiObject* &rhs) const**

Comparison operator for inequality

Parameters **const** – *cpsapiObject* & rhs

Returns bool notEqual

cpsapiObjectData::Type **getType()** const

Retrieve the type of the object

Returns *cpsapiObjectData::Type* type

operator bool() const

Conversion to bool indicating whether the underlying COPASI CDataObject is accessible. The underlying CDataObject might have been deleted, e.g., a species if the parent compartment has been deleted

virtual bool **isValid()** const

Check whether the object is valid

Returns bool valid

template<typename **Visitor**>

void **accept**(*Visitor* &visitor)

Accept the given visitor

Template Parameters **Visitor** –

Parameters **Visitor** – & visitor

bool **setProperty**(const *Property* &property, const *cpsapiData* &value, const CCore::Framework
&framework = CCore::Framework::__SIZE)

Set a property of the object to the provided value under the given framework. The value must match the underlying value of the property. The default framework is unspecified

Parameters

- **const** – Property & property
- **const** – *cpsapiData* & value
- **const** – CCore::Framework & framework (default: CCore::Framework::__SIZE)

Returns bool success

bool **setProperty**(const std::string &property, const *cpsapiData* &value, const std::string &framework = "")

Set a property of the object to the provided value under the given framework. The property must be the string in *cpsapiProperty::Names* The value must match the underlying value of the property. The framework must be string in CCore::FrameworkNames

Parameters

- **const** – std::string & property
- **const** – *cpsapiData* & value
- **const** – std::string & framework (default: "")

Returns bool success

cpsapiData **getProperty**(const *Property* &property, const CCore::Framework &framework = CCore::Framework::__SIZE) const

Retrieve a property of the object to the provided value under the given framework. The default framework is unspecified

Parameters

- **const** – Property & property
- **const** – CCore::Framework & framework (default: CCore::Framework::__SIZE)

Returns *cpsapiData* property

cpsapiData **getProperty**(const std::string &property, const std::string &framework = "") const

Retrieve a property of the object to the provided value under the given framework. The property must be the string in cpsapiProperty::Names The framework must be string in CCore::FrameworkNames

Parameters

- **const** – std::string & property
- **const** – std::string & framework (default: “”)

Returns *cpsapiData* property

CCommonName **getDataCN**(const *Reference* &reference, const CCore::Framework &framework = CCore::Framework::__SIZE) const

Retrieve a data reference of the object under the given framework. The default framework is unspecified

Parameters

- **const** – Reference & reference
- **const** – CCore::Framework & framework (default: CCore::Framework::__SIZE)

Returns CCommonName

CCommonName **getDataCN**(const std::string &reference, const std::string &framework = "") const

Retrieve a property of the object the given framework. The property must be the string in cpsapiReferences::Names The framework must be string in CCore::FrameworkNames

Parameters

- **const** – std::string & reference
- **const** – std::string & framework (default: “”)

Returns CCommonName

```
template<class Visitor> CPSAPI_NAMESPACE_END CPSAPI_NAMESPACE_BEGIN void accept (Visitor &visitor)
```

```
template<>
```

```
inline cpsapiObject::Properties supportedProperties()
```

```
template<>
```

```
inline cpsapiObject::References supportedReferences()
```

Public Static Functions

template<class **CType**>

static *Properties* **supportedProperties()**

Retrieve the effective list of supported properties

Template Parameters **class** – CType *cpsapiObject* or a derived class

Returns Properties SupportedProperties

template<class **CType**>

static bool **isSupportedProperty**(const *cpsapiProperty::Type* &property)

Check whether the property is supported

Template Parameters **class** – CType *cpsapiObject* or a derived class

Parameters **const** – *cpsapiProperty::Type* & property

Returns bool supported

template<class **CType**>

static *References* **supportedReferences()**

Retrieve the effective list of supported references

Template Parameters **class** – CType *cpsapiObject* or a derived class

Returns References SupportedReferences

template<class **CType**>

static bool **isSupportedReference**(const *cpsapiReference::Type* &reference)

Check whether the reference is supported

Template Parameters **class** – CType *cpsapiObject* or a derived class

Parameters **const** – *cpsapiReference::Type* & reference

Returns bool supported

Public Static Attributes

static const CCommonName **Invalid**

static const *Properties* **SupportedProperties**

Static set of supported properties

static const *Properties* **HiddenProperties**

Static set of hidden properties

static const *References* **SupportedReferences**

Static set of supported references

static const *References* **HiddenReferences**

Static set of hidden references

Protected Functions

cpsapiObject(CDataObject *pObject = nullptr, const *cpsapiObjectData::Type* &type = *cpsapiObjectData::Type::Object*)

Specific constructor

Parameters

- **CDataObject** – * pObject (default: nullptr)
- **const** – Type & type (default: Type::Object)

CDataObject *get**Object**() const

Retrieve the pointer to the underlying COPASI CDataObject.

Returns CDataObject * pObject

virtual bool **setProperty**(const *cpsapiProperty::Type* &property, const *cpsapiData* &value, const CCore::Framework &framework)

Set the property

Parameters

- **const** – cpsapiProperty::Type & property
- **const** – *cpsapiData* & value
- **const** – CCore::Framework &framework

Returns bool success

virtual *cpsapiData* **getProperty**(const *cpsapiProperty::Type* &property, const CCore::Framework &framework) const

Retrieve the property

Parameters

- **const** – cpsapiProperty::Type & property
- **const** – CCore::Framework &framework

Returns *cpsapiData* property

virtual CCommonName **getDataCN**(const *cpsapiReference::Type* &reference, const CCore::Framework &framework) const

Retrieve the data reference

Parameters

- **const** – cpsapiReference::Type & reference
- **const** – CCore::Framework &framework

Returns CCommonName

Protected Attributes

Data::Pointer **mpData**

Protected Static Functions

```
template<class CType>
static bool isImplementedProperty(const cpsapiProperty::Type &property)
```

Check whether the given property is supported by the class

Template Parameters **class** – CType *cpsapiObject* or a derived class

Parameters **const** – *cpsapiProperty::Type* & property

Returns bool supported

```
template<class CType>
static bool isHiddenProperty(const cpsapiProperty::Type &property)
```

Check whether the given property is hidden by the class

Template Parameters **class** – CType *cpsapiObject* or a derived class

Parameters **const** – *cpsapiProperty::Type* & property

Returns bool supported

```
template<class CType>
static bool isImplementedReference(const cpsapiReference::Type &reference)
```

Check whether the given reference is supported by the class

Template Parameters **class** – CType *cpsapiObject* or a derived class

Parameters **const** – *cpsapiReference::Type* & reference

Returns bool supported

```
template<class CType>
static bool isHiddenReference(const cpsapiReference::Type &reference)
```

Check whether the given reference is hidden by the class

Template Parameters **class** – CType *cpsapiObject* or a derived class

Parameters **const** – *cpsapiReference::Type* & reference

Returns bool hidden

Friends

```
friend class cpsapiData
```

```
class cpsapiObjectData
```

#include <*cpsapiObjectData.h*> The *cpsapiObject* class is the base class for all COPASI CDataObjects exposed in the *cpsapi*.

Subclassed by *cpsapiCompartment::Data*, *cpsapiEvent::Data*, *cpsapiModel::Data*, *cpsapiReaction::Data*

Public Types

enum **Type**

Values:

enumerator **Object**

enumerator **Value**

enumerator **Container**

enumerator **Vector**

enumerator **ModelEntity**

enumerator **Model**

enumerator **Compartment**

enumerator **Species**

enumerator **GlobalQuantity**

enumerator **Reaction**

enumerator **ReactionParameter**

enumerator **Event**

enumerator **EventAssignment**

enumerator **DataModel**

enumerator **Parameter**

enumerator **Group**

enumerator **Method**

enumerator **Problem**

enumerator **Task**

enumerator **__SIZE**

typedef std::shared_ptr<*cpsapiObjectData*> **Pointer**

typedef std::map<const CDataObject*, *Pointer*> **Map**

Public Functions

cpsapiObjectData(const *cpsapiObjectData*&) = default

cpsapiObjectData(CDataObject *pObject, const *Type* &type)

inline virtual ~**cpsapiObjectData**()

template<>

inline void **assertDataType**(*Pointer* &data)

Public Static Functions

static void **release**()

static void **erase**(const CDataObject *pObject)

static void **deleted**(const CDataObject *pObject)

template<class **CType**>

static void **assertDataType**(*Pointer* &baseData)

Assert that mpData points of the proper class

Tparam

Public Static Attributes

static const CEnumAnnotation<std::string, *Type*> **TypeName**

static *Map* **Manager**

Protected Attributes

CDataObject ***mpObject** = nullptr

Type **mType** = *Type::Object*

Private Functions

cpsapiObjectData() = default

cpsapiObjectData(*cpsapiObjectData*&&) = default

Friends

friend class cpsapiObject

class **cpsapiParameter** : public *cpsapiContainer*

Subclassed by *cpsapiGroup*

Public Types

enum **Property**

Enumeration of the exposed properties

Values:

enumerator **PARAMETER_VALUE**

enumerator **NAME**

enumerator **OBJECT_UNIQUE_NAME**

enumerator **CN**

enum **Reference**

Enumeration of the exposed references

Values:

enumerator **PARAMETER_VALUE**

enumerator **NAME**

enumerator **OBJECT_UNIQUE_NAME**

typedef *cpsapiParameter* **self**

The class

typedef *cpsapiContainer* **base**

The base class

typedef CCopasiParameter **wrapped**

The wrapped COPASI class

Public Functions

cpsapiParameter(*wrapped* *pWrapped = nullptr, const *cpsapiObjectData::Type* &type = *cpsapiObjectData::Type::Parameter*)

Specific constructor

Parameters

- **wrapped** – * pWrapped (default: nullptr)
- **const** – Type & type (default: Type::cpsapiParameter)

virtual **~cpsapiParameter**()

Destructor

template<typename **Visitor**>
void **accept**(*Visitor* &visitor)

Accept the given visitor

Template Parameters **Visitor** –

Parameters Visitor – & visitor

bool **setProperty**(const *Property* &property, const *cpsapiData* &value, const CCore::Framework &framework = CCore::Framework::__SIZE)

Set a property of the object to the provided value under the given framework. The value must match the underlying value of the property. The default framework is unspecified

Parameters

- **const** – Property & property
- **const** – *cpsapiData* & value
- **const** – CCore::Framework & framework (default: CCore::Framework::__SIZE)

Returns bool success

cpsapiData **getProperty**(const *Property* &property, const CCore::Framework &framework = CCore::Framework::__SIZE) const

Set a property of the object to the provided value under the given framework. The value must match the underlying value of the property. The default framework is unspecified

Parameters

- **const** – Property & property
- **const** – CCore::Framework & framework (default: CCore::Framework::__SIZE)

Returns *cpsapiData* property

CCommonName **getDataCN**(const *Reference* &reference, const CCore::Framework &framework = CCore::Framework::__SIZE) const

Retrieve a property of the object to the provided value under the given framework. The default framework is unspecified

Parameters

- **const** – Reference & reference
- **const** – CCore::Framework & framework (default: CCore::Framework::__SIZE)

Returns CCommonName

Public Static Attributes

static const Properties **SupportedProperties**

Static set of supported properties

static const Properties **HiddenProperties**

Static set of hidden properties

static const References **SupportedReferences**

Static set of supported references

static const References **HiddenReferences**

Static set of hidden references

Protected Functions

virtual bool **setProperty**(const *cpsapiProperty::Type* &property, const *cpsapiData* &value, const CCore::Framework &framework) override

Set the property

Parameters

- **const** – *cpsapiProperty::Type* & property
- **const** – *cpsapiData* & value
- **const** – CCore::Framework &framework

Returns bool success

virtual *cpsapiData* **getProperty**(const *cpsapiProperty::Type* &property, const CCore::Framework &framework) const override

Retrieve the property

Parameters

- **const** – *cpsapiProperty::Type* & property
- **const** – CCore::Framework &framework

Returns *cpsapiData* property

virtual CCommonName **getDataCN**(const *cpsapiReference::Type* &reference, const CCore::Framework &framework) const override

Retrieve the data reference

Parameters

- **const** – *cpsapiReference::Type* & reference
- **const** – CCore::Framework &framework

