hypre Reference Manual: Babel-based Interface in C

— Version 2.9.1a —

Contents

1 Matrix and Vector Views (Conceptual Interfaces)

Names

1.1 IJ Matrix View

Names

1.1.1

struct **bHYPRE_IJMatrixView__object**

Symbol "bHYPREIJMatrixView" (version 100)

This interface represents a linear-algebraic conceptual view of a linear system. The 'I' and 'J' in the name are meant to be mnemonic for the traditional matrix notation A(I,J).

1.1.2

bHYPRE_IJMatrixView
bHYPRE_IJMatrixView__connect (const char* , sidl_BaseInterface* _ex)

RMI connector function for the class(addrefs)

1.1.3

SIDL_C_INLINE_DECL int32_t
bHYPRE_IJMatrixView_SetLocalRange (bHYPRE_IJMatrixView self, int32_t ilower, int32_t iupper, int32_t jlower, int32_t jupper, sidl_BaseInterface* _ex)

Set the local range for a matrix object. Each process owns some unique consecutive range of rows, indicated by the global row indices `ilower` and `iupper`. The row data is required to be such that the value of `ilower` on any process p be exactly one more than the value of `iupper` on process $p-1$. Note that the first row of the global matrix may start with any integer value. In particular, one may use zero- or one-based indexing.

For square matrices, `jlower` and `jupper` typically should match `ilower` and `iupper`, respectively. For rectangular matrices, `jlower` and `jupper` should define a partitioning of the columns. This partitioning must be used for any vector v that will be used in matrix-vector products with the rectangular matrix. The matrix data structure may use `jlower` and `jupper` to store the diagonal blocks (rectangular in general) of the matrix separately from the rest of the matrix.

Collective.

1.1.4

int32_t
bHYPRE_IJMatrixView_SetValues (bHYPRE_IJMatrixView self, int32_t nrows, int32_t* ncols, int32_t* rows, int32_t* cols, double* values, int32_t nnonzeros, sidl_BaseInterface* _ex)

Sets values for `nrows` of the matrix. The arrays `ncols` and `rows` are of dimension `nrows` and contain the number of columns in each row and the row indices, respectively. The array `cols` contains the column indices for each of the `rows`, and is ordered by rows. The data in the `values` array corresponds directly to the column entries in `cols`. The last argument is the size of the cols and values arrays, i.e. the total number of nonzeros being provided, i.e. the sum of all values in ncols. This functin erases any previous values at the specified locations and replaces them with new ones, or, if there was no value there before, inserts a new one.

Not collective.

1.1.5

int32_t
bHYPRE_IJMatrixView_AddToValues (bHYPRE_IJMatrixView self, int32_t nrows, int32_t* ncols, int32_t* rows, int32_t* cols, double* values, int32_t nnonzeros, sidl_BaseInterface* _ex)

Adds to values for `nrows` of the matrix. Usage details are analogous to `SetValues`. Adds to any previous values at the specified locations, or, if there was no value there before, inserts a new one.

Not collective.

1.1.6

SIDL_C_INLINE_DECL int32_t
bHYPRE_IJMatrixView_GetLocalRange (bHYPRE_IJMatrixView self, int32_t* ilower, int32_t* iupper, int32_t* jlower, int32_t* jupper, sidl_BaseInterface* _ex)

Gets range of rows owned by this processor and range of column partitioning for this processor

1.1.7

int32_t
bHYPRE_IJMatrixView_GetRowCounts (bHYPRE_IJMatrixView self, int32_t nrows, int32_t* rows, int32_t* ncols, sidl_BaseInterface* _ex)

Gets number of nonzeros elements for `nrows` rows specified in `rows` and returns them in `ncols`, which needs to be allocated by the user

1.1.8

int32_t
bHYPRE_IJMatrixView_GetValues (bHYPRE_IJMatrixView self, int32_t nrows, int32_t* ncols, int32_t* rows, int32_t* cols, double* values, int32_t nnonzeros, sidl_BaseInterface* _ex)

Gets values for `nrows` rows or partial rows of the matrix. Usage details are analogous to `SetValues`.

1.1.9

SIDL_C_INLINE_DECL int32_t
bHYPRE_IJMatrixView_SetRowSizes (bHYPRE_IJMatrixView self, int32_t* sizes, int32_t nrows, sidl_BaseInterface* _ex)

(Optional) Set the max number of nonzeros to expect in each row. The array `sizes` contains estimated sizes for each row on this process. The integer nrows is the number of rows in the local matrix. This call can significantly improve the efficiency of matrix construction, and should always be utilized if possible.

Not collective.

1.1.10

SIDL_C_INLINE_DECL int32_t
bHYPRE_IJMatrixView_Print (bHYPRE_IJMatrixView self, const char* filename, sidl_BaseInterface* _ex)

Print the matrix to file. This is mainly for debugging purposes.

1.1.11

SIDL_C_INLINE_DECL int32_t
bHYPRE_IJMatrixView_Read (bHYPRE_IJMatrixView self, const char* filename, bHYPRE_MPICommunicator comm, sidl_BaseInterface* _ex)

Read the matrix from file. This is mainly for debugging purposes.

1.1.12

struct bHYPRE_IJMatrixView__object*
bHYPRE_IJMatrixView__cast (void* obj, sidl_BaseInterface* _ex)

Cast method for interface and class type conversions

1.1.13

void*
bHYPRE_IJMatrixView__cast2 (void* obj, const char* type, sidl_BaseInterface* _ex)

String cast method for interface and class type conversions

1.1.14

SIDL_C_INLINE_DECL void
bHYPRE_IJMatrixView__exec (bHYPRE_IJMatrixView self, const char* methodName, sidl_rmi_Call inArgs, sidl_rmi_Return outArgs, sidl_BaseInterface* _ex)

Select and execute a method by name

1.1.15

SIDL_C_INLINE_DECL char*
bHYPRE_IJMatrixView__getURL (bHYPRE_IJMatrixView self,
sidl_BaseInterface* _ex)

Get the URL of the Implementation of this object (for RMI)

1.1.16

SIDL_C_INLINE_DECL void
bHYPRE_IJMatrixView__raddRef (bHYPRE_IJMatrixView self,
sidl_BaseInterface* _ex)

On a remote object, addrefs the remote instance

1.1.17

SIDL_C_INLINE_DECL sidl_bool
bHYPRE_IJMatrixView__isRemote (bHYPRE_IJMatrixView self,
sidl_BaseInterface* _ex)

TRUE if this object is remote, false if local

1.1.18

sidl_bool
bHYPRE_IJMatrixView__isLocal (bHYPRE_IJMatrixView self,
sidl_BaseInterface* _ex)

TRUE if this object is remote, false if local

1.1.19

struct bHYPRE_IJMatrixView__object*
bHYPRE_IJMatrixView__rmicast (void* obj, struct sidl_BaseInterface__object** _ex)

Cast method for interface and class type conversions

1.1.20

struct bHYPRE_IJMatrixView__object*
bHYPRE_IJMatrixView__connectI (const char* url, sidl_bool ar, struct sidl_BaseInterface__object** _ex)

RMI connector function for the class. (no addref)

1.2

IJ Vector View

Names

1.2.1

struct **bHYPRE_IJVectorView__object**

Symbol "bHYPREIJVectorView" (version 100)

1.2.2

bHYPRE_IJVectorView
bHYPRE_IJVectorView__connect (const char* , sidl_BaseInterface* _ex)

RMI connector function for the class(addrefs)

1.2.3

SIDL_C_INLINE_DECL int32_t
bHYPRE_IJVectorView_SetLocalRange (bHYPRE_IJVectorView self, int32_t jlower, int32_t jupper, sidl_BaseInterface* _ex)

Set the local range for a vector object. Each process owns some unique consecutive range of vector unknowns, indicated by the global indices `jlower` and `jupper`. The data is required to be such that the value of `jlower` on any process p be exactly one more than the value of `jupper` on process $p-1$. Note that the first index of the global vector may start with any integer value. In particular, one may use zero- or one-based indexing.

Collective.

1.2.4

int32_t
bHYPRE_IJVectorView_SetValues (bHYPRE_IJVectorView self, int32_t nvalues, int32_t* indices, double* values, sidl_BaseInterface* _ex)

Sets values in vector. The arrays `values` and `indices` are of dimension `nvalues` and contain the vector values to be set and the corresponding global vector indices, respectively. Erases any previous values at the specified locations and replaces them with new ones.

Not collective.

1.2.5

int32_t
bHYPRE_IJVectorView_AddToValues (bHYPRE_IJVectorView self, int32_t nvalues, int32_t* indices, double* values, sidl_BaseInterface* _ex)

Adds to values in vector. Usage details are analogous to `SetValues`.

Not collective.

1.2.6

SIDL_C_INLINE_DECL int32_t
bHYPRE_IJVectorView_GetLocalRange (bHYPRE_IJVectorView self, int32_t* jlower, int32_t* jupper, sidl_BaseInterface* _ex)

Returns range of the part of the vector owned by this processor

1.2.7

int32_t
bHYPRE_IJVectorView_GetValues (bHYPRE_IJVectorView self, int32_t nvalues, int32_t* indices, double* values, sidl_BaseInterface* _ex)

Gets values in vector. Usage details are analogous to `SetValues`.

Not collective.

1.2.8

SIDL_C_INLINE_DECL int32_t
bHYPRE_IJVectorView_Print (bHYPRE_IJVectorView self, const char* filename, sidl_BaseInterface* _ex)

Print the vector to file. This is mainly for debugging purposes.

1.2.9

SIDL_C_INLINE_DECL int32_t
bHYPRE_IJVectorView_Read (bHYPRE_IJVectorView self, const char* filename, bHYPRE_MPICommunicator comm, sidl_BaseInterface* _ex)

Read the vector from file. This is mainly for debugging purposes.

1.2.10

struct bHYPRE_IJVectorView__object*
bHYPRE_IJVectorView__cast (void* obj, sidl_BaseInterface* _ex)

Cast method for interface and class type conversions

1.2.11

void*
bHYPRE_IJVectorView__cast2 (void* obj, const char* type,
sidl_BaseInterface* _ex)

String cast method for interface and class type conversions

1.2.12

SIDL_C_INLINE_DECL void
bHYPRE_IJVectorView__exec (bHYPRE_IJVectorView self, const char*
methodName, sidl_rmi_Call inArgs, sidl_rmi_Return outArgs, sidl_BaseInterface*
_ex)

Select and execute a method by name

1.2.13

SIDL_C_INLINE_DECL char*
bHYPRE_IJVectorView__getURL (bHYPRE_IJVectorView self,
sidl_BaseInterface* _ex)

Get the URL of the Implementation of this object (for RMI)

1.2.14

SIDL_C_INLINE_DECL void
bHYPRE_IJVectorView__raddRef (bHYPRE_IJVectorView self,
sidl_BaseInterface* _ex)

On a remote object, addrefs the remote instance

1.2.15

SIDL_C_INLINE_DECL sidl_bool
bHYPRE_IJVectorView__isRemote (bHYPRE_IJVectorView self,
sidl_BaseInterface* _ex)

TRUE if this object is remote, false if local

1.2.16

sidl_bool
bHYPRE_IJVectorView__isLocal (bHYPRE_IJVectorView self,
sidl_BaseInterface* _ex)

TRUE if this object is remote, false if local

1.2.17

struct bHYPRE_IJVectorView__object*
bHYPRE_IJVectorView__rmicast (void* obj, struct
sidl_BaseInterface__object** _ex)

Cast method for interface and class type conversions

1.2.18

struct bHYPRE_IJVectorView__object*
bHYPRE_IJVectorView__connectI (const char* url, sidl_bool ar, struct sidl_BaseInterface__object** _ex)

RMI connector function for the class. (no addref)

1.3

Struct Matrix View

Names

bHYPRE_StructMatrixView_SetBoxValues (bHYPRE_StructMatrixView self, int32_t* ilower, int32_t* iupper, int32_t dim, int32_t num_stencil_indices, int32_t* stencil_indices, double* values, int32_t nvalues, sidl_BaseInterface* _ex)

Set matrix values throughout a box in the grid, specified by its lower and upper corners. 24

1.3.7 SIDL_C_INLINE_DECL int32_t
bHYPRE_StructMatrixView_SetNumGhost (bHYPRE_StructMatrixView self, int32_t* num_ghost, int32_t dim2, sidl_BaseInterface* _ex)

Set the number of ghost zones, separately on the lower and upper sides for each dimension. 24

1.3.8 SIDL_C_INLINE_DECL int32_t
bHYPRE_StructMatrixView_SetSymmetric (bHYPRE_StructMatrixView self, int32_t symmetric, sidl_BaseInterface* _ex)

Call SetSymmetric with symmetric=1 to turn on symmetric matrix storage if available. 24

1.3.9 SIDL_C_INLINE_DECL int32_t
bHYPRE_StructMatrixView_SetConstantEntries (bHYPRE_StructMatrixView self, int32_t num_stencil_constant_points, int32_t* stencil_constant_points, sidl_BaseInterface* _ex)

State which stencil entries are constant over the grid. 24

1.3.10 int32_t
bHYPRE_StructMatrixView_SetConstantValues (bHYPRE_StructMatrixView self, int32_t num_stencil_indices, int32_t* stencil_indices, double* values, sidl_BaseInterface* _ex)

Provide values for matrix coefficients which are constant throughout the grid, one value for each stencil point. 25

1.3.11 struct bHYPRE_StructMatrixView__object*
bHYPRE_StructMatrixView__cast (void* obj, sidl_BaseInterface* _ex)

Cast method for interface and class type conversions 25

1.3.12 void*
bHYPRE_StructMatrixView__cast2 (void* obj, const char* type, sidl_BaseInterface* _ex)

String cast method for interface and class type conversions 25

1.3.13 SIDL_C_INLINE_DECL void

1.3.1

struct **bHYPRE_StructMatrixView__object**

Symbol "bHYPREStructMatrixView" (version 100)

1.3.2

bHYPRE_StructMatrixView
bHYPRE_StructMatrixView__connect (const char* , sidl_BaseInterface* _ex)

RMI connector function for the class(addrefs)

1.3.3

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructMatrixView_SetGrid (bHYPRE_StructMatrixView self, bHYPRE_StructGrid grid, sidl_BaseInterface* _ex)

Set the grid on which vectors are defined. This and the stencil determine the matrix structure.

1.3.4

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructMatrixView_SetStencil (bHYPRE_StructMatrixView self, bHYPRE_StructStencil stencil, sidl_BaseInterface* _ex)

Set the stencil. This and the grid determine the matrix structure.

