PGMT Compound List

Here are the classes, structs, unions and interfaces with brief descriptions:

BaseKey	
BaseMap	
BaseRecord	*Class BaseRecord*
FileKey	*Class FileKey*
FileMap	
FileOp	*Class FileOp and methods*
FileRecord	*Class FileRecord defines a history record*
Machine	*Class Machine*
MW_Graph	
NoSuchString	
parm	*Struct for setting parameter values*
STAT	*Typdef struct STAT*

Generated on Mon Oct 7 01:04:29 2002 for PGMT by doxygen 1.2.18

PGMT Documentation

Generated on Mon Oct 7 01:04:29 2002 for PGMT by doxygen 1.2.18

PGMT Compound Index

B

BaseKey

BaseMap

BaseRecord

F

FileKey

FileMap

FileOp

FileRecord

M

Machine

MW_Graph

N

NoSuchString

P

parm

S

STAT

Generated on Mon Oct 7 01:04:29 2002 for PGMT by doxygen 1.2.18

PGMT File List

Here is a list of all files with brief descriptions:

computeNetwork.doxygen.h	**[code]**
history.doxygen.h	**[code]**
Machine.doxygen.rh.h	**[code]**
MW_Graph.doxygen.h	**[code]**

Generated on Mon Oct 7 01:04:29 2002 for PGMT by doxygen 1.2.18

PGMT Compound Members

_ | b | c | d | f | g | h | i | m | n | o | p | r | s | t | w | ~

Here is a list of all class members with links to the class documentation for each member:

- _ -

- _EntryTime : **FileRecord**
- _HostName : **FileKey**, **FileRecord**
- _Mean : **FileRecord**
- _Nsamples : **FileRecord**
- _ProcessorVector : **Machine**
- _Tag : **FileRecord**
- _Variance : **FileRecord**
- _VstringIDs : **FileKey**, **FileRecord**

- b -

- barrierSynch() : **Machine**
- BaseKey::BaseKey() : **BaseKey**
- BaseKey::get__NodeID() : **BaseKey**
- BaseKey::get__ProcID() : **BaseKey**
- BaseKey::set__Key() : **BaseKey**
- BaseKey::set__ProcID() : **BaseKey**
- BaseKey::~BaseKey() : **BaseKey**
- BaseMap : **BaseKey**
- BaseMap::BaseMap() : **BaseMap**
- BaseMap::BMap_to_FMap() : **BaseMap**
- BaseMap::ClearDataBase() : **BaseMap**
- BaseMap::CreateDataBase() : **BaseMap**
- BaseMap::CreateKey() : **BaseMap**
- BaseMap::DeleteKeyRec() : **BaseMap**
- BaseMap::GetKeyRec() : **BaseMap**

- BaseMap::InsertRec() : **BaseMap**
- BaseMap::QueryKeyRec() : **BaseMap**
- BaseMap::ResetKeyRec() : **BaseMap**
- BaseMap::StatUpdate() : **BaseMap**
- BaseMap::UpdateRec() : **BaseMap**
- BaseMap::ViewDataBase() : **BaseMap**
- BaseMap::~BaseMap() : **BaseMap**
- BaseRecord::BaseRecord() : **BaseRecord**
- BaseRecord::get__ExTime() : **BaseRecord**
- BaseRecord::get__Instance() : **BaseRecord**
- BaseRecord::get__Mean() : **BaseRecord**
- BaseRecord::get__NodeID() : **BaseRecord**
- BaseRecord::get__Nsamples() : **BaseRecord**
- BaseRecord::get__Parms() : **BaseRecord**
- BaseRecord::get__ProcID() : **BaseRecord**
- BaseRecord::get__Tag() : **BaseRecord**
- BaseRecord::get__Variance() : **BaseRecord**
- BaseRecord::printRecord() : **BaseRecord**
- BaseRecord::set__ExTime() : **BaseRecord**
- BaseRecord::set__Instance() : **BaseRecord**
- BaseRecord::set__Mean() : **BaseRecord**
- BaseRecord::set__NodeID() : **BaseRecord**
- BaseRecord::set__Nsamples() : **BaseRecord**
- BaseRecord::set__Parms() : **BaseRecord**
- BaseRecord::set__ProcID() : **BaseRecord**
- BaseRecord::set__Tag() : **BaseRecord**
- BaseRecord::set__Variance() : **BaseRecord**
- BaseRecord::set_all() : **BaseRecord**
- BaseRecord::~BaseRecord() : **BaseRecord**
- bRcvCPMessage() : **Machine**
- bRcvCPParameters() : **Machine**
- bRcvPepMessage() : **Machine**
- build_MsgHeader_MPI_datatype() : **MW_Graph**
- BuildCommunicator() : **Machine**
- BuildProcessorSpeedVector() : **Machine**
- BuildProcessorVector() : **Machine**
- busyLoop() : **Machine**

- c -

- CatFile() : **FileOp**
- CloseReadFile() : **FileOp**
- CloseWriteFile() : **FileOp**
- CreateGraphComm() : **MW_Graph**
- createPepComm() : **Machine**

- d -

- displayProcessorName : **Machine**

- f -

- FileKey::FileKey() : **FileKey**
- FileKey::get__HostName() : **FileKey**
- FileKey::get__Vstring() : **FileKey**
- FileKey::get__VstringIDs() : **FileKey**
- FileKey::set__FileKey() : **FileKey**
- FileKey::set__HostName() : **FileKey**
- FileKey::~FileKey() : **FileKey**
- FileMap : **FileKey**
- FileMap::ClearDataBase() : **FileMap**
- FileMap::CreateDataBase() : **FileMap**
- FileMap::CreateKey() : **FileMap**
- FileMap::DeleteKeyRec() : **FileMap**
- FileMap::FileMap() : **FileMap**
- FileMap::GenerateFileKey() : **FileMap**
- FileMap::GetFkeyRec() : **FileMap**
- FileMap::GetKeyRec() : **FileMap**
- FileMap::InsertRec() : **FileMap**
- FileMap::NormalizeDataBase() : **FileMap**
- FileMap::QueryKeyRec() : **FileMap**
- FileMap::ResetKeyRec() : **FileMap**
- FileMap::SaveDataBase() : **FileMap**
- FileMap::UpdateRec() : **FileMap**
- FileMap::ViewDataBase() : **FileMap**
- FileMap::~FileMap() : **FileMap**

- FileOp : **FileRecord**
- FileOp::FileOp() : **FileOp**
- FileOp::~FileOp() : **FileOp**
- FileRecord::FileRecord() : **FileRecord**
- FileRecord::get__EntryTime() : **FileRecord**
- FileRecord::get__HostName() : **FileRecord**
- FileRecord::get__Mean() : **FileRecord**
- FileRecord::get__Nsamples() : **FileRecord**
- FileRecord::get__Tag() : **FileRecord**
- FileRecord::get__Variance() : **FileRecord**
- FileRecord::get__VstringIDs() : **FileRecord**
- FileRecord::printRecord() : **FileRecord**
- FileRecord::set__EntryTime() : **FileRecord**
- FileRecord::set__HostName() : **FileRecord**
- FileRecord::set__Mean() : **FileRecord**
- FileRecord::set__Nsamples() : **FileRecord**
- FileRecord::set__Tag() : **FileRecord**
- FileRecord::set__Variance() : **FileRecord**
- FileRecord::set__VstringIDs() : **FileRecord**
- FileRecord::set_all() : **FileRecord**
- FileRecord::~FileRecord() : **FileRecord**
- finalize() : **Machine**
- flag : **Machine**
- FLOAT_DELTA : **Machine**
- flushMsgs() : **Machine**
- Fmap : **FileMap**

- g -

- GCL_MW_COMM : **MW_Graph**
- getCurrentProcessID() : **MW_Graph**, **Machine**
- getDisplayProcessorName() : **Machine**
- GetFileNameIn() : **FileOp**
- GetFileNameOut() : **FileOp**
- getHostName() : **Machine**
- getInverseBandwidths() : **Machine**
- getMsgHeader() : **MW_Graph**
- getNextPepMsgType() : **Machine**

- getNumProcesses() : **Machine**
- getProcessorName() : **Machine**
- getProcessorSpeed() : **Machine**
- getTokenMsgSize() : **MW_Graph**

- h -

- hypotheticalSize() : **Machine**
- hypotheticalTime() : **Machine**

- i -

- initialize() : **Machine**

- m -

- Machine() : **Machine**
- meanInvBandwidths : **Machine**
- meanLatencies : **Machine**
- MIDWARE_COMM : **Machine**
- MPI_GROUP_WORLD : **Machine**
- MPI_MsgHeader : **MW_Graph**
- MPI_PEP_REAL : **Machine**
- MW_DATATYPE_NULL : **Machine**
- MW_Graph() : **MW_Graph**
- MW_Root : **Machine**
- MW_Tester : **MW_Graph**

- n -

- nbRcvCPMessage() : **Machine**
- nbRcvCPParameters() : **Machine**
- nbRcvPepMessage() : **Machine**
- noMoreMessages() : **Machine**
- NoSuchString() : **NoSuchString**
- NP : **STAT**
- NRT : **STAT**
- numProcesses : **Machine**

- o -

- OpenReadFile() : **FileOp**
- OpenWriteFile() : **FileOp**

- p -

- PARM_FLAG : **parm**
- PEP_BARRIER_COMM : **Machine**
- PEP_BARRIER_GROUP : **Machine**
- PEP_COMM : **Machine**
- PEP_MW_COMM : **Machine**
- pepBarrierSynch() : **Machine**
- pepCommsCreated : **Machine**
- PP1 : **STAT**
- PP2 : **STAT**
- PP_CYCLES : **parm**
- PP_MSG_SIZE : **parm**
- PPSZ : **STAT**
- predictSize() : **Machine**
- predictTime() : **Machine**
- PrintPepHistoryFile() : **FileOp**
- probeCPMessage() : **Machine**
- probeGCLMessage() : **Machine**
- probePepMessage() : **Machine**
- probePepMWMessage() : **Machine**
- processorName : **Machine**

- r -

- rank : **Machine**
- rcvAssignment() : **Machine**
- rcvDataRequest() : **Machine**
- rcvExtantMessage() : **Machine**
- rcvPepMsg() : **Machine**
- rcvTrackedData() : **Machine**
- ReadRecord() : **FileOp**

- readScheduleStateFile() : **Machine**
- receiveToken() : **MW_Graph**
- recordPerformanceData() : **Machine**

- s -

- SAMPLE_SIZE : **parm**
- SearchString() : **NoSuchString**
- SEED : **parm**
- sendCapacityUpdate() : **MW_Graph**
- sendCPMessage() : **Machine**
- sendCPParameters() : **Machine**
- sendPepAssignmentCmd() : **Machine**
- sendPepMessage() : **Machine**
- sendToken() : **MW_Graph**
- sndAssignment() : **Machine**
- sndDataRequest() : **Machine**
- sndPepMsg() : **Machine**
- sndTrackedData() : **Machine**
- SPSZ : **STAT**
- STAT_FLAG : **parm**
- status : **Machine**
- STV : **STAT**

- t -

- tmpSource : **Machine**
- tmpTag : **Machine**
- TokenMark() : **MW_Graph**
- TokenProcessName : **FileMap**

- w -

- WriteRecord() : **FileOp**
- writeScheduleStateFile() : **Machine**

- ~ -

- ~Machine() : **Machine**
- ~MW_Graph() : **MW_Graph**

Generated on Mon Oct 7 01:04:30 2002 for PGMT by doxygen 1.2.18

PGMT File Members

a | b | c | d | f | m | n | o | p | r | s | u | w

Here is a list of all file members with links to the files they belong to:

- a -

- A_REASSIGN_OFF : **Machine.doxygen.rh.h**
- A_REASSIGN_ON : **Machine.doxygen.rh.h**
- ALARM : **Machine.doxygen.rh.h**

- b -

- BandWidth() : **computeNetwork.doxygen.h**
- build_parm_MPI_datatype() : **computeNetwork.doxygen.h**
- build_SEND_STAT() : **computeNetwork.doxygen.h**

- c -

- ConstPepIter : **Machine.doxygen.rh.h**

- d -

- DATA : **Machine.doxygen.rh.h**
- DATA_REQUEST : **Machine.doxygen.rh.h**

- f -

- FP : **Machine.doxygen.rh.h**
- fulltime : **Machine.doxygen.rh.h**
- FW : **Machine.doxygen.rh.h**

- m -

- Machine_h : **Machine.doxygen.rh.h**
- moment() : **computeNetwork.doxygen.h**
- MPIKeyWords : **Machine.doxygen.rh.h**
- msgWrite() : **Machine.doxygen.rh.h**
- MW_BACK : **Machine.doxygen.rh.h**
- MW_Datatype : **Machine.doxygen.rh.h**
- MW_MASTER : **Machine.doxygen.rh.h**
- MW_QUIT : **Machine.doxygen.rh.h**
- MW_SYNC : **Machine.doxygen.rh.h**
- MW_TIME : **Machine.doxygen.rh.h**
- MW_TRIP : **Machine.doxygen.rh.h**
- MW_WAIT : **Machine.doxygen.rh.h**

- n -

- NONE : **Machine.doxygen.rh.h**

- o -

- oo_Bgn : **Machine.doxygen.rh.h**
- oo_dSR : **Machine.doxygen.rh.h**
- oo_dZ : **Machine.doxygen.rh.h**
- oo_LdR : **Machine.doxygen.rh.h**
- oo_LdS : **Machine.doxygen.rh.h**
- oo_MW_clock_skew : **Machine.doxygen.rh.h**
- oo_myTick : **Machine.doxygen.rh.h**
- oo_noTime : **Machine.doxygen.rh.h**
- oo_Rk_Sk : **Machine.doxygen.rh.h**
- oo_Ro_So : **Machine.doxygen.rh.h**
- oo_Zk : **Machine.doxygen.rh.h**
- operator<() : **history.doxygen.h**

- p -

- Pep_PriorityType : **Machine.doxygen.rh.h**
- PepIter : **Machine.doxygen.rh.h**
- PepMsgType : **Machine.doxygen.rh.h**

- PGMT_ANY_SOURCE : **Machine.doxygen.rh.h**
- PGMT_ANY_TAG : **Machine.doxygen.rh.h**
- PGMT_BYTE : **Machine.doxygen.rh.h**
- PGMT_CHAR : **Machine.doxygen.rh.h**
- PGMT_COMM : **Machine.doxygen.rh.h**
- PGMT_DEFAULT_TAG : **Machine.doxygen.rh.h**
- PGMT_DOUBLE : **Machine.doxygen.rh.h**
- PGMT_FLOAT : **Machine.doxygen.rh.h**
- PGMT_INT : **Machine.doxygen.rh.h**
- PGMT_LONG : **Machine.doxygen.rh.h**
- PGMT_LONG_DOUBLE : **Machine.doxygen.rh.h**
- PGMT_SHORT : **Machine.doxygen.rh.h**
- PGMT_UNSIGNED : **Machine.doxygen.rh.h**
- PGMT_UNSIGNED_CHAR : **Machine.doxygen.rh.h**
- PGMT_UNSIGNED_LONG : **Machine.doxygen.rh.h**
- PGMT_UNSIGNED_SHORT : **Machine.doxygen.rh.h**
- PGMTTags : **Machine.doxygen.rh.h**
- PGMTTypes : **Machine.doxygen.rh.h**
- PingPong() : **computeNetwork.doxygen.h**
- PMsgType : **Machine.doxygen.rh.h**

- r -

- ReadPrm() : **computeNetwork.doxygen.h**
- REASSIGNMENT : **Machine.doxygen.rh.h**
- REASSIGNMENT_COMPLETE : **Machine.doxygen.rh.h**
- REASSURANCE : **Machine.doxygen.rh.h**

- s -

- SendVector() : **computeNetwork.doxygen.h**
- StringsInID : **history.doxygen.h**
- SynchConstants : **Machine.doxygen.rh.h**

- u -

- UPDATE_PARAMETERS : **Machine.doxygen.rh.h**
- USER_REASSIGNMENT : **Machine.doxygen.rh.h**

- USER_REASSIGNMENT_COMPLETE : **Machine.doxygen.rh.h**

- w -

- WriteScreen() : **computeNetwork.doxygen.h**

Generated on Mon Oct 7 01:04:31 2002 for PGMT by doxygen 1.2.18

Machine.doxygen.rh.h File Reference

Detailed Description

Definition in file **Machine.doxygen.rh.h**.`#include "mpi.h"`
`#include "Middleware/Mtool_aux.h"`
`#include "global.h"`

Go to the source code of this file.

Compounds

class **Machine**
class Machine More...

class **NoSuchString**

Defines

#define **Machine_h** 1

Typedefs

typedef MPI_Datatype **MW_Datatype**
Wrapper for MPI_Datatype.

typedef int **Pep_PriorityType**
Typedef of int for Pep_PriorityType.

typedef vector< Tracker >::const_iterator **ConstPepIter**
Typedef of vector<Tracker>::const_iterator for ConstPep Iter.

typedef Real(* **FP**)(const Tracker)

Typedef of Real (FP)(const Tracker) for FP.*

typedef Real **PMsgType**
Typedef of Real for PMsgType.

typedef void(* **FW**)(const Real, **PMsgType** *)
Typedef of void ( ) (const Real, PMsgType *) for FW.*

typedef long long int **fulltime**
Typedef of long long int for fulltime.

typedef vector< **PMsgType** >::iterator **PepIter**
Typedef of vector<MsgPair>::iterator for PepIter.