Returns CCommonName

class **cpsapiProperty**

Public Types

enum **Type**

Values:

enumerator **EXPRESSION**

enumerator **VALUE**

enumerator **INTENSIVE_VALUE**

enumerator **RATE**

enumerator **INTENSIVE_RATE**

enumerator **INITIAL_EXPRESSION**

enumerator **INITIAL_VALUE**

enumerator **INITIAL_INTENSIVE_VALUE**

enumerator **INITIAL_RATE**

enumerator **INITIAL_INTENSIVE_RATE**

enumerator **SIMULATION_TYPE**

enumerator **SPATIAL_DIMENSION**

enumerator **ADD_NOISE**

enumerator **NOISE_EXPRESSION**

enumerator **CHEMICAL_EQUATION**

enumerator **KINETIC_LAW**

enumerator **KINETIC_LAW_EXPRESSION**

enumerator **KINETIC_LAW_UNIT_TYPE**

enumerator **KINETIC_LAW_VARIABLE_MAPPING**

enumerator **LOCAL_REACTION_PARAMETERS**

enumerator **SCALING_COMPARTMENT**

enumerator **FLUX**

enumerator **PARTICLE_FLUX**

enumerator **INITIAL_FLUX**

enumerator **INITIAL_PARTICLE_FLUX**

enumerator **OBJECT_UUID**

enumerator **NAME**

enumerator **OBJECT_PARENT_CN**

enumerator **OBJECT_TYPE**

enumerator **OBJECT_FLAG**

enumerator **OBJECT_HASH**

enumerator **OBJECT_INDEX**

enumerator **OBJECT_REFERENCES**

enumerator **OBJECT_REFERENCE**

enumerator **OBJECT_REFERENCE_CN**

enumerator **OBJECT_REFERENCE_INDEX**

enumerator **OBJECT_POINTER**

enumerator **OBJECT_UNIQUE_NAME**

enumerator **CN**

enumerator **EVALUATION_TREE_TYPE**

enumerator **TASK_TYPE**

enumerator **TASK_SCHEDULED**

enumerator **TASK_UPDATE_MODEL**

enumerator **TASK_REPORT**

enumerator **TASK_REPORT_TARGET**

enumerator **TASK_REPORT_APPEND**

enumerator **TASK_REPORT_CONFIRM_OVERWRITE**

enumerator **PROBLEM**

enumerator **METHOD**

enumerator **METHOD_TYPE**

enumerator **PLOT_TYPE**

enumerator **PLOT_ITEM_TYPE**

enumerator **PARAMETER**

enumerator **PARAMETER_TYPE**

enumerator **PARAMETER_ROLE**

enumerator **PARAMETER_USED**

enumerator **PARAMETER_VALUE**

enumerator **PARAMETER_MAPPING**

enumerator **UNIT**

enumerator **VOLUME_UNIT**

enumerator **AREA_UNIT**

enumerator **LENGTH_UNIT**

enumerator **TIME_UNIT**

enumerator **QUANTITY_UNIT**

enumerator **MODEL_TYPE**

enumerator **AVOGADRO_NUMBER**

enumerator **DIMENSIONALITY**

enumerator **ARRAY_ELEMENT_INDEX**

enumerator **REPORT_SEPARATOR**

enumerator **REPORT_IS_TABLE**

enumerator **REPORT_SHOW_TITLE**

enumerator **REPORT_PRECISION**

enumerator **NOTES**

enumerator **MIRIAM_RDF_XML**

enumerator **MIRIAM_PREDICATE**

enumerator **MIRIAM_RESOURCE**

enumerator **MIRIAM_DESCRIPTION**

enumerator **MIRIAM_ID**

enumerator **DATE**

enumerator **GIVEN_NAME**

enumerator **FAMILY_NAME**

enumerator **EMAIL**

enumerator **ORGANIZATION**

enumerator **FRAMEWORK**

enumerator **DELAY_ASSIGNMENT**

enumerator **FIRE_AT_INITIALTIME**

enumerator **PERSISTENT_TRIGGER**

enumerator **TRIGGER_EXPRESSION**

enumerator **DELAY_EXPRESSION**

enumerator **PRIORITY_EXPRESSION**

enumerator **ASSIGNMENTS**

enumerator **VECTOR_CONTENT**

enumerator **UNIT_SYMBOL**

enumerator **UNIT_EXPRESSION**

enumerator **NOISE**

enumerator **PROPENSITY**

enumerator **__SIZE**

Public Static Attributes

static const CEnumAnnotation<std::string, *Type*> **Name**

class **cpsapiReaction** : public *cpsapiContainer*

Public Types

enum **Property**

Enumeration of the exposed properties

Values:

enumerator **CHEMICAL_EQUATION**

enumerator **KINETIC_LAW**

enumerator **KINETIC_LAW_EXPRESSION**

enumerator **KINETIC_LAW_UNIT_TYPE**

enumerator **SCALING_COMPARTMENT**

enumerator **ADD_NOISE**

enumerator **NOISE_EXPRESSION**

enumerator **NAME**

enumerator **OBJECT_UNIQUE_NAME**

enumerator **CN**

enum **Reference**

Enumeration of the exposed references

Values:

enumerator **NAME**

enumerator **OBJECT_UNIQUE_NAME**

enumerator **FLUX**

enumerator **PARTICLE_FLUX**

enumerator **INITIAL_FLUX**

enumerator **INITIAL_PARTICLE_FLUX**

enumerator **NOISE**

enumerator **PROPENSITY**

typedef *cpsapiReaction* **self**

The class

typedef *cpsapiContainer* **base**

The base class

typedef CReaction **wrapped**

The wrapped COPASI class

typedef std::map<std::string, *cpsapiKineticLawVariable*> **VariableManager**

typedef CDataVector<*cpsapiKineticLawVariable::KineticLawVariable*> **VariableVector**

Public Functions

cpsapiReaction(*wrapped* *pWrapped = nullptr)

Specific constructor

Parameters **wrapped** – * pWrapped

virtual ~**cpsapiReaction**()

Destructor

template<typename **Visitor**>

void **accept**(*Visitor* &visitor)

Accept the given visitor

Template Parameters **Visitor** –

Parameters **Visitor** – & visitor

cpsapiKineticLawVariable **variable**(const std::string &name = "")

Retrieve the kinetic law variable with the givent name

Parameters **const** – std::string & name

Returns *cpsapiKineticLawVariable* variable

cpsapiVector<*cpsapiKineticLawVariable*> **variables**()

Retrieve a vector of the kinetic law variables

Returns *cpsapiVector*< *cpsapiKineticLawVariable* > variables

bool **setProperty**(const *Property* &property, const *cpsapiData* &value, const CCore::Framework
&framework = CCore::Framework::__SIZE)

Set a property of the object to the provided value under the given framework. The value must match the underlying value of the property. The default framework is unspecified

Parameters

- **const** – Property & property
- **const** – *cpsapiData* & value
- **const** – CCore::Framework & framework (default: CCore::Framework::__SIZE)

Returns bool success

cpsapiData **getProperty**(const *Property* &property, const CCore::Framework &framework = CCore::Framework::__SIZE) const

Set a property of the object to the provided value under the given framework. The value must match the underlying value of the property. The default framework is unspecified

Parameters

- **const** – Property & property
- **const** – CCore::Framework & framework (default: CCore::Framework::__SIZE)

Returns *cpsapiData* property

CCommonName **getDataCN**(const *Reference* &reference, const CCore::Framework &framework = CCore::Framework::__SIZE) const

Retrieve a property of the object to the provided value under the given framework. The default framework is unspecified

Parameters

- **const** – Reference & reference
- **const** – CCore::Framework & framework (default: CCore::Framework::__SIZE)

Returns CCommonName

Public Static Attributes

static const Properties **SupportedProperties**

Static set of supported properties

static const Properties **HiddenProperties**

Static set of hidden properties

static const References **SupportedReferences**

Static set of supported references

static const References **HiddenReferences**

Static set of hidden references

Protected Functions

virtual bool **setProperty**(const *cpsapiProperty::Type* &property, const *cpsapiData* &value, const CCore::Framework &framework) override

Set the property

Parameters

- **const** – *cpsapiProperty::Type* & property
- **const** – *cpsapiData* & value
- **const** – CCore::Framework &framework

Returns bool success

virtual *cpsapiData* **getProperty**(const *cpsapiProperty::Type* &property, const CCore::Framework &framework) const override

Retrieve the property

Parameters

- **const** – *cpsapiProperty::Type* & property
- **const** – CCore::Framework &framework

Returns *cpsapiData* property

virtual CCommonName **getDataCN**(const *cpsapiReference::Type* &reference, const CCore::Framework &framework) const override

Retrieve the data reference

Parameters

- **const** – *cpsapiReference::Type* & reference
- **const** – CCore::Framework &framework

Returns CCommonName

Private Functions

cpsapiKineticLawVariable::KineticLawVariable ***assertVariable**(const std::string &name)

Assert that the variable manager contains a variable with the given

Parameters **const** – std::string & name

Returns *cpsapiKineticLawVariable::KineticLawVariable**

class **cpsapiSpecies** : public *cpsapiModelEntity*

Public Types

enum **Property**

Enumeration of the exposed properties

Values:

enumerator **EXPRESSION**

enumerator **INITIAL_EXPRESSION**

enumerator **INITIAL_VALUE**

enumerator **SIMULATION_TYPE**

enumerator **ADD_NOISE**

enumerator **NOISE_EXPRESSION**

enumerator **NAME**

enumerator **OBJECT_UNIQUE_NAME**

enumerator **CN**

enumerator **UNIT**

typedef *cpsapiSpecies* **self**

The class

typedef *cpsapiModelEntity* **base**

The base class

typedef CMetab **wrapped**

The wrapped COPASI class

Public Functions

cpsapiSpecies(*wrapped* *pWrapped = nullptr)

Specific constructor

Parameters

- **wrapped** – * pWrapped Specific constructor
- **wrapped** – * pWrapped

virtual ~**cpsapiSpecies**()

template<typename **Visitor**>

void **accept**(*Visitor* &visitor)

Accept the given visitor

Template Parameters **Visitor** –

Parameters **Visitor** – & visitor

bool **setProperty**(const *Property* &property, const *cpsapiData* &value, const CCore::Framework
&framework = CCore::Framework::__SIZE)

cpsapiData **getProperty**(const *Property* &property, const CCore::Framework &framework =
CCore::Framework::__SIZE) const

Public Static Attributes

static const Properties **SupportedProperties**

Static set of supported properties

Protected Functions

virtual bool **setProperty**(const *cpsapiProperty::Type* &property, const *cpsapiData* &value, const
CCore::Framework &framework) override

Set the property

Parameters

- **const** – cpsapiProperty::Type & property
- **const** – *cpsapiData* & value
- **const** – CCore::Framework &framework

Returns bool success

virtual *cpsapiData* **getProperty**(const *cpsapiProperty::Type* &property, const CCore::Framework
&framework) const override

Retrieve the property

Parameters

- **const** – cpsapiProperty::Type & property
- **const** – CCore::Framework &framework

Returns *cpsapiData* property

class **cpsapiTransaction**

Private Types

```
typedef std::map<CModel*, sChangeInfo> Map
```

```
typedef std::map<CModel*, bool> MapStructureChange
```

Private Static Functions

```
static bool beginTransaction(CModel *pModel)
```

```
static bool endTransaction(CModel *pModel)
```

```
static bool synchronize(const CDataObject *pObject, const CCore::Framework &framework)
```

```
static bool beginStructureChange(CModel *pModel)
```

```
static bool endStructureChange(CModel *pModel)
```

Private Static Attributes

```
static Map Transactions
```

```
static MapStructureChange StructureChange
```

Friends

```
friend class cpsapiObject
```

```
friend class cpsapiModelEntity
```

```
friend class cpsapiCompartment
```

```
friend class cpsapiSpecies
```

```
friend class cpsapiGlobalQuantity
```

```
friend class cpsapiReaction
```

```
friend class cpsapiKineticLawVariable
```

```
friend class cpsapiModel
```

```
friend class cpsapiEvent
```

```
friend class cpsapiEventAssignment
```

```
class cpsapiValue : public cpsapiObject
```

