COPASI API

COPASI Team

Jun 10, 2022

CONTENTS:

CHAPTER

ONE

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 documenation 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building [mo]: targets for 0 po files that are out of date
building [html]: targets for 3 source files that are out of date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looking for now-outdated files...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 configuratioon, 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)

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

CHAPTER

TWO

EXAMPLES

CHAPTER

THREE

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* &parameter)

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 cspapiData

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 ***getObject**() 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* “cpsapi/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* “cpsapi/model/cpsapiGlobalQuantity.h”*#include* “cpsapi/model/cpsapiReaction.h”*#include* “cpsapi/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* “cpsapi/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

CHAPTER

FOUR

INDICES AND TABLES

- genindex
- modindex
- search

INDEX

C

D

W