1.3.5

int32_t
bHYPRE_StructMatrixView_SetValues (bHYPRE_StructMatrixView self, int32_t* index, int32_t dim, int32_t num_stencil_indices, int32_t* stencil_indices, double* values, sidl_BaseInterface* _ex)

Set matrix values at grid point, given by "index". You can supply values for one or more positions in the stencil. "index" is an array of size "dim"; and "stencil_indices" and "values" are arrays of size "num_stencil_indices".

1.3.6

int32_t
bHYPRE_StructMatrixView_SetBoxValues (bHYPRE_StructMatrixView self, int32_t* ilower, int32_t* iupper, int32_t dim, int32_t num_stencil_indices, int32_t* stencil_indices, double* values, int32_t nvalues, sidl_BaseInterface* _ex)

Set matrix values throughout a box in the grid, specified by its lower and upper corners. You can supply these values for one or more positions in the stencil. Thus the total number of matrix values you supply, "nvalues", is num_stencil_indices x box_size, where box_size is the number of grid points in the box. The values array should be organized so all values for a given box point are together (i.e., the stencil index is the most rapidly varying). "ilower" and "iupper" are arrays of size "dim", "stencil_indices" is an array of size "num_stencil_indices", and "values" is an array of size "nvalues".

1.3.7

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructMatrixView_SetNumGhost (bHYPRE_StructMatrixView self, int32_t* num_ghost, int32_t dim2, sidl_BaseInterface* _ex)

Set the number of ghost zones, separately on the lower and upper sides for each dimension. "num_ghost" is an array of size "dim2", twice the number of dimensions

1.3.8

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructMatrixView_SetSymmetric (bHYPRE_StructMatrixView self, int32_t symmetric, sidl_BaseInterface* _ex)

Call SetSymmetric with symmetric=1 to turn on symmetric matrix storage if available.

1.3.9

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructMatrixView_SetConstantEntries (
bHYPRE_StructMatrixView self, int32_t num_stencil_constant_points, int32_t* stencil_constant_points, sidl_BaseInterface* _ex)

State which stencil entries are constant over the grid. Supported options are: (i) none (the default), (ii) all (stencil_constant_points should include all stencil points) (iii) all entries but the diagonal.

1.3.10

int32_t
bHYPRE_StructMatrixView_SetConstantValues (
bHYPRE_StructMatrixView self, int32_t num_stencil_indices, int32_t*
stencil_indices, double* values, sidl_BaseInterface* _ex)

Provide values for matrix coefficients which are constant throughout the grid, one value for each stencil point. "stencil_indices" and "values" is each an array of length "num_stencil_indices"

1.3.11

struct bHYPRE_StructMatrixView__object*
bHYPRE_StructMatrixView__cast (void* obj, sidl_BaseInterface* _ex)

Cast method for interface and class type conversions

1.3.12

void*
bHYPRE_StructMatrixView__cast2 (void* obj, const char* type,
sidl_BaseInterface* _ex)

String cast method for interface and class type conversions

1.3.13

SIDL_C_INLINE_DECL void
bHYPRE_StructMatrixView__exec (bHYPRE_StructMatrixView self, const
char* methodName, sidl_rmi_Call inArgs, sidl_rmi_Return outArgs,
sidl_BaseInterface* _ex)

Select and execute a method by name

1.3.14

SIDL_C_INLINE_DECL char*
bHYPRE_StructMatrixView__getURL (bHYPRE_StructMatrixView self,
sidl_BaseInterface* _ex)

Get the URL of the Implementation of this object (for RMI)

1.3.15

SIDL_C_INLINE_DECL void
bHYPRE_StructMatrixView__raddRef (bHYPRE_StructMatrixView self,
sidl_BaseInterface* _ex)

On a remote object, addrefs the remote instance

1.3.16

SIDL_C_INLINE_DECL sidl_bool
bHYPRE_StructMatrixView__isRemote (bHYPRE_StructMatrixView self,
sidl_BaseInterface* _ex)

TRUE if this object is remote, false if local

1.3.17

sidl_bool
bHYPRE_StructMatrixView__isLocal (bHYPRE_StructMatrixView self,
sidl_BaseInterface* _ex)

TRUE if this object is remote, false if local

1.3.18

struct bHYPRE_StructMatrixView__object*
bHYPRE_StructMatrixView__rmicast (void* obj, struct sidl_BaseInterface__object** _ex)

Cast method for interface and class type conversions

1.3.19

struct bHYPRE_StructMatrixView__object*
bHYPRE_StructMatrixView__connectI (const char* url, sidl_bool ar, struct sidl_BaseInterface__object** _ex)

RMI connector function for the class. (no addref)

1.4

Struct Vector View

Names

1.4.1

struct **bHYPRE_StructVectorView__object**

Symbol "bHYPREStructVectorView" (version 100)

1.4.2

bHYPRE_StructVectorView
bHYPRE_StructVectorView__connect (const char* , sidl_BaseInterface* _ex)

RMI connector function for the class(addrefs)

1.4.3

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructVectorView_SetGrid (bHYPRE_StructVectorView self, bHYPRE_StructGrid grid, sidl_BaseInterface* _ex)

Set the grid on which vectors are defined.

1.4.4

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructVectorView_SetNumGhost (bHYPRE_StructVectorView self, int32_t* num_ghost, int32_t dim2, sidl_BaseInterface* _ex)

Set the number of ghost zones, separately on the lower and upper sides for each dimension. "num_ghost" is an array of size "dim2", twice the number of dimensions.

1.4.5

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructVectorView_SetValue (bHYPRE_StructVectorView self, int32_t* grid_index, int32_t dim, double value, sidl_BaseInterface* _ex)

Set the value of a single vector coefficient, given by "grid_index". "grid_index" is an array of size "dim", where dim is the number of dimensions.

1.4.6

int32_t
bHYPRE_StructVectorView_SetBoxValues (bHYPRE_StructVectorView self, int32_t* ilower, int32_t* iupper, int32_t dim, double* values, int32_t nvalues, sidl_BaseInterface* _ex)

Set the values of all vector coefficient for grid points in a box. The box is defined by its lower and upper corners in the grid. "ilower" and "iupper" are arrays of size "dim", where dim is the number of dimensions. The "values" array has size "nvalues", which is the number of grid points in the box.

1.4.7

struct bHYPRE_StructVectorView__object*
bHYPRE_StructVectorView__cast (void* obj, sidl_BaseInterface* _ex)

Cast method for interface and class type conversions

1.4.8

void*
bHYPRE_StructVectorView__cast2 (void* obj, const char* type, sidl_BaseInterface* _ex)

String cast method for interface and class type conversions

1.4.9

SIDL_C_INLINE_DECL void
bHYPRE_StructVectorView__exec (bHYPRE_StructVectorView self, const char* methodName, sidl_rmi_Call inArgs, sidl_rmi_Return outArgs, sidl_BaseInterface* _ex)

Select and execute a method by name

1.4.10

SIDL_C_INLINE_DECL char*
bHYPRE_StructVectorView__getURL (bHYPRE_StructVectorView self, sidl_BaseInterface* _ex)

Get the URL of the Implementation of this object (for RMI)

1.4.11

SIDL_C_INLINE_DECL void
bHYPRE_StructVectorView__raddRef (bHYPRE_StructVectorView self, sidl_BaseInterface* _ex)

On a remote object, addrefs the remote instance

1.4.12

SIDL_C_INLINE_DECL sidl_bool
bHYPRE_StructVectorView__isRemote (bHYPRE_StructVectorView self,
sidl_BaseInterface* _ex)

TRUE if this object is remote, false if local

1.4.13

sidl_bool
bHYPRE_StructVectorView__isLocal (bHYPRE_StructVectorView self,
sidl_BaseInterface* _ex)

TRUE if this object is remote, false if local

1.4.14

struct bHYPRE_StructVectorView__object*
bHYPRE_StructVectorView__rmicast (void* obj, struct
sidl_BaseInterface__object** _ex)

Cast method for interface and class type conversions

1.4.15

struct bHYPRE_StructVectorView__object*
bHYPRE_StructVectorView__connectI (const char* url, sidl_bool ar, struct
sidl_BaseInterface__object** _ex)

RMI connector function for the class. (no addref)

1.5

SemiStructured Matrix View

Names

1.5.1

struct **bHYPRE_SStructMatrixView__object**

Symbol "bHYPRESStructMatrixView" (version 100)

1.5.2

bHYPRE_SStructMatrixView **bHYPRE_SStructMatrixView__connect** (const char* , sidl_BaseInterface* _ex)

RMI connector function for the class(addrefs)

1.5.3

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructMatrixView_SetGraph (bHYPRE_SStructMatrixView self, bHYPRE_SStructGraph graph, sidl_BaseInterface* _ex)

Set the matrix graph. DEPRECATED Use Create

1.5.4

int32_t
bHYPRE_SStructMatrixView_SetValues (bHYPRE_SStructMatrixView self, int32_t part, int32_t* index, int32_t dim, int32_t var, int32_t nentries, int32_t* entries, double* values, sidl_BaseInterface* _ex)

Set matrix coefficients index by index.

NOTE: Users are required to set values on all processes that own the associated variables. This means that some data will be multiply defined.

NOTE: The entries in this routine must all be of the same type: either stencil or non-stencil, but not both. Also, if they are stencil entries, they must all represent couplings to the same variable type (there are no such restrictions for non-stencil entries).

If the matrix is complex, then `values` consists of pairs of doubles representing the real and imaginary parts of each complex value.

1.5.5

int32_t
bHYPRE_SStructMatrixView_SetBoxValues (bHYPRE_SStructMatrixView self, int32_t part, int32_t* ilower, int32_t* iupper, int32_t dim, int32_t var, int32_t nentries, int32_t* entries, double* values, int32_t nvalues, sidl_BaseInterface* _ex)

Set matrix coefficients a box at a time.

NOTE: Users are required to set values on all processes that own the associated variables. This means that some data will be multiply defined.

NOTE: The entries in this routine must all be of the same type: either stencil or non-stencil, but not both. Also, if they are stencil entries, they must all represent couplings to to the same variable type (there are no such restrictions for non-stencil entries).

If the matrix is complex, then `values` consists of pairs of doubles representing the real and imaginary parts of each complex value.

1.5.6

int32_t
bHYPRE_SStructMatrixView_AddToValues (bHYPRE_SStructMatrixView self, int32_t part, int32_t* index, int32_t dim, int32_t var, int32_t nentries, int32_t* entries, double* values, sidl_BaseInterface* _ex)

Add to matrix coefficients index by index.

NOTE: Users are required to set values on all processes that own the associated variables. This means that some data will be multiply defined.

NOTE: The entries in this routine must all be of the same type: either stencil or non-stencil, but not both. Also, if they are stencil entries, they must all represent couplings to to the same variable type.

If the matrix is complex, then `values` consists of pairs of doubles representing the real and imaginary parts of each complex value.

1.5.7

int32_t
bHYPRE_SStructMatrixView_AddToBoxValues (
bHYPRE_SStructMatrixView self, int32_t part, int32_t* ilower, int32_t* iupper, int32_t dim, int32_t var, int32_t nentries, int32_t* entries, double* values, int32_t nvalues, sidl_BaseInterface* _ex)

Add to matrix coefficients a box at a time.

NOTE: Users are required to set values on all processes that own the associated variables. This means that some data will be multiply defined.

NOTE: The entries in this routine must all be of stencil type. Also, they must all represent couplings to the same variable type.

If the matrix is complex, then `values` consists of pairs of doubles representing the real and imaginary parts of each complex value.

1.5.8

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructMatrixView_SetSymmetric (bHYPRE_SStructMatrixView self, int32_t part, int32_t var, int32_t to_var, int32_t symmetric, sidl_BaseInterface* _ex)

Define symmetry properties for the stencil entries in the matrix. The boolean argument `symmetric` is applied to stencil entries on part `part` that couple variable `var` to variable `to_var`. A value of -1 may be used for `part`, `var`, or `to_var` to specify "all". For example, if `part` and `to_var` are set to -1, then the boolean is applied to stencil entries on all parts that couple variable `var` to all other variables.

By default, matrices are assumed to be nonsymmetric. Significant storage savings can be made if the matrix is symmetric.

1.5.9

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructMatrixView_SetNSSymmetric (
bHYPRE_SStructMatrixView self, int32_t symmetric, sidl_BaseInterface* _ex)

Define symmetry properties for all non-stencil matrix entries

1.5.10

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructMatrixView_SetComplex (bHYPRE_SStructMatrixView self, sidl_BaseInterface* _ex)

Set the matrix to be complex

1.5.11

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructMatrixView_Print (bHYPRE_SStructMatrixView self, const char* filename, int32_t all, sidl_BaseInterface* _ex)

Print the matrix to file. This is mainly for debugging purposes.

1.5.12

struct bHYPRE_SStructMatrixView__object*
bHYPRE_SStructMatrixView__cast (void* obj, sidl_BaseInterface* _ex)

Cast method for interface and class type conversions

1.5.13

void*
bHYPRE_SStructMatrixView__cast2 (void* obj, const char* type,
sidl_BaseInterface* _ex)

String cast method for interface and class type conversions

1.5.14

SIDL_C_INLINE_DECL void
bHYPRE_SStructMatrixView__exec (bHYPRE_SStructMatrixView self,
const char* methodName, sidl_rmi_Call inArgs, sidl_rmi_Return outArgs,
sidl_BaseInterface* _ex)

Select and execute a method by name

1.5.15

SIDL_C_INLINE_DECL char*
bHYPRE_SStructMatrixView__getURL (bHYPRE_SStructMatrixView self,
sidl_BaseInterface* _ex)

Get the URL of the Implementation of this object (for RMI)

1.5.16

SIDL_C_INLINE_DECL void
bHYPRE_SStructMatrixView__raddRef (bHYPRE_SStructMatrixView self, sidl_BaseInterface* _ex)

On a remote object, addrefs the remote instance

1.5.17

SIDL_C_INLINE_DECL sidl_bool
bHYPRE_SStructMatrixView__isRemote (bHYPRE_SStructMatrixView self, sidl_BaseInterface* _ex)

TRUE if this object is remote, false if local

1.5.18

sidl_bool
bHYPRE_SStructMatrixView__isLocal (bHYPRE_SStructMatrixView self, sidl_BaseInterface* _ex)

TRUE if this object is remote, false if local

1.5.19

struct bHYPRE_SStructMatrixView__object*
bHYPRE_SStructMatrixView__rmicast (void* obj, struct sidl_BaseInterface__object** _ex)

Cast method for interface and class type conversions

1.5.20

struct bHYPRE_SStructMatrixView__object*
bHYPRE_SStructMatrixView__connectI (const char* url, sidl_bool ar, struct sidl_BaseInterface__object** _ex)

RMI connector function for the class. (no addref)

1.6

SemiStructured Vector View

Names

1.6.1

struct **bHYPRE_SStructVectorView__object**

Symbol "bHYPRESStructVectorView" (version 100)

1.6.2

bHYPRE_SStructVectorView
bHYPRE_SStructVectorView__connect (const char* , sidl_BaseInterface* _ex)

RMI connector function for the class(addrefs)

1.6.3

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructVectorView_SetGrid (bHYPRE_SStructVectorView self, bHYPRE_SStructGrid grid, sidl_BaseInterface* _ex)

Set the vector grid

1.6.4

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructVectorView_SetValues (bHYPRE_SStructVectorView self, int32_t part, int32_t* index, int32_t dim, int32_t var, double value, sidl_BaseInterface* _ex)

Set vector coefficients index by index.