Public Types

enum **Property**

Enumeration of the exposed properties

Values:

enumerator **VALUE**

enumerator **NAME**

enumerator **OBJECT_UNIQUE_NAME**

enumerator **CN**

enum **Reference**

Enumeration of the exposed references

Values:

enumerator **VALUE**

enumerator **NAME**

enumerator **OBJECT_UNIQUE_NAME**

typedef *cpsapiValue* **self**

The class

typedef *cpsapiObject* **base**

The base class

typedef CDataObject **wrapped**

The wrapped COPASI class

Public Functions

cpsapiValue(*wrapped* *pWrapped = nullptr)

Specific constructor

Parameters **wrapped** – * pWrapped (default: nullptr)

virtual **~cpsapiValue**()

Destructor

template<typename **Visitor**>

void **accept**(*Visitor* &visitor)

Accept the given visitor

Template Parameters **Visitor** –

Parameters **Visitor** – & visitor

bool **setProperty**(const *Property* &property, const *cpsapiData* &value, const CCore::Framework &framework = CCore::Framework::__SIZE)

Set a property of the object to the provided value under the given framework. The value must match the underlying value of the property. The default framework is unspecified

Parameters

- **const** – Property & property
- **const** – *cpsapiData* & value
- **const** – CCore::Framework & framework (default: CCore::Framework::__SIZE)

Returns bool success

cpsapiData **getProperty**(const *Property* &property, const CCore::Framework &framework = CCore::Framework::__SIZE) const

Set a property of the object to the provided value under the given framework. The value must match the underlying value of the property. The default framework is unspecified

Parameters

- **const** – Property & property
- **const** – CCore::Framework & framework (default: CCore::Framework::__SIZE)

Returns *cpsapiData* property

CCommonName **getDataCN**(const *Reference* &reference, const CCore::Framework &framework = CCore::Framework::__SIZE) const

Retrieve a property of the object to the provided value under the given framework. The default framework is unspecified

Parameters

- **const** – Reference & reference
- **const** – CCore::Framework & framework (default: CCore::Framework::__SIZE)

Returns CCommonName

operator *cpsapiData*() const

Convert the contained value into *cpsapiData*

Returns *cpsapiData*

bool **valid**() const

Check whether it contains valid value

Returns bool valid

Returns

Public Static Attributes

static const Properties **SupportedProperties**

Static set of supported properties

static const Properties **HiddenProperties**

Static set of hidden properties

static const References **SupportedReferences**

Static set of supported references

static const References **HiddenReferences**

Static set of hidden references

Protected Functions

virtual bool **setProperty**(const *cpsapiProperty::Type* &property, const *cpsapiData* &value, const CCore::Framework &framework) override

Set the property

Parameters

- **const** – *cpsapiProperty::Type* & property
- **const** – *cpsapiData* & value
- **const** – CCore::Framework &framework

Returns bool success

virtual *cpsapiData* **getProperty**(const *cpsapiProperty::Type* &property, const CCore::Framework &framework) const override

Retrieve the property

Parameters

- **const** – *cpsapiProperty::Type* & property
- **const** – CCore::Framework &framework

Returns *cpsapiData* property

virtual CCommonName **getDataCN**(const *cpsapiReference::Type* &reference, const CCore::Framework &framework) const override

Retrieve the data reference

Parameters

- **const** – *cpsapiReference::Type* & reference
- **const** – CCore::Framework &framework

Returns CCommonName

template<class **Object**>

class **cpsapiVector** : public *cpsapiContainer*

Public Types

enum **Property**

Enumeration of the exposed properties

Values:

typedef *cpsapiVector*<*Object*> **self**

The class

typedef *cpsapiContainer* **base**

The base class

typedef CDataVector<typename *Object*::wrapped> **wrapped**

The wrapped COPASI class

typedef std::map<typename *Object*::wrapped*, *Object*> **ObjectMap**

We need to keep the cpsapiObjects around since the iterator returns references or pointers.

Public Functions

cpsapiVector(*wrapped* *pWrapped = nullptr, const *cpsapiObjectData*::Type &type = *cpsapiObjectData*::Type::Vector)

Specific constructor

Parameters

- **wrapped** – * pWrapped (default: nullptr)
- **const** – Type & type (default: Type::cpsapiVector)

virtual ~**cpsapiVector**()

Destructor

template<typename **Visitor**>

void **accept**(*Visitor* &visitor)

Accept the given visitor

Template Parameters **Visitor** –

Parameters **Visitor** – & visitor

size_t **size**() const

iterator **begin**()

iterator **end**()

Object &**operator**[] (const size_t &index)

Object &**operator**[] (const std::string &name)

size_t **index**(const std::string &name) const

template<>

inline size_t **index**(const std::string &name) const

Public Static Attributes

```
static const Properties SupportedProperties = { }  
    Static set of supported properties
```

```
struct cpsapiVisitor
```

Public Static Functions

```
template<typename Visitor, typename Visited>  
static inline void acceptIfVisitable(Visitor &visitor, Visited *pVisited)  
  
template<class Visitor>  
static void accept(Visitor &visitor, CDataObject *pObject)
```

Private Static Functions

```
template<typename visitor, typename visited, typename Z =  
decltype(std::declval<visitor>().visit(std::declval<visited*>()))>  
static inline void doVisit(visitor &v, visited *pV, int)  
  
template<typename visitor, typename visited>  
static inline void doVisit(visitor, visited*, ...)
```

```
class cpsapiGroup::Data : public Data
```

Public Functions

```
inline Data(const base::Data &data)  
  
inline virtual ~Data()
```

Public Members

```
cpsapiParameter mDefaultParameter
```

```
class cpsapiDataModel::Data : public Data
```

Public Functions

```
inline Data(const base::Data &data)
```

```
inline virtual ~Data()
```

Public Members

```
cpsapiModel mModel
```

```
cpsapiTask mDefaultTask
```

```
CReportDefinition *mpDefaultReportDefinition
```

```
CPlotSpecification *mpDefaultPlotSpecification
```

```
class cpsapiVector::Data : public Data
```

Public Functions

```
inline Data(const base::Data &data)
```

```
inline virtual ~Data()
```

Public Members

```
ObjectMap mMap
```

```
class cpsapiReaction::Data : public cpsapiObjectData
```

Public Functions

```
inline Data(const base::Data &data)
```

```
inline virtual ~Data()
```

Public Members

```
VariableManager mVariableManager
```

```
VariableVector *mpVector
```

```
cpsapiKineticLawVariable mDefaultVariable
```

```
class cpsapiCompartment::Data : public cpsapiObjectData
```

Public Functions

inline **Data**(const *base*::Data &data)

inline virtual ~**Data**()

Public Members

cpsapiSpecies **mDefaultSpecies**

class *cpsapiEvent*::**Data** : public *cpsapiObjectData*

Public Functions

inline **Data**(const *base*::Data &data)

inline virtual ~**Data**()

Public Members

cpsapiEventAssignment **mDefaultEventAssignment**

class *cpsapiModel*::**Data** : public *cpsapiObjectData*

Public Functions

inline **Data**(const *base*::Data &data)

inline virtual ~**Data**()

Public Members

cpsapiCompartment **mDefaultCompartment**

cpsapiReaction **mDefaultReaction**

cpsapiGlobalQuantity **mDefaultGlobalQuantity**

cpsapiEvent **mDefaultEvent**

class *cpsapiVector*::**iterator** : public *iterator*

Public Functions

```

inline iterator()

inline iterator(const typename wrapped::iterator &src, ObjectMap &map)

inline iterator(ObjectMap &map)

inline ~iterator()

inline Object &operator*() const

inline Object *operator->() const

inline operator Object*() const

inline iterator &operator++()

inline iterator operator++(int)

inline iterator &operator--()

inline iterator operator--(int)

inline iterator &operator+=(const typename wrapped::iterator::difference_type &n)

inline iterator operator+(const typename wrapped::iterator::difference_type &n) const

inline iterator &operator-=(const typename wrapped::iterator::difference_type &n)

inline iterator operator-(const typename wrapped::iterator::difference_type &n) const

```

Protected Attributes

ObjectMap *mpMap

```

class cpsapiKineticLawVariable : KineticLawVariable : public CDataObject
    #include <cpsapiKineticLawVariable.h> A fake CDataObject representing the the variables of the kinetic law

```

Public Functions

```
KineticLawVariable(const KineticLawVariable &src, CDataContainer *pParent = nullptr)
```

Copy constructor

Parameters

- **const** – *KineticLawVariable* & src,
- **CDataContainer** – * pParent (default: nullptr)

```
virtual ~KineticLawVariable()
```

Destructor

```
void updateMappedObject()
```

Public Members

cpsapiObject *mpMappedObject

Public Static Functions

static *KineticLawVariable* ***fromData**(const CData &data, CUndoObjectInterface *pParent)

Required for objects to be insterted into CDataVector

Parameters

- **const** – CData & data
- **CUndoObjectInterface** – * pParent

Returns *KineticLawVariable* * pData

Private Functions

KineticLawVariable()

Default constructor (not implemented)

KineticLawVariable(CReaction *pReaction, const std::string &name = "")

Construct a new Fake Data object

Parameters

- **CReaction** – * pReaction
- **const** – std::string & name

Friends

friend class cpsapiReaction

template<class **cpsapi**, class **copasi**>

struct *cpsapiFactory::Part* : public *cpsapiFactory::PartInterface*

Public Functions

inline **Part**(const *PartType* &partType = *PartType::__SIZE*)

inline virtual ~**Part**()

virtual *cpsapiObject* ***create**(CDataObject *pDataObject) const override

virtual *cpsapiObject* ***copy**(const *cpsapiObject* &src) const override

virtual const std::type_info &**cpsapiType**() const override

virtual const std::type_info &**copasiType**() const override

```
virtual void accept(CDataObject *pDataObject, const cpsapiVisitor::VisitorInterface &visitor) const
    override
```

```
template<>
inline virtual cpsapiObject *create(CDataObject*) const
```

```
template<>
inline virtual cpsapiObject *copy(const cpsapiObject&) const
```

```
template<>
inline virtual void accept(CDataObject*, const cpsapiVisitor::VisitorInterface&) const
```

```
struct cpsapiFactory::PartInterface
```

```
    Subclassed by cpsapiFactory::Part< cpsapi, copasi >
```

Public Functions

```
inline PartInterface(const PartType &partType)
```

```
inline virtual ~PartInterface()
```

```
virtual cpsapiObject *create(CDataObject*) const = 0
```

```
virtual cpsapiObject *copy(const cpsapiObject&) const = 0
```

```
virtual const std::type_info &cpsapiType() const = 0
```

```
virtual const std::type_info &copasiType() const = 0
```

```
virtual void accept(CDataObject*, const cpsapiVisitor::VisitorInterface&) const = 0
```

Public Members

```
const PartType type
```

```
struct cpsapiTransaction::sChangeInfo
```

Public Members

```
std::set<const CDataObject*> changed
```

```
CCore::Framework framework = CCore::Framework::__SIZE
```

```
template<class cpsapi, class Visitor>
```

```
struct cpsapiVisitor::VisitorImplementation : public cpsapiVisitor::VisitorInterface
```

Public Functions

```
inline VisitorImplementation(Visitor &visitor)

inline virtual ~VisitorImplementation()

inline virtual void visit(cpsapiObject *pObject) const override
```

Public Members

Visitor &**mVisitor**

```
struct cpsapiVisitor::VisitorInterface
    Subclassed by cpsapiVisitor::VisitorImplementation< cpsapi, Visitor >
```

Public Functions

```
inline virtual ~VisitorInterface()

virtual void visit(cpsapiObject *pObject) const = 0
```

```
file cpsapiContainer.h
    #include "cpsapi/core/cpsapiObject.h"#include <>

file cpsapiData.h
    #include <>#include <>#include <>#include <>#include <>#include <>

file cpsapiDataCollector.h
    #include <>#include "cpsapi/core/cpsapiObject.h"

file cpsapiDataModel.h
    #include "cpsapi/core/cpsapiContainer.h"#include "cpsapi/model/cpsapiModel.h"#include ""#include <>

file cpsapiFactory.h
    #include <>#include <>#include <>#include <>#include <>#include <>#include <>#include <>

file cpsapiGroup.h
    #include "cpsapi/core/cpsapiParameter.h"#include <>

file cpsapiObject.h
    #include <>#include <>#include <>#include <>#include "cpsapi/core/cpsapiProperty.h"#include
    "cpsapi/core/cpsapiObjectData.h"#include "cpsapi/core/cpsapiVisitor.h"#include "cp-
    sapi/core/cpsapiFactory.h"#include "cpsapi/core/cpsapiData.h"
```


