Roccom Developers Guide

Version 3

Xiangmin Jiao, Gengbin Zheng

July 22, 2008

Contents

1 Overview

Roccom (Roc∗ Component Object Manager) is a mechanism for inter-component data and function access in a parallel simulation code, designed for easing the integration of application codes among themselves and with other software tools.

In the Roccom framework, there are three types of components (modules): application modules, service modules, and Roccom runtime systems. An application module usually is a physical component containing specific physical models and typically is in Fortran 90. A service module is a computer science module that provides some specific services to application modules, such as IO or data transfer. Typically, a services module is written in C++. A Roccom runtime system is the network among the aforementioned two types of modules.

This documentation is targeted to developers of service modules and implementors of Roccom runtime systems. It explains the data organization of Roccom, its programming interface for service modules, and its implementation requirements. The reader is assumed to be familiar with Roccom User's Guide and C++.

2 Roccom Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

3 Roccom Namespace Documentation

3.1 COM Namespace Reference

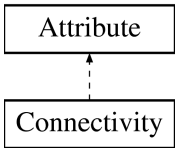
The name space for Roccom.

4 Roccom Class Documentation

4.1 Attribute Class Reference

An Attribute object is a data member of a window.

Inheritance diagram for Attribute::



Constructors and destructors

- Attribute ()

  Default constructor.

- Attribute (Pane *pane, const std::string &name, int id, Shorter_size loc, int type, const int ncomp, const std::string &unit)

  Create an attribute with name n in window w.

- Attribute (Pane *pane, Attribute *parent, const std::string &name, int id)

  Inherit an attribute from another.

- ~Attribute ()

  Destructors.

- Attribute (Pane *pane, int i)

  Constructor for keywords. The default nitems for keywords is 0.

Access methods

- const Pane * pane () const

  Obtain a constant pointer to the owner pane of the attribute.

- Pane * pane ()

  Obtain a modifiable pointer to the owner pane of the attribute.

- const Window * window () const

  Obtain a constant pointer to the parent window of the attribute.

- Window * window ()

  Obtain a modifiable pointer to the parent window of the attribute.

- Shorter_size location () const

  Obtain the location of the attribute.

- bool is_windowed () const

  Checks whether the attribute is associated with the window.

- bool is_panel () const

  Checks whether the attribute is associated with a pane.

- bool is_elemental () const

  Checks whether the attribute is associated with an element.

- bool is_nodal () const

  Checks whether the attribute is associated with a node.

- COM_Type data_type () const

  Obtain the data type of each component of the attribute.

- const std::string & unit () const

  Obtain the unit of the attribute.

- int size_of_components () const

  Obtain the number of components in the attribute.

- int size_of_items () const

  Obtain the number of items in the attribute.

- int maxsize_of_items () const

  Obtain the maximum allowed number of items in the attribute.

- int size_of_ghost_items () const

  Obtain the number of ghost items in the attribute.

- int maxsize_of_ghost_items () const

  Obtain the maximum allowed number of items in the attribute.

- int size_of_real_items () const

  Obtain the number of real items in the attribute.

- int maxsize_of_real_items () const

  Obtain the maximum allowed number of real items in the attribute.

- bool empty () const

  Check whether the number of items of the attribute is zero.

- int capacity () const

  Obtain the capacity of the array.

- int stride () const

  Obtain the stride of the attribute in base datatype.

- int stride_in_bytes () const

  Obtain the stride of the attribute in bytes.

- int status () const

  Obtain the status of the attribute.

- bool initialized () const

  Returns whether the array for the attribute has been set or allocated.

- bool size_set () const

  Returns whether the size for the attribute has been set.

- bool allocated () const

  Returns whether the array for the attribute has been set or allocated.

- bool is_const () const

  Returns whether the array is set to be read-only.

- bool is_staggered () const

Check how the attribute values are organized.

- static int **get_sizeof** (COM_Type type, int count=1)
- static bool **compatible_types** (COM_Type t1, COM_Type t2)
- static bool **is_digit** (char c)

Public Types

- enum **Copy_dir** { **COPY_IN**, **COPY_OUT** }
- typedef unsigned char Shorter_size

  One byte unsighed int.

- typedef unsigned int Size

  Unsighed int.

Public Member Functions

- void set_size (int nitems, int ngitems=0) throw (COM_exception)

  Set the size of items and ghost items.

- void * allocate (int strd, int cap, bool force) throw (COM_exception)

  Allocate memory for the attribute.

- int deallocate () throw (COM_exception)

  Deallocate memory if it was allocated by allocate().

- void **copy_array** (void *buf, int strd, int nitem, int offset=0, int direction=COPY_IN) throw (COM_exception)
- void **append_array** (const void *from, int strd, int nitem) throw (COM_-exception)

Identity

- const std::string & name () const

  Obtain the name of the attribute.

- std::string fullname () const
  Obtain the full name of the attribute including window name suitable for printing out error messages.
- int id () const
  Obtain the id (or index) of the attribute.
- Attribute * parent ()
  Parent attribute used by this object.
- const Attribute * **parent** () const
- Attribute * root ()
  Root of use-inheritance.
- const Attribute * **root** () const

Physical address

- const void * pointer () const
  Obtain a constant pointer to the physical address.
- void * pointer () throw (COM_exception)
  Obtain a modifiable pointer to the physical address.
- const void * get_addr (int i, int j=0) const throw (COM_exception)
  Obtain the address of the jth component of the ith item, where 0<=i<size_of_items.
- void * **get_addr** (int i, int j=0) throw (COM_exception)

Protected Types

- enum {
  STATUS_NOT_INITIALIZED = 0, **STATUS_SET** = 1, **STATUS_SET_CONST** = 2, **STATUS_USE** = 3,
  STATUS_ALLOCATED = 4 }

Protected Member Functions

- void set_pointer (void *p, int strd, int cap, int offset, bool is_const) throw (COM_exception)

  Set the physical address of the attribute values.

- void inherit (Attribute *a, bool clone, bool withghost, int depth=0) throw (COM_exception)

  Inherit from parent. If depth>0, then the procedure is for the subcomponents.

Protected Attributes

- Pane * _pane

  Pointer to its owner pane.

- Attribute * _parent

  Parent attribute being used.

- std::string _name

  Name of the attribute.

- int _id

  Id field data.

- Shorter_size _loc

  Location.

- int _ncomp

  Number of components.

- COM_Type _type

  Base data type of the attribute.

- std::string _unit

  Unit of the attribute.

- int _nitems

Size of total items. Default value is -1.

- int _ngitems
  Size of ghost items.
- int _gap
  Gap between the IDs of real and ghost items.
- Shorter_size _status
  Indicating whether it has been initialized.
- void * _ptr
  Physical address of the attribute.
- int _strd
  Stride.
- int _nbytes_strd
  Number of bytes of the stride.
- int _cap
  Capacity.

Static Protected Attributes

Keywords

- static const char * _keywords [COM_NUM_KEYWORDS]
  List of keywords.
- static const char _keylocs [COM_NUM_KEYWORDS]
  Default locations.
- static const COM_Type _keytypes [COM_NUM_KEYWORDS]
  Default data types.
- static const int _keysizes [COM_NUM_KEYWORDS]
  Default sizes.

4.1.1 Detailed Description

It can be associated with a window, a pane, nodes, or elements. An attribute can be a vector of length size_of_items() with size_of_components() components.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 Attribute::Attribute (Pane * *pane*, const std::string & *name*, int *id*, Shorter_size *loc*, int *type*, const int *ncomp*, const std::string & *unit*) [inline]

Parameters:

pane pointer to its owner pane object.

name attribute name.

id attribute index.

loc location ('w', 'p', 'n', or 'e').

type base data type.

ncomp number of components.

unit unit of the attribute.

4.1.2.2 Attribute::Attribute (Pane * *pane*, Attribute * *parent*, const std::string & *name*, int *id*)

Parameters:

pane pointer to its owner pane object.

parent parent attribute (for supporting inheritance).

name attribute name.

id attribute index.

4.1.3 Member Function Documentation

4.1.3.1 std::string Attribute::fullname () const

4.1.3.2 Attribute* Attribute::parent () [inline]

It determines the the meta-data of the attribute. If the array of this attribute is not set, then this attribute also uses the pointer of its parant.

Reimplemented in Connectivity.

4.1.3.3 Attribute* Attribute::root () [inline]

The basic properties of the attribute (location, data type, number of components, and unit) are copied from the root during inheritance and hence later changes in the root (by renewing the root) will not be reflected in sub-attribute. Other types of information (sizes, pointer, stride, and capacity) always use those of the root attribute.

Reimplemented in Connectivity.