Enumerations

enum **PGMTTags** { **PGMT_ANY_TAG** = MPI_ANY_TAG, **PGMT_DEFAULT_TAG** = 32111, **UPDATE_PARAMETERS** }
Enumerated type for the various message tags used by PGMT. More...

enum **MPIKeyWords** { **PGMT_ANY_SOURCE** = MPI_ANY_SOURCE, **PGMT_COMM** = MPI_COMM_WORLD }
Enumerated type to act as wrapper for MPI_ANY_SOURCE and MPI_COMM_WORLD. More...

enum **PGMTTypes** {
PGMT_INT = MPI_INT, **PGMT_SHORT** = MPI_SHORT, **PGMT_CHAR** = MPI_CHAR, **PGMT_LONG** = MPI_LONG,
PGMT_UNSIGNED_CHAR = MPI_UNSIGNED_CHAR,
PGMT_UNSIGNED_SHORT = MPI_UNSIGNED_SHORT, **PGMT_UNSIGNED** = MPI_UNSIGNED, **PGMT_UNSIGNED_LONG** = MPI_UNSIGNED_LONG,
PGMT_FLOAT = MPI_FLOAT, **PGMT_DOUBLE** = MPI_DOUBLE,
PGMT_LONG_DOUBLE = MPI_LONG_DOUBLE, **PGMT_BYTE** = MPI_BYTE
}
Enumerated type to act a wrapper for MPI's datatypes (e.g. MPI_SHORT). More...

enum **PepMsgType** {
NONE = 0, **ALARM**, **REASSURANCE**, **REASSIGNMENT_COMPLETE**,
USER_REASSIGNMENT_COMPLETE, **DATA_REQUEST**, **DATA**, **REASSIGNMENT**,
USER_REASSIGNMENT, **A_REASSIGN_ON**, **A_REASSIGN_OFF**
}
Enumerated type to indicate the type of Pep message that has been received. More...

```
enum SynchConstants {
    MW_MASTER = 1, MW_TIME, MW_QUIT, MW_BACK,
    MW_SYNC, MW_WAIT, MW_TRIP, oo_noTime = 10,
    oo_Bgn, oo_LdS, oo_LdR, oo_dZ,
    oo_Zk, oo_MW_clock_skew, oo_dSR, oo_Ro_So,
    oo_Rk_Sk, oo_myTick
  }
```

Functions

void **msgWrite** (Real val, **PMsgType** *loc)

Define Documentation

#define Machine_h 1

Definition at line **44** of file **Machine.doxygen.rh.h**.

Typedef Documentation

typedef vector<Tracker>::const_iterator ConstPepIter

Typedef of vector<Tracker>::const_iterator for ConstPep Iter.

Definition at line **84** of file **Machine.doxygen.rh.h**.

typedef Real(* FP)(const Tracker)

Typedef of Real (* FP)(const Tracker) for FP.

Definition at line **87** of file **Machine.doxygen.rh.h**.

typedef long long int fulltime

Typedef of long long int for fulltime.

Definition at line **96** of file **Machine.doxygen.rh.h**.

typedef void(* FW)(const Real, PMsgType *)

Typedef of void (*) (const Real, PMsgType *) for FW.

Definition at line **93** of file **Machine.doxygen.rh.h**.

typedef MPI_Datatype MW_Datatype

Wrapper for MPI_Datatype.

Definition at line **72** of file **Machine.doxygen.rh.h**.

typedef int Pep_PriorityType

Typedef of int for Pep_PriorityType.

Definition at line **75** of file **Machine.doxygen.rh.h**.

typedef vector<PMsgType>::iterator PepIter

Typedef of vector<MsgPair>::iterator for PepIter.

Definition at line **120** of file **Machine.doxygen.rh.h**.

typedef Real PMsgType

Typedef of Real for PMsgType.

Definition at line **90** of file **Machine.doxygen.rh.h**.

Referenced by **msgWrite()**.

Enumeration Type Documentation

enum MPIKeyWords

Enumerated type to act as wrapper for MPI_ANY_SOURCE and MPI_COMM_WORLD.

Enumeration values:

PGMT_ANY_SOURCE
PGMT_COMM

Definition at line **57** of file **Machine.doxygen.rh.h**.

```
00057                { PGMT_ANY_SOURCE=MPI_ANY_SOURCE,
00058                  PGMT_COMM=MPI_COMM_WORLD } MPIKeyWords;
```

enum PepMsgType

Enumerated type to indicate the type of Pep message that has been received.

Enumeration values:

NONE
ALARM
REASSURANCE
REASSIGNMENT_COMPLETE
USER_REASSIGNMENT_COMPLETE
DATA_REQUEST
DATA
REASSIGNMENT
USER_REASSIGNMENT
A_REASSIGN_ON
A_REASSIGN_OFF

Definition at line **78** of file **Machine.doxygen.rh.h**.

```
00078             { NONE=0, ALARM, REASSURANCE, REASSIGNMENT_COMPLETE,
00079                 USER_REASSIGNMENT_COMPLETE, DATA_REQUEST, DATA,
00080                 REASSIGNMENT, USER_REASSIGNMENT,
00081                 A_REASSIGN_ON, A_REASSIGN_OFF } PepMsgType;
```

enum PGMTTags

Enumerated type for the various message tags used by PGMT.

Enumeration values:

PGMT_ANY_TAG
PGMT_DEFAULT_TAG
UPDATE_PARAMETERS

Definition at line **53** of file **Machine.doxygen.rh.h**.

```
00053              { PGMT_ANY_TAG=MPI_ANY_TAG, PGMT_DEFAULT_TAG=32111,
00054                UPDATE_PARAMETERS } PGMTTags;
```

enum PGMTTypes

Enumerated type to act a wrapper for MPI's datatypes (e.g. MPI_SHORT).

Enumeration values:

PGMT_INT
PGMT_SHORT
PGMT_CHAR
PGMT_LONG
PGMT_UNSIGNED_CHAR
PGMT_UNSIGNED_SHORT
PGMT_UNSIGNED
PGMT_UNSIGNED_LONG
PGMT_FLOAT
PGMT_DOUBLE
PGMT_LONG_DOUBLE
PGMT_BYTE

Definition at line **61** of file **Machine.doxygen.rh.h**.

```
00061                { PGMT_INT=MPI_INT, PGMT_SHORT=MPI_SHORT,
00062                  PGMT_CHAR=MPI_CHAR, PGMT_LONG=MPI_LONG,
00063                  PGMT_UNSIGNED_CHAR=MPI_UNSIGNED_CHAR,
00064                  PGMT_UNSIGNED_SHORT=MPI_UNSIGNED_SHORT,
00065                  PGMT_UNSIGNED=MPI_UNSIGNED,
00066                  PGMT_UNSIGNED_LONG=MPI_UNSIGNED_LONG,
00067                  PGMT_FLOAT=MPI_FLOAT, PGMT_DOUBLE=MPI_DOUBLE,
00068                  PGMT_LONG_DOUBLE=MPI_LONG_DOUBLE,
00069                  PGMT_BYTE=MPI_BYTE } PGMTTypes;
```

enum SynchConstants

Enumeration values:

MW_MASTER
MW_TIME
MW_QUIT
MW_BACK
MW_SYNC
MW_WAIT
MW_TRIP
oo_noTime
oo_Bgn
oo_LdS
oo_LdR
oo_dZ

oo_Zk
oo_MW_clock_skew
oo_dSR
oo_Ro_So
oo_Rk_Sk
oo_myTick

Definition at line **98** of file **Machine.doxygen.rh.h**.

```
00098                { MW_MASTER=1, MW_TIME, MW_QUIT, MW_BACK, MW_SYNC, MW_WAIT,
00099                  MW_TRIP, oo_noTime=10, oo_Bgn, oo_LdS, oo_LdR, oo_dZ,
00100                  oo_Zk, oo_MW_clock_skew, oo_dSR, oo_Ro_So, oo_Rk_Sk,
00101                  oo_myTick } SynchConstants;
```

Function Documentation

void msgWrite(Real *val,*
PMsgType * *loc*
) [inline]

Definition at line **820** of file **Machine.doxygen.rh.h**.

References **PMsgType**.

```
00820 {*loc = val;}
```

Generated on Mon Oct 7 01:04:29 2002 for PGMT by doxygen 1.2.18

Machine Class Reference

`#include <Machine.doxygen.rh.h>`

List of all members.

Detailed Description

class Machine

Serves as interface for all middleware functions. One of the primary functions of middleware is to perform message passing for the CommandProgram, the Graph and the Pep. Additionally, the middleware provides the capability of performing barrier synchronizations. All details of the underlying implementation (MPI in this case) are to remain hidden. In the current implementation two seperate communicators are used for sending messages, one for the Command Program processes and one for the Pep processes. Both communicators include all the MPI processes in their MPI Groups. A third communicator needs to be implemented that would only include the processes in the GraphRanks vector which can be used for barrier synchronization across the Pep processes. Throughout the following methods the parameter rsource is a OUT parameter indicating the source of the message just received and rtag is an OUT parameter indicating the tag of the message just received.

Definition at line **143** of file **Machine.doxygen.rh.h**.

Static Public Methods

void **getInverseBandwidths** (vector< vector< Real > > &nanosecPerByte, vector< vector< Real > > &Variances, vector< vector< Real > > &latencyTime, vector< vector< Real > > &latencyVariances)
function void getInverseBandwidths(...)

PepMsgType getNextPepMsgType ()
*function PepMsgType **getNextPepMsgType()**;*

MW_ProcessID **rcvTrackedData** (int numContainers, vector< **PepIter** > &iter, const vector< **FW** > &realWriter)
function MW_ProcessID rcvTrackedData(...)

void **sndTrackedData** (**PepMsgType** pepMsgType, int numContainers, const vector< int > &numInContainer, vector< **ConstPepIter** > &iter, const vector< **FP** > &realReader, const vector< MW_ProcessID > &recipients)
void **rcvAssignment** (vector< MW_ProcessID > &assignment, vector< **Pep_PriorityType** > &priorities, vector< pair< int, int > > &gene)
function void rcvAssignment(...)

void **sndAssignment** (**PepMsgType** msgType, const vector< MW_ProcessID > &assignment, const vector< **Pep_PriorityType** > &priorities, const vector< pair< int, int > > &gene, const vector< MW_ProcessID > &recipients)
function void sndAssignment(...)

int **rcvPepMsg** ()
*function int **rcvPepMsg()**;*

void **sndPepMsg** (**PepMsgType** msgType, int msg, const vector< MW_ProcessID > &recipients)
function void sndPepMsg(...);

Real **rcvDataRequest** ()
*function void **rcvDataRequest()**;*

void **sndDataRequest** (**PepMsgType** pepMsgType, const vector< Real > &times, const vector< MW_ProcessID > &recipients)
function void sndDataRequest(...)'

bool **flushMsgs** ()
called in Machine::Finalize() to flush extant messages

bool **rcvExtantMessage** (MPI_Comm comm)
*called by **flushMsgs()**;*

const int **getNumProcesses** ()
*function const int **getNumProcesses()***

void **initialize** (int *argc, char **argv[])
function void initialize(...)

void **sendPepAssignmentCmd** (char *fname, int dest, int tag)
void **sendPepMessage** (void *buf, int count, int datatype, int dest, int tag)
function sendPepMessage(...)

void **sendPepMessage** (void *buf, int count, int datatype, int dest)
function void sendPepMessage(...)

bool **nbRcvPepMessage** (void *buf, int count, int datatype, int source, int tag, int *rtag, int *rsource)
function bool nbRcvPepMessage(...)

bool **nbRcvPepMessage** (void *buf, int count, int datatype, int *rtag, int *rsource)
function nbRcvPepMessage(...)

void **bRcvPepMessage** (void *buf, int count, int datatype, int source, int tag, int *rtag, int *rsource)
function void bRcvPepMessage(...)

void **bRcvPepMessage** (void *buf, int count, int datatype, int *rtag, int *rsource)
function void bRcvPepMessage(...)

bool **probePepMessage** (int source, int tag, int *rtag, int *rsource)
function bool probePepMessage(...)

bool **probePepMessage** (int *rtag, int *rsource)
function bool probePepMessage(...)

void **barrierSynch** (void *buf, int count, int datatype, int dest, int tag, int comm)
function barrierSynch(...)

void **pepBarrierSynch** (void *buf, int count, int datatype, int dest, int tag, int comm)
function void pepBarrierSynch(...)

void **sendCPMessage** (void *buf, int count, int datatype, int dest, int tag)
function void sendCPMessage(...)

void **sendCPMessage** (void *buf, int count, int datatype, int dest)
function void send CPMessage(...)

void **sendCPParameters** (void *buf, int dest)
bool **nbRcvCPParameters** (void *buf, int source, int *rsource)
void **bRcvCPParameters** (void *buf, int source, int *rsource)
bool **nbRcvCPMessage** (void *buf, int count, int datatype, int source, int tag, int *rtag, int *rsource)
function bool nbRcvCPMessage(...)

bool **nbRcvCPMessage** (void *buf, int count, int datatype, int *rtag, int *rsource)
function bool nbRcvCPMessage(...)

void **bRcvCPMessage** (void *buf, int count, int datatype, int source, int tag, int *rtag, int *rsource)
function void bRcvCPMessage(...)

string & **getProcessorName** ()
*function string& **getProcessorName()***

void **bRcvCPMessage** (void *buf, int count, int datatype, int *rtag, int *rsource)
function void bRcvCPMessage(...)

bool **probeCPMessage** (int source, int tag, int *rtag, int *rsource)
function bool probeCPMessage(...)

bool **probeCPMessage** (int *rtag, int *rsource)
function bool probeCPMessage(...)

void **createPepComm** (MW_ProcessID ranks[], int size)
function void createPepComm(...)

void **finalize** ()
*function void **finalize()***

Used for debuggin to test if there are any outstanding messages for a particular process on any of its communicators This will probably not be made available to the Command Program writer bool **noMoreMessages** ()
Test to be sure queue is flushed before shutting down.

string & **getDisplayProcessorName** ()
MW_ProcessID **getCurrentProcessID** ()
Get the MPI_Rank of the current process.

void **predictTime** (Real &Time, Real &Variance, unsigned int &Nsamples, MW_ProcessID ProcID, GCL_NodeID NodeID)
function int sendToken(...)

void **predictTime** (Real &Time, Real &Variance, unsigned int &Nsamples, MW_ProcessID ProcID, GCL_NodeID NodeID, vector< Real > parameterVector)
void **hypotheticalTime** (Real Time, MW_ProcessID ProcID, GCL_NodeID NodeID, unsigned int instance, const vector< Real > parameterVector)
void **predictSize** (Real &Size, Real &Variance, unsigned int &Nsamples, GCL_NodeID NodeID)
void **hypotheticalSize** (Real Size, GCL_NodeID NodeID, unsigned int instance)
void **recordPerformanceData** ()
void **writeScheduleStateFile** (vector< MW_ProcessID > assignment, vector< **Pep_PriorityType** > priorities, char *outFileName)
void **readScheduleStateFile** (vector< MW_ProcessID > &assignment, vector< **Pep_PriorityType** > &priorities, char *inFileName)
bool **probePepMWMessage** (int *rtag, int *rsource)
bool probePepMWMessage(...)

bool **probeGCLMessage** (int *rtag, int *rsource)
function bool probeGCLMessage(...)

void **BuildProcessorVector** ()
void **BuildProcessorSpeedVector** ()
*function void **BuildProcessorSpeedVector()**;*

void **BuildCommunicator** (MPI_Comm *MPI_COMM)
string **getHostName** (int **rank**)
function string getHostName(rank)

double **getProcessorSpeed** (int **rank**)
double **busyLoop** (double &result)

Static Public Attributes

MW_ProcessID **rank**
MPI_Comm **MIDWARE_COMM**
int **numProcesses**
string * **processorName**
string * **displayProcessorName**
const **MW_Datatype** **MW_DATATYPE_NULL**
vector< vector< Real > > **meanLatencies**
vector< vector< Real > > **meanInvBandwidths**
const int **MW_Root**
const MPI_Datatype **MPI_PEP_REAL**
const Real **FLOAT_DELTA**
vector< string > **_ProcessorVector**

Protected Methods

Machine ()
~Machine ()

Static Protected Attributes

int **flag**
MPI_Status **status**
MPI_Comm **PEP_COMM**
MPI_Comm **PEP_MW_COMM**
MPI_Group **MPI_GROUP_WORLD**
MPI_Group **PEP_BARRIER_GROUP**
MPI_Comm **PEP_BARRIER_COMM**
bool **pepCommsCreated**
int **tmpTag**
int **tmpSource**

Constructor & Destructor Documentation

Machine::Machine( ) [protected]

Machine::~Machine( ) [protected]

Member Function Documentation

```
void Machine::barrierSynch( void *  buf,
                            int     count,
                            int     datatype,
                            int     dest,
                            int     tag,
                            int     comm
                          ) [static]
```

function barrierSynch(...)