Defines

DATA

WRAPPED

Functions

```
CPSAPI_NAMESPACE_END std::ostream & operator<< (std::ostream &os,
const CPSAPI_NAMESPACE_QUALIFIER cpsapiObject &object)
```

file **cpsapiObjectData.h**

```
#include <>#include <>#include <>#include <>#include <>
```

file **cpsapiParameter.h**

```
#include "cpsapi/core/cpsapiContainer.h"#include <>
```

file **cpsapiProperty.h**

```
#include <>#include <>#include <>
```

Typedefs

```
typedef cpsapiProperty cpsapiReference
```

file **cpsapiRoot.h**

```
#include <>#include "cpsapi/core/cpsapiContainer.h"#include "cpsapi/model/cpsapiModel.h"#include
"cpsapi/model/cpsapiCompartment.h"#include "cpsapi/model/cpsapiSpecies.h"#include "cp-
sapi/model/cpsapiGlobalQuantity.h"#include "cpsapi/model/cpsapiReaction.h"#include "cp-
sapi/model/cpsapiEvent.h"#include "cpsapi/model/cpsapiEventAssignment.h"
```

file **cpsapiValue.h**

```
#include "cpsapi/core/cpsapiObject.h"#include <>#include <>
```

file **cpsapiVector.h**

```
#include "cpsapi/core/cpsapiContainer.h"#include <>
```

file **cpsapiVisitor.h**

```
#include <>#include <>#include <>#include <>
```

file **cpsapiCompartment.h**

```
#include "cpsapi/core/cpsapiVector.h"#include "cpsapi/model/cpsapiModelEntity.h"#include "cp-
sapi/model/cpsapiSpecies.h"#include <>#include <>
```

file **cpsapiEvent.h**

`#include "cpsapi/core/cpsapiVector.h"#include "cpsapi/model/cpsapiEventAssignment.h"`

file **cpsapiEventAssignment.h**

`#include "cpsapi/core/cpsapiContainer.h"`

file **cpsapiGlobalQuantity.h**

`#include "cpsapi/model/cpsapiModelEntity.h"`

file **cpsapiKineticLawVariable.h**

`#include "cpsapi/core/cpsapiContainer.h"#include "cpsapi/core/cpsapiVector.h"#include <>`

file **cpsapiModel.h**

`#include "cpsapi/model/cpsapiModelEntity.h"#include "cpsapi/model/cpsapiCompartment.h"#include "cpsapi/model/cpsapiSpecies.h"#include "cpsapi/model/cpsapiGlobalQuantity.h"#include "cpsapi/model/cpsapiReaction.h"#include "cpsapi/model/cpsapiEvent.h"#include <>`

file **cpsapiModelEntity.h**

`#include "cpsapi/core/cpsapiContainer.h"#include <>`

file **cpsapiReaction.h**

`#include "cpsapi/model/cpsapiKineticLawVariable.h"#include "cpsapi/core/cpsapiContainer.h"#include "cpsapi/core/cpsapiVector.h"`

file **cpsapiSpecies.h**

`#include "cpsapi/model/cpsapiModelEntity.h"`

file **cpsapiTransaction.h**

`#include <>#include <>#include ""#include <>`

dir

`/home/docs/checkouts/readthedocs.org/user_builds/copasi-api/checkouts/latest/cpsapi/core`

dir `/home/docs/checkouts/readthedocs.org/user_builds/copasi-api/checkouts/latest/cpsapi`

dir

`/home/docs/checkouts/readthedocs.org/user_builds/copasi-api/checkouts/latest/cpsapi/model`

INDICES AND TABLES

- `genindex`
- `modindex`
- `search`

C

cpsapi (C++ class), 9
 cpsapi::~~cpsapi (C++ function), 9
 cpsapi::addCompartment (C++ function), 10
 cpsapi::addDataModel (C++ function), 9
 cpsapi::addEvent (C++ function), 10
 cpsapi::addGlobalQuantity (C++ function), 10
 cpsapi::addReaction (C++ function), 10
 cpsapi::addSpecies (C++ function), 10
 cpsapi::base (C++ type), 9
 cpsapi::beginTransaction (C++ function), 10
 cpsapi::compartment (C++ function), 10
 cpsapi::cpsapi (C++ function), 9
 cpsapi::dataModel (C++ function), 9
 cpsapi::DataModels (C++ member), 11
 cpsapi::DefaultDataModel (C++ member), 11
 cpsapi::deleteCompartment (C++ function), 10
 cpsapi::deleteDataModel (C++ function), 9
 cpsapi::deleteEvent (C++ function), 10
 cpsapi::deleteGlobalQuantity (C++ function), 10
 cpsapi::deleteReaction (C++ function), 10
 cpsapi::deleteSpecies (C++ function), 10
 cpsapi::endTransaction (C++ function), 10
 cpsapi::event (C++ function), 10
 cpsapi::getCompartments (C++ function), 10
 cpsapi::getDataModels (C++ function), 9
 cpsapi::getEvents (C++ function), 10
 cpsapi::getGlobalQuantities (C++ function), 10
 cpsapi::getReactions (C++ function), 10
 cpsapi::getSpecies (C++ function), 10
 cpsapi::globalQuantity (C++ function), 10
 cpsapi::init (C++ function), 9
 cpsapi::listModelNames (C++ function), 9
 cpsapi::loadFromFile (C++ function), 9
 cpsapi::loadFromString (C++ function), 9
 cpsapi::model (C++ function), 10
 cpsapi::pDefaultFunction (C++ member), 11
 cpsapi::pDefaultUnitDefinition (C++ member), 11
 cpsapi::reaction (C++ function), 10
 cpsapi::release (C++ function), 9
 cpsapi::self (C++ type), 9
 cpsapi::species (C++ function), 10
 cpsapiCompartment (C++ class), 11
 cpsapiCompartment::__species (C++ function), 13
 cpsapiCompartment::~~cpsapiCompartment (C++ function), 12
 cpsapiCompartment::accept (C++ function), 12
 cpsapiCompartment::addSpecies (C++ function), 12
 cpsapiCompartment::base (C++ type), 11
 cpsapiCompartment::cpsapiCompartment (C++ function), 12
 cpsapiCompartment::Data (C++ class), 71
 cpsapiCompartment::Data::~Data (C++ function), 72
 cpsapiCompartment::Data::Data (C++ function), 72
 cpsapiCompartment::Data::mDefaultSpecies (C++ member), 72
 cpsapiCompartment::deleteSpecies (C++ function), 12
 cpsapiCompartment::getProperty (C++ function), 12, 13
 cpsapiCompartment::getSpecies (C++ function), 12
 cpsapiCompartment::Property (C++ enum), 11
 cpsapiCompartment::Property::ADD_NOISE (C++ enumerator), 11
 cpsapiCompartment::Property::CN (C++ enumerator), 11
 cpsapiCompartment::Property::DIMENSIONALITY (C++ enumerator), 11
 cpsapiCompartment::Property::EXPRESSION (C++ enumerator), 11
 cpsapiCompartment::Property::INITIAL_EXPRESSION (C++ enumerator), 11
 cpsapiCompartment::Property::INITIAL_VALUE (C++ enumerator), 11
 cpsapiCompartment::Property::NAME (C++ enumerator), 11
 cpsapiCompartment::Property::NOISE_EXPRESSION (C++ enumerator), 11
 cpsapiCompartment::Property::OBJECT_UNIQUE_NAME (C++ enumerator), 11
 cpsapiCompartment::Property::SIMULATION_TYPE (C++ enumerator), 11

cpsapiCompartment::Property::UNIT (C++ *enumerator*), 11
 cpsapiCompartment::self (C++ *type*), 11
 cpsapiCompartment::setProperty (C++ *function*), 12
 cpsapiCompartment::species (C++ *function*), 12
 cpsapiCompartment::SupportedProperties (C++ *member*), 12
 cpsapiCompartment::updateDefaultSpecies (C++ *function*), 13
 cpsapiCompartment::wrapped (C++ *type*), 12
 cpsapiContainer (C++ *class*), 13
 cpsapiContainer::~~cpsapiContainer (C++ *function*), 13
 cpsapiContainer::accept (C++ *function*), 13
 cpsapiContainer::base (C++ *type*), 13
 cpsapiContainer::cpsapiContainer (C++ *function*), 14
 cpsapiContainer::self (C++ *type*), 13
 cpsapiContainer::wrapped (C++ *type*), 13
 cpsapiData (C++ *class*), 14
 cpsapiData::~~cpsapiData (C++ *function*), 15
 cpsapiData::base (C++ *type*), 16
 cpsapiData::cpsapiData (C++ *function*), 15
 cpsapiData::getType (C++ *function*), 15
 cpsapiData::operator= (C++ *function*), 15
 cpsapiData::toBool (C++ *function*), 15
 cpsapiData::toCommonName (C++ *function*), 15
 cpsapiData::toData (C++ *function*), 15
 cpsapiData::toDouble (C++ *function*), 15
 cpsapiData::toInt32 (C++ *function*), 15
 cpsapiData::toObject (C++ *function*), 15
 cpsapiData::toSizeType (C++ *function*), 15
 cpsapiData::toString (C++ *function*), 15
 cpsapiData::toUnsignedInt32 (C++ *function*), 15
 cpsapiData::Type (C++ *enum*), 14
 cpsapiData::Type::__SIZE (C++ *enumerator*), 14
 cpsapiData::Type::Bool (C++ *enumerator*), 14
 cpsapiData::Type::CommonName (C++ *enumerator*), 14
 cpsapiData::Type::Double (C++ *enumerator*), 14
 cpsapiData::Type::Int32 (C++ *enumerator*), 14
 cpsapiData::Type::Object (C++ *enumerator*), 14
 cpsapiData::Type::SizeType (C++ *enumerator*), 14
 cpsapiData::Type::String (C++ *enumerator*), 14
 cpsapiData::Type::UnsignedInt32 (C++ *enumerator*), 14
 cpsapiData::Type::Vector (C++ *enumerator*), 14
 cpsapiData::Vector (C++ *type*), 14
 cpsapiDataCollector (C++ *class*), 16
 cpsapiDataCollector::~~cpsapiDataCollector (C++ *function*), 16
 cpsapiDataCollector::addDataReferenceAfter (C++ *function*), 16
 cpsapiDataCollector::addDataReferenceBefore (C++ *function*), 16
 cpsapiDataCollector::addDataReferenceDuring (C++ *function*), 16
 cpsapiDataCollector::clearData (C++ *function*), 16
 cpsapiDataCollector::clearReferences (C++ *function*), 16
 cpsapiDataCollector::collect (C++ *function*), 17
 cpsapiDataCollector::compile (C++ *function*), 16
 cpsapiDataCollector::finish (C++ *function*), 16
 cpsapiDataCollector::generateValues (C++ *function*), 17
 cpsapiDataCollector::getDataAfter (C++ *function*), 16
 cpsapiDataCollector::getDataBefore (C++ *function*), 16
 cpsapiDataCollector::getDataDuring (C++ *function*), 16
 cpsapiDataCollector::getDataReferencesAfter (C++ *function*), 16
 cpsapiDataCollector::getDataReferencesBefore (C++ *function*), 16
 cpsapiDataCollector::getDataReferencesDuring (C++ *function*), 16
 cpsapiDataCollector::getDataTypesAfter (C++ *function*), 16
 cpsapiDataCollector::getDataTypesBefore (C++ *function*), 16
 cpsapiDataCollector::getDataTypesDuring (C++ *function*), 16
 cpsapiDataCollector::mCnAsAfter (C++ *member*), 17
 cpsapiDataCollector::mCnAsBefore (C++ *member*), 17
 cpsapiDataCollector::mCnAsDuring (C++ *member*), 17
 cpsapiDataCollector::mDataAfter (C++ *member*), 17
 cpsapiDataCollector::mDataBefore (C++ *member*), 17
 cpsapiDataCollector::mDataDuring (C++ *member*), 17
 cpsapiDataCollector::mValuesAfter (C++ *member*), 17
 cpsapiDataCollector::mValuesBefore (C++ *member*), 17
 cpsapiDataCollector::mValuesDuring (C++ *member*), 17
 cpsapiDataCollector::output (C++ *function*), 16
 cpsapiDataCollector::separate (C++ *function*), 16
 cpsapiDataModel (C++ *class*), 17
 cpsapiDataModel::__task (C++ *function*), 19
 cpsapiDataModel::~~cpsapiDataModel (C++ *func-*