4.1.3.4 const void * Attribute::get_addr (int *i*, int *j* = 0) const throw (COM_exception)

This function is recursive and relatively expensive, and hence should be used only for performance-insenstive tasks.

Reimplemented in Connectivity.

4.1.3.5 Shorter_size Attribute::location () const [inline]

It is encoded as follows: 'w' for windowed attribute, 'p' for panal attribute, 'n' for nodal attribute, and 'e' for elemental attribute.

4.1.3.6 int Attribute::maxsize_of_items () const

Reserved for Roccom3.1

4.1.3.7 int Attribute::maxsize_of_ghost_items () const

Reserved for Roccom3.1

4.1.3.8 int Attribute::maxsize_of_real_items () const

Reserved for Roccom3.1

4.1.3.9 bool Attribute::is_staggered () const [inline]

It returns true if the components of the attribute associated with

4.1.3.10 void Attribute::set_size (int *nitems*, int *ngitems* = 0) throw (COM_exception)

Can be changed only if the attribute is a root.

Reimplemented in Connectivity.

4.1.3.11 void * Attribute::allocate (int *strd*, int *cap*, bool *force*) throw (COM_exception)

The attribute must be a root if the attribute is to be allocated. If from is not NULL, copy its value to the newly allocated array.

Reimplemented in Connectivity.

4.1.3.12 int Attribute::deallocate () throw (COM_exception)

Return 0 if deallocation is successful.

4.1.4 Member Data Documentation

4.1.4.1 Shorter_size Attribute::_loc [protected]

'w' for windowed attribute, 'p' for panal attribute, 'n' for nodal attribute, 'e' for elemental attribute

4.1.4.2 const char * Attribute::_keywords [static, protected]

Initial value:

```
{ "nc", "1-nc", "2-nc", "3-nc", "conn", "mesh",
  "pconn", "ridges", "1-ridges", "2-ridges", "pmesh", "atts", "all" }
```

The names of the keywords.

4.1.4.3 const char Attribute::_keylocs [static, protected]

Initial value:

```
{ 'n', 'n', 'n', 'n', 'e', 'p', 'p', 'p', 'p', 'p', 'p', 'p', 'p'}
```

The locations of keywords.

4.1.4.4 const COM_Type Attribute::_keytypes [static, protected]

Initial value:

```
{ COM_DOUBLE, COM_DOUBLE, COM_DOUBLE, COM_DOUBLE, COM_METADATA,
  COM_METADATA, COM_INT, COM_INT, COM_INT, COM_INT, COM_METADATA,
  COM_METADATA, COM_METADATA }
```

The datatypes of keywords.

4.1.4.5 const COM_Type Attribute::_keysizes [static, protected]

Initial value:

```
{ 3, 1, 1, 1, 0, 0, 1, 2, 1, 1, 0, 0, 0 }
```

The sizes of keywords.

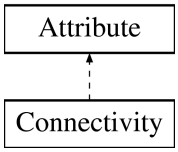
The documentation for this class was generated from the following files:

- Attribute.h
- Attribute.C

4.2 Connectivity Class Reference

Encapsulates an element-connectivity of a mesh.

Inheritance diagram for Connectivity::



Size information

- Size size_of_corners_pe () const

  Get the number of corners per element of the current connectivity table.

- Size size_of_nodes_pe () const

  Get the number of nodes per element of the current connectivity table.

- Size size_of_edges_pe () const

  Get the number of edges per element of the current connectivity table.

- Size size_of_faces_pe () const

  Get the number of faces per element of the current connectivity table.

- Size size_of_elements () const

  Get the total number of elements (including ghost elements) in the table.

- Size size_of_ghost_elements () const

  Get the number of ghost elements.

- Size size_of_real_elements () const

  Get the number of real elements.

- Size size_of_nodes () const

  Get the total number of nodes (including ghost nodes) of the owner pane.

- Size size_of_ghost_nodes () const

  Get the number of ghost nodes of the owner pane.

- Size size_of_real_nodes () const

  Get the number of real nodes of the owner pane.

- Size index_offset () const

  Get the index of the first element.

- Size size_i () const

  Get the number of nodes in i-dimension if the mesh is structured.

- Size size_j () const

  Get the number of nodes in j-dimension if the mesh is structured.

- Size size_k () const

  Get the number of nodes in k-dimension if the mesh is structured.

- static Size size_of_corners_pe (int type)

  Get the number of corners per element of the given type of element.

- static Size size_of_nodes_pe (int type)

  Get the number of nodes per element of the given type of element.

- static Size size_of_edges_pe (int type)

  Get the number of edges per element of the given type of element.

- static Size size_of_faces_pe (int type)

  Get the number of faces per element of the given type of element.

Helpers

- void * allocate (int strd, int cap, bool force) throw (COM_exception)

  Allocate memory for unstructured mesh.

- void set_size (int nitems, int ngitems=0) throw (COM_exception)

  Set the size of items and ghost items.

- static bool **is_element_name** (const std::string &aname)
- static const int * get_size_info (const std::string &aname)

  Obtain the size info of pre-defined connectivity.

Public Types

- enum **Connectivity_type** {

  ST1, **ST2**, **ST3**, **BAR2**,

  BAR3, **TRI3**, **TRI6**, **QUAD4**,

QUAD8, **QUAD9**, **TET4**, **TET10**,
PYRIMID5, **PYRIMID14**, **PRISM6**, **PRISM15**,
PRISM18, **HEX8**, **HEX20**, **HEX27**,
TYPE_MAX_CONN }

- enum **Connectivity_info** {
  TYPE_ID, **SIZE_DIM**, **ORDER**, **SIZE_NNODES**,
  SIZE_NCORN, **SIZE_NEDGES**, **SIZE_NFACES**, **SIZE_MAX_CONN**
  }

Public Member Functions

- bool is_const () const
  Returns whether the array is set to be read-only.

Constructors and destructor

- Connectivity (Pane *pane, const std::string &name, int id, const int sizes[ ], int type=COM_INT)
  Create an attribute with name n in window w.
- Connectivity (Pane *pane, Connectivity *con, const std::string &name, int id)
  Construct from another connectivity table.

Identity

- Connectivity * parent ()
  Parent attribute being used.
- const Connectivity * **parent** () const
- Connectivity * root ()
  Root of use-inheritance.
- const Connectivity * **root** () const
- void inherit (Connectivity *parent, bool clone, bool withghost) throw (COM_exception)

Inherit a connectivity table.

- int element_type () const
  Obtain element type ID.
- int dimension () const
  Get the dimension of the mesh.
- bool is_structured () const
  Determine whether the mesh is quadratic.
- bool is_quadratic () const
  Determine whether the element type is quadratic.

Array information

- const int * pointer () const
  Get a constant pointer to the connectivity array.
- int * pointer ()
  Get a pointer to the connectivity array.
- const int * get_addr (int i, int j=0) const throw (COM_exception)
  Obtain the address of the jth component of the ith item, where 0<=i<size_of_items.
- int * **get_addr** (int i, int j=0) throw (COM_exception)

Protected Member Functions

- void set_pointer (void *p, int strd, int cap, bool is_const) throw (COM_exception)
  Set pointer of connectivity table.
- void set_offset (Size offset) throw (COM_exception)
  Set the index of the first element.

Protected Attributes

- int _offset

  Offset of the first element.

- const int * **_size_info**

Static Protected Attributes

- static const int **_sizes** [TYPE_MAX_CONN][SIZE_MAX_CONN]

4.2.1 Detailed Description

It supports both structured and unstructured meshes.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 Connectivity::Connectivity (Pane * *pane*, const std::string & *name*, int *id*, const int *sizes*[ ], int *type* = COM_INT) [inline]

Parameters:

pane pointer to its owner pane object.

parent parent connectivity (for supporting inheritance).

name connectivity name.

id connectivity ID (always negative).

sizes size information about the element type.

type base data type.

4.2.3 Member Function Documentation

4.2.3.1 const int * Connectivity::get_addr (int *i*, int *j* = 0) const throw (COM_exception)

This function is recursive and relatively expensive, and hence should be used only for performance-insenstive tasks.

Reimplemented from Attribute.

4.2.3.2 void Connectivity::set_size (int *nitems*, int *ngitems* = 0) throw (COM_exception)

Can be changed only if the attribute is a root.

Reimplemented from Attribute.