NOTE: Users are required to set values on all processes that own the associated variables. This means that some data will be multiply defined.

If the vector is complex, then `value` consists of a pair of doubles representing the real and imaginary parts of the complex value.

1.6.5

int32_t
bHYPRE_SStructVectorView_SetBoxValues (bHYPRE_SStructVectorView self, int32_t part, int32_t* ilower, int32_t* iupper, int32_t dim, int32_t var, double* values, int32_t nvalues, sidl_BaseInterface* _ex)

Set vector coefficients a box at a time.

NOTE: Users are required to set values on all processes that own the associated variables. This means that some data will be multiply defined.

If the vector is complex, then `values` consists of pairs of doubles representing the real and imaginary parts of each complex value.

1.6.6

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructVectorView_AddToValues (bHYPRE_SStructVectorView self, int32_t part, int32_t* index, int32_t dim, int32_t var, double value, sidl_BaseInterface* _ex)

Set vector coefficients index by index.

NOTE: Users are required to set values on all processes that own the associated variables. This means that some data will be multiply defined.

If the vector is complex, then `value` consists of a pair of doubles representing the real and imaginary parts of the complex value.

1.6.7

int32_t
bHYPRE_SStructVectorView_AddToBoxValues (
bHYPRE_SStructVectorView self, int32_t part, int32_t* ilower, int32_t* iupper, int32_t dim, int32_t var, double* values, int32_t nvalues, sidl_BaseInterface* _ex)

Set vector coefficients a box at a time.

NOTE: Users are required to set values on all processes that own the associated variables. This means that some data will be multiply defined.

If the vector is complex, then `values` consists of pairs of doubles representing the real and imaginary parts of each complex value.

1.6.8

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructVectorView_Gather (bHYPRE_SStructVectorView self, sidl_BaseInterface* _ex)

Gather vector data before calling `GetValues`

1.6.9

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructVectorView_GetValues (bHYPRE_SStructVectorView self, int32_t part, int32_t* index, int32_t dim, int32_t var, double* value, sidl_BaseInterface* _ex)

Get vector coefficients index by index.

NOTE: Users may only get values on processes that own the associated variables.

If the vector is complex, then `value` consists of a pair of doubles representing the real and imaginary parts of the complex value.

1.6.10

int32_t
bHYPRE_SStructVectorView_GetBoxValues (bHYPRE_SStructVectorView self, int32_t part, int32_t* ilower, int32_t* iupper, int32_t dim, int32_t var, double* values, int32_t nvalues, sidl_BaseInterface* _ex)

Get vector coefficients a box at a time.

NOTE: Users may only get values on processes that own the associated variables.

If the vector is complex, then `values` consists of pairs of doubles representing the real and imaginary parts of each complex value.

1.6.11

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructVectorView_SetComplex (bHYPRE_SStructVectorView self, sidl_BaseInterface* _ex)

Set the vector to be complex

1.6.12

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructVectorView_Print (bHYPRE_SStructVectorView self, const char* filename, int32_t all, sidl_BaseInterface* _ex)

Print the vector to file. This is mainly for debugging purposes.

1.6.13

struct bHYPRE_SStructVectorView__object*
bHYPRE_SStructVectorView__cast (void* obj, sidl_BaseInterface* _ex)

Cast method for interface and class type conversions

1.6.14

void*
bHYPRE_SStructVectorView__cast2 (void* obj, const char* type, sidl_BaseInterface* _ex)

String cast method for interface and class type conversions

1.6.15

SIDL_C_INLINE_DECL void
bHYPRE_SStructVectorView__exec (bHYPRE_SStructVectorView self, const char* methodName, sidl_rmi_Call inArgs, sidl_rmi_Return outArgs, sidl_BaseInterface* _ex)

Select and execute a method by name

1.6.16

SIDL_C_INLINE_DECL char*
bHYPRE_SStructVectorView__getURL (bHYPRE_SStructVectorView self, sidl_BaseInterface* _ex)

Get the URL of the Implementation of this object (for RMI)

1.6.17

SIDL_C_INLINE_DECL void
bHYPRE_SStructVectorView__raddRef (bHYPRE_SStructVectorView self, sidl_BaseInterface* _ex)

On a remote object, addrefs the remote instance

1.6.18

SIDL_C_INLINE_DECL sidl_bool
bHYPRE_SStructVectorView__isRemote (bHYPRE_SStructVectorView self, sidl_BaseInterface* _ex)

TRUE if this object is remote, false if local

1.6.19

sidl_bool
bHYPRE_SStructVectorView__isLocal (bHYPRE_SStructVectorView self, sidl_BaseInterface* _ex)

TRUE if this object is remote, false if local

1.6.20

struct bHYPRE_SStructVectorView__object*
bHYPRE_SStructVectorView__rmicast (void* obj, struct sidl_BaseInterface__object** _ex)

Cast method for interface and class type conversions

1.6.21

struct bHYPRE_SStructVectorView__object*
bHYPRE_SStructVectorView__connectI (const char* url, sidl_bool ar, struct sidl_BaseInterface__object** _ex)

RMI connector function for the class. (no addref)

2

Operator Interface

Names

2.1

struct **bHYPRE_Operator__object**

Symbol "bHYPREOperator" (version 100)

An Operator is anything that maps one Vector to another. The terms `Setup` and `Apply` are reserved for Operators. The implementation is allowed to assume that supplied parameter arrays will not be destroyed.

2.2

bHYPRE_Operator
bHYPRE_Operator__connect (const char* , sidl_BaseInterface* _ex)

RMI connector function for the class(addrefs)

2.3

SIDL_C_INLINE_DECL int32_t
bHYPRE_Operator_SetCommunicator (bHYPRE_Operator self, bHYPRE_MPICommunicator mpi_comm, sidl_BaseInterface* _ex)

Set the MPI Communicator. DEPRECATED, use Create:

2.4

SIDL_C_INLINE_DECL void
bHYPRE_Operator_Destroy (bHYPRE_Operator self, sidl_BaseInterface* _ex)

The Destroy function doesn't necessarily destroy anything. It is just another name for deleteRef. Thus it decrements the object's reference count. The Babel memory management system will destroy the object if the reference count goes to zero.

2.5

SIDL_C_INLINE_DECL int32_t
bHYPRE_Operator_SetIntParameter (bHYPRE_Operator self, const char* name, int32_t value, sidl_BaseInterface* _ex)

Set the int parameter associated with `name`

2.6

SIDL_C_INLINE_DECL int32_t
bHYPRE_Operator_SetDoubleParameter (bHYPRE_Operator self, const char* name, double value, sidl_BaseInterface* _ex)

Set the double parameter associated with `name`

2.7

SIDL_C_INLINE_DECL int32_t
bHYPRE_Operator_SetStringParameter (bHYPRE_Operator self, const char* name, const char* value, sidl_BaseInterface* _ex)

Set the string parameter associated with `name`

2.8

SIDL_C_INLINE_DECL int32_t
bHYPRE_Operator_SetIntArray1Parameter (bHYPRE_Operator self, const char* name, int32_t* value, int32_t nvalues, sidl_BaseInterface* _ex)

Set the int 1-D array parameter associated with `name`

2.9

SIDL_C_INLINE_DECL int32_t
bHYPRE_Operator_SetIntArray2Parameter (bHYPRE_Operator self, const char* name, struct sidl_int__array* value, sidl_BaseInterface* _ex)

Set the int 2-D array parameter associated with `name`

2.10

SIDL_C_INLINE_DECL int32_t
bHYPRE_Operator_SetDoubleArray1Parameter (bHYPRE_Operator self, const char* name, double* value, int32_t nvalues, sidl_BaseInterface* _ex)

Set the double 1-D array parameter associated with `name`

2.11

SIDL_C_INLINE_DECL int32_t
bHYPRE_Operator_SetDoubleArray2Parameter (bHYPRE_Operator self, const char* name, struct sidl_double__array* value, sidl_BaseInterface* _ex)

Set the double 2-D array parameter associated with `name`

2.12

SIDL_C_INLINE_DECL int32_t
bHYPRE_Operator_GetIntValue (bHYPRE_Operator self, const char* name, int32_t* value, sidl_BaseInterface* _ex)

Set the int parameter associated with `name`

2.13

SIDL_C_INLINE_DECL int32_t
bHYPRE_Operator_GetDoubleValue (bHYPRE_Operator self, const char* name, double* value, sidl_BaseInterface* _ex)

Get the double parameter associated with `name`

2.14

SIDL_C_INLINE_DECL int32_t
bHYPRE_Operator_Setup (bHYPRE_Operator self, bHYPRE_Vector b, bHYPRE_Vector x, sidl_BaseInterface* _ex)

(Optional) Do any preprocessing that may be necessary in order to execute `Apply`

2.15

SIDL_C_INLINE_DECL int32_t
bHYPRE_Operator_Apply (bHYPRE_Operator self, bHYPRE_Vector b, bHYPRE_Vector* x, sidl_BaseInterface* _ex)

Apply the operator to `b`, returning `x`

2.16

SIDL_C_INLINE_DECL int32_t
bHYPRE_Operator_ApplyAdjoint (bHYPRE_Operator self, bHYPRE_Vector b, bHYPRE_Vector* x, sidl_BaseInterface* _ex)

Apply the adjoint of the operator to `b`, returning `x`

2.17

struct bHYPRE_Operator__object*
bHYPRE_Operator__cast (void* obj, sidl_BaseInterface* _ex)

Cast method for interface and class type conversions

2.18

void*
bHYPRE_Operator__cast2 (void* obj, const char* type, sidl_BaseInterface* _ex)

String cast method for interface and class type conversions

2.19

SIDL_C_INLINE_DECL void
bHYPRE_Operator__exec (bHYPRE_Operator self, const char* methodName, sidl_rmi_Call inArgs, sidl_rmi_Return outArgs, sidl_BaseInterface* _ex)

Select and execute a method by name

2.20

SIDL_C_INLINE_DECL char*
bHYPRE_Operator__getURL (bHYPRE_Operator self, sidl_BaseInterface* _ex)

Get the URL of the Implementation of this object (for RMI)

2.21

SIDL_C_INLINE_DECL void
bHYPRE_Operator__raddRef (bHYPRE_Operator self, sidl_BaseInterface* _ex)

On a remote object, addrefs the remote instance

2.22

SIDL_C_INLINE_DECL sidl_bool
bHYPRE_Operator__isRemote (bHYPRE_Operator self, sidl_BaseInterface* _ex)

TRUE if this object is remote, false if local

2.23

sidl_bool
bHYPRE_Operator__isLocal (bHYPRE_Operator self, sidl_BaseInterface* _ex)

TRUE if this object is remote, false if local

2.24

struct bHYPRE_Operator__object*
bHYPRE_Operator__rmicast (void* obj, struct sidl_BaseInterface__object** _ex)

Cast method for interface and class type conversions

2.25

struct bHYPRE_Operator__object*
bHYPRE_Operator__connectI (const char* url, sidl_bool ar, struct sidl_BaseInterface__object** _ex)

RMI connector function for the class. (no addref)

3

Vector Interface

Names

3.1

struct **bHYPRE_Vector__object**

Symbol "bHYPREVector" (version 100)

3.2

bHYPRE_Vector
bHYPRE_Vector__connect (const char* , sidl_BaseInterface* _ex)

RMI connector function for the class(addrefs)

3.3

SIDL_C_INLINE_DECL int32_t
bHYPRE_Vector_Clear (bHYPRE_Vector self, sidl_BaseInterface* _ex)

Set `self` to 0

3.4

SIDL_C_INLINE_DECL int32_t
bHYPRE_Vector_Copy (bHYPRE_Vector self, bHYPRE_Vector x, sidl_BaseInterface* _ex)

Copy data from x into `self`

3.5

SIDL_C_INLINE_DECL int32_t
bHYPRE_Vector_Clone (bHYPRE_Vector self, bHYPRE_Vector* x, sidl_BaseInterface* _ex)

Create an `x` compatible with `self`. The new vector's data is not specified.

NOTE: When this method is used in an inherited class, the cloned `Vector` object can be cast to an object with the inherited class type.

3.6

SIDL_C_INLINE_DECL int32_t
bHYPRE_Vector_Scale (bHYPRE_Vector self, double a, sidl_BaseInterface* _ex)

Scale `self` by `a`

3.7

SIDL_C_INLINE_DECL int32_t
bHYPRE_Vector_Dot (bHYPRE_Vector self, bHYPRE_Vector x, double* d, sidl_BaseInterface* _ex)

Compute `d`, the inner-product of `self` and `x`

3.8

SIDL_C_INLINE_DECL int32_t
bHYPRE_Vector_Axpy (bHYPRE_Vector self, double a, bHYPRE_Vector x, sidl_BaseInterface* _ex)

Add `ax` to `self`

3.9

struct bHYPRE_Vector__object*
bHYPRE_Vector__cast (void* obj, sidl_BaseInterface* _ex)

Cast method for interface and class type conversions

3.10

void*
bHYPRE_Vector__cast2 (void* obj, const char* type, sidl_BaseInterface* _ex)

String cast method for interface and class type conversions

3.11

SIDL_C_INLINE_DECL void
bHYPRE_Vector__exec (bHYPRE_Vector self, const char* methodName,
sidl_rmi_Call inArgs, sidl_rmi_Return outArgs, sidl_BaseInterface* _ex)

Select and execute a method by name

3.12

SIDL_C_INLINE_DECL char*
bHYPRE_Vector__getURL (bHYPRE_Vector self, sidl_BaseInterface* _ex)

Get the URL of the Implementation of this object (for RMI)

3.13

SIDL_C_INLINE_DECL void
bHYPRE_Vector__raddRef (bHYPRE_Vector self, sidl_BaseInterface* _ex)

On a remote object, addrefs the remote instance

3.14

SIDL_C_INLINE_DECL sidl_bool
bHYPRE_Vector__isRemote (bHYPRE_Vector self, sidl_BaseInterface* _ex)

TRUE if this object is remote, false if local

3.15

sidl_bool
bHYPRE_Vector__isLocal (bHYPRE_Vector self, sidl_BaseInterface* _ex)

TRUE if this object is remote, false if local

3.16

struct bHYPRE_Vector__object*
bHYPRE_Vector__rmicast (void* obj, struct sidl_BaseInterface__object** _ex)

Cast method for interface and class type conversions

3.17

struct bHYPRE_Vector__object*
bHYPRE_Vector__connectI (const char* url, sidl_bool ar, struct sidl_BaseInterface__object** _ex)

RMI connector function for the class. (no addref)

4

Matrices and Vectors

Names

4.1

IJParCSR Matrix

Names

4.1.1

struct **bHYPRE_IJParCSRMatrix__object**

Symbol "bHYPREIJParCSRMatrix" (version 100)

The IJParCSR matrix class.