tion), 18
 cpsapiDataModel::accept (C++ function), 18
 cpsapiDataModel::addCompartment (C++ function), 18
 cpsapiDataModel::addEvent (C++ function), 19
 cpsapiDataModel::addGlobalQuantity (C++ function), 19
 cpsapiDataModel::addReaction (C++ function), 19
 cpsapiDataModel::addSpecies (C++ function), 18
 cpsapiDataModel::base (C++ type), 17
 cpsapiDataModel::beginTransaction (C++ function), 18
 cpsapiDataModel::compartment (C++ function), 18
 cpsapiDataModel::cpsapiDataModel (C++ function), 18
 cpsapiDataModel::Data (C++ class), 70
 cpsapiDataModel::Data::~Data (C++ function), 71
 cpsapiDataModel::Data::Data (C++ function), 71
 cpsapiDataModel::Data::mDefaultTask (C++ member), 71
 cpsapiDataModel::Data::mModel (C++ member), 71
 cpsapiDataModel::Data::mpDefaultPlotSpecification (C++ member), 71
 cpsapiDataModel::Data::mpDefaultReportDefinition (C++ member), 71
 cpsapiDataModel::deleteCompartment (C++ function), 18
 cpsapiDataModel::deleteEvent (C++ function), 19
 cpsapiDataModel::deleteGlobalQuantity (C++ function), 19
 cpsapiDataModel::deleteReaction (C++ function), 19
 cpsapiDataModel::deleteSpecies (C++ function), 18
 cpsapiDataModel::endTransaction (C++ function), 18
 cpsapiDataModel::event (C++ function), 19
 cpsapiDataModel::getCompartments (C++ function), 18
 cpsapiDataModel::getEvents (C++ function), 19
 cpsapiDataModel::getGlobalQuantities (C++ function), 19
 cpsapiDataModel::getReactions (C++ function), 19
 cpsapiDataModel::getSpecies (C++ function), 18
 cpsapiDataModel::getTasks (C++ function), 19
 cpsapiDataModel::globalQuantity (C++ function), 19
 cpsapiDataModel::loadFromFile (C++ function), 18
 cpsapiDataModel::loadFromString (C++ function), 18
 cpsapiDataModel::method (C++ function), 19
 cpsapiDataModel::model (C++ function), 18
 cpsapiDataModel::problem (C++ function), 19
 cpsapiDataModel::Property (C++ enum), 17
 cpsapiDataModel::reaction (C++ function), 19
 cpsapiDataModel::self (C++ type), 17
 cpsapiDataModel::species (C++ function), 18
 cpsapiDataModel::task (C++ function), 19
 cpsapiDataModel::wrapped (C++ type), 18
 cpsapiEvent (C++ class), 19
 cpsapiEvent::__eventAssignment (C++ function), 23
 cpsapiEvent::~cpsapiEvent (C++ function), 20
 cpsapiEvent::accept (C++ function), 20
 cpsapiEvent::addEventAssignment (C++ function), 21
 cpsapiEvent::base (C++ type), 20
 cpsapiEvent::cpsapiEvent (C++ function), 20
 cpsapiEvent::Data (C++ class), 72
 cpsapiEvent::Data::~Data (C++ function), 72
 cpsapiEvent::Data::Data (C++ function), 72
 cpsapiEvent::Data::mDefaultEventAssignment (C++ member), 72
 cpsapiEvent::deleteEventAssignment (C++ function), 21
 cpsapiEvent::eventAssignment (C++ function), 21
 cpsapiEvent::getDataCN (C++ function), 21, 22
 cpsapiEvent::getEventAssignments (C++ function), 21
 cpsapiEvent::getProperty (C++ function), 21, 22
 cpsapiEvent::HiddenProperties (C++ member), 22
 cpsapiEvent::HiddenReferences (C++ member), 22
 cpsapiEvent::Property (C++ enum), 19
 cpsapiEvent::Property::CN (C++ enumerator), 20
 cpsapiEvent::Property::DELAY_ASSIGNMENT (C++ enumerator), 19
 cpsapiEvent::Property::DELAY_EXPRESSION (C++ enumerator), 20
 cpsapiEvent::Property::FIRE_AT_INITIALTIME (C++ enumerator), 19
 cpsapiEvent::Property::NAME (C++ enumerator), 20
 cpsapiEvent::Property::OBJECT_UNIQUE_NAME (C++ enumerator), 20
 cpsapiEvent::Property::PERSISTENT_TRIGGER (C++ enumerator), 19
 cpsapiEvent::Property::PRIORITY_EXPRESSION (C++ enumerator), 20
 cpsapiEvent::Property::TRIGGER_EXPRESSION (C++ enumerator), 19
 cpsapiEvent::Reference (C++ enum), 20
 cpsapiEvent::Reference::NAME (C++ enumerator), 20
 cpsapiEvent::Reference::OBJECT_UNIQUE_NAME (C++ enumerator), 20
 cpsapiEvent::self (C++ type), 20
 cpsapiEvent::setProperty (C++ function), 21, 22

cpsapiEvent::SupportedProperties (C++ member), 22
 cpsapiEvent::SupportedReferences (C++ member), 22
 cpsapiEvent::updateDefaultEventAssignment (C++ function), 23
 cpsapiEvent::wrapped (C++ type), 20
 cpsapiEventAssignment (C++ class), 23
 cpsapiEventAssignment::~~cpsapiEventAssignment (C++ function), 24
 cpsapiEventAssignment::accept (C++ function), 24
 cpsapiEventAssignment::base (C++ type), 24
 cpsapiEventAssignment::cpsapiEventAssignment (C++ function), 24
 cpsapiEventAssignment::getDataCN (C++ function), 25, 26
 cpsapiEventAssignment::getProperty (C++ function), 24, 25
 cpsapiEventAssignment::HiddenProperties (C++ member), 25
 cpsapiEventAssignment::HiddenReferences (C++ member), 25
 cpsapiEventAssignment::Property (C++ enum), 23
 cpsapiEventAssignment::Property::CN (C++ enumerator), 23
 cpsapiEventAssignment::Property::EXPRESSION (C++ enumerator), 23
 cpsapiEventAssignment::Property::NAME (C++ enumerator), 23
 cpsapiEventAssignment::Property::OBJECT_REFERENCE_CN (C++ enumerator), 23
 cpsapiEventAssignment::Property::OBJECT_UNIQUE_NAME (C++ enumerator), 23
 cpsapiEventAssignment::Reference (C++ enum), 23
 cpsapiEventAssignment::Reference::NAME (C++ enumerator), 23
 cpsapiEventAssignment::Reference::OBJECT_UNIQUE_NAME (C++ enumerator), 23
 cpsapiEventAssignment::self (C++ type), 24
 cpsapiEventAssignment::setProperty (C++ function), 24, 25
 cpsapiEventAssignment::SupportedProperties (C++ member), 25
 cpsapiEventAssignment::SupportedReferences (C++ member), 25
 cpsapiEventAssignment::wrapped (C++ type), 24
 cpsapiFactory (C++ class), 26
 cpsapiFactory::accept (C++ function), 28
 cpsapiFactory::callDerived (C++ function), 28
 cpsapiFactory::copasiMap (C++ member), 29
 cpsapiFactory::CopasiMap (C++ type), 28
 cpsapiFactory::copy (C++ function), 28
 cpsapiFactory::create (C++ function), 28
 cpsapiFactory::free_unique (C++ function), 27, 28
 cpsapiFactory::free_unique_t (C++ type), 27
 cpsapiFactory::getDataType (C++ function), 28
 cpsapiFactory::info (C++ function), 28
 cpsapiFactory::init (C++ function), 28
 cpsapiFactory::insert (C++ function), 28
 cpsapiFactory::make (C++ function), 28
 cpsapiFactory::make_shared (C++ function), 27, 28
 cpsapiFactory::make_unique (C++ function), 27, 28
 cpsapiFactory::Part (C++ struct), 74
 cpsapiFactory::Part::~~Part (C++ function), 74
 cpsapiFactory::Part::accept (C++ function), 74, 75
 cpsapiFactory::Part::copasiType (C++ function), 74
 cpsapiFactory::Part::copy (C++ function), 74, 75
 cpsapiFactory::Part::cpsapiType (C++ function), 74
 cpsapiFactory::Part::create (C++ function), 74, 75
 cpsapiFactory::Part::Part (C++ function), 74
 cpsapiFactory::PartInterface (C++ struct), 75
 cpsapiFactory::PartInterface::~~PartInterface (C++ function), 75
 cpsapiFactory::PartInterface::accept (C++ function), 75
 cpsapiFactory::PartInterface::copasiType (C++ function), 75
 cpsapiFactory::PartInterface::copy (C++ function), 75
 cpsapiFactory::PartInterface::cpsapiType (C++ function), 75
 cpsapiFactory::PartInterface::create (C++ function), 75
 cpsapiFactory::PartInterface::PartInterface (C++ function), 75
 cpsapiFactory::PartInterface::type (C++ member), 75
 cpsapiFactory::PartType (C++ enum), 26
 cpsapiFactory::PartType::__SIZE (C++ enumerator), 27
 cpsapiFactory::PartType::compartment (C++ enumerator), 26
 cpsapiFactory::PartType::dataModel (C++ enumerator), 27
 cpsapiFactory::PartType::event (C++ enumerator), 27
 cpsapiFactory::PartType::eventAssignment (C++ enumerator), 27
 cpsapiFactory::PartType::globalQuantity (C++ enumerator), 26
 cpsapiFactory::PartType::group (C++ enumerator), 27
 cpsapiFactory::PartType::method (C++ enumerator), 27