Perform a barrier sychronization across all MPI processes. No process will exit this method until all the processes have entered it.

```
void Machine::bRcvCPMessage( void *  buf,
                             int     count,
                             int     datatype,
                             int *   rtag,
                             int *   rsource
                           ) [static]
```

function void bRcvCPMessage(...)

Receive a CommandProgram message from any source with any tag sent by **sendCPMessage()**. The receive is blocking. By this we mean that the function will not return until a message is received.

```
void Machine::bRcvCPMessage( void *   buf,
                             int      count,
                             int      datatype,
                             int      source,
                             int      tag,
                             int *    rtag,
                             int *    rsource
                            ) [static]
```

function void bRcvCPMessage(...)

Receive a CommandProgram message from a particular source with a particular tag sent by **sendCPMessage()**. The receive is blocking. By this we mean that the function will not return until a message from the appropriate source with the appropriate tag is received. Source or tag can be wild-carded by passing "PGMT_ANY_SOURCE" or "PGMT_ANY_TAG".

```
void Machine::bRcvCPParameters( void *   buf,
                                int      source,
                                int *    rsource
                               ) [static]
```

```
void Machine::bRcvPepMessage( void *   buf,
                              int      count,
                              int      datatype,
                              int *    rtag,
                              int *    rsource
                             ) [static]
```

function void bRcvPepMessage(...)

Receive a Pep message from any source with any tag sent by **sendPepMessage()**. The receive is blocking. By this we mean that the function will not return until a message is received.

```
void Machine::bRcvPepMessage( void *   buf,
                              int      count,
                              int      datatype,
                              int      source,
                              int      tag,
                              int *    rtag,
                              int *    rsource
                             ) [static]
```

function void bRcvPepMessage(...)

Receive a Pep message from a particular source with a particular tag sent by **sendPepMessage()**. The receive is blocking. By this we mean that the function will not return until a message from the appropriate source with the appropriate tag is received. Source or tag can be wild-carded by passing "PGMT_ANY_SOURCE" or "PGMT_ANY_ TAG".

```
void Machine::BuildCommunicator( MPI_Comm *   MPI_COMM) [static]
```

```
void Machine::BuildProcessorSpeedVector( ) [static]
```

function void **BuildProcessorSpeedVector()**;

```
void Machine::BuildProcessorVector( ) [static]
```

```
double Machine::busyLoop( double &   result) [static]
```

void Machine::createPepComm(MW_ProcessID *ranks*[], int *size*) [static]

function void createPepComm(...)

Create the Pep communicator for the Pep. This needs work! It should not be called from Pep::initialize() but rather from the Graph constructor (as long as every process builds a copy of the (same hopefully) graph. Note: MPI_Comm_create() creates a communicator from an existing communicator that has a Group of processes that is a superset of the processes desired for the new communicator. BUT, it performs a barrier sychronization to create the new communicator over all the processes in the OLD communicator (not just those in the new communicator).

void Machine::finalize( ) [static]

function void **finalize()**

Destroy the middleware layer. Call MPI_Finalize() among other things.

bool Machine::flushMsgs( ) [static]

called in Machine::Finalize() to flush extant messages

MW_ProcessID Machine::getCurrentProcessID( ) [static]

Get the MPI_Rank of the current process.

string& Machine::getDisplayProcessorName( ) [static]

string Machine::getHostName(int *rank*) [static]

function string getHostName(rank)

This function returns the name of the machine host as a function of the process rank.

void Machine::getInverseBandwidths(vector< vector< Real > > & *nanosecPerByte*, vector< vector< Real > > & *Variances*, vector< vector< Real > > & *latencyTime*, vector< vector< Real > > & *latencyVariances*) [static]

function void getInverseBandwidths(...)

Assumptions for the function getInverseBandwidths(..).
1. The values returned for latency and inverse bandwidths are independent of the calling context. They may be called from any process after the middleware is initialized. Latency is defined as the time for passing a message in the limit of small messages. The units are nanoseconds (per message) and nanoseconds per byte, respectively.
2. The matrix (vector<vector<Real>) is indexed on both dimensions by MW_ProcessID's of the communicating processes.
3. The values will be stored into the matrices ("vector<vector<Real> >") which are supplied by reference. The vectors may be resized to zero and the values entered via the method push_back().

PepMsgType Machine::getNextPepMsgType( ) [static]

function PepMsgType **getNextPepMsgType()**;

Assumptions for **getNextPepMsgType()**: 1. The underlying assumption is that all the messages sent/received by the methods below will arrive at the destination and go into a single message queue. This call is made to determine the next message to be received. It will notify the recipient only of the messages sent by methods detailed below and will be called prior to calling one of the receiving methods below.
2. Each of the sending methods below will pass a value for the pep message type which **getNextPepMsgType()** will return when called.
3. Returns NULL if there is no next message.

This function probes any incoming messages on the communicator PEP_MW_COMM, and returns PepMsgType, which is encoded as the message tag. NULL is returned if there is no incoming message.

const int Machine::getNumProcesses( ) [inline, static]

function const int **getNumProcesses()**

Get the number of MPI processes in the application (this is a fixed number for the life of the application). Any processes spawned by the Command Program writer (e.g. JFrontEnd) are not MPI processes and do not affect the value returned by get_numProcesses().

Definition at line **284** of file **Machine.doxygen.rh.h**.

References **numProcesses**.

```
00284                                   {
00285     return numProcesses;
00286   }
```

string& Machine::getProcessorName( ) [static]

function string& **getProcessorName()**

Calls MPI_Get_processor_name(). It is currently invoked in CP::run().

double Machine::getProcessorSpeed(int *rank***) [static]**

void Machine::hypotheticalSize(Real *Size*,
GCL_NodeID *NodeID*,
unsigned int *instance*
) [static]

void Machine::hypotheticalTime(Real *Time*,
MW_ProcessID *ProcID*,
GCL_NodeID *NodeID*,
unsigned int *instance*,
const vector< Real > *parameterVector*
) [static]

void Machine::initialize(int * *argc*,
char **** *argv*[]
) [static]

function void initialize(...)

Initialize the middleware layer. It is called in main before the Pep is initialized and before the Command Program is instantiated. It initializes MPI.

bool Machine::nbRcvCPMessage(void * *buf*,
int *count*,
int *datatype*,
int * *rtag*,
int * *rsource*
) [static]

function bool nbRcvCPMessage(...)

Receive a CommandProgram message from any source with any tag sent by **sendPepMessage()**. The receive is non-blocking. By this we mean that the function will return immediately if there are currently no messages in the queue.

```
bool Machine::nbRcvCPMessage( void *  buf,
                              int     count,
                              int     datatype,
                              int     source,
                              int     tag,
                              int *   rtag,
                              int *   rsource
                             ) [static]
```

function bool nbRcvCPMessage(...)

Receive a CommandProgram message from a particular source with a particular tag sent by **sendCPMessage()**. The receive is non-blocking. By this we mean that the function will return immediately if there are currently no messages in the queue. Source or tag can be wild-carded by passing "PGMT_ANY_SOURCE" or "PGMT_ANY_ TAG".

```
bool Machine::nbRcvCPParameters( void *  buf,
                                 int     source,
                                 int *   rsource
                                ) [static]
```

```
bool Machine::nbRcvPepMessage( void *  buf,
                               int     count,
                               int     datatype,
                               int *   rtag,
                               int *   rsource
                              ) [static]
```

function nbRcvPepMessage(...)

Receive a Pep message from any source with any tag sent by **sendPepMessage()**. The receive is non-blocking. By this we mean that the function will return immediately if there are currently no messages in the queue.

```
bool Machine::nbRcvPepMessage( void *  buf,
                               int     count,
                               int     datatype,
                               int     source,
                               int     tag,
                               int *   rtag,
                               int *   rsource
                              ) [static]
```

function bool nbRcvPepMessage(...)

Receive a Pep message from a particular source with a particular tag sent by **sendPepMessage()**. The receive is non-blocking. By this we mean that the function will return immediately if there are currently no messages in the queue. Source or tag can be wild-carded by passing "PGMT_ANY_SOURCE" or "PGMT_ANY_TAG".

Used for debuggin to test if there are any outstanding messages for a particular process on any of its communicators This will probably not be made available to the Command Program writer bool Machine::noMoreMessages( ) [static]

Test to be sure queue is flushed before shutting down.

```
void Machine::pepBarrierSynch( void *  buf,
                               int     count,
                               int     datatype,
                               int     dest,
                               int     tag,
                               int     comm
                              ) [static]
```

function void pepBarrierSynch(...)

Perform a barrier sychronization across all Pep processes. No process will exit this method until all the Pep processes have entered it. Not yet implemented.

```
void Machine::predictSize( Real &          Size,
                           Real &          Variance,
                           unsigned int &  Nsamples,
                           GCL_NodeID      NodeID
                          ) [static]
```

```
void Machine::predictTime( Real &          Time,
                           Real &          Variance,
                           unsigned int &  Nsamples,
                           MW_ProcessID    ProcID,
                           GCL_NodeID      NodeID,
                           vector< Real >  parameterVector
                          ) [static]
```

```
void Machine::predictTime( Real &          Time,
                           Real &          Variance,
                           unsigned int &  Nsamples,
                           MW_ProcessID    ProcID,
                           GCL_NodeID      NodeID
                          ) [static]
```

function int sendToken(...)

```
bool Machine::probeCPMessage( int *  rtag,
                              int *  rsource
                             ) [static]
```

function bool probeCPMessage(...)

Check to see if there is a CommandProgram message in the queue regardless of source or tag. Return value indicates whether or not there is a message. As elsewhere, rsource is an OUT parameter indicating the source of the message (if there is one) and rtag indicates its tag.

```
bool Machine::probeCPMessage( int    source,
                              int    tag,
                              int *  rtag,
                              int *  rsource
                             ) [static]
```

function bool probeCPMessage(...)

Check to see if there is a CommandProgram message in the queue with a particular source and a particular tag. Return value indicates whether or not there is a message. As elsewhere, rsource is an OUT parameter indicating the source of the message (if there is one) and rtag indicates its tag. Source or tag can be wild-carded by passing "PGMT_ANY_SOURCE" or "PGMT_ANY_ TAG".

```
bool Machine::probeGCLMessage( int *  rtag,
                               int *  rsource
                              ) [static]
```

function bool probeGCLMessage(...)

Check to see if there is a Pep message in the queue regardless of source or tag. Return value indicates whether or not there is a message. As elsewhere, rsource is an OUT parameter indicating the source of the message (if there is one) and rtag indicates its tag.

```
bool Machine::probePepMessage( int *  rtag,
                               int *  rsource
                              ) [static]
```

function bool probePepMessage(...)

Check to see if there is a Pep message in the queue regardless of source or tag. Return value indicates whether or not there is a message. As elsewhere, rsource is an OUT parameter indicating the source of the message (if there is one) and rtag indicates its tag.

```
bool Machine::probePepMessage( int     source,
                               int     tag,
                               int *   rtag,
                               int *   rsource
                             ) [static]
```

function bool probePepMessage(...)

Check to see if there is a Pep message in the queue with a particular source and a particular tag. Return value indicates whether or not there is a message. As elsewhere, rsource is an OUT parameter indicating the source of the message (if there is one) and rtag indicates its tag. Source or tag can be wild-carded by passing "PGMT_ANY_SOURCE" or "PGMT_ANY_TAG".

```
bool Machine::probePepMWMessage( int *   rtag,
                                 int *   rsource
                               ) [static]
```

bool probePepMWMessage(...)

```
void Machine::rcvAssignment( vector< MW_ProcessID > &      assignment,
                             vector< Pep_PriorityType > &  priorities,
                             vector< pair< int, int > > &  gene
                           ) [static]
```

function void rcvAssignment(...)

Notes:
1. This receive method receives a single integer message.
2. A variable number of recipients may be specified. All receive the same msg.
3. The single integer message is returned to the recipient via rcvPepMsg.

```
Real Machine::rcvDataRequest( ) [static]
```

function void **rcvDataRequest()**;

Notes on **rcvDataRequest()** and **sndDataRequest()**:
1. This pair of functions passes a Real (double) to each recipient.
2. A different Real is passed to each recipient. The two vectors passed to sndDataRequest have identical numbers of elements. They are passed by copy so that the Middleware can mutate/mutilate them or whatever.

```
bool Machine::rcvExtantMessage( MPI_Comm   comm) [static]
```

called by **flushMsgs()**;

```
int Machine::rcvPepMsg( ) [static]
```

function int **rcvPepMsg()**;

Notes on **rcvPepMsg()** and **sndPepMsg()**:
1. This send method sends a single integer message.
2. A variable number of recipients may be specified. All receive the same msg.
3. The single integer message is returned to [i.e. received by] the recipient via rcvPepMsg.

```
MW_ProcessID Machine::rcvTrackedData( int                      numContainers,
                                      vector< PepIter > &      iter,
                                      const vector< FW > &     realWriter
                                    ) [static]
```

function MW_ProcessID rcvTrackedData(...)

Note: Typedef of void (*) (const Real, PMsgType *) for FW i.e. typedef void (* FW) (const Real, PMsgType *);

Also note:
1. "numContainers" indicates the size of the vectors "numInContainer", "iter", "realReader", and "realWriter".
2. "iter[i]" is a forward iterator of the i'th Pep container object. It points to the first item in the container. In the code, iter[i] is always iterated within the ith container.
3. rcvTrackedData(...) returns the ID of the sending process.

```
void Machine::readScheduleStateFile( vector< MW_ProcessID > &      assignment,
                                     vector< Pep_PriorityType > &  priorities,
                                     char *                        inFileName
                                   ) [static]
```

```
void Machine::recordPerformanceData( ) [static]
```

```
void Machine::sendCPMessage( void *  buf,
                             int     count,
                             int     datatype,
                             int     dest
                           ) [static]
```

function void send CPMessage(...)

Send a message to a CommandProgram process using a particular tag. This method must only be called by a CommandProgram method.

```
void Machine::sendCPMessage( void *  buf,
                             int     count,
                             int     datatype,
                             int     dest,
                             int     tag
                           ) [static]
```

function void sendCPMessage(...)

Send a message to a CommandProgram process using a particular tag. This method must only be called by a CommandProgram method.

```
void Machine::sendCPParameters( void *  buf,
                                int     dest
                              ) [static]
```

```
void Machine::sendPepAssignmentCmd( char *  fname,
                                    int     dest,
                                    int     tag
                                  ) [static]
```

```
void Machine::sendPepMessage( void *  buf,
                              int     count,
                              int     datatype,
                              int     dest
                            ) [static]
```

function void sendPepMessage(...)

Send a message to a Pep process without a particular tag. This method must only be called by a Pep method. It may be called by a process that is acting as a Command Program process. For example, when a CommandProgram process calls Pep::stopGraph(), the Pep method uses send PepMessage() even though no instance of the Pep is instantiated on that process and control is returned to the CommandProgram on that process as soon as the send has completed.

```
void Machine::sendPepMessage( void *  buf,
                              int     count,
                              int     datatype,
                              int     dest,
                              int     tag
                            ) [static]
```

function sendPepMessage(...)

Send a message to a Pep process using a particular tag. This method must only be called by a Pep method. It may be called by a process that is acting as a Command Program process. For example, when a CommandProgram process calls Pep::stopGraph(), the Pep method uses send PepMessage() even though no instance of the Pep is instantiated on that process and control is returned to the CommandProgram on that process as soon as the send has completed.

```
void Machine::sndAssignment( PepMsgType                          msgType,
                             const vector< MW_ProcessID > &      assignment,
                             const vector< Pep_PriorityType > &  priorities,
                             const vector< pair< int, int > > &  gene,
                             const vector< MW_ProcessID > &      recipients
                           ) [static]
```

function void sndAssignment(...)

```
void Machine::sndDataRequest( PepMsgType                        pepMsgType,
                              const vector< Real > &            times,
                              const vector< MW_ProcessID > &    recipients
                            ) [static]
```

function void sndDataRequest(...)'

rcvDataRequest();

```
void Machine::sndPepMsg( PepMsgType                        msgType,
                         int                               msg,
                         const vector< MW_ProcessID > &    recipients
                       ) [static]
```

function void sndPepMsg(...);

Notes: 1. This send method sends a single integer message.
2. A variable number of recipients may be specified. All receive the same msg.
3. The single integer message is returned to [i.e. received by] the recipient by calling rcvPepMsg.

```
void Machine::sndTrackedData( PepMsgType                        pepMsgType,
                              int                               numContainers,
                              const vector< int > &             numInContainer,
                              vector< ConstPepIter > &          iter,
                              const vector< FP > &              realReader,
                              const vector< MW_ProcessID > &    recipients
                            ) [static]
```

```
void Machine::writeScheduleStateFile( vector< MW_ProcessID >    assignment,
                                      vector< Pep_PriorityType > priorities,
                                      char *                     outFileName
                                    ) [static]
```

Member Data Documentation

vector< string > Machine::_ProcessorVector `[static]`

Definition at line **807** of file **Machine.doxygen.rh.h**.

string* Machine::displayProcessorName `[static]`

Definition at line **676** of file **Machine.doxygen.rh.h**.

int Machine::flag `[static, protected]`

Definition at line **705** of file **Machine.doxygen.rh.h**.

const Real Machine::FLOAT_DELTA `[static]`

Definition at line **688** of file **Machine.doxygen.rh.h**.

vector< vector< Real > > Machine::meanInvBandwidths `[static]`

Definition at line **682** of file **Machine.doxygen.rh.h**.

vector< vector< Real > > Machine::meanLatencies `[static]`

Definition at line **680** of file **Machine.doxygen.rh.h**.