Objects of this type can be cast to IJMatrixView, Operator, or CoefficientAccess objects using the `__cast` methods.

4.1.2

struct bHYPRE_IJParCSRMatrix__object*
bHYPRE_IJParCSRMatrix__create (sidl_BaseInterface* _ex)

Constructor function for the class

4.1.3

bHYPRE_IJParCSRMatrix
bHYPRE_IJParCSRMatrix__createRemote (const char* url,
sidl_BaseInterface* _ex)

RMI constructor function for the class

4.1.4

bHYPRE_IJParCSRMatrix
bHYPRE_IJParCSRMatrix__wrapObj (void* data, sidl_BaseInterface* _ex)

Wraps up the private data struct pointer (struct bHYPRE_IJParCSRMatrix__data) passed in rather than running the constructor

4.1.5

bHYPRE_IJParCSRMatrix
bHYPRE_IJParCSRMatrix__connect (const char* , sidl_BaseInterface* _ex)

RMI connector function for the class(addrefs)

4.1.6

bHYPRE_IJParCSRMatrix
bHYPRE_IJParCSRMatrix_Create (bHYPRE_MPICommunicator
mpi_comm, int32_t ilower, int32_t iupper, int32_t jlower, int32_t jupper,
sidl_BaseInterface* _ex)

This function is the preferred way to create an IJParCSR Matrix.

4.1.7

bHYPRE_IJParCSRMatrix
bHYPRE_IJParCSRMatrix_GenerateLaplacian (
bHYPRE_MPICommunicator mpi_comm, int32_t nx, int32_t ny, int32_t nz,
int32_t Px, int32_t Py, int32_t Pz, int32_t p, int32_t q, int32_t r, double* values,
int32_t nvalues, int32_t discretization, sidl_BaseInterface* _ex)

Method: GenerateLaplacian[]

4.1.8

int32_t
bHYPRE_IJParCSRMatrix_SetDiagOffdSizes (bHYPRE_IJParCSRMatrix
self, int32_t* diag_sizes, int32_t* offdiag_sizes, int32_t local_nrows,
sidl_BaseInterface* _ex)

(Optional) Set the max number of nonzeros to expect in each row of the diagonal and off-diagonal blocks. The diagonal block is the submatrix whose column numbers correspond to rows owned by this process, and the off-diagonal block is everything else. The arrays `diag_sizes` and `offdiag_sizes` contain estimated sizes for each row of the diagonal and off-diagonal blocks, respectively. This routine can significantly improve the efficiency of matrix construction, and should always be utilized if possible.

Not collective.

4.1.9

SIDL_C_INLINE_DECL int32_t
bHYPRE_IJParCSRMatrix_SetLocalRange (bHYPRE_IJParCSRMatrix
self, int32_t ilower, int32_t iupper, int32_t jlower, int32_t jupper,
sidl_BaseInterface* _ex)

Set the local range for a matrix object. Each process owns some unique consecutive range of rows, indicated by the global row indices `ilower` and `iupper`. The row data is required to be such that the value of `ilower` on any process p be exactly one more than the value of `iupper` on process $p-1$. Note that the first row of the global matrix may start with any integer value. In particular, one may use zero- or one-based indexing.

For square matrices, `jlower` and `jupper` typically should match `ilower` and `iupper`, respectively. For rectangular matrices, `jlower` and `jupper` should define a partitioning of the columns. This partitioning must be used for any vector v that will be used in matrix-vector products with the rectangular matrix. The

matrix data structure may use `jlower` and `jupper` to store the diagonal blocks (rectangular in general) of the matrix separately from the rest of the matrix.

Collective.

4.1.10

int32_t
bHYPRE_IJParCSRMatrix_SetValues (bHYPRE_IJParCSRMatrix self, int32_t nrows, int32_t* ncols, int32_t* rows, int32_t* cols, double* values, int32_t nnonzeros, sidl_BaseInterface* _ex)

Sets values for `nrows` of the matrix. The arrays `ncols` and `rows` are of dimension `nrows` and contain the number of columns in each row and the row indices, respectively. The array `cols` contains the column indices for each of the `rows`, and is ordered by rows. The data in the `values` array corresponds directly to the column entries in `cols`. The last argument is the size of the cols and values arrays, i.e. the total number of nonzeros being provided, i.e. the sum of all values in ncols. This functin erases any previous values at the specified locations and replaces them with new ones, or, if there was no value there before, inserts a new one.

Not collective.

4.1.11

int32_t
bHYPRE_IJParCSRMatrix_AddToValues (bHYPRE_IJParCSRMatrix self, int32_t nrows, int32_t* ncols, int32_t* rows, int32_t* cols, double* values, int32_t nnonzeros, sidl_BaseInterface* _ex)

Adds to values for `nrows` of the matrix. Usage details are analogous to `SetValues`. Adds to any previous values at the specified locations, or, if there was no value there before, inserts a new one.

Not collective.

4.1.12

SIDL_C_INLINE_DECL int32_t
bHYPRE_IJParCSRMatrix_GetLocalRange (bHYPRE_IJParCSRMatrix self, int32_t* ilower, int32_t* iupper, int32_t* jlower, int32_t* jupper, sidl_BaseInterface* _ex)

Gets range of rows owned by this processor and range of column partitioning for this processor

4.1.13

int32_t
bHYPRE_IJParCSRMatrix_GetRowCounts (bHYPRE_IJParCSRMatrix self, int32_t nrows, int32_t* rows, int32_t* ncols, sidl_BaseInterface* _ex)

Gets number of nonzeros elements for `nrows` rows specified in `rows` and returns them in `ncols`, which needs to be allocated by the user

4.1.14

int32_t
bHYPRE_IJParCSRMatrix_GetValues (bHYPRE_IJParCSRMatrix self, int32_t nrows, int32_t* ncols, int32_t* rows, int32_t* cols, double* values, int32_t nnonzeros, sidl_BaseInterface* _ex)

Gets values for `nrows` rows or partial rows of the matrix. Usage details are analogous to `SetValues`.

4.1.15

SIDL_C_INLINE_DECL int32_t
bHYPRE_IJParCSRMatrix_SetRowSizes (bHYPRE_IJParCSRMatrix self, int32_t* sizes, int32_t nrows, sidl_BaseInterface* _ex)

(Optional) Set the max number of nonzeros to expect in each row. The array `sizes` contains estimated sizes for each row on this process. The integer nrows is the number of rows in the local matrix. This call can significantly improve the efficiency of matrix construction, and should always be utilized if possible.

Not collective.

4.1.16

SIDL_C_INLINE_DECL int32_t
bHYPRE_IJParCSRMatrix_Print (bHYPRE_IJParCSRMatrix self, const char* filename, sidl_BaseInterface* _ex)

Print the matrix to file. This is mainly for debugging purposes.

4.1.17

SIDL_C_INLINE_DECL int32_t
bHYPRE_IJParCSRMatrix_Read (bHYPRE_IJParCSRMatrix self, const char* filename, bHYPRE_MPICommunicator comm, sidl_BaseInterface* _ex)

Read the matrix from file. This is mainly for debugging purposes.

4.1.18

SIDL_C_INLINE_DECL int32_t
bHYPRE_IJParCSRMatrix_SetCommunicator (bHYPRE_IJParCSRMatrix self, bHYPRE_MPICommunicator mpi_comm, sidl_BaseInterface* _ex)

Set the MPI Communicator. DEPRECATED, Use Create()

4.1.19

SIDL_C_INLINE_DECL void
bHYPRE_IJParCSRMatrix_Destroy (bHYPRE_IJParCSRMatrix self, sidl_BaseInterface* _ex)

The Destroy function doesn't necessarily destroy anything. It is just another name for deleteRef. Thus it decrements the object's reference count. The Babel memory management system will destroy the object if the reference count goes to zero.

4.1.20

SIDL_C_INLINE_DECL int32_t
bHYPRE_IJParCSRMatrix_Initialize (bHYPRE_IJParCSRMatrix self,
sidl_BaseInterface* _ex)

Prepare an object for setting coefficient values, whether for the first time or subsequently

4.1.21

SIDL_C_INLINE_DECL int32_t
bHYPRE_IJParCSRMatrix_Assemble (bHYPRE_IJParCSRMatrix self,
sidl_BaseInterface* _ex)

Finalize the construction of an object before using, either for the first time or on subsequent uses. `Initialize` and `Assemble` always appear in a matched set, with Initialize preceding Assemble. Values can only be set in between a call to Initialize and Assemble.

4.1.22

SIDL_C_INLINE_DECL int32_t
bHYPRE_IJParCSRMatrix_SetIntParameter (bHYPRE_IJParCSRMatrix
self, const char* name, int32_t value, sidl_BaseInterface* _ex)

Set the int parameter associated with `name`

4.1.23

SIDL_C_INLINE_DECL int32_t
bHYPRE_IJParCSRMatrix_SetDoubleParameter (
bHYPRE_IJParCSRMatrix self, const char* name, double value,
sidl_BaseInterface* _ex)

Set the double parameter associated with `name`

4.1.24

SIDL_C_INLINE_DECL int32_t
bHYPRE_IJParCSRMatrix_SetStringParameter (
bHYPRE_IJParCSRMatrix self, const char* name, const char* value,
sidl_BaseInterface* _ex)

Set the string parameter associated with `name`

4.1.25

SIDL_C_INLINE_DECL int32_t
bHYPRE_IJParCSRMatrix_SetIntArray1Parameter (
bHYPRE_IJParCSRMatrix self, const char* name, int32_t* value, int32_t nvalues,
sidl_BaseInterface* _ex)

Set the int 1-D array parameter associated with `name`

4.1.26

SIDL_C_INLINE_DECL int32_t
bHYPRE_IJParCSRMatrix_SetIntArray2Parameter (
bHYPRE_IJParCSRMatrix self, const char* name, struct sidl_int__array* value,
sidl_BaseInterface* _ex)

Set the int 2-D array parameter associated with `name`

4.1.27

SIDL_C_INLINE_DECL int32_t
bHYPRE_IJParCSRMatrix_SetDoubleArray1Parameter (
bHYPRE_IJParCSRMatrix self, const char* name, double* value, int32_t nvalues,
sidl_BaseInterface* _ex)

Set the double 1-D array parameter associated with `name`

4.1.28

SIDL_C_INLINE_DECL int32_t
bHYPRE_IJParCSRMatrix_SetDoubleArray2Parameter (
bHYPRE_IJParCSRMatrix self, const char* name, struct sidl_double__array*
value, sidl_BaseInterface* _ex)

Set the double 2-D array parameter associated with `name`

4.1.29

SIDL_C_INLINE_DECL int32_t
bHYPRE_IJParCSRMatrix_GetIntValue (bHYPRE_IJParCSRMatrix self,
const char* name, int32_t* value, sidl_BaseInterface* _ex)

Set the int parameter associated with `name`

4.1.30

SIDL_C_INLINE_DECL int32_t
bHYPRE_IJParCSRMatrix_GetDoubleValue (bHYPRE_IJParCSRMatrix
self, const char* name, double* value, sidl_BaseInterface* _ex)

Get the double parameter associated with `name`

4.1.31

SIDL_C_INLINE_DECL int32_t
bHYPRE_IJParCSRMatrix_Setup (bHYPRE_IJParCSRMatrix self,
bHYPRE_Vector b, bHYPRE_Vector x, sidl_BaseInterface* _ex)

(Optional) Do any preprocessing that may be necessary in order to execute `Apply`

4.1.32

SIDL_C_INLINE_DECL int32_t
bHYPRE_IJParCSRMatrix_Apply (bHYPRE_IJParCSRMatrix self, bHYPRE_Vector b, bHYPRE_Vector* x, sidl_BaseInterface* _ex)

Apply the operator to `b`, returning `x`

4.1.33

SIDL_C_INLINE_DECL int32_t
bHYPRE_IJParCSRMatrix_ApplyAdjoint (bHYPRE_IJParCSRMatrix self, bHYPRE_Vector b, bHYPRE_Vector* x, sidl_BaseInterface* _ex)

Apply the adjoint of the operator to `b`, returning `x`

4.1.34

SIDL_C_INLINE_DECL int32_t
bHYPRE_IJParCSRMatrix_GetRow (bHYPRE_IJParCSRMatrix self, int32_t row, int32_t* size, struct sidl_int__array** col_ind, struct sidl_double__array** values, sidl_BaseInterface* _ex)

The GetRow method will allocate space for its two output arrays on the first call. The space will be reused on subsequent calls. Thus the user must not delete them, yet must not depend on the data from GetRow to persist beyond the next GetRow call.