tor), 27
 cpsapiFactory::PartType::model (C++ enumerator), 26
 cpsapiFactory::PartType::parameter (C++ enumerator), 27
 cpsapiFactory::PartType::problem (C++ enumerator), 27
 cpsapiFactory::PartType::reaction (C++ enumerator), 27
 cpsapiFactory::PartType::reactionParameter (C++ enumerator), 27
 cpsapiFactory::PartType::species (C++ enumerator), 26
 cpsapiFactory::PartType::task (C++ enumerator), 27
 cpsapiFactory::PartType::value (C++ enumerator), 26
 cpsapiFactory::PartType::vectorCompartment (C++ enumerator), 26
 cpsapiFactory::PartType::vectorDataModel (C++ enumerator), 26
 cpsapiFactory::PartType::vectorEvent (C++ enumerator), 26
 cpsapiFactory::PartType::vectorEventAssignment (C++ enumerator), 26
 cpsapiFactory::PartType::vectorGlobalQuantity (C++ enumerator), 26
 cpsapiFactory::PartType::vectorReaction (C++ enumerator), 26
 cpsapiFactory::PartType::vectorReactionParameter (C++ enumerator), 26
 cpsapiFactory::PartType::vectorSpecies (C++ enumerator), 26
 cpsapiFactory::PartType::vectorTask (C++ enumerator), 26
 cpsapiGlobalQuantity (C++ class), 29
 cpsapiGlobalQuantity::~~cpsapiGlobalQuantity (C++ function), 30
 cpsapiGlobalQuantity::accept (C++ function), 30
 cpsapiGlobalQuantity::base (C++ type), 29
 cpsapiGlobalQuantity::cpsapiGlobalQuantity (C++ function), 30
 cpsapiGlobalQuantity::getProperty (C++ function), 30
 cpsapiGlobalQuantity::Property (C++ enum), 29
 cpsapiGlobalQuantity::Property::ADD_NOISE (C++ enumerator), 29
 cpsapiGlobalQuantity::Property::CN (C++ enumerator), 29
 cpsapiGlobalQuantity::Property::EXPRESSION (C++ enumerator), 29
 cpsapiGlobalQuantity::Property::INITIAL_EXPRESSION (C++ enumerator), 29
 cpsapiGlobalQuantity::Property::INITIAL_VALUE (C++ enumerator), 29
 cpsapiGlobalQuantity::Property::NAME (C++ enumerator), 29
 cpsapiGlobalQuantity::Property::NOISE_EXPRESSION (C++ enumerator), 29
 cpsapiGlobalQuantity::Property::OBJECT_UNIQUE_NAME (C++ enumerator), 29
 cpsapiGlobalQuantity::Property::SIMULATION_TYPE (C++ enumerator), 29
 cpsapiGlobalQuantity::Property::UNIT (C++ enumerator), 29
 cpsapiGlobalQuantity::self (C++ type), 29
 cpsapiGlobalQuantity::setProperty (C++ function), 30
 cpsapiGlobalQuantity::SupportedProperties (C++ member), 30
 cpsapiGlobalQuantity::wrapped (C++ type), 29
 cpsapiGroup (C++ class), 30
 cpsapiGroup::__parameter (C++ function), 33
 cpsapiGroup::~~cpsapiGroup (C++ function), 31
 cpsapiGroup::accept (C++ function), 31
 cpsapiGroup::addGroup (C++ function), 32
 cpsapiGroup::addParameter (C++ function), 31
 cpsapiGroup::base (C++ type), 31
 cpsapiGroup::cpsapiGroup (C++ function), 31
 cpsapiGroup::Data (C++ class), 70
 cpsapiGroup::Data::~~Data (C++ function), 70
 cpsapiGroup::Data::Data (C++ function), 70
 cpsapiGroup::Data::mDefaultParameter (C++ member), 70
 cpsapiGroup::deleteParameter (C++ function), 32
 cpsapiGroup::getDataCN (C++ function), 33, 34
 cpsapiGroup::getParameters (C++ function), 32
 cpsapiGroup::getProperty (C++ function), 32, 33
 cpsapiGroup::HiddenProperties (C++ member), 33
 cpsapiGroup::HiddenReferences (C++ member), 33
 cpsapiGroup::parameter (C++ function), 32
 cpsapiGroup::Property (C++ enum), 31
 cpsapiGroup::Property::CN (C++ enumerator), 31
 cpsapiGroup::Property::NAME (C++ enumerator), 31
 cpsapiGroup::Property::OBJECT_UNIQUE_NAME (C++ enumerator), 31
 cpsapiGroup::Property::PARAMETER_VALUE (C++ enumerator), 31
 cpsapiGroup::Reference (C++ enum), 31
 cpsapiGroup::Reference::NAME (C++ enumerator), 31
 cpsapiGroup::Reference::OBJECT_UNIQUE_NAME (C++ enumerator), 31
 cpsapiGroup::self (C++ type), 31
 cpsapiGroup::setProperty (C++ function), 32, 33
 cpsapiGroup::SupportedProperties (C++ member), 33

cpsapiGroup::SupportedReferences (C++ member), 33
 cpsapiGroup::updateDefaultParameter (C++ function), 33
 cpsapiGroup::wrapped (C++ type), 31
 cpsapiKineticLawVariable (C++ class), 34
 cpsapiKineticLawVariable::~cpsapiKineticLawVariable (C++ function), 35
 cpsapiKineticLawVariable::accept (C++ function), 35
 cpsapiKineticLawVariable::base (C++ type), 34
 cpsapiKineticLawVariable::cpsapiKineticLawVariable (C++ function), 35
 cpsapiKineticLawVariable::getDataCN (C++ function), 35, 36
 cpsapiKineticLawVariable::getProperty (C++ function), 35, 36
 cpsapiKineticLawVariable::HiddenProperties (C++ member), 36
 cpsapiKineticLawVariable::HiddenReferences (C++ member), 36
 cpsapiKineticLawVariable::isValid (C++ function), 35
 cpsapiKineticLawVariable::KineticLawVariable (C++ class), 73
 cpsapiKineticLawVariable::KineticLawVariable::~KineticLawVariable (C++ function), 73
 cpsapiKineticLawVariable::KineticLawVariable::KineticLawVariable (C++ function), 74
 cpsapiKineticLawVariable::KineticLawVariable::KineticLawVariable::KineticLawVariable (C++ function), 73, 74
 cpsapiKineticLawVariable::KineticLawVariable::KineticLawVariable::KineticLawVariable (C++ member), 74
 cpsapiKineticLawVariable::KineticLawVariable::KineticLawVariable::KineticLawVariable (C++ function), 73
 cpsapiKineticLawVariable::Property (C++ enum), 34
 cpsapiKineticLawVariable::Property::CN (C++ enumerator), 34
 cpsapiKineticLawVariable::Property::MAPPING (C++ enumerator), 34
 cpsapiKineticLawVariable::Property::NAME (C++ enumerator), 34
 cpsapiKineticLawVariable::Property::OBJECT_UNIQUE_NAME (C++ enumerator), 34
 cpsapiKineticLawVariable::Property::ROLE (C++ enumerator), 34
 cpsapiKineticLawVariable::Property::VALUE (C++ enumerator), 34
 cpsapiKineticLawVariable::Reference (C++ enum), 34
 cpsapiKineticLawVariable::Reference::NAME (C++ enumerator), 34
 cpsapiKineticLawVariable::Reference::OBJECT_UNIQUE_NAME (C++ enumerator), 34
 cpsapiKineticLawVariable::Reference::VALUE (C++ enumerator), 34
 cpsapiKineticLawVariable::self (C++ type), 34
 cpsapiKineticLawVariable::setProperty (C++ function), 35, 36
 cpsapiKineticLawVariable::SupportedProperties (C++ member), 36
 cpsapiKineticLawVariable::SupportedReferences (C++ member), 36
 cpsapiKineticLawVariable::wrapped (C++ type), 31
 cpsapiModel (C++ class), 37
 cpsapiModel::__compartment (C++ function), 40
 cpsapiModel::__EVENT (C++ function), 40
 cpsapiModel::__globalQuantity (C++ function), 40
 cpsapiModel::__reaction (C++ function), 40
 cpsapiModel::__species (C++ function), 40
 cpsapiModel::~cpsapiModel (C++ function), 38
 cpsapiModel::accept (C++ function), 38
 cpsapiModel::addCompartment (C++ function), 38
 cpsapiModel::addEvent (C++ function), 39
 cpsapiModel::addEventAssignment (C++ function), 39
 cpsapiModel::addGlobalQuantity (C++ function), 39
 cpsapiModel::addReaction (C++ function), 38
 cpsapiModel::addSpecies (C++ function), 38
 cpsapiModel::base (C++ type), 37
 cpsapiModel::cancelTransaction (C++ function), 38
 cpsapiModel::compartment (C++ function), 38
 cpsapiModel::compile (C++ function), 38
 cpsapiModel::cpsapiModel (C++ function), 38
 cpsapiModel::Data (C++ class), 72
 cpsapiModel::Data::~Data (C++ function), 72
 cpsapiModel::Data::Data (C++ function), 72
 cpsapiModel::Data::mDefaultCompartment (C++ member), 72
 cpsapiModel::Data::mDefaultEvent (C++ member), 72
 cpsapiModel::Data::mDefaultGlobalQuantity (C++ member), 72
 cpsapiModel::Data::mDefaultReaction (C++ member), 72
 cpsapiModel::deleteAllDependents (C++ function), 39
 cpsapiModel::deleteCompartment (C++ function), 38
 cpsapiModel::deleteDependents (C++ function), 40
 cpsapiModel::deleteEvent (C++ function), 39
 cpsapiModel::deleteEventAssignment (C++ function), 39
 cpsapiModel::deleteGlobalQuantity (C++ function), 38

cpsapiModel::deleteReaction (C++ function), 38
 cpsapiModel::deleteSpecies (C++ function), 38
 cpsapiModel::endTransaction (C++ function), 38
 cpsapiModel::event (C++ function), 39
 cpsapiModel::eventAssignment (C++ function), 39
 cpsapiModel::getCompartments (C++ function), 38
 cpsapiModel::getEventAssignments (C++ function), 39
 cpsapiModel::getEvents (C++ function), 39
 cpsapiModel::getGlobalQuantities (C++ function), 38
 cpsapiModel::getProperty (C++ function), 39
 cpsapiModel::getReactions (C++ function), 39
 cpsapiModel::getSpecies (C++ function), 38
 cpsapiModel::globalQuantity (C++ function), 38
 cpsapiModel::HiddenProperties (C++ member), 39
 cpsapiModel::Property (C++ enum), 37
 cpsapiModel::Property::AREA_UNIT (C++ enumerator), 37
 cpsapiModel::Property::AVOGADRO_NUMBER (C++ enumerator), 37
 cpsapiModel::Property::CN (C++ enumerator), 37
 cpsapiModel::Property::INITIAL_VALUE (C++ enumerator), 37
 cpsapiModel::Property::LENGTH_UNIT (C++ enumerator), 37
 cpsapiModel::Property::MODEL_TYPE (C++ enumerator), 37
 cpsapiModel::Property::NAME (C++ enumerator), 37
 cpsapiModel::Property::OBJECT_UNIQUE_NAME (C++ enumerator), 37
 cpsapiModel::Property::QUANTITY_UNIT (C++ enumerator), 37
 cpsapiModel::Property::TIME_UNIT (C++ enumerator), 37
 cpsapiModel::Property::UNIT (C++ enumerator), 37
 cpsapiModel::Property::VOLUME_UNIT (C++ enumerator), 37
 cpsapiModel::reaction (C++ function), 38
 cpsapiModel::self (C++ type), 37
 cpsapiModel::setProperty (C++ function), 39
 cpsapiModel::species (C++ function), 38
 cpsapiModel::SupportedProperties (C++ member), 39
 cpsapiModel::synchronize (C++ function), 38
 cpsapiModel::updateDefaultCompartment (C++ function), 40
 cpsapiModel::updateDefaultEvent (C++ function), 40
 cpsapiModel::updateDefaultGlobalQuantity (C++ function), 40
 cpsapiModel::updateDefaultReaction (C++ function), 40
 cpsapiModel::wrapped (C++ type), 38
 cpsapiModelEntity (C++ class), 40
 cpsapiModelEntity::~cpsapiModelEntity (C++ function), 41
 cpsapiModelEntity::accept (C++ function), 41
 cpsapiModelEntity::base (C++ type), 41
 cpsapiModelEntity::cpsapiModelEntity (C++ function), 41, 42
 cpsapiModelEntity::getProperty (C++ function), 41, 42
 cpsapiModelEntity::Property (C++ enum), 40
 cpsapiModelEntity::Property::ADD_NOISE (C++ enumerator), 41
 cpsapiModelEntity::Property::CN (C++ enumerator), 41
 cpsapiModelEntity::Property::EXPRESSION (C++ enumerator), 40
 cpsapiModelEntity::Property::INITIAL_EXPRESSION (C++ enumerator), 40
 cpsapiModelEntity::Property::INITIAL_VALUE (C++ enumerator), 40
 cpsapiModelEntity::Property::NAME (C++ enumerator), 41
 cpsapiModelEntity::Property::NOISE_EXPRESSION (C++ enumerator), 41
 cpsapiModelEntity::Property::OBJECT_UNIQUE_NAME (C++ enumerator), 41
 cpsapiModelEntity::Property::SIMULATION_TYPE (C++ enumerator), 40
 cpsapiModelEntity::self (C++ type), 41
 cpsapiModelEntity::setProperty (C++ function), 41, 42
 cpsapiModelEntity::SupportedProperties (C++ member), 41
 cpsapiObject (C++ class), 42
 cpsapiObject::~cpsapiObject (C++ function), 43
 cpsapiObject::accept (C++ function), 44
 cpsapiObject::cpsapiObject (C++ function), 43, 47
 cpsapiObject::Data (C++ type), 43
 cpsapiObject::getDataCN (C++ function), 45, 47
 cpsapiObject::getObject (C++ function), 47
 cpsapiObject::getProperty (C++ function), 45, 47
 cpsapiObject::getType (C++ function), 44
 cpsapiObject::HiddenProperties (C++ member), 46
 cpsapiObject::HiddenReferences (C++ member), 46
 cpsapiObject::Invalid (C++ member), 46
 cpsapiObject::isHiddenProperty (C++ function), 48
 cpsapiObject::isHiddenReference (C++ function), 48