MPI_Comm Machine::MIDWARE_COMM `[static]`

Definition at line **663** of file **Machine.doxygen.rh.h**.

MPI_Group Machine::MPI_GROUP_WORLD `[static, protected]`

Definition at line **726** of file **Machine.doxygen.rh.h**.

const MPI_Datatype Machine::MPI_PEP_REAL `[static]`

Definition at line **686** of file **Machine.doxygen.rh.h**.

const MW_Datatype Machine::MW_DATATYPE_NULL `[static]`

Definition at line **678** of file **Machine.doxygen.rh.h**.

const int Machine::MW_Root `[static]`

Definition at line **684** of file **Machine.doxygen.rh.h**.

int Machine::numProcesses `[static]`

Definition at line **668** of file **Machine.doxygen.rh.h**.

Referenced by **getNumProcesses()**.

MPI_Comm Machine::PEP_BARRIER_COMM `[static, protected]`

Definition at line **734** of file **Machine.doxygen.rh.h**.

MPI_Group Machine::PEP_BARRIER_GROUP `[static, protected]`

Definition at line **729** of file **Machine.doxygen.rh.h**.

MPI_Comm Machine::PEP_COMM `[static, protected]`

Definition at line **715** of file **Machine.doxygen.rh.h**.

MPI_Comm Machine::PEP_MW_COMM `[static, protected]`

Definition at line **723** of file **Machine.doxygen.rh.h**.

bool Machine::pepCommsCreated `[static, protected]`

Definition at line **737** of file **Machine.doxygen.rh.h**.

string* Machine::processorName `[static]`

Definition at line **672** of file **Machine.doxygen.rh.h**.

MW_ProcessID Machine::rank `[static]`

Definition at line **657** of file **Machine.doxygen.rh.h**.

MPI_Status Machine::status `[static, protected]`

Definition at line **709** of file **Machine.doxygen.rh.h**.

int Machine::tmpSource `[static, protected]`

Definition at line **743** of file **Machine.doxygen.rh.h**.

int Machine::tmpTag `[static, protected]`

Definition at line **740** of file **Machine.doxygen.rh.h**.

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

- **Machine.doxygen.rh.h**

Generated on Mon Oct 7 01:04:30 2002 for PGMT by doxygen 1.2.18

Machine Member List

This is the complete list of members for **Machine**, including all inherited members.

_ProcessorVector	**Machine**	[static]
barrierSynch(void *buf, int count, int datatype, int dest, int tag, int comm)	**Machine**	[static]
bRcvCPMessage(void *buf, int count, int datatype, int source, int tag, int *rtag, int *rsource)	**Machine**	[static]
bRcvCPMessage(void *buf, int count, int datatype, int *rtag, int *rsource)	**Machine**	[static]
bRcvCPParameters(void *buf, int source, int *rsource)	**Machine**	[static]
bRcvPepMessage(void *buf, int count, int datatype, int source, int tag, int *rtag, int *rsource)	**Machine**	[static]
bRcvPepMessage(void *buf, int count, int datatype, int *rtag, int *rsource)	**Machine**	[static]
BuildCommunicator(MPI_Comm *MPI_COMM)	**Machine**	[static]
BuildProcessorSpeedVector()	**Machine**	[static]
BuildProcessorVector()	**Machine**	[static]
busyLoop(double &result)	**Machine**	[static]
createPepComm(MW_ProcessID ranks[], int size)	**Machine**	[static]
displayProcessorName	**Machine**	[static]
finalize()	**Machine**	[static]
flag	**Machine**	[protected, static]
FLOAT_DELTA	**Machine**	[static]
flushMsgs()	**Machine**	[static]
getCurrentProcessID()	**Machine**	[static]
getDisplayProcessorName()	**Machine**	[static]
getHostName(int rank)	**Machine**	[static]

getInverseBandwidths(vector< vector< Real > > &nanosecPerByte, vector< vector< Real > > &Variances, vector< vector< Real > > &latencyTime, vector< vector< Real > > &latencyVariances)	**Machine**	[static]
getNextPepMsgType()	**Machine**	[static]
getNumProcesses()	**Machine**	[inline, static]
getProcessorName()	**Machine**	[static]
getProcessorSpeed(int rank)	**Machine**	[static]
hypotheticalSize(Real Size, GCL_NodeID NodeID, unsigned int instance)	**Machine**	[static]
hypotheticalTime(Real Time, MW_ProcessID ProcID, GCL_NodeID NodeID, unsigned int instance, const vector< Real > parameterVector)	**Machine**	[static]
initialize(int *argc, char **argv[])	**Machine**	[static]
Machine()	**Machine**	[protected]
meanInvBandwidths	**Machine**	[static]
meanLatencies	**Machine**	[static]
MIDWARE_COMM	**Machine**	[static]
MPI_GROUP_WORLD	**Machine**	[protected, static]
MPI_PEP_REAL	**Machine**	[static]
MW_DATATYPE_NULL	**Machine**	[static]
MW_Root	**Machine**	[static]
nbRcvCPMessage(void *buf, int count, int datatype, int source, int tag, int *rtag, int *rsource)	**Machine**	[static]
nbRcvCPMessage(void *buf, int count, int datatype, int *rtag, int *rsource)	**Machine**	[static]
nbRcvCPParameters(void *buf, int source, int *rsource)	**Machine**	[static]
nbRcvPepMessage(void *buf, int count, int datatype, int source, int tag, int *rtag, int *rsource)	**Machine**	[static]
nbRcvPepMessage(void *buf, int count, int datatype, int *rtag, int *rsource)	**Machine**	[static]
noMoreMessages()	**Machine**	[static]
numProcesses	**Machine**	[static]
PEP_BARRIER_COMM	**Machine**	[protected, static]

PEP_BARRIER_GROUP	**Machine**	[protected, static]
PEP_COMM	**Machine**	[protected, static]
PEP_MW_COMM	**Machine**	[protected, static]
pepBarrierSynch(void *buf, int count, int datatype, int dest, int tag, int comm)	**Machine**	[static]
pepCommsCreated	**Machine**	[protected, static]
predictSize(Real &Size, Real &Variance, unsigned int &Nsamples, GCL_NodeID NodeID)	**Machine**	[static]
predictTime(Real &Time, Real &Variance, unsigned int &Nsamples, MW_ProcessID ProcID, GCL_NodeID NodeID)	**Machine**	[static]
predictTime(Real &Time, Real &Variance, unsigned int &Nsamples, MW_ProcessID ProcID, GCL_NodeID NodeID, vector< Real > parameterVector)	**Machine**	[static]
probeCPMessage(int source, int tag, int *rtag, int *rsource)	**Machine**	[static]
probeCPMessage(int *rtag, int *rsource)	**Machine**	[static]
probeGCLMessage(int *rtag, int *rsource)	**Machine**	[static]
probePepMessage(int source, int tag, int *rtag, int *rsource)	**Machine**	[static]
probePepMessage(int *rtag, int *rsource)	**Machine**	[static]
probePepMWMessage(int *rtag, int *rsource)	**Machine**	[static]
processorName	**Machine**	[static]
rank	**Machine**	[static]
rcvAssignment(vector< MW_ProcessID > &assignment, vector< Pep_PriorityType > &priorities, vector< pair< int, int > > &gene)	**Machine**	[static]
rcvDataRequest()	**Machine**	[static]
rcvExtantMessage(MPI_Comm comm)	**Machine**	[static]
rcvPepMsg()	**Machine**	[static]
rcvTrackedData(int numContainers, vector< PepIter > &iter, const vector< FW > &realWriter)	**Machine**	[static]
readScheduleStateFile(vector< MW_ProcessID > &assignment, vector< Pep_PriorityType > &priorities, char *inFileName)	**Machine**	[static]
recordPerformanceData()	**Machine**	[static]

sendCPMessage(void *buf, int count, int datatype, int dest, int tag)	**Machine**	[static]
sendCPMessage(void *buf, int count, int datatype, int dest)	**Machine**	[static]
sendCPParameters(void *buf, int dest)	**Machine**	[static]
sendPepAssignmentCmd(char *fname, int dest, int tag)	**Machine**	[static]
sendPepMessage(void *buf, int count, int datatype, int dest, int tag)	**Machine**	[static]
sendPepMessage(void *buf, int count, int datatype, int dest)	**Machine**	[static]
sndAssignment(PepMsgType msgType, const vector< MW_ProcessID > &assignment, const vector< Pep_PriorityType > &priorities, const vector< pair< int, int > > &gene, const vector< MW_ProcessID > &recipients)	**Machine**	[static]
sndDataRequest(PepMsgType pepMsgType, const vector< Real > &times, const vector< MW_ProcessID > &recipients)	**Machine**	[static]
sndPepMsg(PepMsgType msgType, int msg, const vector< MW_ProcessID > &recipients)	**Machine**	[static]
sndTrackedData(PepMsgType pepMsgType, int numContainers, const vector< int > &numInContainer, vector< ConstPepIter > &iter, const vector< FP > &realReader, const vector< MW_ProcessID > &recipients)	**Machine**	[static]
status	**Machine**	[protected, static]
tmpSource	**Machine**	[protected, static]
tmpTag	**Machine**	[protected, static]
writeScheduleStateFile(vector< MW_ProcessID > assignment, vector< Pep_PriorityType > priorities, char *outFileName)	**Machine**	[static]
~Machine()	**Machine**	[protected]

NoSuchString Class Reference

`#include <Machine.doxygen.rh.h>`

List of all members.

Public Methods

NoSuchString (string str)
string **SearchString** ()

Constructor & Destructor Documentation

NoSuchString::NoSuchString(string *str*) [inline]

Definition at line **107** of file **Machine.doxygen.rh.h**.

```
00107 :_SearchString( str ){}
```

Member Function Documentation

string NoSuchString::SearchString( ) [inline]

Definition at line **109** of file **Machine.doxygen.rh.h**.

```
00109                          {
00110     return _SearchString;
00111   }
```

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

- **Machine.doxygen.rh.h**

Generated on Mon Oct 7 01:04:30 2002 for PGMT by doxygen 1.2.18

NoSuchString Member List

This is the complete list of members for **NoSuchString**, including all inherited members.

NoSuchString(string str)	**NoSuchString**	`[inline]`
SearchString()	**NoSuchString**	`[inline]`

Generated on Mon Oct 7 01:04:30 2002 for PGMT by doxygen 1.2.18

computeNetwork.doxygen.h File Reference

Detailed Description

Definition in file **computeNetwork.doxygen.h**.`#include "mpi.h"`

```
#include "Machine.h"
#include <iostream>
#include <stdlib.h>
#include <string>
```

Go to the source code of this file.

Compounds

struct **parm**
Struct for setting parameter values. More...

struct **STAT**
typdef struct STAT More...

Functions

int **ReadPrm** (**parm** ***FP**)
ReadPrm(parm* FP) *reads the parameter file for the bandwidth tests.*

MPI_Datatype **build_parm_MPI_datatype** ()
MPI_Datatype ***build_parm_MPI_datatype()****;*.

STAT BandWidth (int WorldRank, int WorldSize, int SampleSize, int PP1, int PP2, int Cycles, const int PmsgSize, const int LatencySwitch, const int SEED)
Bandwidth(...).

void **PingPong** (MPI_Datatype SEND_STAT, int sample_size, int PP1, int PP2, int cycles, const int pmsg_size, const int LatencySwitch, const int SEED)
void PingPong(...)

void **build_SEND_STAT** (**STAT** *InBuffer, MPI_Datatype *SEND_STAT)
void build_SEND_STAT(...)

void **moment** (double data[], int n, double *ave, double *adev, double *sdev, double *svar, double *skew, double *curt, double *dmax, double *dmin)
void Moment(...)

int **WriteScreen** (**STAT** &Report)
WriteScreen(STAT &Report) *writes the statistics report to the screen.*

int **SendVector** (int WorldRank, int WorldSize, vector< vector< Real > > &StatVec)
Function SendVector(...).

Function Documentation

```
STAT BandWidth( int        WorldRank,
                int        WorldSize,
                int        SampleSize,
                int        PP1,
                int        PP2,
                int        Cycles,
                const int  PmsgSize,
                const int  LatencySwitch,
                const int  SEED
              )
```

Bandwidth(...).

The function **BandWidth()** measures the bandwidth between the process PP1 and PP2. The statistical Report is returned to all processes, but only the Report returned to process MW_Root contains data. On MW_Root, the results of each test are stored in the arrays nanosecsPerByte and Variances; then copies of these arrays are sent to all processes.

This function measures the time required to send a message of PmsgSize from PP1 to PP2, and receive a 1-word acknowledgement. The function **PingPong()** executes SampleSize measurements for a process pair PP1-PP2. The moments of these measurements are calculated, and the moments are sent in a **STAT** struct to MW_Root. The Reports are received in Bandwidth(), and returned to the calling program. In the calling program, the Reports for each pair of processes must be collated and scaled to build a table of interprocess bandwidths or latencies for all process pairs.

If PmsgSize is large, the average time (in secs) to send PmsgSize bytes is effectively measured, uncorrected for latency of each one-way transmission. If PmsgSize = 1, the average latency is effectively measured for two round trips one-way transmissions.

The message size for the bandwidth test in bytes is specified in the parameter file. For the latency test, we normally send a one-word message of type float. The size of the latency test message in bytes is specified by LATENCY_MSG_SIZE.

Parameters:

WorldRank:	MPI Rank in MPI_COMM_WORLD.
WorldSize:	# Processes in MPI_COMM_WORLD.
SampleSize:	Number of timing tests per process pair.
PP1:	MPI rank of sending process.
PP2:	MPI rank of receiving process.
Cycles:	Number of round-trips per ping-pong test. Cycles is set to be greater than 1 if it is necessary to correct for poor clock resolution.
PmsgSize:	The size of the message sent from PP1 to PP2. The acknowledgement from PP2 to PP1 is always 1 float.
LatencySwitch:	0 => BandWidth Test; !0 => Latency Test.
SEED:	seed for random number generator for generating upstream message.

MPI_Datatype build_parm_MPI_datatype( )

MPI_Datatype **build_parm_MPI_datatype()**;.

This function builds the MPI datatype for send the parameter vector from the parm file between MPI processes.

```
void build_SEND_STAT( STAT *          InBuffer,
                      MPI_Datatype *  SEND_STAT
                    )
```

void build_SEND_STAT(...)

This function builds the MPI datatype required for sending the statistical report computed by moment(..)between processors.

```
void moment( double    data[],
             int       n,
             double *  ave,
             double *  adev,
             double *  sdev,
             double *  svar,
             double *  skew,
             double *  curt,
             double *  dmax,
             double *  dmin
           )
```

void Moment(...)

moment(...) is a revised C function from numerical recipes. The output variables are returned as type double, and the minimum and maximum values in the array are also calculated. See 'Numerical Recipes in C', section 14.1, p. 613.

The original function used 1-based indexing. This has been changed to zero-based indexing.

```
void PingPong( MPI_Datatype  SEND_STAT,
               int           sample_size,
               int           PP1,
               int           PP2,
               int           cycles,
               const int     pmsg_size,
               const int     LatencySwitch,
               const int     SEED
             )
```

void PingPong(...)

PingPong sends 'cycles' instances of a message of pmsg_size [in bytes] from PP1 to PP2 [MW Process Ranks]. Then the sending process PP1 waits for the proper number of acknowledgements. The average per bandwidth is computed, and returns the average time for a one-way message transmission to the root MW process via an MPI message. Note the assymetry in the lengths of the messages sent and received.

int ReadPrm(parm * *FP*)

ReadPrm(parm* FP) reads the parameter file for the bandwidth tests.

```
int SendVector( int                          WorldRank,
                int                          WorldSize,
                vector< vector< Real > > &   StatVec
              )
```

Function SendVector(...).

This function broadcasts a vector of vectors from MW_Root to all of the other processes in **Machine::MIDWARE_COMM**.

int WriteScreen(STAT & *Report*)

WriteScreen(STAT &Report) writes the statistics report to the screen.

Generated on Mon Oct 7 01:04:29 2002 for PGMT by doxygen 1.2.18

parm Struct Reference

`#include <computeNetwork.doxygen.h>`

List of all members.

Detailed Description

Struct for setting parameter values.

The parm parameters are read in from ParmInverseBandwidth.prm. Refer to this file for recommended values.

Definition at line **53** of file **computeNetwork.doxygen.h**.