4.2.4 Member Data Documentation

4.2.4.1 const int Connectivity::_sizes [static, protected]

Initial value:

```
{

  {ST1,       1, 1,     2,       2,      1,      0},
  {ST2,       2, 1,     4,       4,      4,      1},
  {ST3,       3, 1,     8,       8,     12,      6},
  {BAR2,      1, 1,     2,       2,      1,      0},
  {BAR3,      1, 2,     3,       2,      1,      0},
  {TRI3,      2, 1,     3,       3,      3,      1},
  {TRI6,      2, 2,     6,       3,      3,      1},
  {QUAD4,     2, 1,     4,       4,      4,      1},
  {QUAD8,     2, 2,     8,       4,      4,      1},
  {QUAD9,     2, 2,     9,       4,      4,      1},
  {TET4,      3, 1,     4,       4,      6,      4},
  {TET10,     3, 2,     10,      4,      6,      4},
  {PYRIMID5,  3, 1,     5,       5,      8,      5},
  {PYRIMID14,3, 2,      14,      5,      8,      5},
  {PRISM6,    3, 1,     6,       6,      9,      5},
  {PRISM15,   3, 2,     15,      6,      9,      5},
  {PRISM18,   3, 2,     18,      6,      9,      5},
  {HEX8,      3, 1,     8,       8,     12,      6},
  {HEX20,     3, 2,     20,      8,     12,      6},
  {HEX27,     3, 2,     27,      8,     12,      6}
}
```

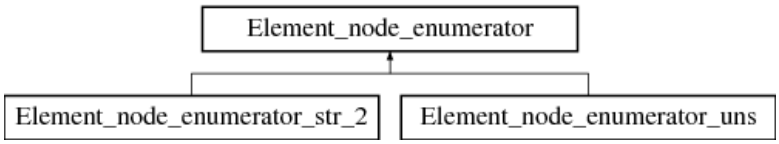
The documentation for this class was generated from the following files:

- Connectivity.h
- Connectivity.C

4.3 Element_node_enumerator Class Reference

An adaptor for enumerating node IDs of an element.

Inheritance diagram for Element_node_enumerator::



Public Member Functions

- Element_node_enumerator (const Pane *pane, int i, const Connectivity *conn=NULL)

  Constructor for an element in a structured or an unstructured mesh.

- Element_node_enumerator (const Pane *pane, const std::pair< int, int > &id)

  A constructor for an element in a structured mesh.

- void next ()

  Go to the next element within the connectivity tables of a pane.

- int type () const

  Obtain the type of the element.

- int is_quadratic () const

  Check whether the element is quadratic.

- int size_of_nodes () const

  Number of nodes per element.

- int size_of_corners () const

  Number of corners per element.

- void get_nodes (std::vector< int > &nodes)

  Get a vector of all of the nodes in the element.

- int size_of_edges () const

Number of edges per element.

- int size_of_faces () const

  Number of faces per element.

- int dimension () const

  Get the dimension of the base pane.

- int id () const

  Get the local id of the element within the pane.

- int vertex (int lvid, bool level=false) const

  Get the vertex index of an vertex within the element.

- int operator[ ] (int i) const

  Obtain the pane-scope node ID from its element-scope index.

- const Pane * **pane** () const

Protected Attributes

- const Pane * **_pane**
- const Connectivity * **_conn**
- union {
  const int * **_start**
  int **_base**
  };

- int **_res**

4.3.1 Constructor & Destructor Documentation

4.3.1.1 Element_node_enumerator::Element_node_enumerator (const Pane * *pane*, int *i*, const Connectivity * *conn* = NULL)

If conn==NULL, then i is an element index local to the pane. If conn!=NULL, then i is an element index local to the connectivity.

4.3.1.2 Element_node_enumerator::Element_node_enumerator (const Pane * *pane*, const std::pair< int, int > & *id*)

Parameters:

pane a pointer to the owner pane.

r the row id of the element (starting from 1).

c the column id of the element (starting from 1).

4.3.2 Member Function Documentation

4.3.2.1 int Element_node_enumerator::operator[ ] (int *i*) const [inline]

Parameters:

i the element-scope index of a node (starting from 0).

Reimplemented in Element_node_enumerator_str_2, and Element_node_-enumerator_uns.

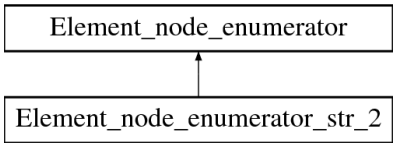
The documentation for this class was generated from the following files:

- Element_accessors.h
- Element_accessors.C

4.4 Element_node_enumerator_str_2 Class Reference

Optimized version for 2-D structured meshes.

Inheritance diagram for Element_node_enumerator_str_2::



Public Member Functions

- **Element_node_enumerator_str_2** (const Pane *pane, int i)
- **Element_node_enumerator_str_2** (const Pane *pane, const std::pair< int, int > &id)
- void next ()

  Go to the next element within the connectivity tables of a pane.

- int operator[ ] (int i) const

  Obtain the pane-scope node ID from its element-scope index.

4.4.1 Member Function Documentation

4.4.1.1 int Element_node_enumerator_str_2::operator[ ] (int *i*) const [inline]

Parameters:

i the element-scope index of a node (starting from 0).

Reimplemented from Element_node_enumerator.

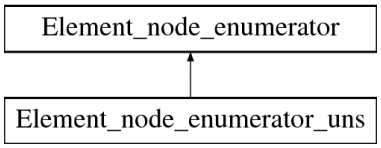
The documentation for this class was generated from the following file:

- Element_accessors.h

4.5 Element_node_enumerator_uns Class Reference

Optimized version for unstructured meshes.

Inheritance diagram for Element_node_enumerator_uns::



Public Member Functions

- **Element_node_enumerator_uns** (const Pane *pane, int i, const Connectivity *conn=NULL)
- void next ()

  Go to the next element within the connectivity tables of a pane.

- int operator[ ] (int i) const

  Obtain the pane-scope node ID from its element-scope index.

- int size_of_nodes () const

  Number of nodes per element.

4.5.1 Member Function Documentation

4.5.1.1 int Element_node_enumerator_uns::operator[ ] (int *i*) const [inline]

Parameters:

i the element-scope index of a node (starting from 0).

Reimplemented from Element_node_enumerator.

The documentation for this class was generated from the following file:

- Element_accessors.h

4.6 Element_node_vectors_k_const< Value > Class Template Reference

This is a helper class for accessing nodal data.

Inherited by Element_node_vectors_k< Value >.

Public Types

- enum { **MAX_NODES** = 9 }
- typedef unsigned int **Size**

Public Member Functions

- Element_node_vectors_k_const ()
  Default constructor.
- void set (const Value *p, Element_node_enumerator &ene, int strd)
  initialize the accessor with a pointer and a specific stride.
- void set (const Attribute *a, Element_node_enumerator &ene)
  initialize the accessor from an attribute assiciated with a pane.
- const Value & **operator()** (int i, int j) const
- const Value & **operator()** (int i) const
- const Value & **operator[ ]** (int i) const

Protected Attributes

- const Attribute * **_attr**
- union {
  int **_offsets** [MAX_NODES]
  const Value * **_vs** [MAX_NODES]
  };

template<class Value> class Element_node_vectors_k_const< Value >

The documentation for this class was generated from the following file:

- Element_accessors.h

4.7 Element_node_vectors_k_const< Value > Class Template Reference

This is a helper class for accessing nodal data.

Inherited by Element_node_vectors_k< Value >.

Public Types

- enum { **MAX_NODES** = 9 }
- typedef unsigned int **Size**

Public Member Functions

- Element_node_vectors_k_const ()
  Default constructor.
- void set (const Value *p, Element_node_enumerator &ene, int strd)
  initialize the accessor with a pointer and a specific stride.
- void set (const Attribute *a, Element_node_enumerator &ene)
  initialize the accessor from an attribute assiciated with a pane.
- const Value & **operator()** (int i, int j) const
- const Value & **operator()** (int i) const
- const Value & **operator[ ]** (int i) const

Protected Attributes

- const Attribute * **_attr**
- union {
  int **_offsets** [MAX_NODES]
  const Value * **_vs** [MAX_NODES]
  };

template<class Value> class Element_node_vectors_k_const< Value >

The documentation for this class was generated from the following file:

- Element_accessors.h

4.8 COM_exception Struct Reference

Encapsulates the states of an exception.

Public Member Functions

- COM_exception (Error_code i, const std::string &m=std::string())
  Constructor from an error code and optionally an error message.
- COM_exception (const COM_exception &e)
  Copy constructor.

Public Attributes

- Error_code ierr
  Error code.
- std::string msg
  Error message.

4.8.1 Detailed Description

The documentation for this struct was generated from the following file:

- roccom_exception.h

4.9 Facet_node_enumerator Class Reference

Adaptor for enumerating node IDs of a facet of an element.