cpsapiObject::isImplementedProperty (C++ function), 48
 cpsapiObject::isImplementedReference (C++ function), 48
 cpsapiObject::isSupportedProperty (C++ function), 46
 cpsapiObject::isSupportedReference (C++ function), 46
 cpsapiObject::isValid (C++ function), 44
 cpsapiObject::mpData (C++ member), 48
 cpsapiObject::operator bool (C++ function), 44
 cpsapiObject::operator!= (C++ function), 44
 cpsapiObject::operator* (C++ function), 43
 cpsapiObject::operator= (C++ function), 43
 cpsapiObject::operator== (C++ function), 43
 cpsapiObject::operator-> (C++ function), 43
 cpsapiObject::Properties (C++ type), 43
 cpsapiObject::Property (C++ enum), 42
 cpsapiObject::Property::CN (C++ enumerator), 42
 cpsapiObject::Property::NAME (C++ enumerator), 42
 cpsapiObject::Property::OBJECT_UNIQUE_NAME (C++ enumerator), 42
 cpsapiObject::Reference (C++ enum), 42
 cpsapiObject::Reference::NAME (C++ enumerator), 43
 cpsapiObject::Reference::OBJECT_UNIQUE_NAME (C++ enumerator), 43
 cpsapiObject::References (C++ type), 43
 cpsapiObject::self (C++ type), 43
 cpsapiObject::setProperty (C++ function), 44, 47
 cpsapiObject::supportedProperties (C++ function), 45, 46
 cpsapiObject::SupportedProperties (C++ member), 46
 cpsapiObject::supportedReferences (C++ function), 45, 46
 cpsapiObject::SupportedReferences (C++ member), 46
 cpsapiObjectData (C++ class), 48
 cpsapiObjectData::~cpsapiObjectData (C++ function), 50
 cpsapiObjectData::assertDataType (C++ function), 50
 cpsapiObjectData::cpsapiObjectData (C++ function), 50, 51
 cpsapiObjectData::deleted (C++ function), 50
 cpsapiObjectData::erase (C++ function), 50
 cpsapiObjectData::Manager (C++ member), 50
 cpsapiObjectData::Map (C++ type), 50
 cpsapiObjectData::mpObject (C++ member), 50
 cpsapiObjectData::mType (C++ member), 50
 cpsapiObjectData::Pointer (C++ type), 50
 cpsapiObjectData::release (C++ function), 50
 cpsapiObjectData::Type (C++ enum), 49
 cpsapiObjectData::Type::__SIZE (C++ enumerator), 49
 cpsapiObjectData::Type::Compartment (C++ enumerator), 49
 cpsapiObjectData::Type::Container (C++ enumerator), 49
 cpsapiObjectData::Type::DataModel (C++ enumerator), 49
 cpsapiObjectData::Type::Event (C++ enumerator), 49
 cpsapiObjectData::Type::EventAssignment (C++ enumerator), 49
 cpsapiObjectData::Type::GlobalQuantity (C++ enumerator), 49
 cpsapiObjectData::Type::Group (C++ enumerator), 49
 cpsapiObjectData::Type::Method (C++ enumerator), 49
 cpsapiObjectData::Type::Model (C++ enumerator), 49
 cpsapiObjectData::Type::ModelEntity (C++ enumerator), 49
 cpsapiObjectData::Type::Object (C++ enumerator), 49
 cpsapiObjectData::Type::Parameter (C++ enumerator), 49
 cpsapiObjectData::Type::Problem (C++ enumerator), 49
 cpsapiObjectData::Type::Reaction (C++ enumerator), 49
 cpsapiObjectData::Type::ReactionParameter (C++ enumerator), 49
 cpsapiObjectData::Type::Species (C++ enumerator), 49
 cpsapiObjectData::Type::Task (C++ enumerator), 49
 cpsapiObjectData::Type::Value (C++ enumerator), 49
 cpsapiObjectData::Type::Vector (C++ enumerator), 49
 cpsapiObjectData::TypeName (C++ member), 50
 cpsapiParameter (C++ class), 51
 cpsapiParameter::~cpsapiParameter (C++ function), 52
 cpsapiParameter::accept (C++ function), 52
 cpsapiParameter::base (C++ type), 51
 cpsapiParameter::cpsapiParameter (C++ function), 52
 cpsapiParameter::getDataCN (C++ function), 52, 53
 cpsapiParameter::getProperty (C++ function), 52, 53
 cpsapiParameter::HiddenProperties (C++ member), 53

cpsapiParameter::HiddenReferences (C++ member), 53
 cpsapiParameter::Property (C++ enum), 51
 cpsapiParameter::Property::CN (C++ enumerator), 51
 cpsapiParameter::Property::NAME (C++ enumerator), 51
 cpsapiParameter::Property::OBJECT_UNIQUE_NAME (C++ enumerator), 51
 cpsapiParameter::Property::PARAMETER_VALUE (C++ enumerator), 51
 cpsapiParameter::Reference (C++ enum), 51
 cpsapiParameter::Reference::NAME (C++ enumerator), 51
 cpsapiParameter::Reference::OBJECT_UNIQUE_NAME (C++ enumerator), 51
 cpsapiParameter::Reference::PARAMETER_VALUE (C++ enumerator), 51
 cpsapiParameter::self (C++ type), 51
 cpsapiParameter::setProperty (C++ function), 52, 53
 cpsapiParameter::SupportedProperties (C++ member), 53
 cpsapiParameter::SupportedReferences (C++ member), 53
 cpsapiParameter::wrapped (C++ type), 51
 cpsapiProperty (C++ class), 53
 cpsapiProperty::Name (C++ member), 58
 cpsapiProperty::Type (C++ enum), 54
 cpsapiProperty::Type::__SIZE (C++ enumerator), 58
 cpsapiProperty::Type::ADD_NOISE (C++ enumerator), 54
 cpsapiProperty::Type::AREA_UNIT (C++ enumerator), 56
 cpsapiProperty::Type::ARRAY_ELEMENT_INDEX (C++ enumerator), 57
 cpsapiProperty::Type::ASSIGNMENTS (C++ enumerator), 58
 cpsapiProperty::Type::AVOGADRO_NUMBER (C++ enumerator), 57
 cpsapiProperty::Type::CHEMICAL_EQUATION (C++ enumerator), 54
 cpsapiProperty::Type::CN (C++ enumerator), 55
 cpsapiProperty::Type::DATE (C++ enumerator), 57
 cpsapiProperty::Type::DELAY_ASSIGNMENT (C++ enumerator), 58
 cpsapiProperty::Type::DELAY_EXPRESSION (C++ enumerator), 58
 cpsapiProperty::Type::DIMENSIONALITY (C++ enumerator), 57
 cpsapiProperty::Type::EMAIL (C++ enumerator), 57
 cpsapiProperty::Type::EVALUATION_TREE_TYPE (C++ enumerator), 55
 cpsapiProperty::Type::EXPRESSION (C++ enumerator), 54
 cpsapiProperty::Type::FAMILY_NAME (C++ enumerator), 57
 cpsapiProperty::Type::FIRE_AT_INITIALTIME (C++ enumerator), 58
 cpsapiProperty::Type::FLUX (C++ enumerator), 55
 cpsapiProperty::Type::FRAMEWORK (C++ enumerator), 58
 cpsapiProperty::Type::GIVEN_NAME (C++ enumerator), 57
 cpsapiProperty::Type::INITIAL_EXPRESSION (C++ enumerator), 54
 cpsapiProperty::Type::INITIAL_FLUX (C++ enumerator), 55
 cpsapiProperty::Type::INITIAL_INTENSIVE_RATE (C++ enumerator), 54
 cpsapiProperty::Type::INITIAL_INTENSIVE_VALUE (C++ enumerator), 54
 cpsapiProperty::Type::INITIAL_PARTICLE_FLUX (C++ enumerator), 55
 cpsapiProperty::Type::INITIAL_RATE (C++ enumerator), 54
 cpsapiProperty::Type::INITIAL_VALUE (C++ enumerator), 54
 cpsapiProperty::Type::INTENSIVE_RATE (C++ enumerator), 54
 cpsapiProperty::Type::INTENSIVE_VALUE (C++ enumerator), 54
 cpsapiProperty::Type::KINETIC_LAW (C++ enumerator), 54
 cpsapiProperty::Type::KINETIC_LAW_EXPRESSION (C++ enumerator), 54
 cpsapiProperty::Type::KINETIC_LAW_UNIT_TYPE (C++ enumerator), 54
 cpsapiProperty::Type::KINETIC_LAW_VARIABLE_MAPPING (C++ enumerator), 54
 cpsapiProperty::Type::LENGTH_UNIT (C++ enumerator), 56
 cpsapiProperty::Type::LOCAL_REACTION_PARAMETERS (C++ enumerator), 54
 cpsapiProperty::Type::METHOD (C++ enumerator), 56
 cpsapiProperty::Type::METHOD_TYPE (C++ enumerator), 56
 cpsapiProperty::Type::MIRIAM_DESCRIPTION (C++ enumerator), 57
 cpsapiProperty::Type::MIRIAM_ID (C++ enumerator), 57
 cpsapiProperty::Type::MIRIAM_PREDICATE (C++ enumerator), 57
 cpsapiProperty::Type::MIRIAM_RDF_XML (C++ enumerator), 57

cpsapiProperty::Type::MIRIAM_RESOURCE (C++
 enumerator), 57
 cpsapiProperty::Type::MODEL_TYPE (C++ enumerator), 57
 cpsapiProperty::Type::NAME (C++ enumerator), 55
 cpsapiProperty::Type::NOISE (C++ enumerator), 58
 cpsapiProperty::Type::NOISE_EXPRESSION (C++
 enumerator), 54
 cpsapiProperty::Type::NOTES (C++ enumerator), 57
 cpsapiProperty::Type::OBJECT_FLAG (C++ enumerator), 55
 cpsapiProperty::Type::OBJECT_HASH (C++ enumerator), 55
 cpsapiProperty::Type::OBJECT_INDEX (C++ enumerator), 55
 cpsapiProperty::Type::OBJECT_PARENT_CN (C++
 enumerator), 55
 cpsapiProperty::Type::OBJECT_POINTER (C++
 enumerator), 55
 cpsapiProperty::Type::OBJECT_REFERENCE (C++
 enumerator), 55
 cpsapiProperty::Type::OBJECT_REFERENCE_CN
 (C++ enumerator), 55
 cpsapiProperty::Type::OBJECT_REFERENCE_INDEX
 (C++ enumerator), 55
 cpsapiProperty::Type::OBJECT_REFERENCES
 (C++ enumerator), 55
 cpsapiProperty::Type::OBJECT_TYPE (C++ enumerator), 55
 cpsapiProperty::Type::OBJECT_UNIQUE_NAME
 (C++ enumerator), 55
 cpsapiProperty::Type::OBJECT_UUID (C++ enumerator), 55
 cpsapiProperty::Type::ORGANIZATION (C++ enumerator), 57
 cpsapiProperty::Type::PARAMETER (C++ enumerator), 56
 cpsapiProperty::Type::PARAMETER_MAPPING
 (C++ enumerator), 56
 cpsapiProperty::Type::PARAMETER_ROLE (C++
 enumerator), 56
 cpsapiProperty::Type::PARAMETER_TYPE (C++
 enumerator), 56
 cpsapiProperty::Type::PARAMETER_USED (C++
 enumerator), 56
 cpsapiProperty::Type::PARAMETER_VALUE (C++
 enumerator), 56
 cpsapiProperty::Type::PARTICLE_FLUX (C++ enumerator), 55
 cpsapiProperty::Type::PERSISTENT_TRIGGER
 (C++ enumerator), 58
 cpsapiProperty::Type::PLOT_ITEM_TYPE (C++
 enumerator), 56
 cpsapiProperty::Type::PLOT_TYPE (C++ enumerator), 56
 cpsapiProperty::Type::PRIORITY_EXPRESSION
 (C++ enumerator), 58
 cpsapiProperty::Type::PROBLEM (C++ enumerator), 56
 cpsapiProperty::Type::PROPENSITY (C++ enumerator), 58
 cpsapiProperty::Type::QUANTITY_UNIT (C++ enumerator), 57
 cpsapiProperty::Type::RATE (C++ enumerator), 54
 cpsapiProperty::Type::REPORT_IS_TABLE (C++
 enumerator), 57
 cpsapiProperty::Type::REPORT_PRECISION (C++
 enumerator), 57
 cpsapiProperty::Type::REPORT_SEPARATOR (C++
 enumerator), 57
 cpsapiProperty::Type::REPORT_SHOW_TITLE
 (C++ enumerator), 57
 cpsapiProperty::Type::SCALING_COMPARTMENT
 (C++ enumerator), 55
 cpsapiProperty::Type::SIMULATION_TYPE (C++
 enumerator), 54
 cpsapiProperty::Type::SPATIAL_DIMENSION
 (C++ enumerator), 54
 cpsapiProperty::Type::TASK_REPORT (C++ enumerator), 56
 cpsapiProperty::Type::TASK_REPORT_APPEND
 (C++ enumerator), 56
 cpsapiProperty::Type::TASK_REPORT_CONFIRM_OVERWRITE
 (C++ enumerator), 56
 cpsapiProperty::Type::TASK_REPORT_TARGET
 (C++ enumerator), 56
 cpsapiProperty::Type::TASK_SCHEDULED (C++
 enumerator), 56
 cpsapiProperty::Type::TASK_TYPE (C++ enumerator), 55
 cpsapiProperty::Type::TASK_UPDATE_MODEL
 (C++ enumerator), 56
 cpsapiProperty::Type::TIME_UNIT (C++ enumerator), 57
 cpsapiProperty::Type::TRIGGER_EXPRESSION
 (C++ enumerator), 58
 cpsapiProperty::Type::UNIT (C++ enumerator), 56
 cpsapiProperty::Type::UNIT_EXPRESSION (C++
 enumerator), 58
 cpsapiProperty::Type::UNIT_SYMBOL (C++ enumerator), 58
 cpsapiProperty::Type::VALUE (C++ enumerator), 54
 cpsapiProperty::Type::VECTOR_CONTENT (C++
 enumerator), 58
 cpsapiProperty::Type::VOLUME_UNIT (C++ enu-