Public Attributes

int **SAMPLE_SIZE**
int **PP_CYCLES**
int **PP_MSG_SIZE**
int **SEED**
int **STAT_FLAG**
int **PARM_FLAG**

Member Data Documentation

int parm::PARM_FLAG

Definition at line **58** of file **computeNetwork.doxygen.h**.

int parm::PP_CYCLES

Definition at line **54** of file **computeNetwork.doxygen.h**.

int parm::PP_MSG_SIZE

Definition at line **55** of file **computeNetwork.doxygen.h**.

int parm::SAMPLE_SIZE

Definition at line **53** of file **computeNetwork.doxygen.h**.

int parm::SEED

Definition at line **56** of file **computeNetwork.doxygen.h**.

int parm::STAT_FLAG

Definition at line **57** of file **computeNetwork.doxygen.h**.

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

- **computeNetwork.doxygen.h**

Generated on Mon Oct 7 01:04:30 2002 for PGMT by doxygen 1.2.18

parm Member List

This is the complete list of members for **parm**, including all inherited members.

PARM_FLAG	**parm**
PP_CYCLES	**parm**
PP_MSG_SIZE	**parm**
SAMPLE_SIZE	**parm**
SEED	**parm**
STAT_FLAG	**parm**

Generated on Mon Oct 7 01:04:30 2002 for PGMT by doxygen 1.2.18

STAT Struct Reference

`#include <computeNetwork.doxygen.h>`

List of all members.

Detailed Description

typdef struct STAT

Statistical Report Structure:

Parameters:

NP := total number of processes
PP1 := rank of ping-process 1
PP2 := rank of ping-process 2
NRT := number of round-trips per ping-pong timing loop
PPSZ := message size for ping-pong processes
SPSZ := sample size (round trip tests)

ave := average time/PingPong loop
adev := average deviation
sdev := standard deviation
svar := variance
skew := skew
curt := kurtosis
dmax := maximum value
dmin := minimum value

Definition at line **80** of file **computeNetwork.doxygen.h**.

Public Attributes

int **NP**
int **PP1**
int **PP2**
int **NRT**
int **PPSZ**
int **SPSZ**
double **STV** [8]

Member Data Documentation

int STAT::NP

Definition at line **80** of file **computeNetwork.doxygen.h**.

int STAT::NRT

Definition at line **80** of file **computeNetwork.doxygen.h**.

int STAT::PP1

Definition at line **80** of file **computeNetwork.doxygen.h**.

int STAT::PP2

Definition at line **80** of file **computeNetwork.doxygen.h**.

int STAT::PPSZ

Definition at line **80** of file **computeNetwork.doxygen.h**.

int STAT::SPSZ

Definition at line **81** of file **computeNetwork.doxygen.h**.

double STAT::STV[8]

Definition at line **81** of file **computeNetwork.doxygen.h**.

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

- **computeNetwork.doxygen.h**

Generated on Mon Oct 7 01:04:31 2002 for PGMT by doxygen 1.2.18

STAT Member List

This is the complete list of members for **STAT**, including all inherited members.

NP	**STAT**
NRT	**STAT**
PP1	**STAT**
PP2	**STAT**
PPSZ	**STAT**
SPSZ	**STAT**
STV	**STAT**

Generated on Mon Oct 7 01:04:31 2002 for PGMT by doxygen 1.2.18

history.doxygen.h File Reference

Detailed Description

Definition in file **history.doxygen.h**.`#include "global.h"`
`#include "GCL/GCL_StringIDs.h"`
`#include "GCL/GCL_Graph.h"`

Go to the source code of this file.

Compounds

class **BaseKey**
class **BaseMap**
class **BaseRecord**
class BaseRecord More...

struct **ProcID_and_Mean**
class **FileKey**
class FileKey More...

class **FileMap**
class **FileOp**
Class FileOp and methods. More...

class **FileRecord**
class FileRecord defines a history record. More...

Functions

bool **operator<** (const **BaseKey** &a, const **BaseKey** &b)
bool **operator<** (const **FileKey** &a, const **FileKey** &b)

Variables

const int **StringsInID** = 5
 # strings in GCL_StringIDs class

Function Documentation

```
bool operator<( const FileKey &  a,
                const FileKey &  b
              )
```

```
bool operator<( const BaseKey &  a,
                const BaseKey &  b
              )
```

Variable Documentation

```
const int StringsInID = 5
```

strings in GCL_StringIDs class

Author:
Roger Hillson

Definition at line **38** of file **history.doxygen.h**.

Generated on Mon Oct 7 01:04:29 2002 for PGMT by doxygen 1.2.18

history.doxygen.h

Go to the documentation of this file.

```
00001
00007 /*********************************************************************
00008 *                                                                    *
00009 * Notice:                                                            *
00010 *  This software module was developed at the Processing Systems Section *
00011 *  (Code 5583) of the Naval Research Laboratory for release to the   *
00012 *  public.  It was funded by the Office of Naval Research, the Navy  *
00013 *  Signal  Processing Office, and the Advanced Systems Technology    *
00014 *  Office (USW).  It is available for unlimited distribution at no   *
00015 *  charge under the condition that this permission notice appears in *
00016 *  all copies of its derivative works, or modified portions thereof. It *
00017 *  is provided "as is"; no warranty is given.                       *
00018 *                                                                    *
00019 *  Authors:     Roger Hillson                                        *
00020 *               Naval Research Laboratory                            *
00021 *                                                                    *
00022 *  Module:      Middleware                                           *
00023 *                                                                    *
00024 *  File:         history.h                                           *
00025 *                                                                    *
00026 * Purpose:                                                           *
00027 *                                                                    *
00028 *********************************************************************/
00029
00030 #ifndef HISTORY_H
00031 #define HISTORY_H
00032
00033 #include "global.h"
00034 #include "GCL/GCL_StringIDs.h"
00035 #include "GCL/GCL_Graph.h"
00036
00038 const int StringsInID = 5;
00040
00049 class BaseRecord
00050 {
00051
00052 public:
00054
00055  BaseRecord::BaseRecord();
00056  BaseRecord::~BaseRecord(){};
00057
00062  BaseRecord::BaseRecord( const GCL_NodeID, const MW_ProcessID, const Real,
```

```
00063     const vector< Real >&, const int, const int,
00064     const Real, const Real, const int);
00065
00069  BaseRecord::BaseRecord( const BaseRecord &right );
00070
00071  void  BaseRecord::printRecord();
00072
00074
00075  const GCL_NodeID      BaseRecord::get__NodeID () const;
00076  const MW_ProcessID    BaseRecord::get__ProcID () const;
00077  const Real        BaseRecord::get__ExTime () const;
00078  const vector< Real >& BaseRecord::get__Parms  () const;
00079  const int        BaseRecord::get__Instance () const;
00080
00081  const int        BaseRecord::get__Nsamples () const;
00082  const Real        BaseRecord::get__Mean  () const;
00083  const Real        BaseRecord::get__Variance () const;
00084  const int        BaseRecord::get__Tag  () const;
00085
00086  void BaseRecord::set_all( const GCL_NodeID, const MW_ProcessID, const Real,
00087     const vector< Real >&, const int, const int,
00088     const Real, const Real, const int);
00089  void BaseRecord::set__NodeID ( GCL_NodeID value);
00090  void BaseRecord::set__ProcID ( MW_ProcessID value );
00091  void BaseRecord::set__ExTime ( Real value );
00092  void BaseRecord::set__Parms  ( vector < Real >& value );
00093  void BaseRecord::set__Instance ( int value );
00094  void BaseRecord::set__Nsamples ( int value );
00095  void BaseRecord::set__Mean ( Real value );
00096  void BaseRecord::set__Variance ( Real value );
00097  void BaseRecord::set__Tag ( int  value );
00098
00099 private:
00100
00103
00104
00108 GCL_NodeID _NodeID;
00109
00111
00116 MW_ProcessID _ProcID;
00118 Real  _ExTime;
00120 vector< Real > _Parms;
00122 int  _Instance;
00126 int        _Nsamples;
00127
00128 Real        _Mean;
00129
00130 Real        _Variance;
00131
00132 int        _Tag;
00133
```

```
00134
00135 };  // End BaseRecord Class
00136
00137 //-----------------------------------
00138
00139 class BaseKey  {
00140 friend class BaseMap;
00141
00142 public:
00143   BaseKey::BaseKey();
00144   BaseKey::~BaseKey(){};
00145   BaseKey::BaseKey(GCL_NodeID, MW_ProcessID);
00146   const  GCL_NodeID    BaseKey::get__NodeID () const;
00147   const  MW_ProcessID  BaseKey::get__ProcID () const;
00148   void BaseKey::set__ProcID(MW_ProcessID);
00149   void BaseKey::set__Key(GCL_NodeID, MW_ProcessID);
00150
00151 private:
00152   GCL_NodeID _NodeID;
00153    MW_ProcessID _ProcID;
00154 };
00155
00156 //-----------------------------------
00157
00158 class BaseMap {
00159
00164 public:
00165   BaseMap::BaseMap(){};
00166   BaseMap::~BaseMap(){};
00167
00169
00172   static void  BaseMap::CreateDataBase();
00173   static void  BaseMap::ClearDataBase();
00174   static void  BaseMap::ViewDataBase();
00175
00176   static BaseKey     BaseMap::CreateKey  (BaseRecord&);
00177
00178
00180
00185   static BaseRecord& BaseMap::GetKeyRec  (BaseKey&);
00186   static bool      BaseMap::QueryKeyRec(BaseKey&);
00187
00189
00192   static bool      BaseMap::InsertRec   (BaseRecord&);
00193   static bool      BaseMap::DeleteKeyRec(BaseKey&);
00194
00195   static bool      BaseMap::UpdateRec   (BaseRecord&);
00197   static bool      BaseMap::ResetKeyRec (BaseKey&);
00199
00214     static bool        BaseMap::StatUpdate(BaseKey::BaseKey &, double, int);
```

```
00215
00216     /* Perhaps this should be private, but called from a public
00217  method which cleans things up, and writes the FileMap to
00218  disk? */
00219
00220
00222
00238 static unsigned int BaseMap::BMap_to_FMap();
00239
00240 private:
00241
00242 static map < BaseKey::BaseKey, BaseRecord::BaseRecord > Bmap;
00243 static vector< vector< unsigned int > > StringIDCrossIndex;
00244 static vector< vector< double > > ScaleFactor;
00245 static vector< unsigned int >    NodeFillVector;
00246
00247 struct ProcID_and_Mean {
00248   MW_ProcessID ProcID;
00249   double       Mean; };
00250
00251 static int    BaseMap::CreateBlankRecords();
00252
00253 static int    BaseMap::InitializeFromFmap();
00254
00256  /*  Initialize the cross-table of nodes which have
00257      the same GCL_StringIDs, the sizexsize vector
00258      of processor speed ratios, and the NodeFillVector.    */
00259
00260 static void   BaseMap::CrossIndex();
00261 static int    BaseMap::Extrapolate();
00262
00264
00285 static int    BaseMap::FillData(GCL_NodeID);
00286
00287 };  // end class BaseMap
00288
00290
00291 bool operator < ( const BaseKey& a, const BaseKey& b );
00292
00294
00296
00304 class FileRecord {
00305
00306 //  History record
00307
00308       public:
00309 // constructors
00310
00311  FileRecord::FileRecord();
00312  FileRecord::~FileRecord(){};
00314  FileRecord::FileRecord( const vector< string >&, const string, const time_t,
```

```
00315     const int, const Real, const Real, const int);
00316  FileRecord::FileRecord( const FileRecord& right );
00317
00318 // helper functions
00319
00320      void  FileRecord::printRecord();
00321
00322 const vector< string >& FileRecord::get__VstringIDs () const;
00323 const string  FileRecord::get__HostName () const;
00324 const time_t  FileRecord::get__EntryTime () const;
00325 const int  FileRecord::get__Nsamples () const;
00326 const Real  FileRecord::get__Mean () const;
00327 const Real  FileRecord::get__Variance () const;
00328 const int  FileRecord::get__Tag () const;
00329
00330 void  FileRecord::set_all(const vector< string >&, const string, const time_t,
00331      const int, const Real, const Real, const int);
00332 void   FileRecord::set__VstringIDs ( const vector< string >& value);
00333 void   FileRecord::set__HostName ( string value );
00334 void   FileRecord::set__EntryTime ( time_t value );
00335 void   FileRecord::set__Nsamples ( int value );
00336 void   FileRecord::set__Mean ( Real value );
00337 void   FileRecord::set__Variance ( Real value );
00338 void   FileRecord::set__Tag ( int value );
00339
00340 //private:
00341
00342 friend class FileOp;
00343
00344 vector< string >   _VstringIDs;
00345 string     _HostName;
00346 time_t     _EntryTime;
00347 int     _Nsamples;
00348 Real     _Mean;
00349 Real     _Variance;
00350 int     _Tag;
00351
00352 }; //  End FileRecord Class
00353
00355
00357
00360 class FileOp {
00361
00364 public:
00365 FileOp::FileOp(){};
00366 FileOp::~FileOp(){};
00367
00368 static bool  OpenWriteFile(const char*, ofstream&);
00369 static bool  OpenReadFile(const char*, ifstream&);
00370 static bool  CloseWriteFile(ofstream&);
```

```
00371 static bool  CloseReadFile(ifstream&);
00372
00374
00379 static bool  WriteRecord(ofstream&, FileRecord&);
00380
00381
00383
00384 static bool  ReadRecord(ifstream&,  FileRecord&);
00385
00386 static const char* GetFileNameIn();
00387 static const char* GetFileNameOut();
00388 static void  CatFile(const char *, char *);
00389
00391
00396 static void  PrintPepHistoryFile();
00397
00398 private:
00399 // define delimiter for disk FileRecords
00400 static const char*  MagicMarker;
00401
00406 static const char* FileNameIn;
00407 static const char* FileNameOut;
00408
00409 };  // End Class FileOp
00410
00412
00414
00417 //-----------------------------------
00418
00419 class FileKey  {
00420 friend class FileMap;
00421
00422 public:
00423   FileKey::FileKey();
00424   FileKey::~FileKey(){};
00425   FileKey::FileKey(vector<string>, string);
00426   const string&  FileKey::get__Vstring(int) const;
00427   const string&  FileKey::get__HostName()   const;
00428   void   FileKey::set__FileKey(const GCL_StringIDs&, const string&);
00429   void   FileKey::set__HostName( const string& );
00430   const vector <string>& FileKey::get__VstringIDs() const;
00431
00432   //private:
00433   vector < string > _VstringIDs;
00434     string   _HostName;
00435 };
00436
00437 //-------------------------------------------------------------
00438
00439 class FileMap {
```

```
00440
00446 public:
00447   FileMap::FileMap(){};
00448   FileMap::~FileMap(){};
00449
00450   static int   FileMap::CreateDataBase();
00451   static int   FileMap::SaveDataBase();
00452   static void  FileMap::ClearDataBase();
00453   static void  FileMap::ViewDataBase();
00454
00456
00462   static void  FileMap::NormalizeDataBase();
00463
00464   static FileKey     FileMap::CreateKey  (FileRecord&);
00465   static FileRecord& FileMap::GetKeyRec  (FileKey&);
00466
00467
00469
00470   static bool      FileMap::QueryKeyRec(FileKey&);
00471
00473
00486   static void      FileMap::GenerateFileKey(BaseKey&, FileKey&);
00487
00491
00492
00498   static bool      FileMap::InsertRec   (FileRecord&);
00499
00500
00502
00505   static bool      FileMap::DeleteKeyRec(FileKey&);
00506
00507   static bool      FileMap::UpdateRec   (FileRecord&);
00509   static bool      FileMap::ResetKeyRec (FileKey&);
00511
00518   static int      FileMap::GetFkeyRec( FileKey::FileKey&,
00519        map < BaseKey::BaseKey, BaseRecord::BaseRecord >::iterator );
00520
00521   static const string TokenProcessName;
00522
00523   //private:
00524
00525 static map < FileKey::FileKey, FileRecord::FileRecord > Fmap;
00526
00527 };  // end class FileMap
00528
00529 bool operator < ( const FileKey& a, const FileKey& b );
00530
00532 //  Function Bodies for BaseRecord Class
00534
00536
```

```
00537 inline const GCL_NodeID
00538 BaseRecord::get__NodeID () const {
00539   return _NodeID;
00540 }
00541
00542 inline const MW_ProcessID
00543 BaseRecord::get__ProcID () const {
00544   return _ProcID;
00545 }
00546
00547 inline const Real
00548 BaseRecord::get__ExTime () const {
00549   return _ExTime;
00550 }
00551
00552 inline const vector< Real > &
00553 BaseRecord::get__Parms () const {
00554   return _Parms;
00555 }
00556
00557 inline const int
00558 BaseRecord::get__Instance () const {
00559   return _Instance;
00560 }
00561
00562 inline const int
00563 BaseRecord::get__Nsamples () const {
00564   return _Nsamples;
00565 }
00566
00567 inline const Real
00568 BaseRecord::get__Mean () const {
00569   return _Mean;
00570 }
00571 inline const Real
00572 BaseRecord::get__Variance () const {
00573   return _Variance;
00574 }
00575 inline const int
00576 BaseRecord::get__Tag () const {
00577   return _Tag;
00578 }
00580
00581 inline void
00582 BaseRecord::set_all(   const GCL_NodeID nodeID, const MW_ProcessID procID,
00583          const Real ExTime, const vector< Real >& Parms,
00584          const int instance, const int Nsamples, const Real Mean,
00585          const Real Variance, const int Tag)
00586 {
00587   _NodeID = nodeID;
00588   _ProcID = procID;
```

```
00589   _ExTime = ExTime;
00590   _Parms  = Parms;
00591   _Instance = instance;
00592   _Nsamples = Nsamples;
00593   _Mean     = Mean;
00594   _Variance = Variance;
00595   _Tag     = Tag;
00596 };
00597
00599
00600
00601 inline void
00602 BaseRecord::set__NodeID ( GCL_NodeID value) {
00603   _NodeID = value;
00604 }
00605
00606 inline void
00607 BaseRecord::set__ProcID ( MW_ProcessID value ) {
00608   _ProcID = value;
00609 }
00610
00611 inline void
00612 BaseRecord::set__ExTime ( Real value ) {
00613   _ExTime = value;
00614 }
00615
00616 inline void
00617 BaseRecord::set__Parms ( vector < Real >& value ) {
00618   _Parms = value;
00619 }
00620
00621 inline void
00622 BaseRecord::set__Instance ( int value ) {
00623   _Instance = value;
00624 }
00625
00626 inline void
00627 BaseRecord::set__Nsamples ( int value ) {
00628   _Nsamples = value;
00629 }
00630
00631 inline void
00632 BaseRecord::set__Mean ( Real value ) {
00633   _Mean = value;
00634 }
00635 inline void
00636 BaseRecord::set__Variance ( Real value ) {
00637    _Variance = value;
00638 }
00639 inline void
00640 BaseRecord::set__Tag ( int value ) {
```

```
00641    _Tag = value;
00642 }
00644 //  Function bodies for FileRecord class
00646
00647 //  (1) get functions
00648
00649 inline const vector< string >&
00650 FileRecord::get__VstringIDs () const {
00651   return _VstringIDs;
00652 }
00653
00654 inline const string
00655 FileRecord::get__HostName () const {
00656   return _HostName;
00657 }
00658
00659 inline const time_t
00660 FileRecord::get__EntryTime () const {
00661   return _EntryTime;
00662 }
00663
00664 inline const int
00665 FileRecord::get__Nsamples () const {
00666   return _Nsamples;
00667 }
00668
00669 inline const Real
00670 FileRecord::get__Mean () const {
00671   return _Mean;
00672 }
00673
00674 inline const Real
00675 FileRecord::get__Variance () const {
00676   return _Variance;
00677 }
00678
00679 inline const int
00680 FileRecord::get__Tag () const {
00681   return _Tag;
00682 }
00683
00684 //  (2)  set functions
00685
00686 inline void
00687 FileRecord::set_all(const vector< string >& VstringIDs, const string HostName,
00688       const time_t EntryTime, const int Nsamples, const Real Mean,
00689       const Real Variance, const int Tag)
00690 {
00691 _VstringIDs = VstringIDs;
00692 _HostName   = HostName;
00693 _EntryTime  = EntryTime;
```

```
00694 _Nsamples   = Nsamples;
00695 _Mean     = Mean;
00696 _Variance   = Variance;
00697 _Tag     = Tag;
00698 }
00699
00700 inline void
00701 FileRecord::set__VstringIDs ( const vector< string >& value) {
00702   _VstringIDs = value;
00703 }
00704
00705 inline void
00706 FileRecord::set__HostName ( string value ) {
00707   _HostName  = value;
00708 }
00709
00710 inline void
00711 FileRecord::set__EntryTime ( time_t value ) {
00712   _EntryTime = value;
00713 }
00714
00715 inline void
00716 FileRecord::set__Nsamples ( int value ) {
00717   _Nsamples = value;
00718 }
00719
00720 inline void
00721 FileRecord::set__Mean ( Real value ) {
00722   _Mean = value;
00723 }
00724
00725 inline void
00726 FileRecord::set__Variance ( Real value ) {
00727   _Variance = value;
00728 }
00729
00730 inline void
00731 FileRecord::set__Tag ( int value ) {
00732   _Tag = value;
00733 }
00734
00736 //  Other InLine Function bodies
00738
00739 inline const char*
00740 FileOp::GetFileNameIn () {
00741   return FileOp::FileNameIn; }
00742
00744
00745 inline const char*
00746 FileOp::GetFileNameOut () {
00747   return FileOp::FileNameOut; }
```

```
00748
00749
00751
00752 #endif
```