Public Member Functions

- **Facet_node_enumerator** (const Element_node_enumerator *ene, int k)
- int **size_of_nodes** () const
- int **size_of_corners** () const
- int **size_of_edges** () const
- void **get_nodes** (std::vector< int > &nodes, bool quad_ret)
- int **operator[ ]** (int i) const

Protected Attributes

- const Element_node_enumerator ∗const **_ene**
- const int **_k**
- const int **_ne**
- const int ∗ **_fn_list**

Static Protected Attributes

- static const int **_face_node_lists_tets** [4][6]
- static const int **_face_node_lists_pyra** [5][8]
- static const int **_face_node_lists_pris** [5][9]
- static const int **_face_node_lists_hexa** [6][9]

4.9.1 Member Data Documentation

4.9.1.1 const int Facet_node_enumerator::_face_node_lists_tets

`[static, protected]`

Initial value:

```
{ {0, 2, 1, 6, 5, 4}, { 0, 1, 3, 4, 8, 7},
  {1, 2, 3, 5, 9, 8}, {2, 0, 3, 6, 7, 9}}
```

4.9.1.2 const int Facet_node_enumerator::_face_node_lists_pyra

`[static, protected]`

Initial value:

```
{ {0,3,2,1,8,7,6,5}, {0,1,4,5,10,9,-1,-1}, {1,2,4,6,11,10,-1,-1},
  {2,3,4,7,12,11,-1,-1}, {3,0,4,8,9,12,-1,-1}}
```

4.9.1.3 const int Facet_node_enumerator::_face_node_lists_pris

`[static, protected]`

Initial value:

```
{ {0,1,4,3,6,10,12,9,15}, {1,2,5,4,7,11,13,10,16}, {2,0,3,5,8,9,14,11,17},
  {0,2,1,8,7,6,-1,-1,-1}, {3,4,5,12,13,14,-1,-1,-1}}
```

4.9.1.4 const int Facet_node_enumerator::_face_node_lists_hexa [static, protected]

Initial value:

```
{ {0,3,2,1,11,10,9, 8, 20}, {0,1,5,4,8, 13,16,12,21},
  {1,2,6,5,9, 14,17,13,22}, {2,3,7,6,10,15,18,14,23},
  {0,4,7,3,12,19,15,11,24}, {4,5,6,7,16,17,18,19,25}}
```

The documentation for this class was generated from the following files:

- Element_accessors.h
- Element_accessors.C

4.10 Function Class Reference

A Function object corresponds to a function member of a window.

Public Types

- enum { **MAX_NUMARG** = 7 }
- enum { **F_FUNC**, **C_FUNC**, **CPP_MEMBER** }

Public Member Functions

Constructors

- Function ()
  Default constructor.
- Function (Func_ptr p, const std::string &s, const int *t, Attribute *a, bool b=false)
  Create a function object with physical address p.
- **Function** (Member_func_ptr p, const std::string &s, const int *t, Attribute *a)

Access methods

- Func_ptr pointer ()
  Get physical address of the function.
- int num_of_args () const
  Get the number of arguments.
- bool is_input (int i) const
  Check whether the ith argument is for input.
- bool is_output (int i) const
  Check whether the ith argument is for output.
- bool is_literal (int i) const
  Check whether the ith argument is literal type.
- bool is_optional (int i) const
  Check whether the ith argument is optional.
- bool is_rawdata (int i) const
  Check whether the ith argument is raw.
- bool is_metadata (int i) const
  Check whether the ith argument is meta.
- bool **is_fortran** () const
- COM_Type **data_type** (int i) const
- char **intent** (int i) const
- void **set_communicator** (MPI_Comm c)
- MPI_Comm **communicator** () const
- Attribute * **attribute** ()

Invocation

- void operator() (int n, void **ps) throw (COM_exception)
  invoke with an array of arguments

4.10.1 Detailed Description

It can take up to MAX_NUMARG arguments of void* type (excluding the implicit arguments) and return no value.

4.10.2 Constructor & Destructor Documentation

4.10.2.1 Function::Function (Func_ptr *p*, const std::string & *s*, const int * *t*, Attribute * *a*, bool *b* = `false`) `[inline]`

Parameters:

- ***p*** physical address of the function.
- ***s*** the intentions of the arguments. Its length indicates the number of arguments. Each entry indicates the intention of its corresponding argument: 'i'/'I' for input-only; 'o'/'O' for output-only; 'b'/'B' for input and output. Uppercase indicates optional arguments.
- ***t*** the data types of the arguments. If it is COM_METADATA, then the argument should be a pointer to an attribute. If it is COM_RAWDATA, then argument should be the physical address of the values of an attribute. Otherwise, it is literal type and the argument should be a pointer to corresponding data type.
- ***a*** attribute with which a member functions is associated.
- ***b*** whether the function is a Fortran subroutine

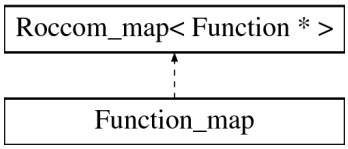
The documentation for this class was generated from the following file:

- Function.h

4.11 Function_map Class Reference

A map functions.

Inheritance diagram for Function_map::



Public Member Functions

- int add_object (const std::string &n, Function *t)

Insert a function into the table.

Public Attributes

- std::vector< char > verbs
  Whether verbose is on.
- std::vector< double > wtimes_self
  Accumulator of wall-clock time spent by itself excluding functions called by it.
- std::vector< double > wtimes_tree
  Accumulator of wall-clock time spent by itself and those functions called by it.
- std::vector< int > counts
  Counts of the number of calls.

4.11.1 Detailed Description

Supports quickly finding a function object from a function handle. Also contains timing information of functions to support profiling.

The documentation for this class was generated from the following file:

- maps.h

4.12 Pane Class Reference

A Pane object contains a mesh, pane attribute, and field variables.

Inherited by Window::Pane_friend.

Public Types

- enum **OP_Init** {
  OP_SET = 1, **OP_SET_CONST**, **OP_ALLOC**, **OP_RESIZE**,
  OP_DEALLOC }

- enum **Inherit_Modes** { **INHERIT_USE** = 0, **INHERIT_CLONE**, **INHERIT_COPY** }
- typedef std::vector< Attribute * > Attr_set

  Vector of attributs.

- typedef std::vector< Connectivity * > Cnct_set

  Vector of connectivities.

- typedef unsigned int Size

  Unsighed int.

Public Member Functions

Constructor and destructor

- Pane (Window *w, int i)

  Create a pane in window w with ID i.

- Pane (Pane *p, int i)

  Create a pane by copying from attribute descriptions from another pane.

- virtual ~Pane ()

  Default destructor.

Initialization

- void init_done () throw (COM_exception)

  Finalize the initialization of a pane.

Identification

- const Window * window () const

  Obtain a constant pointer to its owner window object.

- Window * window ()

  Obtain a pointer to its owner window object.

- int id () const
  Get the ID of the pane.
- int dimension () const
  Dimension of the pane.
- bool is_unstructured () const
  Is mesh of the pane unstructured?
- bool is_mixed () const
  Does the pane contain more than one type of elements?
- bool is_structured () const
  Is mesh of the pane structured?

Mesh management

- Size size_of_nodes () const
  Get the total number of nodes in the pane (including ghost nodes).
- Size maxsize_of_nodes () const
  Get the maximum number of real nodes in the pane (excluding ghost nodes).
- Size size_of_ghost_nodes () const
  Get the number of ghost nodes.
- Size maxsize_of_ghost_nodes () const
  Get the maximum number of real nodes in the pane (excluding ghost nodes).
- Size size_of_real_nodes () const
  Get the number of real nodes in the pane (excluding ghost nodes).
- Size maxsize_of_real_nodes () const
  Get the maximum number of real nodes in the pane (excluding ghost nodes).
- Size size_of_elements () const
  Get the total number of elements in the pane (including ghost elements).

- Size maxsize_of_elements () const

  Get the maximum number of elements allowed in the pane (including ghost elements).

- Size size_of_ghost_elements () const

  Get the total number of ghost elements.

- Size maxsize_of_ghost_elements () const

  Get the maximum number of elements allowed in the pane (including ghost elements).

- Size size_of_real_elements () const

  Get the number of real elements in the pane (excluding ghost elements).

- Size maxsize_of_real_elements () const

  Get the maximum number of real elements allowed in the pane (excluding ghost elements).

- double * coordinates ()

  Get the pointer to the values of the coordinates.

- double * x_coordinates ()

  Get the pointer to the values of the x-coordinates, if the coordinates are staggered.

- double * y_coordinates ()

  Get the pointer to the values of the y-coordinates, if the coordinates are staggered.