4.1.35

struct bHYPRE_IJParCSRMatrix__object*
bHYPRE_IJParCSRMatrix__cast (void* obj, sidl_BaseInterface* _ex)

Cast method for interface and class type conversions

4.1.36

void*
bHYPRE_IJParCSRMatrix__cast2 (void* obj, const char* type,
sidl_BaseInterface* _ex)

String cast method for interface and class type conversions

4.1.37

SIDL_C_INLINE_DECL void
bHYPRE_IJParCSRMatrix__exec (bHYPRE_IJParCSRMatrix self, const
char* methodName, sidl_rmi_Call inArgs, sidl_rmi_Return outArgs,
sidl_BaseInterface* _ex)

Select and execute a method by name

4.1.38

SIDL_C_INLINE_DECL char*
bHYPRE_IJParCSRMatrix__getURL (bHYPRE_IJParCSRMatrix self,
sidl_BaseInterface* _ex)

Get the URL of the Implementation of this object (for RMI)

4.1.39

SIDL_C_INLINE_DECL void
bHYPRE_IJParCSRMatrix__raddRef (bHYPRE_IJParCSRMatrix self,
sidl_BaseInterface* _ex)

On a remote object, addrefs the remote instance

4.1.40

SIDL_C_INLINE_DECL sidl_bool
bHYPRE_IJParCSRMatrix__isRemote (bHYPRE_IJParCSRMatrix self,
sidl_BaseInterface* _ex)

TRUE if this object is remote, false if local

4.1.41

sidl_bool
bHYPRE_IJParCSRMatrix__isLocal (bHYPRE_IJParCSRMatrix self,
sidl_BaseInterface* _ex)

TRUE if this object is remote, false if local

4.1.42

struct bHYPRE_IJParCSRMatrix__object*
bHYPRE_IJParCSRMatrix__rmicast (void* obj, struct
sidl_BaseInterface__object** _ex)

Cast method for interface and class type conversions

4.1.43

struct bHYPRE_IJParCSRMatrix__object*
bHYPRE_IJParCSRMatrix__connectI (const char* url, sidl_bool ar, struct
sidl_BaseInterface__object** _ex)

RMI connector function for the class. (no addref)

4.2

IJParCSR Vector

Names

4.2.1

struct **bHYPRE_IJParCSRVector__object**

Symbol "bHYPREIJParCSRVector" (version 100)

The IJParCSR vector class.

Objects of this type can be cast to IJVectorView or Vector objects using the `__cast` methods.

4.2.2

struct bHYPRE_IJParCSRVector__object*
bHYPRE_IJParCSRVector__create (sidl_BaseInterface* _ex)

Constructor function for the class

4.2.3

bHYPRE_IJParCSRVector
bHYPRE_IJParCSRVector__createRemote (const char* url,
sidl_BaseInterface* _ex)

RMI constructor function for the class

4.2.4

bHYPRE_IJParCSRVector
bHYPRE_IJParCSRVector__wrapObj (void* data, sidl_BaseInterface* _ex)

Wraps up the private data struct pointer (struct bHYPRE_IJParCSRVector__data) passed in rather than running the constructor

4.2.5

bHYPRE_IJParCSRVector
bHYPRE_IJParCSRVector__connect (const char* , sidl_BaseInterface* _ex)

RMI connector function for the class(addrefs)

4.2.6

bHYPRE_IJParCSRVector
bHYPRE_IJParCSRVector_Create (bHYPRE_MPICommunicator mpi_comm,
int32_t jlower, int32_t jupper, sidl_BaseInterface* _ex)

This function is the preferred way to create an IJParCSR Vector.

4.2.7

SIDL_C_INLINE_DECL int32_t
bHYPRE_IJParCSRVector_SetLocalRange (bHYPRE_IJParCSRVector self, int32_t jlower, int32_t jupper, sidl_BaseInterface* _ex)

Set the local range for a vector object. Each process owns some unique consecutive range of vector unknowns, indicated by the global indices `jlower` and `jupper`. The data is required to be such that the value of `jlower` on any process p be exactly one more than the value of `jupper` on process $p-1$. Note that the first index of the global vector may start with any integer value. In particular, one may use zero- or one-based indexing.

Collective.

4.2.8

int32_t
bHYPRE_IJParCSRVector_SetValues (bHYPRE_IJParCSRVector self, int32_t nvalues, int32_t* indices, double* values, sidl_BaseInterface* _ex)

Sets values in vector. The arrays `values` and `indices` are of dimension `nvalues` and contain the vector values to be set and the corresponding global vector indices, respectively. Erases any previous values at the specified locations and replaces them with new ones.

Not collective.

4.2.9

int32_t
bHYPRE_IJParCSRVector_AddToValues (bHYPRE_IJParCSRVector self, int32_t nvalues, int32_t* indices, double* values, sidl_BaseInterface* _ex)

Adds to values in vector. Usage details are analogous to `SetValues`.

Not collective.

4.2.10

SIDL_C_INLINE_DECL int32_t
bHYPRE_IJParCSRVector_GetLocalRange (bHYPRE_IJParCSRVector self, int32_t* jlower, int32_t* jupper, sidl_BaseInterface* _ex)

Returns range of the part of the vector owned by this processor

4.2.11

int32_t
bHYPRE_IJParCSRVector_GetValues (bHYPRE_IJParCSRVector self, int32_t nvalues, int32_t* indices, double* values, sidl_BaseInterface* _ex)

Gets values in vector. Usage details are analogous to `SetValues`.

Not collective.

4.2.12

SIDL_C_INLINE_DECL int32_t
bHYPRE_IJParCSRVector_Print (bHYPRE_IJParCSRVector self, const char* filename, sidl_BaseInterface* _ex)

Print the vector to file. This is mainly for debugging purposes.

4.2.13

SIDL_C_INLINE_DECL int32_t
bHYPRE_IJParCSRVector_Read (bHYPRE_IJParCSRVector self, const char* filename, bHYPRE_MPICommunicator comm, sidl_BaseInterface* _ex)

Read the vector from file. This is mainly for debugging purposes.

4.2.14

SIDL_C_INLINE_DECL int32_t
bHYPRE_IJParCSRVector_SetCommunicator (bHYPRE_IJParCSRVector self, bHYPRE_MPICommunicator mpi_comm, sidl_BaseInterface* _ex)

Set the MPI Communicator. DEPRECATED, Use Create()

4.2.15

SIDL_C_INLINE_DECL void
bHYPRE_IJParCSRVector_Destroy (bHYPRE_IJParCSRVector self, sidl_BaseInterface* _ex)

The Destroy function doesn't necessarily destroy anything. It is just another name for deleteRef. Thus it decrements the object's reference count. The Babel memory management system will destroy the object if the reference count goes to zero.

4.2.16

SIDL_C_INLINE_DECL int32_t
bHYPRE_IJParCSRVector_Initialize (bHYPRE_IJParCSRVector self, sidl_BaseInterface* _ex)

Prepare an object for setting coefficient values, whether for the first time or subsequently

4.2.17

SIDL_C_INLINE_DECL int32_t
bHYPRE_IJParCSRVector_Assemble (bHYPRE_IJParCSRVector self, sidl_BaseInterface* _ex)

Finalize the construction of an object before using, either for the first time or on subsequent uses. `Initialize` and `Assemble` always appear in a matched set, with Initialize preceding Assemble. Values can only be set in between a call to Initialize and Assemble.

4.2.18

SIDL_C_INLINE_DECL int32_t
bHYPRE_IJParCSRVector_Clear (bHYPRE_IJParCSRVector self,
sidl_BaseInterface* _ex)

Set `self` to 0

4.2.19

SIDL_C_INLINE_DECL int32_t
bHYPRE_IJParCSRVector_Copy (bHYPRE_IJParCSRVector self,
bHYPRE_Vector x, sidl_BaseInterface* _ex)

Copy data from x into `self`

4.2.20

SIDL_C_INLINE_DECL int32_t
bHYPRE_IJParCSRVector_Clone (bHYPRE_IJParCSRVector self,
bHYPRE_Vector* x, sidl_BaseInterface* _ex)

Create an `x` compatible with `self`. The new vector's data is not specified.

NOTE: When this method is used in an inherited class, the cloned `Vector` object can be cast to an object with the inherited class type.

4.2.21

SIDL_C_INLINE_DECL int32_t
bHYPRE_IJParCSRVector_Scale (bHYPRE_IJParCSRVector self, double a,
sidl_BaseInterface* _ex)

Scale `self` by `a`

4.2.22

SIDL_C_INLINE_DECL int32_t
bHYPRE_IJParCSRVector_Dot (bHYPRE_IJParCSRVector self,
bHYPRE_Vector x, double* d, sidl_BaseInterface* _ex)

Compute `d`, the inner-product of `self` and `x`

4.2.23

SIDL_C_INLINE_DECL int32_t
bHYPRE_IJParCSRVector_Axpy (bHYPRE_IJParCSRVector self, double a,
bHYPRE_Vector x, sidl_BaseInterface* _ex)

Add `ax` to `self`

4.2.24

struct bHYPRE_IJParCSRVector__object*
bHYPRE_IJParCSRVector__cast (void* obj, sidl_BaseInterface* _ex)

Cast method for interface and class type conversions

4.2.25

void*
bHYPRE_IJParCSRVector__cast2 (void* obj, const char* type,
sidl_BaseInterface* _ex)

String cast method for interface and class type conversions

4.2.26

SIDL_C_INLINE_DECL void
bHYPRE_IJParCSRVector__exec (bHYPRE_IJParCSRVector self, const char* methodName, sidl_rmi_Call inArgs, sidl_rmi_Return outArgs, sidl_BaseInterface* _ex)

Select and execute a method by name

4.2.27

SIDL_C_INLINE_DECL char*
bHYPRE_IJParCSRVector__getURL (bHYPRE_IJParCSRVector self, sidl_BaseInterface* _ex)

Get the URL of the Implementation of this object (for RMI)

4.2.28

SIDL_C_INLINE_DECL void
bHYPRE_IJParCSRVector__raddRef (bHYPRE_IJParCSRVector self, sidl_BaseInterface* _ex)

On a remote object, addrefs the remote instance

4.2.29

SIDL_C_INLINE_DECL sidl_bool
bHYPRE_IJParCSRVector__isRemote (bHYPRE_IJParCSRVector self, sidl_BaseInterface* _ex)

TRUE if this object is remote, false if local

4.2.30

sidl_bool
bHYPRE_IJParCSRVector__isLocal (bHYPRE_IJParCSRVector self,
sidl_BaseInterface* _ex)

TRUE if this object is remote, false if local

4.2.31

struct bHYPRE_IJParCSRVector__object*
bHYPRE_IJParCSRVector__rmicast (void* obj, struct
sidl_BaseInterface__object** _ex)

Cast method for interface and class type conversions

4.2.32

struct bHYPRE_IJParCSRVector__object*
bHYPRE_IJParCSRVector__connectI (const char* url, sidl_bool ar, struct
sidl_BaseInterface__object** _ex)

RMI connector function for the class. (no addref)

4.3

Struct Matrix

Names

4.3.1

struct **bHYPRE_StructMatrix__object**

Symbol "bHYPREStructMatrix" (version 100)

A single class that implements both a view interface and an operator interface. A StructMatrix is a matrix on a structured grid. One function unique to a StructMatrix is SetConstantEntries. This declares that matrix entries corresponding to certain stencil points (supplied as stencil element indices) will be constant throughout the grid.

4.3.2

struct bHYPRE_StructMatrix__object*
bHYPRE_StructMatrix__create (sidl_BaseInterface* _ex)

Constructor function for the class

4.3.3

bHYPRE_StructMatrix
bHYPRE_StructMatrix__createRemote (const char* url, sidl_BaseInterface* _ex)

RMI constructor function for the class

4.3.4

bHYPRE_StructMatrix
bHYPRE_StructMatrix__wrapObj (void* data, sidl_BaseInterface* _ex)

Wraps up the private data struct pointer (struct bHYPRE_StructMatrix__data) passed in rather than running the constructor

4.3.5

bHYPRE_StructMatrix
bHYPRE_StructMatrix__connect (const char* , sidl_BaseInterface* _ex)

RMI connector function for the class(addrefs)

4.3.6

bHYPRE_StructMatrix
bHYPRE_StructMatrix_Create (bHYPRE_MPICommunicator mpi_comm, bHYPRE_StructGrid grid, bHYPRE_StructStencil stencil, sidl_BaseInterface* _ex)

This function is the preferred way to create a Struct Matrix.

4.3.7

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructMatrix_SetGrid (bHYPRE_StructMatrix self, bHYPRE_StructGrid grid, sidl_BaseInterface* _ex)

Set the grid on which vectors are defined. This and the stencil determine the matrix structure.

4.3.8

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructMatrix_SetStencil (bHYPRE_StructMatrix self, bHYPRE_StructStencil stencil, sidl_BaseInterface* _ex)

Set the stencil. This and the grid determine the matrix structure.

4.3.9

int32_t
bHYPRE_StructMatrix_SetValues (bHYPRE_StructMatrix self, int32_t* index, int32_t dim, int32_t num_stencil_indices, int32_t* stencil_indices, double* values, sidl_BaseInterface* _ex)

Set matrix values at grid point, given by "index". You can supply values for one or more positions in the stencil. "index" is an array of size "dim"; and "stencil_indices" and "values" are arrays of size "num_stencil_indices".

4.3.10

int32_t
bHYPRE_StructMatrix_SetBoxValues (bHYPRE_StructMatrix self, int32_t* ilower, int32_t* iupper, int32_t dim, int32_t num_stencil_indices, int32_t* stencil_indices, double* values, int32_t nvalues, sidl_BaseInterface* _ex)

Set matrix values throughout a box in the grid, specified by its lower and upper corners. You can supply these values for one or more positions in the stencil. Thus the total number of matrix values you supply, "nvalues", is num_stencil_indices x box_size, where box_size is the number of grid points in the box. The values array should be organized so all values for a given box point are together (i.e., the stencil index is the most rapidly varying). "ilower" and "iupper" are arrays of size "dim", "stencil_indices" is an array of size "num_stencil_indices", and "values" is an array of size "nvalues".

4.3.11

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructMatrix_SetNumGhost (bHYPRE_StructMatrix self, int32_t* num_ghost, int32_t dim2, sidl_BaseInterface* _ex)

Set the number of ghost zones, separately on the lower and upper sides for each dimension. "num_ghost" is an array of size "dim2", twice the number of dimensions

4.3.12

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructMatrix_SetSymmetric (bHYPRE_StructMatrix self, int32_t symmetric, sidl_BaseInterface* _ex)

Call SetSymmetric with symmetric=1 to turn on symmetric matrix storage if available.

4.3.13

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructMatrix_SetConstantEntries (bHYPRE_StructMatrix self, int32_t num_stencil_constant_points, int32_t* stencil_constant_points, sidl_BaseInterface* _ex)

State which stencil entries are constant over the grid. Supported options are: (i) none (the default), (ii) all (stencil_constant_points should include all stencil points) (iii) all entries but the diagonal.

4.3.14

int32_t
bHYPRE_StructMatrix_SetConstantValues (bHYPRE_StructMatrix self, int32_t num_stencil_indices, int32_t* stencil_indices, double* values, sidl_BaseInterface* _ex)

Provide values for matrix coefficients which are constant throughout the grid, one value for each stencil point. "stencil_indices" and "values" is each an array of length "num_stencil_indices"

4.3.15

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructMatrix_SetCommunicator (bHYPRE_StructMatrix self, bHYPRE_MPICommunicator mpi_comm, sidl_BaseInterface* _ex)

Set the MPI Communicator. DEPRECATED, Use Create()

4.3.16

SIDL_C_INLINE_DECL void
bHYPRE_StructMatrix_Destroy (bHYPRE_StructMatrix self, sidl_BaseInterface* _ex)

The Destroy function doesn't necessarily destroy anything. It is just another name for deleteRef. Thus it decrements the object's reference count. The Babel memory management system will destroy the object if the reference count goes to zero.