Generated on Mon Oct 7 01:04:29 2002 for PGMT by doxygen 1.2.18

BaseRecord Class Reference

```
#include <history.doxygen.h>
```

List of all members.

Detailed Description

class BaseRecord

Base record objects contain data required to build, maintain, & update the dynamic data base used to tracking transition execution timings and token sizes. Before the dynamic data base is written to a file, base records are converted to FileRecords. BaseRecords which map onto the same processor and the same GCL_StringID will be compressed into a single **FileRecord**.

Definition at line **49** of file **history.doxygen.h**.

Public Methods

	BaseRecord::BaseRecord ()
	constructors
	BaseRecord::~BaseRecord ()
	BaseRecord::BaseRecord (const GCL_NodeID, const MW_ProcessID, const Real, const vector< Real > &, const int, const int, const Real, const Real, const int)
	BaseRecord::BaseRecord (const BaseRecord &right)
void	**BaseRecord::printRecord** ()
const GCL_NodeID	**BaseRecord::get__NodeID** () const
	Methods to set and return data members.

const MW_ProcessID **BaseRecord::get__ProcID** () const
const Real **BaseRecord::get__ExTime** () const
const vector< Real > & **BaseRecord::get__Parms** () const
const int **BaseRecord::get__Instance** () const
const int **BaseRecord::get__Nsamples** () const
const Real **BaseRecord::get__Mean** () const
const Real **BaseRecord::get__Variance** () const
const int **BaseRecord::get__Tag** () const
void **BaseRecord::set_all** (const GCL_NodeID, const MW_ProcessID, const Real, const vector< Real > &, const int, const int, const Real, const Real, const int)
void **BaseRecord::set__NodeID** (GCL_NodeID value)
void **BaseRecord::set__ProcID** (MW_ProcessID value)
void **BaseRecord::set__ExTime** (Real value)
void **BaseRecord::set__Parms** (vector< Real > &value)
void **BaseRecord::set__Instance** (int value)
void **BaseRecord::set__Nsamples** (int value)
void **BaseRecord::set__Mean** (Real value)
void **BaseRecord::set__Variance** (Real value)
void **BaseRecord::set__Tag** (int value)

Constructor & Destructor Documentation

BaseRecord::BaseRecord::~BaseRecord( ) [inline]

Definition at line **56** of file **history.doxygen.h**.

References **BaseRecord::~BaseRecord()**.

Referenced by **BaseRecord::~BaseRecord()**.

```
00056 {};
```

Member Function Documentation

BaseRecord::BaseRecord::BaseRecord(const BaseRecord & *right*)

This constructor builds a record for entry into the internal (i.e. memory) database.

```
BaseRecord::BaseRecord::BaseRecord( const                    GCL_NodeID,
                                    const                    MW_ProcessID,
                                    const                    Real,
                                    const vector< Real > &   ,
                                    const                    int,
                                    const                    int,
                                    const                    Real,
                                    const                    Real,
                                    const                    int
                                  )
```

This constructor builds a record for entry into the internal (i.e. memory) database.

BaseRecord::BaseRecord::BaseRecord( )

constructors

const Real BaseRecord::BaseRecord::get__ExTime( )

const int BaseRecord::BaseRecord::get__Instance( )

const Real BaseRecord::BaseRecord::get__Mean( )

const GCL_NodeID BaseRecord::BaseRecord::get__NodeID( )

Methods to set and return data members.

const int BaseRecord::BaseRecord::get__Nsamples( )

const vector< Real >& BaseRecord::BaseRecord::get__Parms( )

const MW_ProcessID BaseRecord::BaseRecord::get__ProcID( )

const int BaseRecord::BaseRecord::get__Tag( )

const Real BaseRecord::BaseRecord::get__Variance( )

void BaseRecord::BaseRecord::printRecord( )

void BaseRecord::BaseRecord::set__ExTime(Real *value*)

void BaseRecord::BaseRecord::set__Instance(int *value*)

void BaseRecord::BaseRecord::set__Mean(Real *value*)

void BaseRecord::BaseRecord::set__NodeID(GCL_NodeID *value*)

void BaseRecord::BaseRecord::set__Nsamples(int *value*)

void BaseRecord::BaseRecord::set__Parms(vector< Real > & *value*)

void BaseRecord::BaseRecord::set__ProcID(MW_ProcessID *value*)

void BaseRecord::BaseRecord::set__Tag(int *value*)

void BaseRecord::BaseRecord::set__Variance(Real *value*)

```
void BaseRecord::BaseRecord::set_all( const                    GCL_NodeID,
                                      const                    MW_ProcessID,
                                      const                    Real,
                                      const vector< Real > &   ,
                                      const                    int,
                                      const                    int,
                                      const                    Real,
                                      const                    Real,
                                      const                    int
                                    )
```

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

- **history.doxygen.h**

Generated on Mon Oct 7 01:04:30 2002 for PGMT by doxygen 1.2.18

BaseRecord Member List

This is the complete list of members for **BaseRecord**, including all inherited members.

BaseRecord::BaseRecord()	**BaseRecord**	
BaseRecord::BaseRecord(const GCL_NodeID, const MW_ProcessID, const Real, const vector< Real > &, const int, const int, const Real, const Real, const int)	**BaseRecord**	
BaseRecord::BaseRecord(const BaseRecord &right)	**BaseRecord**	
BaseRecord::get__ExTime() const	**BaseRecord**	
BaseRecord::get__Instance() const	**BaseRecord**	
BaseRecord::get__Mean() const	**BaseRecord**	
BaseRecord::get__NodeID() const	**BaseRecord**	
BaseRecord::get__Nsamples() const	**BaseRecord**	
BaseRecord::get__Parms() const	**BaseRecord**	
BaseRecord::get__ProcID() const	**BaseRecord**	
BaseRecord::get__Tag() const	**BaseRecord**	
BaseRecord::get__Variance() const	**BaseRecord**	
BaseRecord::printRecord()	**BaseRecord**	
BaseRecord::set__ExTime(Real value)	**BaseRecord**	
BaseRecord::set__Instance(int value)	**BaseRecord**	
BaseRecord::set__Mean(Real value)	**BaseRecord**	
BaseRecord::set__NodeID(GCL_NodeID value)	**BaseRecord**	
BaseRecord::set__Nsamples(int value)	**BaseRecord**	
BaseRecord::set__Parms(vector< Real > &value)	**BaseRecord**	
BaseRecord::set__ProcID(MW_ProcessID value)	**BaseRecord**	
BaseRecord::set__Tag(int value)	**BaseRecord**	
BaseRecord::set__Variance(Real value)	**BaseRecord**	
BaseRecord::set_all(const GCL_NodeID, const MW_ProcessID, const Real, const vector< Real > &, const int, const int, const Real, const Real, const int)	**BaseRecord**	
BaseRecord::~BaseRecord()	**BaseRecord**	`[inline]`

Generated on Mon Oct 7 01:04:30 2002 for PGMT by doxygen 1.2.18

FileRecord Class Reference

`#include <history.doxygen.h>`

List of all members.

Detailed Description

class FileRecord defines a history record.

The history records are written to and from the disk. When the PEP is shutting down, the base records are converted to file records, and written to the disk. Each saved record includes the GCL_StringID and the name of the physical host processor that the transition was executed on, as well as the basic statistics - average transition timing, number of executions after last reset, etc.

Definition at line **304** of file **history.doxygen.h**.

Public Methods

FileRecord::FileRecord ()
FileRecord::~FileRecord ()
FileRecord::FileRecord (const vector< string > &, const string, const time_t, const int, const Real, const Real, const int)
Constructor for the file records.

FileRecord::FileRecord (const FileRecord &right)
void **FileRecord::printRecord** ()
const vector< string > & **FileRecord::get__VstringIDs** () const
const string **FileRecord::get__HostName** () const
const time_t **FileRecord::get__EntryTime** () const
const int **FileRecord::get__Nsamples** () const

const Real **FileRecord::get__Mean** () const
const Real **FileRecord::get__Variance** () const
const int **FileRecord::get__Tag** () const
void **FileRecord::set_all** (const vector< string > &, const string, const time_t, const int, const Real, const Real, const int)
void **FileRecord::set__VstringIDs** (const vector< string > &value)
void **FileRecord::set__HostName** (string value)
void **FileRecord::set__EntryTime** (time_t value)
void **FileRecord::set__Nsamples** (int value)
void **FileRecord::set__Mean** (Real value)
void **FileRecord::set__Variance** (Real value)
void **FileRecord::set__Tag** (int value)

Public Attributes

vector< string > **_VstringIDs**
5-string vector <-GCL_StringID

string **_HostName**
Processor name <-MW_ProcessID.

time_t **_EntryTime**
time of data entry

int **_Nsamples**
samples in average

Real **_Mean**
mean execution time

Real **_Variance**
variance of execution time

int **_Tag**
token size tag

Friends

class **FileOp**

Constructor & Destructor Documentation

FileRecord::FileRecord::~FileRecord( ) [inline]

Definition at line **312** of file **history.doxygen.h**.

References **FileRecord::~FileRecord()**.

Referenced by **FileRecord::~FileRecord()**.

```
00312 {};
```

Member Function Documentation

FileRecord::FileRecord::FileRecord(const FileRecord & *right*)

**FileRecord::FileRecord::FileRecord(const vector< string > & *,*
const *string,*
const *time_t,*
const *int,*
const *Real,*
const *Real,*
const *int*
)**

Constructor for the file records.

FileRecord::FileRecord::FileRecord( )

const time_t FileRecord::FileRecord::get__EntryTime( )

const string FileRecord::FileRecord::get__HostName( )

const Real FileRecord::FileRecord::get__Mean( )

const int FileRecord::FileRecord::get__Nsamples( )

const int FileRecord::FileRecord::get__Tag( )

const Real FileRecord::FileRecord::get__Variance( )

const vector< string >& FileRecord::FileRecord::get__VstringIDs( )

void FileRecord::FileRecord::printRecord( )

void FileRecord::FileRecord::set__EntryTime(time_t *value*)

void FileRecord::FileRecord::set__HostName(string *value*)

void FileRecord::FileRecord::set__Mean(Real *value*)

void FileRecord::FileRecord::set__Nsamples(int *value*)

void FileRecord::FileRecord::set__Tag(int *value*)

void FileRecord::FileRecord::set__Variance(Real *value*)

void FileRecord::FileRecord::set__VstringIDs(const vector< string > & *value*)

```
void FileRecord::FileRecord::set_all( const vector< string > & ,
                                      const                     string,
                                      const                     time_t,
                                      const                     int,
                                      const                     Real,
                                      const                     Real,
                                      const                     int
                                    )
```

Friends And Related Function Documentation

friend class FileOp `[friend]`

Definition at line **342** of file **history.doxygen.h**.

Member Data Documentation

time_t FileRecord::_EntryTime

time of data entry

Definition at line **346** of file **history.doxygen.h**.

string FileRecord::_HostName

Processor name <-MW_ProcessID.

Definition at line **345** of file **history.doxygen.h**.

Real FileRecord::_Mean

mean execution time

Definition at line **348** of file **history.doxygen.h**.

int FileRecord::_Nsamples

samples in average

Definition at line **347** of file **history.doxygen.h**.

int FileRecord::_Tag

token size tag

Definition at line **350** of file **history.doxygen.h**.

Real FileRecord::_Variance

variance of execution time

Definition at line **349** of file **history.doxygen.h**.

vector< string > FileRecord::_VstringIDs

5-string vector <-GCL_StringID

Definition at line **344** of file **history.doxygen.h**.

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

- **history.doxygen.h**

Generated on Mon Oct 7 01:04:30 2002 for PGMT by doxygen 1.2.18

FileRecord Member List

This is the complete list of members for **FileRecord**, including all inherited members.