- double * z_coordinates ()

  Get the pointer to the values of the z-coordinates, if the coordinates are staggered.

- const double * coordinates () const

  Get a constant pointer to the values of the coordinates.

- const double * x_coordinates () const

  Get a constant pointer to the values of the x-coordinates, if the coordinates are staggered.

- const double * y_coordinates () const

Get a constant pointer to the values of the y-coordinates, if the coordinates are staggered.

- const double * z_coordinates () const
  Get a constant pointer to the values of the z-coordinates, if the coordinates are staggered.

- int * pane_connectivity ()
  Get a pointer to the values of the pane connectivity.

- const int * pane_connectivity () const
  Get a const pointer to the values of the pane connectivity.

- bool **ignore_ghost** () const
- void **set_ignore_ghost** (bool ignore)

Structured meshes

- Size size_of_ghost_layers () const
  Dimension of the pane.

- Size size_i () const
  Get the number of nodes in i-dimension if the mesh is structured.

- Size size_j () const
  Get the number of nodes in j-dimension if the mesh is structured.

- Size size_k () const
  Get the number of nodes in k-dimension if the mesh is structured.

Attribute management

- void attributes (std::vector< Attribute * > &as)
  Obtain all the attributes of the pane.

- void attributes (std::vector< const Attribute * > &as) const
  Obtain all the attributes of the pane.

- Attribute * attribute (const std::string &a)
  Obtain the attribute from given name.

- const Attribute * attribute (const std::string &a) const
  Obtain the attribute from given name.

- Attribute * attribute (int i)
  Obtain the attribute from its ID.

- const Attribute * attribute (int i) const
  Obtain the attribute from its ID.

Connectivity management

- void connectivities (std::vector< Connectivity * > &es)
  Obtain all the element connectivities of the pane.

- void **connectivities** (std::vector< const Connectivity * > &es) const
- Connectivity * connectivity (Size i) throw (COM_exception)
  Obtain the connectivity table containing the element with the given ID.

- const Connectivity * **connectivity** (Size i) const throw (COM_exception)
- void elements (std::vector< Connectivity * > &es)
  Obtain all the element connectivities of the pane.

- void **elements** (std::vector< const Connectivity * > &es) const
- void refresh_connectivity () throw (COM_exception)
  Update offsets and sizes of connectivity of an unstructured mesh.

Protected Member Functions

- Attribute * **new_attribute** (const std::string &aname, int aid, const char loc, const int type, int ncomp, const std::string &unit) throw (COM_exception)
- void insert (Attribute *attr) throw (COM_exception)
  Insert an attribute onto the pane.

- void delete_attribute (int id) throw (COM_exception)
  Delete an existing attribute with given id.

- void **reinit_attr** (int aid, OP_Init op, void **addr, int strd, int cap) throw (COM_exception)
- const Connectivity * connectivity (const std::string &a) const throw (COM_exception)

  Obtain the connectivity with the given name.

- Connectivity * **connectivity** (const std::string &a, bool insert=false) throw (COM_exception)
- void **reinit_conn** (Connectivity *con, OP_Init op, int **addr, int strd, int cap) throw (COM_exception)
- Attribute * inherit (Attribute *from, const std::string &aname, int mode, bool withghost) throw (COM_exception)

  Inherit an attribute from another pane onto the current pane:.

- void set_size (Attribute *a, int nitems, int ng) throw (COM_exception)

  Set the size of an attribute.

- void set_size (Connectivity *con, int nitems, int ng) throw (COM_exception)

  Set the size of a connectivity table.

Protected Attributes

- Window * _window

  Point to the parent window.

- int _id

  Pane id.

- Attr_set _attr_set

  Set of attributes.

- Cnct_set _cnct_set

  Set of element connectivity.

- bool _ignore_ghost

  Whether the ghosts were ignored.

Classes

- class **Attribute_friend**
- class **Connectivity_friend**

4.12.1 Detailed Description

Mesh data include nodal coordinates and element connectivity.

4.12.2 Member Function Documentation

4.12.2.1 double* Pane::coordinates () [inline]

It returns NULL if the coordinates are staggered.

4.12.2.2 double* Pane::x_coordinates () [inline]

4.12.2.3 double* Pane::y_coordinates () [inline]

4.12.2.4 double* Pane::z_coordinates () [inline]

4.12.2.5 const double* Pane::x_coordinates () const [inline]

4.12.2.6 const double* Pane::y_coordinates () const [inline]

4.12.2.7 const double* Pane::z_coordinates () const [inline]

4.12.2.8 Size Pane::size_of_ghost_layers () const [inline]

Get the number of ghost layers for structured mesh

4.12.2.9 void Pane::elements (std::vector< Connectivity * > & *es*) [inline]

Kept for backward compatibility

The documentation for this class was generated from the following files:

- Pane.h
- Pane.C

4.13 Roccom_base Class Reference

This file indirectly includes the following files: iostream, map, string, vector, and roccom_basic.h.

Miscellaneous

- enum { **FPTR_NONE** = 0, **FPTR_INSERT** = 1, **FPTR_APPEND** = 2 }
- int get_error_code () const

  Get the error code.

- void **turn_on_exception** ()
- void **turn_off_exception** ()
- int **f90ptr_treat** () const
- static int get_sizeof (COM_Type type, int count=1)

  Gets the size of the data type given by its index.

Initialization and Finalization

- Roccom_base (int *argc, char ***argv) throw (COM_exception, int)

  Constructor.

- ~Roccom_base ()

  Destructor.

- static void **init** (int *argc, char ***argv) throw (int)
- static void **finalize** () throw (int)
- static void **abort** (int ierr)
- static void set_default_communicator (MPI_Comm comm)

  Set the default communicator of Roccom.

- static MPI_Comm get_default_communicator ()

Get the default communicator of Roccom.

- static void set_roccom (Roccom_base ∗)
  Set the Roccom pointer to the given object.
- static Roccom_base ∗ get_roccom ()
  Get a pointer to the Roccom object.
- static bool initialized ()
  Checks whether Roccom has been initialized.

Public Types

- typedef Window::Pointer_descriptor **Pointer_descriptor**

Public Member Functions

Module management

- void load_module (const std::string &lname, const std::string &wname) throw (int)
  Load a module.
- void unload_module (const std::string &lname, const std::string &wname, int dodl=1) throw (int)
  Unload a module.

Window and pane management

- void new_window (const std::string &wname, MPI_Comm comm) throw (int)
  Creates a window with given name.
- void delete_window (const std::string &wname) throw (int)
  Deletes a window with given name.

- void window_init_done (const std::string &wname, bool panechanged=true) throw (int)
  Marks the end of the registration of a window.

- void delete_pane (const std::string &wname, const int pid) throw (int)
  Deletes a pane and its associated data.

Attribute management

- void new_attribute (const std::string &wa, const char loc, const int data_type, int size, const std::string &unit) throw (int)
  Creates a new attribute for a window.

- void delete_attribute (const std::string &wa) throw (int)
  Delete an existing attribute from a window.

- void set_size (const std::string &wa_str, int pane_id, int nitems, int ng=0) throw (int)
  Set the sizes of an attribute.

- void set_object (const std::string &wa, const int pane_id, void *obj_addr, void *casted_obj) throw (int)
  Associates an object with a specific window.

- void get_object (const std::string &wa, const int pane_id, void **ptr) throw (int)
  Associates an object with a specific window.

- void set_array (const std::string &wa, const int pane_id, void *addr, int strd=0, int cap=0, bool is_const=false) throw (int)
  Associates an array with an attribute for a specific pane.

- template<class T>
  void **set_bounds** (const std::string &wa, const int pane_id, T lbnd, T ubnd) throw (int)
- template<class T>
  void **set_bounds** (const std::string &wa, const int pane_id, const T *lbnd, const T *ubnd) throw (int)
- void **set_bounds** (const std::string &wa, const int pane_id, const void *lbnd, const void *ubnd) throw (int)

- void **get_bounds** (const std::string &wa, const int pane_id, void *lbnd, void *ubnd) throw (int)
- int **check_bounds** (const std::string &wa, int pane_id) throw (int)
- void allocate_array (const std::string &wa, const int pane_id=0, void **addr=NULL, int strd=0, int cap=0) throw (int)

  Allocate space for an attribute on a specific pane and return the address by setting addr.

- void resize_array (const std::string &wa, const int pane_id=0, void **addr=NULL, int strd=-1, int cap=0) throw (int)

  Resize an attribute on a specific pane and return the address by setting addr.

- void append_array (const std::string &wa, const int pane_id, const void *val, int v_strd, int v_size) throw (int)

  Append an array to the end of the attribute on a specific pane and return the new address by setting addr.