4.3.17

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructMatrix_Initialize (bHYPRE_StructMatrix self,
sidl_BaseInterface* _ex)

Prepare an object for setting coefficient values, whether for the first time or subsequently

4.3.18

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructMatrix_Assemble (bHYPRE_StructMatrix self,
sidl_BaseInterface* _ex)

Finalize the construction of an object before using, either for the first time or on subsequent uses. `Initialize` and `Assemble` always appear in a matched set, with Initialize preceding Assemble. Values can only be set in between a call to Initialize and Assemble.

4.3.19

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructMatrix_SetIntParameter (bHYPRE_StructMatrix self,
const char* name, int32_t value, sidl_BaseInterface* _ex)

Set the int parameter associated with `name`

4.3.20

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructMatrix_SetDoubleParameter (bHYPRE_StructMatrix self,
const char* name, double value, sidl_BaseInterface* _ex)

Set the double parameter associated with `name`

4.3.21

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructMatrix_SetStringParameter (bHYPRE_StructMatrix self, const char* name, const char* value, sidl_BaseInterface* _ex)

Set the string parameter associated with `name`

4.3.22

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructMatrix_SetIntArray1Parameter (bHYPRE_StructMatrix self, const char* name, int32_t* value, int32_t nvalues, sidl_BaseInterface* _ex)

Set the int 1-D array parameter associated with `name`

4.3.23

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructMatrix_SetIntArray2Parameter (bHYPRE_StructMatrix self, const char* name, struct sidl_int__array* value, sidl_BaseInterface* _ex)

Set the int 2-D array parameter associated with `name`

4.3.24

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructMatrix_SetDoubleArray1Parameter (bHYPRE_StructMatrix self, const char* name, double* value, int32_t nvalues, sidl_BaseInterface* _ex)

Set the double 1-D array parameter associated with `name`

4.3.25

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructMatrix_SetDoubleArray2Parameter (
bHYPRE_StructMatrix self, const char* name, struct sidl_double__array* value,
sidl_BaseInterface* _ex)

Set the double 2-D array parameter associated with `name`

4.3.26

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructMatrix_GetIntValue (bHYPRE_StructMatrix self, const
char* name, int32_t* value, sidl_BaseInterface* _ex)

Set the int parameter associated with `name`

4.3.27

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructMatrix_GetDoubleValue (bHYPRE_StructMatrix self,
const char* name, double* value, sidl_BaseInterface* _ex)

Get the double parameter associated with `name`

4.3.28

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructMatrix_Setup (bHYPRE_StructMatrix self, bHYPRE_Vector
b, bHYPRE_Vector x, sidl_BaseInterface* _ex)

(Optional) Do any preprocessing that may be necessary in order to execute `Apply`

4.3.29

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructMatrix_Apply (bHYPRE_StructMatrix self, bHYPRE_Vector b, bHYPRE_Vector* x, sidl_BaseInterface* _ex)

Apply the operator to `b`, returning `x`

4.3.30

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructMatrix_ApplyAdjoint (bHYPRE_StructMatrix self, bHYPRE_Vector b, bHYPRE_Vector* x, sidl_BaseInterface* _ex)

Apply the adjoint of the operator to `b`, returning `x`

4.3.31

struct bHYPRE_StructMatrix__object*
bHYPRE_StructMatrix__cast (void* obj, sidl_BaseInterface* _ex)

Cast method for interface and class type conversions

4.3.32

void*
bHYPRE_StructMatrix__cast2 (void* obj, const char* type, sidl_BaseInterface* _ex)

String cast method for interface and class type conversions

4.3.33

SIDL_C_INLINE_DECL void
bHYPRE_StructMatrix__exec (bHYPRE_StructMatrix self, const char* methodName, sidl_rmi_Call inArgs, sidl_rmi_Return outArgs, sidl_BaseInterface* _ex)

Select and execute a method by name

4.3.34

SIDL_C_INLINE_DECL char*
bHYPRE_StructMatrix__getURL (bHYPRE_StructMatrix self, sidl_BaseInterface* _ex)

Get the URL of the Implementation of this object (for RMI)

4.3.35

SIDL_C_INLINE_DECL void
bHYPRE_StructMatrix__raddRef (bHYPRE_StructMatrix self, sidl_BaseInterface* _ex)

On a remote object, addrefs the remote instance

4.3.36

SIDL_C_INLINE_DECL sidl_bool
bHYPRE_StructMatrix__isRemote (bHYPRE_StructMatrix self, sidl_BaseInterface* _ex)

TRUE if this object is remote, false if local

4.3.37

sidl_bool
bHYPRE_StructMatrix__isLocal (bHYPRE_StructMatrix self,
sidl_BaseInterface* _ex)

TRUE if this object is remote, false if local

4.3.38

struct bHYPRE_StructMatrix__object*
bHYPRE_StructMatrix__rmicast (void* obj, struct
sidl_BaseInterface__object** _ex)

Cast method for interface and class type conversions

4.3.39

struct bHYPRE_StructMatrix__object*
bHYPRE_StructMatrix__connectI (const char* url, sidl_bool ar, struct
sidl_BaseInterface__object** _ex)

RMI connector function for the class. (no addref)

4.4

Struct Vector

Names

4.4.1

struct **bHYPRE_StructVector__object**

Symbol "bHYPREStructVector" (version 100)

4.4.2

struct bHYPRE_StructVector__object*
bHYPRE_StructVector__create (sidl_BaseInterface* _ex)

Constructor function for the class

4.4.3

bHYPRE_StructVector
bHYPRE_StructVector__createRemote (const char* url, sidl_BaseInterface* _ex)

RMI constructor function for the class

4.4.4

bHYPRE_StructVector
bHYPRE_StructVector__wrapObj (void* data, sidl_BaseInterface* _ex)

Wraps up the private data struct pointer (struct bHYPRE_StructVector__data) passed in rather than running the constructor

4.4.5

bHYPRE_StructVector
bHYPRE_StructVector__connect (const char* , sidl_BaseInterface* _ex)

RMI connector function for the class(addrefs)

4.4.6

bHYPRE_StructVector
bHYPRE_StructVector_Create (bHYPRE_MPICommunicator mpi_comm, bHYPRE_StructGrid grid, sidl_BaseInterface* _ex)

This function is the preferred way to create a Struct Vector.

4.4.7

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructVector_SetGrid (bHYPRE_StructVector self, bHYPRE_StructGrid grid, sidl_BaseInterface* _ex)

Set the grid on which vectors are defined.

4.4.8

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructVector_SetNumGhost (bHYPRE_StructVector self, int32_t* num_ghost, int32_t dim2, sidl_BaseInterface* _ex)

Set the number of ghost zones, separately on the lower and upper sides for each dimension. "num_ghost" is an array of size "dim2", twice the number of dimensions.

4.4.9

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructVector_SetValue (bHYPRE_StructVector self, int32_t* grid_index, int32_t dim, double value, sidl_BaseInterface* _ex)

Set the value of a single vector coefficient, given by "grid_index". "grid_index" is an array of size "dim", where dim is the number of dimensions.

4.4.10

int32_t
bHYPRE_StructVector_SetBoxValues (bHYPRE_StructVector self, int32_t* ilower, int32_t* iupper, int32_t dim, double* values, int32_t nvalues, sidl_BaseInterface* _ex)

Set the values of all vector coefficient for grid points in a box. The box is defined by its lower and upper corners in the grid. "ilower" and "iupper" are arrays of size "dim", where dim is the number of dimensions. The "values" array has size "nvalues", which is the number of grid points in the box.

4.4.11

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructVector_SetCommunicator (bHYPRE_StructVector self,
bHYPRE_MPICommunicator mpi_comm, sidl_BaseInterface* _ex)

Set the MPI Communicator. DEPRECATED, Use Create()

4.4.12

SIDL_C_INLINE_DECL void
bHYPRE_StructVector_Destroy (bHYPRE_StructVector self,
sidl_BaseInterface* _ex)

The Destroy function doesn't necessarily destroy anything. It is just another name for deleteRef. Thus it decrements the object's reference count. The Babel memory management system will destroy the object if the reference count goes to zero.

4.4.13

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructVector_Initialize (bHYPRE_StructVector self,
sidl_BaseInterface* _ex)

Prepare an object for setting coefficient values, whether for the first time or subsequently

4.4.14

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructVector_Assemble (bHYPRE_StructVector self,
sidl_BaseInterface* _ex)

Finalize the construction of an object before using, either for the first time or on subsequent uses. `Initialize` and `Assemble` always appear in a matched set, with Initialize preceding Assemble. Values can only be set in between a call to Initialize and Assemble.

4.4.15

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructVector_Clear (bHYPRE_StructVector self,
sidl_BaseInterface* _ex)

Set `self` to 0

4.4.16

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructVector_Copy (bHYPRE_StructVector self, bHYPRE_Vector x, sidl_BaseInterface* _ex)

Copy data from x into `self`

4.4.17

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructVector_Clone (bHYPRE_StructVector self, bHYPRE_Vector* x, sidl_BaseInterface* _ex)

Create an `x` compatible with `self`. The new vector's data is not specified.

NOTE: When this method is used in an inherited class, the cloned `Vector` object can be cast to an object with the inherited class type.

4.4.18

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructVector_Scale (bHYPRE_StructVector self, double a, sidl_BaseInterface* _ex)

Scale `self` by `a`

4.4.19

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructVector_Dot (bHYPRE_StructVector self, bHYPRE_Vector x, double* d, sidl_BaseInterface* _ex)

Compute `d`, the inner-product of `self` and `x`

4.4.20

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructVector_Axpy (bHYPRE_StructVector self, double a, bHYPRE_Vector x, sidl_BaseInterface* _ex)

Add `ax` to `self`

4.4.21

struct bHYPRE_StructVector__object*
bHYPRE_StructVector__cast (void* obj, sidl_BaseInterface* _ex)

Cast method for interface and class type conversions

4.4.22

void*
bHYPRE_StructVector__cast2 (void* obj, const char* type, sidl_BaseInterface* _ex)

String cast method for interface and class type conversions

4.4.23

SIDL_C_INLINE_DECL void
bHYPRE_StructVector__exec (bHYPRE_StructVector self, const char* methodName, sidl_rmi_Call inArgs, sidl_rmi_Return outArgs, sidl_BaseInterface* _ex)

Select and execute a method by name

4.4.24

SIDL_C_INLINE_DECL char*
bHYPRE_StructVector__getURL (bHYPRE_StructVector self, sidl_BaseInterface* _ex)

Get the URL of the Implementation of this object (for RMI)

4.4.25

SIDL_C_INLINE_DECL void
bHYPRE_StructVector__raddRef (bHYPRE_StructVector self, sidl_BaseInterface* _ex)

On a remote object, addrefs the remote instance

4.4.26

SIDL_C_INLINE_DECL sidl_bool
bHYPRE_StructVector__isRemote (bHYPRE_StructVector self, sidl_BaseInterface* _ex)

TRUE if this object is remote, false if local

4.4.27

sidl_bool
bHYPRE_StructVector__isLocal (bHYPRE_StructVector self,
sidl_BaseInterface* _ex)

TRUE if this object is remote, false if local

4.4.28

struct bHYPRE_StructVector__object*
bHYPRE_StructVector__rmicast (void* obj, struct sidl_BaseInterface__object**
_ex)

Cast method for interface and class type conversions

4.4.29

struct bHYPRE_StructVector__object*
bHYPRE_StructVector__connectI (const char* url, sidl_bool ar, struct
sidl_BaseInterface__object** _ex)

RMI connector function for the class. (no addref)

4.5

SemiStructured Matrix

Names

4.5.1

struct **bHYPRE_SStructMatrix__object**

Symbol "bHYPRESStructMatrix" (version 100)

The semi-structured grid matrix class.

Objects of this type can be cast to SStructMatrixView or Operator objects using the `__cast` methods.

4.5.2

struct bHYPRE_SStructMatrix__object*
bHYPRE_SStructMatrix__create (sidl_BaseInterface* _ex)

Constructor function for the class

4.5.3

bHYPRE_SStructMatrix
bHYPRE_SStructMatrix__createRemote (const char* url, sidl_BaseInterface* _ex)

RMI constructor function for the class

4.5.4

bHYPRE_SStructMatrix
bHYPRE_SStructMatrix__wrapObj (void* data, sidl_BaseInterface* _ex)

Wraps up the private data struct pointer (struct bHYPRE_SStructMatrix__data) passed in rather than running the constructor

4.5.5

bHYPRE_SStructMatrix
bHYPRE_SStructMatrix__connect (const char* , sidl_BaseInterface* _ex)

RMI connector function for the class(addrefs)

4.5.6

bHYPRE_SStructMatrix
bHYPRE_SStructMatrix_Create (bHYPRE_MPICommunicator mpi_comm, bHYPRE_SStructGraph graph, sidl_BaseInterface* _ex)

This function is the preferred way to create a SStruct Matrix.

4.5.7

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructMatrix_SetObjectType (bHYPRE_SStructMatrix self, int32_t type, sidl_BaseInterface* _ex)

Method: SetObjectType[]

4.5.8

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructMatrix_SetGraph (bHYPRE_SStructMatrix self, bHYPRE_SStructGraph graph, sidl_BaseInterface* _ex)

Set the matrix graph. DEPRECATED Use Create

4.5.9

int32_t
bHYPRE_SStructMatrix_SetValues (bHYPRE_SStructMatrix self, int32_t part, int32_t* index, int32_t dim, int32_t var, int32_t nentries, int32_t* entries, double* values, sidl_BaseInterface* _ex)

Set matrix coefficients index by index.

NOTE: Users are required to set values on all processes that own the associated variables. This means that some data will be multiply defined.

NOTE: The entries in this routine must all be of the same type: either stencil or non-stencil, but not both. Also, if they are stencil entries, they must all represent couplings to the same variable type (there are no such restrictions for non-stencil entries).

If the matrix is complex, then `values` consists of pairs of doubles representing the real and imaginary parts of each complex value.