_EntryTime	**FileRecord**	
_HostName	**FileRecord**	
_Mean	**FileRecord**	
_Nsamples	**FileRecord**	
_Tag	**FileRecord**	
_Variance	**FileRecord**	
_VstringIDs	**FileRecord**	
FileOp class	**FileRecord**	`[friend]`
FileRecord::FileRecord()	**FileRecord**	
FileRecord::FileRecord(const vector< string > &, const string, const time_t, const int, const Real, const Real, const int)	**FileRecord**	
FileRecord::FileRecord(const FileRecord &right)	**FileRecord**	
FileRecord::get__EntryTime() const	**FileRecord**	
FileRecord::get__HostName() const	**FileRecord**	
FileRecord::get__Mean() const	**FileRecord**	
FileRecord::get__Nsamples() const	**FileRecord**	
FileRecord::get__Tag() const	**FileRecord**	
FileRecord::get__Variance() const	**FileRecord**	
FileRecord::get__VstringIDs() const	**FileRecord**	
FileRecord::printRecord()	**FileRecord**	
FileRecord::set__EntryTime(time_t value)	**FileRecord**	
FileRecord::set__HostName(string value)	**FileRecord**	
FileRecord::set__Mean(Real value)	**FileRecord**	
FileRecord::set__Nsamples(int value)	**FileRecord**	
FileRecord::set__Tag(int value)	**FileRecord**	
FileRecord::set__Variance(Real value)	**FileRecord**	

FileRecord::set__VstringIDs(const vector< string > &value)	**FileRecord**	
FileRecord::set_all(const vector< string > &, const string, const time_t, const int, const Real, const Real, const int)	**FileRecord**	
FileRecord::~FileRecord()	**FileRecord**	`[inline]`

Generated on Mon Oct 7 01:04:30 2002 for PGMT by doxygen 1.2.18

BaseKey Class Reference

`#include <history.doxygen.h>`

List of all members.

Public Methods

	BaseKey::BaseKey ()
	BaseKey::~BaseKey ()
	BaseKey::BaseKey (GCL_NodeID, MW_ProcessID)
const GCL_NodeID	**BaseKey::get__NodeID** () const
const MW_ProcessID	**BaseKey::get__ProcID** () const
void	**BaseKey::set__ProcID** (MW_ProcessID)
void	**BaseKey::set__Key** (GCL_NodeID, MW_ProcessID)

Friends

class **BaseMap**

Constructor & Destructor Documentation

BaseKey::BaseKey::~BaseKey( ) [inline]

Definition at line **144** of file **history.doxygen.h**.

References **BaseKey::~BaseKey()**.

Referenced by **BaseKey::~BaseKey()**.

```
00144 {};
```

Member Function Documentation

BaseKey::BaseKey::BaseKey(GCL_NodeID , MW_ProcessID)

BaseKey::BaseKey::BaseKey( )

const GCL_NodeID BaseKey::BaseKey::get__NodeID( )

const MW_ProcessID BaseKey::BaseKey::get__ProcID( )

void BaseKey::BaseKey::set__Key(GCL_NodeID , MW_ProcessID)

void BaseKey::BaseKey::set__ProcID(MW_ProcessID)

Friends And Related Function Documentation

friend class BaseMap `[friend]`

Definition at line **140** of file **history.doxygen.h**.

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

- **history.doxygen.h**

Generated on Mon Oct 7 01:04:30 2002 for PGMT by doxygen 1.2.18

BaseKey Member List

This is the complete list of members for **BaseKey**, including all inherited members.

BaseKey::BaseKey()	**BaseKey**	
BaseKey::BaseKey(GCL_NodeID, MW_ProcessID)	**BaseKey**	
BaseKey::get__NodeID() const	**BaseKey**	
BaseKey::get__ProcID() const	**BaseKey**	
BaseKey::set__Key(GCL_NodeID, MW_ProcessID)	**BaseKey**	
BaseKey::set__ProcID(MW_ProcessID)	**BaseKey**	
BaseKey::~BaseKey()	**BaseKey**	`[inline]`
BaseMap class	**BaseKey**	`[friend]`

Generated on Mon Oct 7 01:04:30 2002 for PGMT by doxygen 1.2.18

BaseMap Class Reference

```
#include <history.doxygen.h>
```

List of all members.

Public Methods

BaseMap::BaseMap ()
BaseMap::~BaseMap ()

Static Public Methods

void **BaseMap::CreateDataBase** ()
Class BaseMap::CreateDataBase().

void **BaseMap::ClearDataBase** ()
Clear the data base Bmap.

void **BaseMap::ViewDataBase** ()
BaseKey **BaseMap::CreateKey** (**BaseRecord** &)
*Extract key from **BaseRecord**.*

BaseRecord & **BaseMap::GetKeyRec** (**BaseKey** &)
Retrieve record by key.

bool **BaseMap::QueryKeyRec** (**BaseKey** &)
Query record by key.

bool **BaseMap::InsertRec** (**BaseRecord** &)
Write record into Bmap.

bool **BaseMap::DeleteKeyRec** (**BaseKey** &)
bool **BaseMap::UpdateRec** (**BaseRecord** &)

bool **BaseMap::ResetKeyRec** (**BaseKey** &)
Set Mean, Variance, and Nsamples of record to zero.

bool **BaseMap::StatUpdate** (BaseKey::BaseKey &, double, int)
Recursively update the mean and variance of a record.

unsigned int **BaseMap::BMap_to_FMap** ()
clear Fmap & copy Bmap records into it.

Constructor & Destructor Documentation

BaseMap::BaseMap::~BaseMap( ) [inline]

Definition at line **166** of file **history.doxygen.h**.

```
00166 {};
```

Member Function Documentation

BaseMap::BaseMap::BaseMap( ) [inline]

Data members and methods for BaseMap. BaseMap corresponds to a base record, and is indexed by a key composed of the GCL_NodeID and the MW_ProcessID.

Definition at line **165** of file **history.doxygen.h**.

References **BaseMap::BaseMap()**.

Referenced by **BaseMap::BaseMap()**.

```
00165 {};
```

unsigned int BaseMap::BaseMap::BMap_to_FMap( ) [static]

clear Fmap & copy Bmap records into it.

This function writes the BaseMap to the **FileMap**. BaseMap records which map onto the same **FileMap** record are averaged i.e. the mean and variances are averaged, and Nsamples is set equal to 1.0. This function is executed prior to saving the **FileMap** to disk.

There are three cases:
(a) BaseMap and **FileMap** record both have valid data:
Average the Mean and Variance; increment Nsamples.
(b) Only the BaseMap record is valid:
Overwrite the **FileMap** record with the BaseMap record.
(c) Neither the BaseMap nor the **FileMap** record is valid:
Leave the **FileRecord** alone.

void BaseMap::BaseMap::ClearDataBase( ) [static]

Clear the data base Bmap.

void BaseMap::BaseMap::CreateDataBase( ) [static]

Class BaseMap::CreateDataBase().

This method reads the history file from disk and creates the Fmap. Return the number of records read from disk.

BaseKey BaseMap::BaseMap::CreateKey(BaseRecord &) [static]

Extract key from **BaseRecord**.

bool BaseMap::BaseMap::DeleteKeyRec(BaseKey &) [static]

Delete record by key Destructive InsertRec() i.e. overwrite any record with same key

BaseRecord& BaseMap::BaseMap::GetKeyRec(BaseKey &) [static]

Retrieve record by key.

If the desired key is not in the map, the key is inserted, and the default **BaseRecord** (built by the **BaseRecord** default constructor) is returned. In other words, a record is ALWAYS returned.

bool BaseMap::BaseMap::InsertRec(BaseRecord &) [static]

Write record into Bmap.

BaseMap::InsertRec returns false if a record with this key already exists

bool BaseMap::BaseMap::QueryKeyRec(BaseKey &) [static]

Query record by key.

bool BaseMap::BaseMap::ResetKeyRec(BaseKey &) [static]

Set Mean, Variance, and Nsamples of record to zero.

bool BaseMap::BaseMap::StatUpdate(BaseKey::BaseKey &, double, int) [static]

Recursively update the mean and variance of a record.

This function recursively updates the mean and variance in the record indexed by Bkey.

IF the boolean variable AutoReset is true, the following reset algorithm is applied:

IF the new mean deviates by NumSD Standard Deviations or more from the old mean, AND there are more than ResetMin samples, the **BaseRecord** variables _Mean and _Variance are reset as follows:

_Mean is set to Mean,
_Variance is set to 0, and
_Nsamples is set to 1.

bool BaseMap::BaseMap::UpdateRec(BaseRecord &) [static]

void BaseMap::BaseMap::ViewDataBase( ) [static]

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

- **history.doxygen.h**

Generated on Mon Oct 7 01:04:30 2002 for PGMT by doxygen 1.2.18

BaseMap Member List

This is the complete list of members for **BaseMap**, including all inherited members.

BaseMap::BaseMap()	**BaseMap**	[inline]
BaseMap::BMap_to_FMap()	**BaseMap**	[static]
BaseMap::ClearDataBase()	**BaseMap**	[static]
BaseMap::CreateDataBase()	**BaseMap**	[static]
BaseMap::CreateKey(BaseRecord &)	**BaseMap**	[static]
BaseMap::DeleteKeyRec(BaseKey &)	**BaseMap**	[static]
BaseMap::GetKeyRec(BaseKey &)	**BaseMap**	[static]
BaseMap::InsertRec(BaseRecord &)	**BaseMap**	[static]
BaseMap::QueryKeyRec(BaseKey &)	**BaseMap**	[static]
BaseMap::ResetKeyRec(BaseKey &)	**BaseMap**	[static]
BaseMap::StatUpdate(BaseKey::BaseKey &, double, int)	**BaseMap**	[static]
BaseMap::UpdateRec(BaseRecord &)	**BaseMap**	[static]
BaseMap::ViewDataBase()	**BaseMap**	[static]
BaseMap::~BaseMap()	**BaseMap**	[inline]

FileMap Class Reference

`#include <history.doxygen.h>`

List of all members.

Public Methods

FileMap::FileMap ()
FileMap::~FileMap ()

Static Public Methods

int **FileMap::CreateDataBase** ()
If possible, read file into Fmap.

int **FileMap::SaveDataBase** ()
*Write **FileMap::Fmap** to disk.*

void **FileMap::ClearDataBase** ()
Clear the data base Fmap.

void **FileMap::ViewDataBase** ()
void **FileMap::NormalizeDataBase** ()
Class FileMap::NormalizeDataBase().

FileKey **FileMap::CreateKey** (**FileRecord** &)
*Extract key from **FileRecord**.*

FileRecord & **FileMap::GetKeyRec** (**FileKey** &)
Get record by key.

bool **FileMap::QueryKeyRec** (**FileKey** &)
Query record by key.

void **FileMap::GenerateFileKey** (**BaseKey** &, **FileKey** &)

*Convert **BaseKey** to **FileKey**.*

bool **FileMap::InsertRec** (**FileRecord** &)
FileMap::InsertRec writes a record into Fmap.

bool **FileMap::DeleteKeyRec** (**FileKey** &)
Delete record by key.

bool **FileMap::UpdateRec** (**FileRecord** &)
bool **FileMap::ResetKeyRec** (**FileKey** &)
Set Mean, Variance, and Nsamples of record to zero.

int **FileMap::GetFkeyRec** (FileKey::FileKey &, map< BaseKey::BaseKey, BaseRecord::BaseRecord >::iterator)
FileMap::GetFkeyRec() is called by BaseMap::InitializeFromFmap().

Static Public Attributes

const string **TokenProcessName**
HostName for Token size records.

map< FileKey::FileKey, FileRecord::FileRecord > **Fmap**

Constructor & Destructor Documentation

FileMap::FileMap::~FileMap( ) [inline]

Definition at line **448** of file **history.doxygen.h**.

```
00448 {};
```

Member Function Documentation

void FileMap::FileMap::ClearDataBase( ) [static]

Clear the data base Fmap.

int FileMap::FileMap::CreateDataBase( ) [static]

If possible, read file into Fmap.

FileKey FileMap::FileMap::CreateKey(FileRecord &) [static]

Extract key from **FileRecord**.

bool FileMap::FileMap::DeleteKeyRec(FileKey &) [static]

Delete record by key.

Delete record by key Destructive InsertRec() i.e. overwrite any record with same key

FileMap::FileMap::FileMap( ) [inline]

Data members and methods for FileMap. FileMap corresponds to a file record, and is indexed by a key composed of the GCL_StringID object [i.e. 5 substrings] and the processor name.

Definition at line **447** of file **history.doxygen.h**.

References **FileMap::FileMap()**.

Referenced by **FileMap::FileMap()**.

```
00447 {};
```

void FileMap::FileMap::GenerateFileKey(BaseKey & , FileKey &) [static]

Convert **BaseKey** to **FileKey**.

Convert a **BaseKey** 'Bkey' to a **FileKey** 'Fkey'.

The **BaseKey** is a relative key which is composed of two scalars: GCL_NodeID & MW_ProcessID. GCL_NodeID is an index to the Nodes in a specific graph, and MW_ProcessID is the MPI rank of the process.

The **FileKey** is an absolute key which is composed of a vector of strings extracted from a GCL_StringIDs object, and a string specifying the name of the processor.

int FileMap::FileMap::GetFkeyRec(FileKey::FileKey &, map< BaseKey::BaseKey, BaseRecord::BaseRecord >::iterator) [static]

FileMap::GetFkeyRec() is called by BaseMap::InitializeFromFmap().

If the desired key is not in the map, the key is inserted, and the default **FileRecord** (built by the **FileRecord** default constructor) is returned.

In other words, a record is ALWAYS returned.

FileRecord& FileMap::FileMap::GetKeyRec(FileKey &) [static]

Get record by key.

bool FileMap::FileMap::InsertRec(FileRecord &) [static]

FileMap::InsertRec writes a record into Fmap.

FileMap::InsertRec inserts a file record into **FileMap::Fmap**. InsertRec returns true is record insertion is successful. This is a non-destructive operation; if the key already exists, the record will not be inserted and the return value is false.

void FileMap::FileMap::NormalizeDataBase( ) [static]

Class FileMap::NormalizeDataBase().

Normalize the *averaged* file records only. Nsample can be greater than one for both averaged and non-averaged records, so an averaged record is implied by a _negative_ value for Nsamples.

bool FileMap::FileMap::QueryKeyRec(FileKey &) [static]

Query record by key.

QueryKeyRec returns true if the key is in the FileMap.

bool FileMap::FileMap::ResetKeyRec(FileKey &) [static]

Set Mean, Variance, and Nsamples of record to zero.

int FileMap::FileMap::SaveDataBase( ) [static]

Write **FileMap::Fmap** to disk.

bool FileMap::FileMap::UpdateRec(FileRecord &) [static]

void FileMap::FileMap::ViewDataBase( ) [static]

Member Data Documentation

map< FileKey::FileKey, FileRecord::FileRecord > FileMap::Fmap [static]

Definition at line **525** of file **history.doxygen.h**.

const string FileMap::TokenProcessName [static]

HostName for Token size records.

Definition at line **521** of file **history.doxygen.h**.

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

- **history.doxygen.h**

Generated on Mon Oct 7 01:04:30 2002 for PGMT by doxygen 1.2.18

FileMap Member List

This is the complete list of members for **FileMap**, including all inherited members.

FileMap::ClearDataBase()	**FileMap**	[static]
FileMap::CreateDataBase()	**FileMap**	[static]
FileMap::CreateKey(FileRecord &)	**FileMap**	[static]
FileMap::DeleteKeyRec(FileKey &)	**FileMap**	[static]
FileMap::FileMap()	**FileMap**	[inline]
FileMap::GenerateFileKey(BaseKey &, FileKey &)	**FileMap**	[static]
FileMap::GetFkeyRec(FileKey::FileKey &, map< BaseKey::BaseKey, BaseRecord::BaseRecord >::iterator)	**FileMap**	[static]
FileMap::GetKeyRec(FileKey &)	**FileMap**	[static]
FileMap::InsertRec(FileRecord &)	**FileMap**	[static]
FileMap::NormalizeDataBase()	**FileMap**	[static]
FileMap::QueryKeyRec(FileKey &)	**FileMap**	[static]
FileMap::ResetKeyRec(FileKey &)	**FileMap**	[static]
FileMap::SaveDataBase()	**FileMap**	[static]
FileMap::UpdateRec(FileRecord &)	**FileMap**	[static]
FileMap::ViewDataBase()	**FileMap**	[static]
FileMap::~FileMap()	**FileMap**	[inline]
Fmap	**FileMap**	[static]
TokenProcessName	**FileMap**	[static]

FileKey Class Reference

`#include <history.doxygen.h>`

List of all members.

Detailed Description

class FileKey

Define key for **FileMap** database. Making this class internal to **FileMap** seems to break the map insertion operation.

Definition at line **419** of file **history.doxygen.h**.

Public Methods

	FileKey::FileKey ()
	FileKey::~FileKey ()
	FileKey::FileKey (vector< string >, string)
const string &	**FileKey::get__Vstring** (int) const
const string &	**FileKey::get__HostName** () const
void	**FileKey::set__FileKey** (const GCL_StringIDs &, const string &)
void	**FileKey::set__HostName** (const string &)
const vector< string > &	**FileKey::get__VstringIDs** () const

Public Attributes

vector< string >	**_VstringIDs**

string **_HostName**

Friends

class **FileMap**

Constructor & Destructor Documentation

FileKey::FileKey::~FileKey( ) [inline]

Definition at line **424** of file **history.doxygen.h**.

References **FileKey::~FileKey()**.

Referenced by **FileKey::~FileKey()**.

```
00424 {};
```

Member Function Documentation

FileKey::FileKey::FileKey(vector< string > ,
string
)

FileKey::FileKey::FileKey( )

const string& FileKey::FileKey::get__HostName( )

const string& FileKey::FileKey::get__Vstring(int) const

const vector<string>& FileKey::FileKey::get__VstringIDs( )

void FileKey::FileKey::set__FileKey(const GCL_StringIDs & ,
const string &
)

void FileKey::FileKey::set__HostName(const string &)

Friends And Related Function Documentation

friend class FileMap `[friend]`

Definition at line **420** of file **history.doxygen.h**.