- void use_attribute (const std::string &wname, const std::string &pwname, int withghost=1, const char *cndname=NULL, int val=0) throw (int)

  Use the subset of panes of another window of which the given pane attribute has value val.

- void clone_attribute (const std::string &wname, const std::string &pwname, int withghost=1, const char *cndname=NULL, int val=0) throw (int)

  Clone the subset of panes of another window of which the given pane attribute has value val.

- void copy_attribute (const std::string &wname, const std::string &pwname, int withghost=1, const char *cndname=NULL, int val=0) throw (int)

  Copy an attribute onto another.

- void copy_attribute (int trg_hdl, int src_hdl, int withghost=1, int ptn_hdl=0, int val=0) throw (int)

  Copy an attribute onto another.

- void deallocate_array (const std::string &wa, const int pid=0) throw (int)

  Deallocate space for an attribute in a pane, asuming the memory was allocated allocate_mesh or allocate_attribute.

- void get_attribute (const std::string &wa_str, char *loc, int *type, int *size, std::string *unit) throw (int)

  Information retrieval Get the information about an attribute.

- void get_size (const std::string &wa_str, int pane_id, int *size, int *ng=0) throw (int)

  Get the sizes of an attribute. The opposite of set_size.

- int get_status (const std::string &wa_str, int pane_id) throw (int)

  Get the status of an attribute.

- void get_array (const std::string &wa, const int pane_id, void **addr, int *strd=NULL, int *cap=0, bool is_const=false) throw (int)

  Get the address for an attribute on a specific pane.

- void get_array (const std::string &wa, const int pane_id, Pointer_-descriptor &addr, int *strd=NULL, int *cap=0, bool is_const=false) throw (int)

  Get the address for an attribute on a specific pane.

- void copy_array (const std::string &wa, const int pane_id, void *val, int v_strd=0, int v_size=0, int offset=0) throw (int)

  Copy an array from an attribute on a specific pane into a given buffer.

- void **set_f90pointer** (const std::string &waname, void *ptr, Func_ptr f, long int l) throw (int)
- void **get_f90pointer** (const std::string &waname, void *ptr, Func_ptr f, long int l) throw (int)

Information retrieval

- MPI_Comm **get_communicator** (const std::string &wname) throw (int)
- void get_panes (const std::string &wname, std::vector< int > &paneids_-vec, int rank=-2, int **pane_ids=NULL) throw (int)

  Obtain the panes of a given window on a specific process.

- void get_attributes (const std::string &wname, int *na, std::string &str, char **names=NULL) throw (int)

  Obtain the user-defined attributes of the given window.

- void get_connectivities (const std::string &wname, int pane_id, int *nc, std::string &str, char **names=NULL) throw (int)
  Obtain the connectivity tables of a pane of the given window.

- void get_parent (const std::string &waname, int pane_id, std::string &str, char **name=NULL) throw (int)
  Obtain the parent attribute's name of a given attribute on a given pane.

- void **free_buffer** (int **buf)
- void **free_buffer** (char **buf)
- int **get_window_handle** (const std::string &wname) throw (int)
- Window * **get_window_object** (int hdl) throw (int)
- const Window * **get_window_object** (int hdl) const throw (int)
- Window * **get_window_object** (const std::string &wname) throw (int)
- const Window * **get_window_object** (const std::string &wname) const throw (int)
- int **get_attribute_handle** (const std::string &waname) throw (int)
- int **get_attribute_handle_const** (const std::string &waname) throw (int)
- int **get_function_handle** (const std::string &wfname) throw (int)

Function management

- void set_function (const std::string &wf, Func_ptr ptr, const std::string &intents, const COM_Type *types, bool ff=false) throw (int)
  Registers a function to the window.

- void **set_member_function** (const std::string &wf, Func_ptr ptr, const std::string &wa, const std::string &intents, const COM_Type *types, bool ff=false) throw (int)
- void **set_member_function** (const std::string &wf, Member_func_ptr ptr, const std::string &wa, const std::string &intents, const COM_Type *types, bool ff=false) throw (int)
- int get_num_arguments (const std::string &wf) throw (COM_exception)
  Get the number of arguments of a given function "window.function".

- int get_num_arguments (const int wf) throw (COM_exception)
  Get the number of arguments of a given function from its handle.

- void call_function (int wf, int count, void **args, const int *lens=NULL, bool from_c=true) throw (int)
  Invoke a function with given arguments.

- void icall_function (int wf, int count, void *args[ ], int *reqid, const int *lens=NULL) throw (int)
  Nonblockingly invoke a function with given arguments.

- void wait (int)
  Wait for the completion of a nonblocking call.

- int test (int)
  Test whether a nonblocking call has finished.

Profiling and tracing tools

- int get_verbose () const
  Determines whether verbose is on.

- void set_verbose (int v)
  Changes the verbose setting.

- void **set_function_verbose** (int i, int level) throw (int)
- void set_profiling (int i)
  This subroutine turns on (or off) profiling if i==1 (or ==0).

- void **set_profiling_barrier** (int hdl, MPI_Comm comm)
- void **print_profile** (const std::string &fname, const std::string &header)

Protected Member Functions

- template<class T>
  void **set_member_function_helper** (const std::string &wf, T ptr, const std::string &wa, const std::string &intents, const COM_Type *types, bool ff=false) throw (int)
- std::pair< int, int > **get_f90pntoffsets** (const Attribute *a)

Window management

- Window & get_window (const std::string &wname) throw (COM_exception)
  Obtains a reference to the Window object from its name.

- const Window & get_window (const std::string &wname) const throw (COM_exception)

  Obtains a constant reference to the Window object from its name.

- Attribute & get_attribute (const int) throw (COM_exception)

  Obtains a reference to an attribute from its handle.

- const Attribute & get_attribute (const int) const throw (COM_exception)

  Obtains a const reference to an attribute from its handle.

- Function & get_function (const int) throw (COM_exception)

  Obtains a reference to an attribute from its handle.

- const Function & get_function (const int) const throw (COM_exception)

  Obtains a const reference to an attribute from its handle.

Miscellaneous

- int split_name (const std::string &wa, std::string &wname, std::string &aname, bool tothrow=true) throw (COM_exception)

  Extracts the window and attribute names from "window.attribute".

- void **proc_exception** (const COM_exception &, const std::string &) throw (int)

Protected Attributes

- Module_map **_module_map**
- Window_map **_window_map**
- Attribute_map **_attr_map**
- Function_map **_func_map**
- std::string _libdir

  Library directory.

- std::vector< double > _timer

  Timers for function calls.

- int _depth

  Depth of procedure calls.

- int _verbose

  Indicates whether verbose is on.

- int _verb1

  Indicates whether to print detailed information.

- MPI_Comm _comm

  Default communicator of Roccom.

- bool _mpi_initialized

  Indicates whether MPI was initialized by Roccom.

- int _errorcode

  Error code.

- bool _exception_on

  Indicates whether Roccom should throw exception.

- bool _profile_on

  Indicates whether should profile.

- int _f90_mangling

  Encoding name mangling.

- int _f90ptr_treat

  Treatement of F90 pointers.

- int _cppobj_casting

  Treatement of C++ objects.

Static Protected Attributes

- static Roccom_base * **roccom_base** = NULL

4.13.1 Detailed Description

The base class for Roccom implementations.

4.13.2 Member Function Documentation

4.13.2.1 static void Roccom_base::set_default_communicator (MPI_Comm *comm*) [inline, static]

This communicator will be used as the default communicator for any new window.

4.13.2.2 void Roccom_base::set_roccom (Roccom_base * *r*) [static]

It was introduced to support processes.

4.13.2.3 void Roccom_base::set_size (const std::string & *wa_str*, int *pane_id*, int *nitems*, int *ng* = 0) throw (int)

Note that for nodal or elemental data, setting sizes for one such attributes affects all other attributes.

4.13.2.4 void Roccom_base::allocate_array (const std::string & *wa*, const int *pane_id* = 0, void ** *addr* = NULL, int *strd* = 0, int *cap* = 0) throw (int)

Allocate for all panes if pane-id is 0, in which case, do not set addr.

4.13.2.5 void Roccom_base::resize_array (const std::string & *wa*, const int *pane_id* = 0, void ** *addr* = NULL, int *strd* = -1, int *cap* = 0) throw (int)

Resize for all panes if pane-id is 0, in which case, do not set addr. The difference between resize and allocate is that resize will reallocate memory only if the current array cannot accomodate the requested capacity.