4.5.10

int32_t
bHYPRE_SStructMatrix_SetBoxValues (bHYPRE_SStructMatrix self, int32_t part, int32_t* ilower, int32_t* iupper, int32_t dim, int32_t var, int32_t nentries, int32_t* entries, double* values, int32_t nvalues, sidl_BaseInterface* _ex)

Set matrix coefficients a box at a time.

NOTE: Users are required to set values on all processes that own the associated variables. This means that some data will be multiply defined.

NOTE: The entries in this routine must all be of the same type: either stencil or non-stencil, but not both. Also, if they are stencil entries, they must all represent couplings to the same variable type (there are no such restrictions for non-stencil entries).

If the matrix is complex, then `values` consists of pairs of doubles representing the real and imaginary parts of each complex value.

4.5.11

int32_t
bHYPRE_SStructMatrix_AddToValues (bHYPRE_SStructMatrix self, int32_t part, int32_t* index, int32_t dim, int32_t var, int32_t nentries, int32_t* entries, double* values, sidl_BaseInterface* _ex)

Add to matrix coefficients index by index.

NOTE: Users are required to set values on all processes that own the associated variables. This means that some data will be multiply defined.

NOTE: The entries in this routine must all be of the same type: either stencil or non-stencil, but not both. Also, if they are stencil entries, they must all represent couplings to the same variable type.

If the matrix is complex, then `values` consists of pairs of doubles representing the real and imaginary parts of each complex value.

4.5.12

int32_t
bHYPRE_SStructMatrix_AddToBoxValues (bHYPRE_SStructMatrix self, int32_t part, int32_t* ilower, int32_t* iupper, int32_t dim, int32_t var, int32_t nentries, int32_t* entries, double* values, int32_t nvalues, sidl_BaseInterface* _ex)

Add to matrix coefficients a box at a time.

NOTE: Users are required to set values on all processes that own the associated variables. This means that some data will be multiply defined.

NOTE: The entries in this routine must all be of stencil type. Also, they must all represent couplings to the same variable type.

If the matrix is complex, then `values` consists of pairs of doubles representing the real and imaginary parts of each complex value.

4.5.13

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructMatrix_SetSymmetric (bHYPRE_SStructMatrix self, int32_t part, int32_t var, int32_t to_var, int32_t symmetric, sidl_BaseInterface* _ex)

Define symmetry properties for the stencil entries in the matrix. The boolean argument `symmetric` is applied to stencil entries on part `part` that couple variable `var` to variable `to_var`. A value of -1 may be used for `part`, `var`, or `to_var` to specify "all". For example, if `part` and `to_var` are set to -1, then the boolean is applied to stencil entries on all parts that couple variable `var` to all other variables.

By default, matrices are assumed to be nonsymmetric. Significant storage savings can be made if the matrix is symmetric.

4.5.14

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructMatrix_SetNSSymmetric (bHYPRE_SStructMatrix self, int32_t symmetric, sidl_BaseInterface* _ex)

Define symmetry properties for all non-stencil matrix entries

4.5.15

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructMatrix_SetComplex (bHYPRE_SStructMatrix self, sidl_BaseInterface* _ex)

Set the matrix to be complex

4.5.16

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructMatrix_Print (bHYPRE_SStructMatrix self, const char* filename, int32_t all, sidl_BaseInterface* _ex)

Print the matrix to file. This is mainly for debugging purposes.

4.5.17

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructMatrix_GetObject (bHYPRE_SStructMatrix self, sidl_BaseInterface* A, sidl_BaseInterface* _ex)

A semi-structured matrix or vector contains a Struct or IJ matrix or vector. GetObject returns it. The returned type is a sidl.BaseInterface. A cast must be used on the returned object to convert it into a known type.

4.5.18

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructMatrix_SetCommunicator (bHYPRE_SStructMatrix self, bHYPRE_MPICommunicator mpi_comm, sidl_BaseInterface* _ex)

Set the MPI Communicator. DEPRECATED, Use Create()

4.5.19

SIDL_C_INLINE_DECL void
bHYPRE_SStructMatrix_Destroy (bHYPRE_SStructMatrix self,
sidl_BaseInterface* _ex)

The Destroy function doesn't necessarily destroy anything. It is just another name for deleteRef. Thus it decrements the object's reference count. The Babel memory management system will destroy the object if the reference count goes to zero.

4.5.20

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructMatrix_Initialize (bHYPRE_SStructMatrix self,
sidl_BaseInterface* _ex)

Prepare an object for setting coefficient values, whether for the first time or subsequently

4.5.21

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructMatrix_Assemble (bHYPRE_SStructMatrix self,
sidl_BaseInterface* _ex)

Finalize the construction of an object before using, either for the first time or on subsequent uses. `Initialize` and `Assemble` always appear in a matched set, with Initialize preceding Assemble. Values can only be set in between a call to Initialize and Assemble.

4.5.22

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructMatrix_SetIntParameter (bHYPRE_SStructMatrix self,
const char* name, int32_t value, sidl_BaseInterface* _ex)

Set the int parameter associated with `name`

4.5.23

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructMatrix_SetDoubleParameter (bHYPRE_SStructMatrix self, const char* name, double value, sidl_BaseInterface* _ex)

Set the double parameter associated with `name`

4.5.24

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructMatrix_SetStringParameter (bHYPRE_SStructMatrix self, const char* name, const char* value, sidl_BaseInterface* _ex)

Set the string parameter associated with `name`

4.5.25

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructMatrix_SetIntArray1Parameter (bHYPRE_SStructMatrix self, const char* name, int32_t* value, int32_t nvalues, sidl_BaseInterface* _ex)

Set the int 1-D array parameter associated with `name`

4.5.26

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructMatrix_SetIntArray2Parameter (bHYPRE_SStructMatrix self, const char* name, struct sidl_int__array* value, sidl_BaseInterface* _ex)

Set the int 2-D array parameter associated with `name`

4.5.27

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructMatrix_SetDoubleArray1Parameter (
bHYPRE_SStructMatrix self, const char* name, double* value, int32_t nvalues,
sidl_BaseInterface* _ex)

Set the double 1-D array parameter associated with `name`

4.5.28

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructMatrix_SetDoubleArray2Parameter (
bHYPRE_SStructMatrix self, const char* name, struct sidl_double__array* value,
sidl_BaseInterface* _ex)

Set the double 2-D array parameter associated with `name`

4.5.29

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructMatrix_GetIntValue (bHYPRE_SStructMatrix self, const
char* name, int32_t* value, sidl_BaseInterface* _ex)

Set the int parameter associated with `name`

4.5.30

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructMatrix_GetDoubleValue (bHYPRE_SStructMatrix self,
const char* name, double* value, sidl_BaseInterface* _ex)

Get the double parameter associated with `name`

4.5.31

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructMatrix_Setup (bHYPRE_SStructMatrix self,
bHYPRE_Vector b, bHYPRE_Vector x, sidl_BaseInterface* _ex)

(Optional) Do any preprocessing that may be necessary in order to execute `Apply`

4.5.32

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructMatrix_Apply (bHYPRE_SStructMatrix self,
bHYPRE_Vector b, bHYPRE_Vector* x, sidl_BaseInterface* _ex)

Apply the operator to `b`, returning `x`

4.5.33

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructMatrix_ApplyAdjoint (bHYPRE_SStructMatrix self,
bHYPRE_Vector b, bHYPRE_Vector* x, sidl_BaseInterface* _ex)

Apply the adjoint of the operator to `b`, returning `x`

4.5.34

struct bHYPRE_SStructMatrix__object*
bHYPRE_SStructMatrix__cast (void* obj, sidl_BaseInterface* _ex)

Cast method for interface and class type conversions

4.5.35

void*
bHYPRE_SStructMatrix__cast2 (void* obj, const char* type,
sidl_BaseInterface* _ex)

String cast method for interface and class type conversions

4.5.36

SIDL_C_INLINE_DECL void
bHYPRE_SStructMatrix__exec (bHYPRE_SStructMatrix self, const char*
methodName, sidl_rmi_Call inArgs, sidl_rmi_Return outArgs, sidl_BaseInterface*
_ex)

Select and execute a method by name

4.5.37

SIDL_C_INLINE_DECL char*
bHYPRE_SStructMatrix__getURL (bHYPRE_SStructMatrix self,
sidl_BaseInterface* _ex)

Get the URL of the Implementation of this object (for RMI)

4.5.38

SIDL_C_INLINE_DECL void
bHYPRE_SStructMatrix__raddRef (bHYPRE_SStructMatrix self,
sidl_BaseInterface* _ex)

On a remote object, addrefs the remote instance

4.5.39

SIDL_C_INLINE_DECL sidl_bool
bHYPRE_SStructMatrix__isRemote (bHYPRE_SStructMatrix self,
sidl_BaseInterface* _ex)

TRUE if this object is remote, false if local

4.5.40

sidl_bool
bHYPRE_SStructMatrix__isLocal (bHYPRE_SStructMatrix self,
sidl_BaseInterface* _ex)

TRUE if this object is remote, false if local

4.5.41

struct bHYPRE_SStructMatrix__object*
bHYPRE_SStructMatrix__rmicast (void* obj, struct
sidl_BaseInterface__object** _ex)

Cast method for interface and class type conversions

4.5.42

struct bHYPRE_SStructMatrix__object*
bHYPRE_SStructMatrix__connectI (const char* url, sidl_bool ar, struct
sidl_BaseInterface__object** _ex)

RMI connector function for the class. (no addref)

4.6

SemiStructured Vector

Names

4.6.1

struct **bHYPRE_SStructVector__object**

Symbol "bHYPRESStructVector" (version 100)

The semi-structured grid vector class.

Objects of this type can be cast to SStructVectorView or Vector objects using the `__cast` methods.

4.6.2

struct bHYPRE_SStructVector__object*
bHYPRE_SStructVector__create (sidl_BaseInterface* _ex)

Constructor function for the class

4.6.3

bHYPRE_SStructVector
bHYPRE_SStructVector__createRemote (const char* url, sidl_BaseInterface* _ex)

RMI constructor function for the class

4.6.4

bHYPRE_SStructVector
bHYPRE_SStructVector__wrapObj (void* data, sidl_BaseInterface* _ex)

Wraps up the private data struct pointer (struct bHYPRE_SStructVector__data) passed in rather than running the constructor

4.6.5

bHYPRE_SStructVector
bHYPRE_SStructVector__connect (const char* , sidl_BaseInterface* _ex)

RMI connector function for the class(addrefs)

4.6.6

bHYPRE_SStructVector
bHYPRE_SStructVector_Create (bHYPRE_MPICommunicator mpi_comm,
bHYPRE_SStructGrid grid, sidl_BaseInterface* _ex)

This function is the preferred way to create a SStruct Vector.

4.6.7

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructVector_SetObjectType (bHYPRE_SStructVector self,
int32_t type, sidl_BaseInterface* _ex)

Method: SetObjectType[]

4.6.8

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructVector_SetGrid (bHYPRE_SStructVector self,
bHYPRE_SStructGrid grid, sidl_BaseInterface* _ex)

Set the vector grid

4.6.9

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructVector_SetValues (bHYPRE_SStructVector self, int32_t part, int32_t* index, int32_t dim, int32_t var, double value, sidl_BaseInterface* _ex)

Set vector coefficients index by index.

NOTE: Users are required to set values on all processes that own the associated variables. This means that some data will be multiply defined.

If the vector is complex, then `value` consists of a pair of doubles representing the real and imaginary parts of the complex value.

4.6.10

int32_t
bHYPRE_SStructVector_SetBoxValues (bHYPRE_SStructVector self, int32_t part, int32_t* ilower, int32_t* iupper, int32_t dim, int32_t var, double* values, int32_t nvalues, sidl_BaseInterface* _ex)

Set vector coefficients a box at a time.

NOTE: Users are required to set values on all processes that own the associated variables. This means that some data will be multiply defined.

If the vector is complex, then `values` consists of pairs of doubles representing the real and imaginary parts of each complex value.

4.6.11

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructVector_AddToValues (bHYPRE_SStructVector self, int32_t part, int32_t* index, int32_t dim, int32_t var, double value, sidl_BaseInterface* _ex)

Set vector coefficients index by index.

NOTE: Users are required to set values on all processes that own the associated variables. This means that some data will be multiply defined.

If the vector is complex, then `value` consists of a pair of doubles representing the real and imaginary parts of the complex value.

4.6.12

int32_t
bHYPRE_SStructVector_AddToBoxValues (bHYPRE_SStructVector self, int32_t part, int32_t* ilower, int32_t* iupper, int32_t dim, int32_t var, double* values, int32_t nvalues, sidl_BaseInterface* _ex)

Set vector coefficients a box at a time.

NOTE: Users are required to set values on all processes that own the associated variables. This means that some data will be multiply defined.

If the vector is complex, then `values` consists of pairs of doubles representing the real and imaginary parts of each complex value.

4.6.13

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructVector_Gather (bHYPRE_SStructVector self, sidl_BaseInterface* _ex)

Gather vector data before calling `GetValues`

4.6.14

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructVector_GetValues (bHYPRE_SStructVector self, int32_t part, int32_t* index, int32_t dim, int32_t var, double* value, sidl_BaseInterface* _ex)

Get vector coefficients index by index.

NOTE: Users may only get values on processes that own the associated variables.

If the vector is complex, then `value` consists of a pair of doubles representing the real and imaginary parts of the complex value.

4.6.15

int32_t
bHYPRE_SStructVector_GetBoxValues (bHYPRE_SStructVector self, int32_t part, int32_t* ilower, int32_t* iupper, int32_t dim, int32_t var, double* values, int32_t nvalues, sidl_BaseInterface* _ex)

Get vector coefficients a box at a time.

NOTE: Users may only get values on processes that own the associated variables.

If the vector is complex, then `values` consists of pairs of doubles representing the real and imaginary parts of each complex value.

4.6.16

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructVector_SetComplex (bHYPRE_SStructVector self, sidl_BaseInterface* _ex)

Set the vector to be complex

4.6.17

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructVector_Print (bHYPRE_SStructVector self, const char* filename, int32_t all, sidl_BaseInterface* _ex)

Print the vector to file. This is mainly for debugging purposes.

4.6.18

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructVector_GetObject (bHYPRE_SStructVector self, sidl_BaseInterface* A, sidl_BaseInterface* _ex)

A semi-structured matrix or vector contains a Struct or IJ matrix or vector. GetObject returns it. The returned type is a sidl.BaseInterface. A cast must be used on the returned object to convert it into a known type.