Member Data Documentation

string FileKey::_HostName

Definition at line **434** of file **history.doxygen.h**.

vector< string > FileKey::_VstringIDs

Definition at line **433** of file **history.doxygen.h**.

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

- **history.doxygen.h**

Generated on Mon Oct 7 01:04:30 2002 for PGMT by doxygen 1.2.18

FileKey Member List

This is the complete list of members for **FileKey**, including all inherited members.

_HostName	**FileKey**	
_VstringIDs	**FileKey**	
FileKey::FileKey()	**FileKey**	
FileKey::FileKey(vector< string >, string)	**FileKey**	
FileKey::get__HostName() const	**FileKey**	
FileKey::get__Vstring(int) const	**FileKey**	
FileKey::get__VstringIDs() const	**FileKey**	
FileKey::set__FileKey(const GCL_StringIDs &, const string &)	**FileKey**	
FileKey::set__HostName(const string &)	**FileKey**	
FileKey::~FileKey()	**FileKey**	`[inline]`
FileMap class	**FileKey**	`[friend]`

FileOp Class Reference

`#include <history.doxygen.h>`

List of all members.

Detailed Description

Class FileOp and methods.

Class FileOp defines the data structures and methods for reading & writing FileRecords to and from disk.

Definition at line **360** of file **history.doxygen.h**.

Public Methods

FileOp::FileOp ()
FileOp::~FileOp ()

Static Public Methods

bool **OpenWriteFile** (const char *, ofstream &)
bool **OpenReadFile** (const char *, ifstream &)
bool **CloseWriteFile** (ofstream &)
bool **CloseReadFile** (ifstream &)
bool **WriteRecord** (ofstream &, **FileRecord** &)
*Method **FileOp::WriteRecord**.*

bool **ReadRecord** (ifstream &, **FileRecord** &)

*Method **FileOp::ReadRecord**.*

const char * **GetFileNameIn** ()
const char * **GetFileNameOut** ()
void **CatFile** (const char *, char *)
void **PrintPepHistoryFile** ()
*Method **FileOp::PrintPepHistoryFile()**;.*

Constructor & Destructor Documentation

FileOp::FileOp::~FileOp( ) [inline]

Definition at line **366** of file **history.doxygen.h**.

```
00366 {};
```

Member Function Documentation

**void FileOp::CatFile(const char * ,
char *
) [static]**

bool FileOp::CloseReadFile(ifstream &) [static]

bool FileOp::CloseWriteFile(ofstream &) [static]

FileOp::FileOp::FileOp( ) [inline]

File routines for reading & writing FileRecords to and from disk.

Definition at line **365** of file **history.doxygen.h**.

References **FileOp::FileOp()**.

Referenced by **FileOp::FileOp()**.

```
00365 {};
```

const char * FileOp::GetFileNameIn( ) [inline, static]

Definition at line **740** of file **history.doxygen.h**.

```
00740                                {
00741   return FileOp::FileNameIn; }
```

const char * FileOp::GetFileNameOut( ) [inline, static]

Definition at line **746** of file **history.doxygen.h**.

```
00746                                 {
00747   return FileOp::FileNameOut; }
```

bool FileOp::OpenReadFile(const char * , ifstream &) [static]

bool FileOp::OpenWriteFile(const char * , ofstream &) [static]

void FileOp::PrintPepHistoryFile( ) [static]

Method **FileOp::PrintPepHistoryFile()**;.

This function writes the contents of the of the Pep Input and Output History Files to stdout -- if the file exists.

bool FileOp::ReadRecord(ifstream & , FileRecord &) [static]

Method **FileOp::ReadRecord**.

Read a single **FileRecord** from the disk.

bool FileOp::WriteRecord(ofstream & , FileRecord &) [static]

Method **FileOp::WriteRecord**.

Write one **FileRecord** to outfile. Note that it is necessary to save (i.e. write) the length of each string before writing the string.

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

- **history.doxygen.h**

Generated on Mon Oct 7 01:04:30 2002 for PGMT by doxygen 1.2.18

FileOp Member List

This is the complete list of members for **FileOp**, including all inherited members.

CatFile(const char *, char *)	**FileOp**	[static]
CloseReadFile(ifstream &)	**FileOp**	[static]
CloseWriteFile(ofstream &)	**FileOp**	[static]
FileOp::FileOp()	**FileOp**	[inline]
FileOp::~FileOp()	**FileOp**	[inline]
GetFileNameIn()	**FileOp**	[inline, static]
GetFileNameOut()	**FileOp**	[inline, static]
OpenReadFile(const char *, ifstream &)	**FileOp**	[static]
OpenWriteFile(const char *, ofstream &)	**FileOp**	[static]
PrintPepHistoryFile()	**FileOp**	[static]
ReadRecord(ifstream &, FileRecord &)	**FileOp**	[static]
WriteRecord(ofstream &, FileRecord &)	**FileOp**	[static]

Generated on Mon Oct 7 01:04:30 2002 for PGMT by doxygen 1.2.18

MW_Graph Class Reference

`#include <MW_Graph.doxygen.h>`

List of all members.

Public Methods

MW_Graph ()
~MW_Graph ()

Static Public Methods

MW_ProcessID **getCurrentProcessID** ()
Return the current MW_ProcessID.

unsigned int **sendToken** (GCL_MsgHeader *header, GCL_BaseType baseType, MW_ProcessID DestProc, unsigned int descriptorSize, unsigned int dataSize, int *descriptor, char *data)
*function unsigned int **sendToken()***

void **sendCapacityUpdate** (GCL_MsgHeader *header, MW_ProcessID DestProc)
function void sendCapacityUpdate(...)

void **TokenMark** (GCL_MsgHeader *header, MW_ProcessID DestProc)
function void TokenMark(...)

unsigned int **getTokenMsgSize** (GCL_BaseType baseType, unsigned int descriptorSize, unsigned int dataSize)
*function **getTokenMsgSize()***

bool **getMsgHeader** (GCL_MsgHeader *header)
*function **getMsgHeader(GCL_MsgHeader * header)***

unsigned int **receiveToken** (GCL_BaseType baseType, int *descriptor, char *data)

function receiveToken(...)

void **CreateGraphComm** (vector< int > &Group)
Create the GCL MW MPI communicator.

void **build_MsgHeader_MPI_datatype** ()
Build the MPI datatype MPI_MsgHeader.

Static Public Attributes

MPI_Comm **GCL_MW_COMM**
MPI_Datatype **MPI_MsgHeader**

Friends

class **MW_Tester**

Constructor & Destructor Documentation

MW_Graph::MW_Graph( ) [inline]

Definition at line **65** of file **MW_Graph.doxygen.h**.

```
00065 {};  // default constructor
```

MW_Graph::~MW_Graph( ) [inline]

Definition at line **66** of file **MW_Graph.doxygen.h**.

```
00066 {};  // destructor
```

Member Function Documentation

void MW_Graph::build_MsgHeader_MPI_datatype( ) [static]

Build the MPI datatype MPI_MsgHeader.

void MW_Graph::CreateGraphComm(vector< int > & *Group*) [static]

Create the GCL MW MPI communicator.

MW_ProcessID MW_Graph::getCurrentProcessID( ) [static]

Return the current MW_ProcessID.

bool MW_Graph::getMsgHeader(GCL_MsgHeader * *header*) [static]

function **getMsgHeader(GCL_MsgHeader * header)**

(1) Throw an exception if a receive is in progress.
(2) If no receive is in progress, probe for an incoming message.
(3) Return false if there is no incoming message.
(4) If there _is_ an incoming message, determine the source and size of the message.
(5) Resize vector<char>TokenRcvBuffer to count (# MPI_PACKED elements) being received.
(6) Receive the entire array of MPI_PACKED elements.
(7) Unpack the lengths of the descriptor and data vectors
(i.e. the packing slip) and the Message Header.
(8) If DesLen and DataLen are zero, this is a token mark.
Reset RcvInProgress to false; reset positionRcv to zero;
and return. Token receive operation is complete.
(9) If DesLen or DataLen are > 0, this is a regular token.

unsigned int MW_Graph::getTokenMsgSize(GCL_BaseType *baseType*, unsigned int *descriptorSize*, unsigned int *dataSize*) [static]

function **getTokenMsgSize()**

This function returns the estimated size in bytes of an MPI token message which has been encoded as an array [i.e. 1-dimensional vector] of type MPI_PACK.

descriptorSize => number of integers in descriptor array dataSize => number of elements of type baseType in data array

unsigned int MW_Graph::receiveToken(	**GCL_BaseType**	***baseType,***
	int *	***descriptor,***
	char *	***data***
	) [static]	

function receiveToken(...)

The title of this function is a bit misleading. When this is called, the token has already been received to the local process as a message of type MPI_PACK. This function unpacks the descriptor and data segments to the addresses passed by the GCL calls.

void MW_Graph::sendCapacityUpdate(	**GCL_MsgHeader ***	***header,***
	MW_ProcessID	***DestProc***
	) [static]	

function void sendCapacityUpdate(...)

send queue capacity update.

unsigned int MW_Graph::sendToken(	**GCL_MsgHeader ***	***header,***
	GCL_BaseType	***baseType,***
	MW_ProcessID	***DestProc,***
	unsigned int	***descriptorSize,***
	unsigned int	***dataSize,***
	int *	***descriptor,***
	char *	***data***
	) [static]	

function unsigned int **sendToken()**

This function packs a PGM token as two components: a descriptor, and the data component proper. These are sent as a message of type MPI_PACK. The function inputs are

Parameters:

header	A pointer to a GCL_MsgHeader
baseType	The GCL_BaseType for the token
DestProc	The MW_ProcessID for the destination
descriptor	The descriptor size
dataSize	The dataSize (as a contignous vector)
descriptor	An integer pointer to the token descriptor
data	A char pointer to the data component.

The entire token is stored dynamically on the heap until MPI can send it to the destination process. Packed tokens will be deleted when the immediate send REQUEST is satisfied. Send completion REQUESTs will be checked everytime this routine is entered.

void MW_Graph::TokenMark(GCL_MsgHeader * *header*, MW_ProcessID *DestProc*) [static]

function void TokenMark(...)

This message is sent during during transition reassignment to delimit the end of a sequence of token transmissions.

Designate the process to which the transition is initially assigned as the "original owner." Designate the process to which the transition is reassigned as the "new owner." Designate the process sending tokens to the old\new owner as the "source process."

(a) This message is sent from the source process to the original owner to indicated that no more tokens will be sent to the source process.
(b) This message is also sent by the new owner to the source process to indicate that all "old" tokens have been received by the new owner from the old owner.

Friends And Related Function Documentation

friend class MW_Tester [friend]

Definition at line **193** of file **MW_Graph.doxygen.h**.

Member Data Documentation

MPI_Comm MW_Graph::GCL_MW_COMM [static]

Definition at line **190** of file **MW_Graph.doxygen.h**.

MPI_Datatype MW_Graph::MPI_MsgHeader [static]

Definition at line **191** of file **MW_Graph.doxygen.h**.

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

- **MW_Graph.doxygen.h**

Generated on Mon Oct 7 01:04:30 2002 for PGMT by doxygen 1.2.18

MW_Graph Member List

This is the complete list of members for **MW_Graph**, including all inherited members.

build_MsgHeader_MPI_datatype()	**MW_Graph**	[static]
CreateGraphComm(vector< int > &Group)	**MW_Graph**	[static]
GCL_MW_COMM	**MW_Graph**	[static]
getCurrentProcessID()	**MW_Graph**	[static]
getMsgHeader(GCL_MsgHeader *header)	**MW_Graph**	[static]
getTokenMsgSize(GCL_BaseType baseType, unsigned int descriptorSize, unsigned int dataSize)	**MW_Graph**	[static]
MPI_MsgHeader	**MW_Graph**	[static]
MW_Graph()	**MW_Graph**	[inline]
MW_Tester class	**MW_Graph**	[friend]
receiveToken(GCL_BaseType baseType, int *descriptor, char *data)	**MW_Graph**	[static]
sendCapacityUpdate(GCL_MsgHeader *header, MW_ProcessID DestProc)	**MW_Graph**	[static]
sendToken(GCL_MsgHeader *header, GCL_BaseType baseType, MW_ProcessID DestProc, unsigned int descriptorSize, unsigned int dataSize, int *descriptor, char *data)	**MW_Graph**	[static]
TokenMark(GCL_MsgHeader *header, MW_ProcessID DestProc)	**MW_Graph**	[static]
~MW_Graph()	**MW_Graph**	[inline]

MW_Graph.doxygen.h File Reference

Detailed Description

Definition in file **MW_Graph.doxygen.h**.`#include "GCL/GCL_MsgHeader.h"`

```
#include "global.h"
#include "Middleware/MW_MPI.h"
#include "Middleware/Machine.h"
#include <vector.h>
#include <list>
#include <fstream.h>
#include <string.h>
#include <mpi.h>
```

Go to the source code of this file.

Compounds

class	**HeapAllocationFailure**
class	**GCL_MsgHeader_Public**
class	**MW_Graph**
struct	**TokenRecord**
class	**SplitReceive**

Doxygen

Introduction

Manual
Mailing Lists
Mail archive
FAQ
ChangeLog
Todo / Wish List
Report bugs
Doxygen users
Intro (in Chinese)
Other Doc
Tools
Donate

CVS
Sources
Binaries
Download manual
Helper tools
RPM Packages
Debian Packages

Doxygen is a documentation system for C++, C, Java, IDL (Corba, Microsoft, and KDE-DCOP flavors) and to some extent PHP.

It can help you in three ways:

1. It can generate an on-line documentation browser (in HTML) and/or an off-line reference manual (in) from a set of documented source files. There is also support for generating output in RTF (MS-Word), PostScript, hyperlinked PDF, compressed HTML, and Unix man pages. The documentation is extracted directly from the sources, which makes it much easier to keep the documentation consistent with the source code.
2. Doxygen can be configured to extract the code structure from undocumented source files. This can be very useful to quickly find your way in large source distributions. The relations between the various elements are be visualized by means of include dependency graphs, inheritance diagrams, and collaboration diagrams, which are all generated automatically.
3. You can even `abuse' doxygen for creating normal documentation (as I did for this manual).

Doxygen is developed under Linux, but is set-up to be highly portable. As a result, it runs on most other Unix flavors as well. Furthermore, executables for Windows 9x/NT and Mac OS X are available.

Doxygen license

Copyright © 1997-2002 by Dimitri van Heesch.

Permission to use, copy, modify, and distribute this software and its documentation under the terms of the GNU General Public License is hereby granted. No representations are made about the suitability of this software for any purpose. It is provided "as is" without express or implied warranty. See the GNU General Public License for more details.

Documents produced by doxygen are derivative works derived from the input used in their production; they are not affected by this license.

Projects using doxygen

I have compiled a list of projects that use doxygen. If you know other projects, let me know and I'll add them.

Future work

Although doxygen is used successfully by a lot of people already, there is always room for improvement. Therefore, I have compiled a todo/wish list of possible and/or requested enhancements.

Donate

The best way to help the doxygen project is to participate as a developer. I realise that not everyone has to time to do this however, so another way to help is to donate some money. This would allow me to cover some of the costs involved in developing doxygen (internet access, equipment, etc.). Especially if you are using doxygen in a commercial environment I ask you to consider this option so doxygen can continue to be free in the future. A donation can be made quite conveniently using PayPal's services, just press on the PayPal button in the side panel. Please contact me if you prefer other methods of payment.

Acknowledgements

Thanks go to:

- Malte Zöckler and Roland Wunderling, authors of DOC++. The first version of doxygen borrowed some code of an old version of DOC++. Although I have rewritten practically all code since then, DOC++ has still given me a good start in writing doxygen.
- All people at Troll Tech, for creating a beautiful GUI Toolkit (which is very useful as a Windows/Unix platform abstraction layer :-)
- My brother Frank for rendering the logos.
- Harm van der Heijden for adding HTML help support.
- Wouter Slegers of Your Creative Solutions for registering the www.doxygen.org domain.
- Parker Waechter for adding the RTF output generator.
- Joerg Baumann, for adding conditional documentation blocks, PDF links, and the configuration generator.
- Matthias Andree for providing a .spec script for building rpms from the sources.
- Tim Mensch for adding the todo command.
- Christian Hammond for redesigning the web-site.
- Ken Wong for providing the HTML tree view code.

- Petr Prikryl for coordinating the internationalisation support. All language maintainers for providing translations into many languages.
- Erik Jan Lingen of Habanera, Mark Roddy, Paul Schwartz, Charles Duffy, Vadym Voznyuk, Philip Walton, Dwight Browne, Andreas Fredriksson, Karel Lindveld, and Ivan Lee for donating money.
- The Comms group of Symbian for donating an ultra cool Revo plus organizer!
- The band Porcupine Tree for providing hours of great music to listen to while coding.
- many, many others for suggestions, patches and bug reports.

You are visitor number **1196303** since 27 October 1997.
This page was last modified on 17 Oct 2002.