4.13.2.6 void Roccom_base::append_array (const std::string & *wa*, const int *pane_id*, const void * *val*, int *v_strd*, int *v_size*) throw (int)

4.13.2.7 void Roccom_base::use_attribute (const std::string & *wname*, const std::string & *pwname*, int *withghost* = 1, const char * *cndname* = NULL, int *val* = 0) throw (int)

4.13.2.8 void Roccom_base::clone_attribute (const std::string & *wname*, const std::string & *pwname*, int *withghost* = 1, const char * *cndname* = NULL, int *val* = 0) throw (int)

4.13.2.9 void Roccom_base::deallocate_array (const std::string & *wa*, const int *pid* = 0) throw (int)

4.13.2.10 void Roccom_base::get_attribute (const std::string & *wa_str*, char * *loc*, int * *type*, int * *size*, std::string * *unit*) throw (int)

The opposite of new_attribute.

4.13.2.11 int Roccom_base::get_status (const std::string & *wa_str*, int *pane_id*) throw (int)

If the attribute name is empty, and pane ID is 0, then checks whether the window exist (return 0 if does and -1 if not); if attribute name is empty and pane ID is >0, then check whether check whether the given pane exists (return 0 if so and -1 if not). Otherwise, it checks the status of an attribute and returns one of the following values: -1: not exist. 0: not yet initialized 1: set by the user. 2: set by the user with const modifier. 3: inherited from (i.e., use) another attribute. 4: allocated by Roccom.

4.13.2.12 void Roccom_base::get_panes (const std::string & *wname*, std::vector< int > & *paneids_vec*, int *rank* = -2, int ** *pane_ids* = NULL) throw (int)

If rank is -2, then the current process is assumed. If rank is -1, then get the panes on all processes within the window's communicator.

4.13.2.13 void Roccom_base::get_parent (const std::string & *waname*, int *pane_id*, std::string & *str*, char ** *name* = NULL) throw (int)

If the attribute has no parent, then set name to empty.

4.13.2.14 void Roccom_base::set_function (const std::string & *wf*, Func_ptr *ptr*, const std::string & *intents*, const COM_Type * *types*, bool *ff* = false) throw (int)

The names of the window and the function is give by wf in the format of "window.function" (null terminated). The address is given by ptr. The last two arguments specifies the number of the arguments, the intentions and types of the arguments.

4.13.2.15 void Roccom_base::call_function (int *wf*, int *count*, void ** *args*, const int * *lens* = NULL, bool *from_c* = true) throw (int)

Parameters:

wf the handle to the function.

count the number of input arguments.

args the addresses to the arguments.

lens the lengths of character strings.

4.13.2.16 void Roccom_base::icall_function (int *wf*, int *count*, void * *args*[ ], int * *reqid*, const int * *lens* = NULL) throw (int) [inline]

Parameters:

wf the handle to the function.

count the number of input arguments.

args the addresses to the arguments.

reqid is set to the request of the current call.

lens the lengths of character strings.

4.13.2.17 int Roccom_base::test (int) [inline]

4.13.2.18 void Roccom_base::set_profiling (int *i*)

It (re-)initializes all profiling info to 0.

4.13.2.19 int Roccom_base::get_sizeof (COM_Type *type*, int *count* = 1) [static]

See also:

DDT

4.13.2.20 int Roccom_base::split_name (const std::string & *wa*, std::string & *wname*, std::string & *aname*, bool *tothrow* = true) throw (COM_exception) [protected]

Returns nonzero if fails.

4.13.3 Member Data Documentation

4.13.3.1 int Roccom_base::_f90_mangling [protected]

-1: Unknown. 0: lower-case without appending 1: upper-case without appending 2: lower-case with appending 3: upper-case with appending

4.13.3.2 int Roccom_base::_cppobj_casting [protected]

-1: Unknown 0: No casting to COM_Object 1: Casting to COM_Object

The documentation for this class was generated from the following files:

- Roccom_base.h
- Roccom_base.C

4.14 Roccom_map< Object > Class Template Reference

Supports mapping from names to handles and vice-versa for a module, window, function, or attribute.

Public Types

- typedef Object **value_type**

Public Member Functions

- int add_object (std::string name, Object t, bool is_const=false) throw (COM_exception)

  Insert an object into the table.

- void remove_object (std::string name, bool is_const=false) throw (COM_exception)

  Remove an object from the table.

- bool is_immutable (int i) const

  whether the object mutable

- const Object & operator[ ] (int i) const throw (COM_exception)

  Access an object using its handle.

- Object & **operator[ ]** (int i) throw (COM_exception)
- const std::string & name (int i) const

  Name of the object.

- int **size** () const
- std::pair< int, Object * > **find** (const std::string &name, bool is_const=false)

Protected Attributes

- I2O i2o

  Mapping from index to objects.

- N2I n2i

  Mapping from names to indices.

- std::list< int > salvaged

  List of salvaged indices.

- std::vector< std::string > names

  Name of the objects.

4.14.1 Detailed Description

template<class Object> class Roccom_map< Object >

The documentation for this class was generated from the following file:

- maps.h

4.15 Window Class Reference

A Window object contains multiple panes and multiple data attributes.

Public Types

- typedef std::map< int, int > **Proc_map**

Public Member Functions

Constructor and destructor

- Window (const std::string &name, MPI_Comm c)
  Create a window with a given name and MPI communicator.
- virtual ~Window ()
  Destructor.

Identity

- const std::string & name () const
  Obtain the window's name.
- MPI_Comm get_communicator () const
  Obtain the communicator of the window.

Function and data management

- void set_function (const std::string &fname, Func_ptr func, const std::string &intents, const COM_Type *types, Attribute *a, bool if_f90=false) throw (COM_exception)

  Initialize a Function record.

- void set_function (const std::string &fname, Member_func_ptr func, const std::string &intents, const COM_Type *types, Attribute *a, bool if_f90=false) throw (COM_exception)

  Initialize a Function record.

- Attribute * new_attribute (const std::string &aname, const char loc, const int type, int ncomp, const std::string &unit) throw (COM_exception)

  Create a new Attribute object with given properties.

- void delete_attribute (const std::string &aname) throw (COM_exception)

  Delete an existing Attribute object.

- void set_size (const std::string &aname, int pane_id, int nitems, int ng=0) throw (COM_exception)

  Set the sizes of an attribute for a specific pane.

- void set_array (const std::string &aname, const int pane_id, void *addr, int strd=0, int cap=0, bool is_const=false) throw (COM_exception)

  Associate an array with an attribute for a specific pane.

- void alloc_array (const std::string &aname, const int pane_id, void **addr, int strd=0, int cap=0) throw (COM_exception)

  Allocate memory for an attribute for a specific pane and set addr to the address.

- void resize_array (const std::string &aname, const int pane_id, void **addr, int strd=-1, int cap=0) throw (COM_exception)

  Resize memory for an attribute for a specific pane and set addr to the address.

- void **resize_array** (Attribute *a, void **addr, int strd=-1, int cap=0) throw (COM_exception)
- void **resize_array** (Connectivity *c, void **addr, int strd=-1, int cap=0) throw (COM_exception)
- void append_array (const std::string &aname, const int pane_id, const void *val, int v_strd, int v_size) throw (COM_exception)

Append the given array to the end of the attribute on a specific pane, and reallocate memory for the attribute if necessary.

- void dealloc_array (const std::string &aname, const int pane_id=0) throw (COM_exception)

  Deallocate memory for an attribute for a specific pane if allocated by Roccom.

- void **dealloc_array** (Attribute *a) throw (COM_exception)
- void **dealloc_array** (Connectivity *c) throw (COM_exception)
- Attribute * inherit (Attribute *from, const std::string &aname, int inherit_mode, bool withghost, const Attribute *cond, int val) throw (COM_exception)

  Inherit the attributes of another window with a different name.

- void copy_attribute (const Attribute *from, Attribute *to) throw (COM_exception)

  Copy an attribute object onto another.

- Attribute * get_attribute (const std::string &aname, char *l, int *t, int *n, std::string *u) const throw (COM_exception)

  Get the meta-information about an attribute.

- void get_size (const std::string &aname, int pane_id, int *nitems, int *ng) const throw (COM_exception)

  Get the sizes of an attribute for a specific pane.

- int get_status (const std::string &aname, int pane_id) const throw (COM_exception)

  Get the status of an attribute or pane.

- void get_parent (const std::string &aname, int pane_id, std::string &name) const throw (COM_exception)

  Get the parent name of an attribute and load into name.

- void get_array (const std::string &aname, const int pane_id, Pointer_descriptor &addr, int *strd=NULL, int *cap=NULL, bool is_const=false) throw (COM_exception)

  Get the address associated with an attribute for a specific pane.