merator), 56
 cpsapiReaction (C++ class), 58
 cpsapiReaction::~cpsapiReaction (C++ function), 60
 cpsapiReaction::accept (C++ function), 60
 cpsapiReaction::assertVariable (C++ function), 62
 cpsapiReaction::base (C++ type), 60
 cpsapiReaction::cpsapiReaction (C++ function), 60
 cpsapiReaction::Data (C++ class), 71
 cpsapiReaction::Data::~Data (C++ function), 71
 cpsapiReaction::Data::Data (C++ function), 71
 cpsapiReaction::Data::mDefaultVariable (C++ member), 71
 cpsapiReaction::Data::mpVector (C++ member), 71
 cpsapiReaction::Data::mVariableManager (C++ member), 71
 cpsapiReaction::getDataCN (C++ function), 61, 62
 cpsapiReaction::getProperty (C++ function), 61, 62
 cpsapiReaction::HiddenProperties (C++ member), 61
 cpsapiReaction::HiddenReferences (C++ member), 61
 cpsapiReaction::Property (C++ enum), 59
 cpsapiReaction::Property::ADD_NOISE (C++ enumerator), 59
 cpsapiReaction::Property::CHEMICAL_EQUATION (C++ enumerator), 59
 cpsapiReaction::Property::CN (C++ enumerator), 59
 cpsapiReaction::Property::KINETIC_LAW (C++ enumerator), 59
 cpsapiReaction::Property::KINETIC_LAW_EXPRESSION (C++ enumerator), 59
 cpsapiReaction::Property::KINETIC_LAW_UNIT_TYPE (C++ enumerator), 59
 cpsapiReaction::Property::NAME (C++ enumerator), 59
 cpsapiReaction::Property::NOISE_EXPRESSION (C++ enumerator), 59
 cpsapiReaction::Property::OBJECT_UNIQUE_NAME (C++ enumerator), 59
 cpsapiReaction::Property::SCALING_COMPARTMENT (C++ enumerator), 59
 cpsapiReaction::Reference (C++ enum), 59
 cpsapiReaction::Reference::FLUX (C++ enumerator), 59
 cpsapiReaction::Reference::INITIAL_FLUX (C++ enumerator), 59
 cpsapiReaction::Reference::INITIAL_PARTICLE_FLUX (C++ enumerator), 59
 cpsapiReaction::Reference::NAME (C++ enumerator), 59
 cpsapiReaction::Reference::NOISE (C++ enumerator), 59
 cpsapiReaction::Reference::OBJECT_UNIQUE_NAME (C++ enumerator), 59
 cpsapiReaction::Reference::PARTICLE_FLUX (C++ enumerator), 59
 cpsapiReaction::Reference::PROPENSITY (C++ enumerator), 60
 cpsapiReaction::self (C++ type), 60
 cpsapiReaction::setProperty (C++ function), 60, 62
 cpsapiReaction::SupportedProperties (C++ member), 61
 cpsapiReaction::SupportedReferences (C++ member), 61
 cpsapiReaction::variable (C++ function), 60
 cpsapiReaction::VariableManager (C++ type), 60
 cpsapiReaction::variables (C++ function), 60
 cpsapiReaction::VariableVector (C++ type), 60
 cpsapiReaction::wrapped (C++ type), 60
 cpsapiReference (C++ type), 77
 cpsapiSpecies (C++ class), 62
 cpsapiSpecies::~cpsapiSpecies (C++ function), 63
 cpsapiSpecies::accept (C++ function), 64
 cpsapiSpecies::base (C++ type), 63
 cpsapiSpecies::cpsapiSpecies (C++ function), 63
 cpsapiSpecies::getProperty (C++ function), 64
 cpsapiSpecies::Property (C++ enum), 63
 cpsapiSpecies::Property::ADD_NOISE (C++ enumerator), 63
 cpsapiSpecies::Property::CN (C++ enumerator), 63
 cpsapiSpecies::Property::EXPRESSION (C++ enumerator), 63
 cpsapiSpecies::Property::INITIAL_EXPRESSION (C++ enumerator), 63
 cpsapiSpecies::Property::INITIAL_VALUE (C++ enumerator), 63
 cpsapiSpecies::Property::NAME (C++ enumerator), 63
 cpsapiSpecies::Property::NOISE_EXPRESSION (C++ enumerator), 63
 cpsapiSpecies::Property::OBJECT_UNIQUE_NAME (C++ enumerator), 63
 cpsapiSpecies::Property::SIMULATION_TYPE (C++ enumerator), 63
 cpsapiSpecies::Property::UNIT (C++ enumerator), 63
 cpsapiSpecies::self (C++ type), 63
 cpsapiSpecies::setProperty (C++ function), 64
 cpsapiSpecies::SupportedProperties (C++ member), 64

cpsapiSpecies::wrapped (C++ type), 63
 cpsapiTransaction (C++ class), 64
 cpsapiTransaction::beginStructureChange (C++ function), 65
 cpsapiTransaction::beginTransaction (C++ function), 65
 cpsapiTransaction::endStructureChange (C++ function), 65
 cpsapiTransaction::endTransaction (C++ function), 65
 cpsapiTransaction::Map (C++ type), 65
 cpsapiTransaction::MapStructureChange (C++ type), 65
 cpsapiTransaction::sChangeInfo (C++ struct), 75
 cpsapiTransaction::sChangeInfo::changed (C++ member), 75
 cpsapiTransaction::sChangeInfo::framework (C++ member), 75
 cpsapiTransaction::StructureChange (C++ member), 65
 cpsapiTransaction::synchronize (C++ function), 65
 cpsapiTransaction::Transactions (C++ member), 65
 cpsapiValue (C++ class), 65
 cpsapiValue::~cpsapiValue (C++ function), 66
 cpsapiValue::accept (C++ function), 66
 cpsapiValue::base (C++ type), 66
 cpsapiValue::cpsapiValue (C++ function), 66
 cpsapiValue::getDataCN (C++ function), 67, 68
 cpsapiValue::getProperty (C++ function), 67, 68
 cpsapiValue::HiddenProperties (C++ member), 68
 cpsapiValue::HiddenReferences (C++ member), 68
 cpsapiValue::operator cpsapiData (C++ function), 67
 cpsapiValue::Property (C++ enum), 66
 cpsapiValue::Property::CN (C++ enumerator), 66
 cpsapiValue::Property::NAME (C++ enumerator), 66
 cpsapiValue::Property::OBJECT_UNIQUE_NAME (C++ enumerator), 66
 cpsapiValue::Property::VALUE (C++ enumerator), 66
 cpsapiValue::Reference (C++ enum), 66
 cpsapiValue::Reference::NAME (C++ enumerator), 66
 cpsapiValue::Reference::OBJECT_UNIQUE_NAME (C++ enumerator), 66
 cpsapiValue::Reference::VALUE (C++ enumerator), 66
 cpsapiValue::self (C++ type), 66
 cpsapiValue::setProperty (C++ function), 67, 68
 cpsapiValue::SupportedProperties (C++ member), 68
 cpsapiValue::SupportedReferences (C++ member), 68
 cpsapiValue::valid (C++ function), 67
 cpsapiValue::wrapped (C++ type), 66
 cpsapiVector (C++ class), 68
 cpsapiVector::~cpsapiVector (C++ function), 69
 cpsapiVector::accept (C++ function), 69
 cpsapiVector::base (C++ type), 69
 cpsapiVector::begin (C++ function), 69
 cpsapiVector::cpsapiVector (C++ function), 69
 cpsapiVector::Data (C++ class), 71
 cpsapiVector::Data::~Data (C++ function), 71
 cpsapiVector::Data::Data (C++ function), 71
 cpsapiVector::Data::mMap (C++ member), 71
 cpsapiVector::end (C++ function), 69
 cpsapiVector::index (C++ function), 69
 cpsapiVector::iterator (C++ class), 72
 cpsapiVector::iterator::~iterator (C++ function), 73
 cpsapiVector::iterator::iterator (C++ function), 73
 cpsapiVector::iterator::mpMap (C++ member), 73
 cpsapiVector::iterator::operator Object* (C++ function), 73
 cpsapiVector::iterator::operator* (C++ function), 73
 cpsapiVector::iterator::operator+ (C++ function), 73
 cpsapiVector::iterator::operator++ (C++ function), 73
 cpsapiVector::iterator::operator+= (C++ function), 73
 cpsapiVector::iterator::operator- (C++ function), 73
 cpsapiVector::iterator::operator-= (C++ function), 73
 cpsapiVector::iterator::operator-- (C++ function), 73
 cpsapiVector::iterator::operator-> (C++ function), 73
 cpsapiVector::ObjectMap (C++ type), 69
 cpsapiVector::operator[] (C++ function), 69
 cpsapiVector::Property (C++ enum), 69
 cpsapiVector::self (C++ type), 69
 cpsapiVector::size (C++ function), 69
 cpsapiVector::SupportedProperties (C++ member), 70
 cpsapiVector::wrapped (C++ type), 69
 cpsapiVisitor (C++ struct), 70
 cpsapiVisitor::accept (C++ function), 70
 cpsapiVisitor::acceptIfVisitable (C++ function), 70
 cpsapiVisitor::doVisit (C++ function), 70

`cpsapiVisitor::VisitorImplementation` (C++
 struct), [75](#)
`cpsapiVisitor::VisitorImplementation::~~VisitorImplementation`
 (C++ *function*), [76](#)
`cpsapiVisitor::VisitorImplementation::mVisitor`
 (C++ *member*), [76](#)
`cpsapiVisitor::VisitorImplementation::visit`
 (C++ *function*), [76](#)
`cpsapiVisitor::VisitorImplementation::VisitorImplementation`
 (C++ *function*), [76](#)
`cpsapiVisitor::VisitorInterface` (C++ *struct*),
 [76](#)
`cpsapiVisitor::VisitorInterface::~~VisitorInterface`
 (C++ *function*), [76](#)
`cpsapiVisitor::VisitorInterface::visit` (C++
 function), [76](#)

D

`DATA` (C *macro*), [77](#)

W

`WRAPPED` (C *macro*), [77](#)