- void copy_array (const std::string &aname, const int pane_id, void *val, int v_strd=0, int v_size=0, int offset=0) const throw (COM_exception)

Copy an attribute on a specific pane into a given array.

- void init_done (bool pane_changed=true) throw (COM_exception)
  Perform some final checking of the window.

Pane management

- int size_of_panes () const
  Obtain the number of local panes in the window.
- int size_of_panes_global () const
  Obtain the total number of panes in the window on all processes.
- int owner_rank (const int pane_id) const
  Obtain the process rank that owns a given pane.
- int last_attribute_id () const
  Return the last attribute id.
- const Proc_map & proc_map () const
  Obtain the process map.
- void delete_pane (const int pane_id) throw (COM_exception)
  Remove the pane with given ID.

Miscellaneous

- Pane & pane (const int pane_id, bool insert=false) throw (COM_exception)
  Find the pane with given ID. If not found, insert a pane with given ID.
- const Pane & **pane** (const int pane_id) const throw (COM_exception)
- void panes (std::vector< int > &ps, int rank=-2)
  Obtain all the local panes of the window.
- void panes (std::vector< Pane * > &ps)
  Obtain all the local panes of the window.
- void panes (std::vector< const Pane * > &ps) const

Obtain all the local panes of the window.

- void attributes (std::vector< Attribute * > &as)
  Obtain all the attributes of the pane.

- void attributes (std::vector< const Attribute * > &as) const
  Obtain all the attributes of the pane.

- Attribute * attribute (const std::string &a) throw (COM_exception)
  Obtain a pointer to the attribute metadata from its name.

- const Attribute * **attribute** (const std::string &a) const throw (COM_exception)
- Attribute * attribute (int i) throw (COM_exception)
  Obtain a pointer to the attribute metadata from its index.

- const Attribute * **attribute** (int i) const throw (COM_exception)
- Function * function (const std::string &f)
  Obtain the function pointer from its name.

- const Function * **function** (const std::string &f) const

Protected Types

- enum { **STATUS_SHRUNK**, **STATUS_CHANGED**, **STATUS_NOCHANGE** }

Protected Member Functions

- void reinit_attr (Attribute *attr, OP_Init op, void **addr=NULL, int strd=0, int cap=0) throw (COM_exception)

  Implementation for setting (op==OP_SET or OP_SET_CONST), allocating (op==OP_ALLOC), resizing (op==OP_RESIZE) and deallocating (op==OP_DEALLOC) an array for a specific attribute.

- void reinit_conn (Connectivity *con, OP_Init op, int **addr=NULL, int strd=0, int cap=0) throw (COM_exception)

  Template implementation for setting (op==OP_SET or OP_SET_CONST), allocating (op==OP_ALLOC), resizing (op==OP_RESIZE) and deallocating (op==OP_DEALLOC) an array for a specific connectivity table.

Protected Attributes

- Pane _dummy

  Dummy pane.

- std::string _name

  Name of the window.

- Attr_map _attr_map

  Map from attribute names to their metadata.

- Func_map _func_map

  Map from function names to their metadata.

- Pane_map _pane_map

  Map from pane ID to their metadata.

- Proc_map _proc_map

  Map from pane ID to process ranks.

- int _last_id

  The last used attribute index.

- MPI_Comm _comm

  the MPI communicator of the window.

- int _status

  Status of the window.

Classes

- class **Pane_friend**
- struct **Pointer_descriptor**

4.15.1 Constructor & Destructor Documentation

4.15.1.1 Window::Window (const std::string & *name*, MPI_Comm *c*)

Parameters:

name name of the window

c MPI communicator where the window resides

4.15.2 Member Function Documentation

4.15.2.1 Attribute * Window::new_attribute (const std::string & *aname*, const char *loc*, const int *type*, int *ncomp*, const std::string & *unit*) throw (COM_exception)

Parameters:

aname attribute name.

loc location ('w', 'p', 'n', or 'e').

type base data type.

ncomp number of components.

unit unit of the attribute.

4.15.2.2 void Window::delete_attribute (const std::string & *aname*) throw (COM_exception)

4.15.2.3 void Window::set_size (const std::string & *aname*, int *pane_id*, int *nitems*, int *ng* = 0) throw (COM_exception)

Parameters:

aname attribute name

pane_id pane ID

nitems total number of items (including ghosts)

ng number of ghosts

4.15.2.4 void Window::set_array (const std::string & *aname*, const int *pane_id*, void * *addr*, int *strd* = 0, int *cap* = 0, bool *is_const* = false) throw (COM_exception)

Parameters:

aname attribute name

pane_id pane ID

addr address of the array

strd Stride between two items of each component

cap capacity of the array alloc_array, resize_array

4.15.2.5 void Window::alloc_array (const std::string & *aname*, const int *pane_id*, void ** *addr*, int *strd* = 0, int *cap* = 0) throw (COM_exception)

alloc_array, resize_array, append_array

4.15.2.6 void Window::resize_array (const std::string & *aname*, const int *pane_id*, void ** *addr*, int *strd* = -1, int *cap* = 0) throw (COM_exception)

set_array, alloc_array, append_array

4.15.2.7 void Window::append_array (const std::string & *aname*, const int *pane_id*, const void * *val*, int *v_strd*, int *v_size*) throw (COM_exception)

set_array, alloc_array, resize_array

4.15.2.8 void Window::dealloc_array (const std::string & *aname*, const int *pane_id* = 0) throw (COM_exception)

alloc_array, resize_array

4.15.2.9 Attribute * Window::inherit (Attribute * *from*, const std::string & *aname*, int *inherit_mode*, bool *withghost*, const Attribute * *cond*, int *val*) throw (COM_exception)

Returns the corresponding value.

Parameters:

from attribute being copied from

aname new name of the attribute

cond an integer pane-attribute

val value to be compared against cond

inherit_mode mode of inheritance

withghost wheather ghost nodes/elements should be ignored

4.15.2.10 Attribute * Window::get_attribute (const std::string & *aname*, char * *l*, int * *t*, int * *n*, std::string * *u*) const throw (COM_exception)

Parameters:

aname attribute name

l location

t data type

n number of components

u unit

4.15.2.11 void Window::get_size (const std::string & *aname*, int *pane_id*, int * *nitems*, int * *ng*) const throw (COM_exception)

Parameters:

aname attribute name

pane_id pane ID

nitems total number of items (including ghosts)

ng number of ghosts

4.15.2.12 int Window::get_status (const std::string & *aname*, int *pane_id*) const throw (COM_exception)

Roccom_base::get_status()

4.15.2.13 void Window::get_parent (const std::string & *aname*, int *pane_id*, std::string & *name*) const throw (COM_exception)

If the attribute has no parent, then name is empty.

4.15.2.14 void Window::get_array (const std::string & *aname*, const int *pane_id*, Pointer_descriptor & *addr*, int * *strd* = NULL, int * *cap* = NULL, bool *is_const* = false) throw (COM_exception)

Parameters:

aname attribute name

pane_id pane ID

addr address of the array

strd Stride between two items of each component

cap capacity of the array alloc_array, resize_array, copy_array

4.15.2.15 void Window::copy_array (const std::string & *aname*, const int *pane_id*, void * *val*, int *v_strd* = 0, int *v_size* = 0, int *offset* = 0) const throw (COM_exception)

Parameters:

aname attribute name

pane_id pane ID

val address of the user array

v_strd stride of user array. 0 (the default) indicates number of components.

v_size number of items to be copied. 0 (the default) indicates number of items of the attribute.

offset starting item to be copied in the attribute. alloc_array, resize_array, get_array

4.15.2.16 int Window::owner_rank (const int *pane_id*) const

Returns -1 if the pane cannot be found in the process map.

4.15.2.17 void Window::reinit_attr (Attribute * *attr*, OP_Init *op*, void ** *addr* = NULL, int *strd* = 0, int *cap* = 0) throw (COM_exception) [protected]

Parameters:

attr attribute

op Operation (OP_SET, OP_SET_CONST, OP_ALLOC, OP_RESIZE)

addr address

strd stride

cap capacity

4.15.2.18 void Window::reinit_conn (Connectivity * *con*, OP_Init *op*, int ** *addr* = NULL, int *strd* = 0, int *cap* = 0) throw (COM_exception) [protected]

Parameters:

attr connectivity table

op Operation (OP_SET, OP_SET_CONST, OP_ALLOC, OP_RESIZE)

addr address

strd stride

cap capacity

4.15.3 Member Data Documentation

4.15.3.1 Attr_map Window::_attr_map [protected]

It does not contain individual components.

4.15.3.2 int Window::_last_id [protected]

The next available one is _last_id+1.

The documentation for this class was generated from the following files:

- Window.h
- Window.C

Index