4.6.19

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructVector_SetCommunicator (bHYPRE_SStructVector self,
bHYPRE_MPICommunicator mpi_comm, sidl_BaseInterface* _ex)

Set the MPI Communicator. DEPRECATED, Use Create()

4.6.20

SIDL_C_INLINE_DECL void
bHYPRE_SStructVector_Destroy (bHYPRE_SStructVector self,
sidl_BaseInterface* _ex)

The Destroy function doesn't necessarily destroy anything. It is just another name for deleteRef. Thus it decrements the object's reference count. The Babel memory management system will destroy the object if the reference count goes to zero.

4.6.21

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructVector_Initialize (bHYPRE_SStructVector self,
sidl_BaseInterface* _ex)

Prepare an object for setting coefficient values, whether for the first time or subsequently

4.6.22

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructVector_Assemble (bHYPRE_SStructVector self,
sidl_BaseInterface* _ex)

Finalize the construction of an object before using, either for the first time or on subsequent uses. `Initialize` and `Assemble` always appear in a matched set, with Initialize preceding Assemble. Values can only be set in between a call to Initialize and Assemble.

4.6.23

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructVector_Clear (bHYPRE_SStructVector self,
sidl_BaseInterface* _ex)

Set `self` to 0

4.6.24

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructVector_Copy (bHYPRE_SStructVector self,
bHYPRE_Vector x, sidl_BaseInterface* _ex)

Copy data from x into `self`

4.6.25

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructVector_Clone (bHYPRE_SStructVector self,
bHYPRE_Vector* x, sidl_BaseInterface* _ex)

Create an `x` compatible with `self`. The new vector's data is not specified.

NOTE: When this method is used in an inherited class, the cloned `Vector` object can be cast to an object with the inherited class type.

4.6.26

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructVector_Scale (bHYPRE_SStructVector self, double a, sidl_BaseInterface* _ex)

Scale `self` by `a`

4.6.27

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructVector_Dot (bHYPRE_SStructVector self, bHYPRE_Vector x, double* d, sidl_BaseInterface* _ex)

Compute `d`, the inner-product of `self` and `x`

4.6.28

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructVector_Axpy (bHYPRE_SStructVector self, double a, bHYPRE_Vector x, sidl_BaseInterface* _ex)

Add `ax` to `self`

4.6.29

struct bHYPRE_SStructVector__object*
bHYPRE_SStructVector__cast (void* obj, sidl_BaseInterface* _ex)

Cast method for interface and class type conversions

4.6.30

void*
bHYPRE_SStructVector__cast2 (void* obj, const char* type,
sidl_BaseInterface* _ex)

String cast method for interface and class type conversions

4.6.31

SIDL_C_INLINE_DECL void
bHYPRE_SStructVector__exec (bHYPRE_SStructVector self, const char*
methodName, sidl_rmi_Call inArgs, sidl_rmi_Return outArgs, sidl_BaseInterface*
_ex)

Select and execute a method by name

4.6.32

SIDL_C_INLINE_DECL char*
bHYPRE_SStructVector__getURL (bHYPRE_SStructVector self,
sidl_BaseInterface* _ex)

Get the URL of the Implementation of this object (for RMI)

4.6.33

SIDL_C_INLINE_DECL void
bHYPRE_SStructVector__raddRef (bHYPRE_SStructVector self,
sidl_BaseInterface* _ex)

On a remote object, addrefs the remote instance

4.6.34

SIDL_C_INLINE_DECL sidl_bool
bHYPRE_SStructVector__isRemote (bHYPRE_SStructVector self,
sidl_BaseInterface* _ex)

TRUE if this object is remote, false if local

4.6.35

sidl_bool
bHYPRE_SStructVector__isLocal (bHYPRE_SStructVector self,
sidl_BaseInterface* _ex)

TRUE if this object is remote, false if local

4.6.36

struct bHYPRE_SStructVector__object*
bHYPRE_SStructVector__rmicast (void* obj, struct
sidl_BaseInterface__object** _ex)

Cast method for interface and class type conversions

4.6.37

struct bHYPRE_SStructVector__object*
bHYPRE_SStructVector__connectI (const char* url, sidl_bool ar, struct
sidl_BaseInterface__object** _ex)

RMI connector function for the class. (no addref)

4.7

SemiStructured ParCSR Matrix

Names

4.7.1

struct **bHYPRE_SStructParCSRMatrix__object**

Symbol "bHYPRESStructParCSRMatrix" (version 100)

The SStructParCSR matrix class.

Objects of this type can be cast to SStructMatrixView or Operator objects using the `__cast` methods.

4.7.2

struct bHYPRE_SStructParCSRMatrix__object*
bHYPRE_SStructParCSRMatrix__create (sidl_BaseInterface* _ex)

Constructor function for the class

4.7.3

bHYPRE_SStructParCSRMatrix
bHYPRE_SStructParCSRMatrix__createRemote (const char* url, sidl_BaseInterface* _ex)

RMI constructor function for the class

4.7.4

bHYPRE_SStructParCSRMatrix
bHYPRE_SStructParCSRMatrix__wrapObj (void* data, sidl_BaseInterface* _ex)

Wraps up the private data struct pointer (struct bHYPRE_SStructParCSRMatrix__data) passed in rather than running the constructor

4.7.5

bHYPRE_SStructParCSRMatrix
bHYPRE_SStructParCSRMatrix__connect (const char* , sidl_BaseInterface* _ex)

RMI connector function for the class(addrefs)

4.7.6

bHYPRE_SStructParCSRMatrix
bHYPRE_SStructParCSRMatrix_Create (bHYPRE_MPICommunicator mpi_comm, bHYPRE_SStructGraph graph, sidl_BaseInterface* _ex)

This function is the preferred way to create a SStruct ParCSR Matrix.

4.7.7

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructParCSRMatrix_SetGraph (
bHYPRE_SStructParCSRMatrix self, bHYPRE_SStructGraph graph,
sidl_BaseInterface* _ex)

Set the matrix graph. DEPRECATED Use Create

4.7.8

int32_t
bHYPRE_SStructParCSRMatrix_SetValues (
bHYPRE_SStructParCSRMatrix self, int32_t part, int32_t* index, int32_t dim,
int32_t var, int32_t nentries, int32_t* entries, double* values, sidl_BaseInterface*
_ex)

Set matrix coefficients index by index.

NOTE: Users are required to set values on all processes that own the associated variables. This means that some data will be multiply defined.

NOTE: The entries in this routine must all be of the same type: either stencil or non-stencil, but not both. Also, if they are stencil entries, they must all represent couplings to the same variable type (there are no such restrictions for non-stencil entries).

If the matrix is complex, then `values` consists of pairs of doubles representing the real and imaginary parts of each complex value.

4.7.9

int32_t
bHYPRE_SStructParCSRMatrix_SetBoxValues (
bHYPRE_SStructParCSRMatrix self, int32_t part, int32_t* ilower, int32_t*
iupper, int32_t dim, int32_t var, int32_t nentries, int32_t* entries, double* values,
int32_t nvalues, sidl_BaseInterface* _ex)

Set matrix coefficients a box at a time.

NOTE: Users are required to set values on all processes that own the associated variables. This means that some data will be multiply defined.

NOTE: The entries in this routine must all be of the same type: either stencil or non-stencil, but not both. Also, if they are stencil entries, they must all represent couplings to the same variable type (there are no such restrictions for non-stencil entries).

If the matrix is complex, then `values` consists of pairs of doubles representing the real and imaginary parts of each complex value.

4.7.10

int32_t
bHYPRE_SStructParCSRMatrix_AddToValues (
bHYPRE_SStructParCSRMatrix self, int32_t part, int32_t* index, int32_t dim, int32_t var, int32_t nentries, int32_t* entries, double* values, sidl_BaseInterface* _ex)

Add to matrix coefficients index by index.

NOTE: Users are required to set values on all processes that own the associated variables. This means that some data will be multiply defined.

NOTE: The entries in this routine must all be of the same type: either stencil or non-stencil, but not both. Also, if they are stencil entries, they must all represent couplings to the same variable type.

If the matrix is complex, then `values` consists of pairs of doubles representing the real and imaginary parts of each complex value.

4.7.11

int32_t
bHYPRE_SStructParCSRMatrix_AddToBoxValues (
bHYPRE_SStructParCSRMatrix self, int32_t part, int32_t* ilower, int32_t* iupper, int32_t dim, int32_t var, int32_t nentries, int32_t* entries, double* values, int32_t nvalues, sidl_BaseInterface* eex)

Add to matrix coefficients a box at a time.

NOTE: Users are required to set values on all processes that own the associated variables. This means that some data will be multiply defined.

NOTE: The entries in this routine must all be of stencil type. Also, they must all represent couplings to the same variable type.

If the matrix is complex, then `values` consists of pairs of doubles representing the real and imaginary parts of each complex value.

4.7.12

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructParCSRMatrix_SetSymmetric (
bHYPRE_SStructParCSRMatrix self, int32_t part, int32_t var, int32_t to_var,
int32_t symmetric, sidl_BaseInterface* _ex)

Define symmetry properties for the stencil entries in the matrix. The boolean argument `symmetric` is applied to stencil entries on part `part` that couple variable `var` to variable `to_var`. A value of -1 may be used for `part`, `var`, or `to_var` to specify "all". For example, if `part` and `to_var` are set to -1, then the boolean is applied to stencil entries on all parts that couple variable `var` to all other variables.

By default, matrices are assumed to be nonsymmetric. Significant storage savings can be made if the matrix is symmetric.

4.7.13

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructParCSRMatrix_SetNSSymmetric (
bHYPRE_SStructParCSRMatrix self, int32_t symmetric, sidl_BaseInterface* _ex)

Define symmetry properties for all non-stencil matrix entries

4.7.14

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructParCSRMatrix_SetComplex (
bHYPRE_SStructParCSRMatrix self, sidl_BaseInterface* _ex)

Set the matrix to be complex

4.7.15

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructParCSRMatrix_Print (bHYPRE_SStructParCSRMatrix self, const char* filename, int32_t all, sidl_BaseInterface* _ex)

Print the matrix to file. This is mainly for debugging purposes.

4.7.16

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructParCSRMatrix_GetObject (
bHYPRE_SStructParCSRMatrix self, sidl_BaseInterface* A, sidl_BaseInterface* _ex)

A semi-structured matrix or vector contains a Struct or IJ matrix or vector. GetObject returns it. The returned type is a sidl.BaseInterface. A cast must be used on the returned object to convert it into a known type.

4.7.17

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructParCSRMatrix_SetCommunicator (
bHYPRE_SStructParCSRMatrix self, bHYPRE_MPICommunicator mpi_comm, sidl_BaseInterface* _ex)

Set the MPI Communicator. DEPRECATED, Use Create()

4.7.18

SIDL_C_INLINE_DECL void
bHYPRE_SStructParCSRMatrix_Destroy (bHYPRE_SStructParCSRMatrix self, sidl_BaseInterface* _ex)

The Destroy function doesn't necessarily destroy anything. It is just another name for deleteRef. Thus it decrements the object's reference count. The Babel memory management system will destroy the object if the reference count goes to zero.

4.7.19

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructParCSRMatrix_Initialize (
bHYPRE_SStructParCSRMatrix self, sidl_BaseInterface* _ex)

Prepare an object for setting coefficient values, whether for the first time or subsequently

4.7.20

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructParCSRMatrix_Assemble (
bHYPRE_SStructParCSRMatrix self, sidl_BaseInterface* _ex)

Finalize the construction of an object before using, either for the first time or on subsequent uses. `Initialize` and `Assemble` always appear in a matched set, with Initialize preceding Assemble. Values can only be set in between a call to Initialize and Assemble.

4.7.21

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructParCSRMatrix_SetIntParameter (
bHYPRE_SStructParCSRMatrix self, const char* name, int32_t value,
sidl_BaseInterface* _ex)

Set the int parameter associated with `name`

4.7.22

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructParCSRMatrix_SetDoubleParameter (
bHYPRE_SStructParCSRMatrix self, const char* name, double value,
sidl_BaseInterface* _ex)

Set the double parameter associated with `name`

4.7.23

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructParCSRMatrix_SetStringParameter (
bHYPRE_SStructParCSRMatrix self, const char* name, const char* value,
sidl_BaseInterface* _ex)

Set the string parameter associated with `name`

4.7.24

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructParCSRMatrix_SetIntArray1Parameter (
bHYPRE_SStructParCSRMatrix self, const char* name, int32_t* value, int32_t
nvalues, sidl_BaseInterface* _ex)

Set the int 1-D array parameter associated with `name`

4.7.25

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructParCSRMatrix_SetIntArray2Parameter (
bHYPRE_SStructParCSRMatrix self, const char* name, struct sidl_int__array*
value, sidl_BaseInterface* _ex)

Set the int 2-D array parameter associated with `name`

4.7.26

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructParCSRMatrix_SetDoubleArray1Parameter (
bHYPRE_SStructParCSRMatrix self, const char* name, double* value, int32_t
nvalues, sidl_BaseInterface* _ex)

Set the double 1-D array parameter associated with `name`

4.7.27

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructParCSRMatrix_SetDoubleArray2Parameter (
bHYPRE_SStructParCSRMatrix self, const char* name, struct sidl_double__array*
value, sidl_BaseInterface* _ex)

Set the double 2-D array parameter associated with `name`

4.7.28

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructParCSRMatrix_GetIntValue (
bHYPRE_SStructParCSRMatrix self, const char* name, int32_t* value,
sidl_BaseInterface* _ex)

Set the int parameter associated with `name`

4.7.29

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructParCSRMatrix_GetDoubleValue (
bHYPRE_SStructParCSRMatrix self, const char* name, double* value,
sidl_BaseInterface* _ex)

Get the double parameter associated with `name`

4.7.30

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructParCSRMatrix_Setup (bHYPRE_SStructParCSRMatrix
self, bHYPRE_Vector b, bHYPRE_Vector x, sidl_BaseInterface* _ex)

(Optional) Do any preprocessing that may be necessary in order to execute `Apply`

4.7.31

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructParCSRMatrix_Apply (bHYPRE_SStructParCSRMatrix self, bHYPRE_Vector b, bHYPRE_Vector* x, sidl_BaseInterface* _ex)

Apply the operator to `b`, returning `x`

4.7.32

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructParCSRMatrix_ApplyAdjoint (
bHYPRE_SStructParCSRMatrix self, bHYPRE_Vector b, bHYPRE_Vector* x,
sidl_BaseInterface* _ex)

Apply the adjoint of the operator to `b`, returning `x`

4.7.33

struct bHYPRE_SStructParCSRMatrix__object*
bHYPRE_SStructParCSRMatrix__cast (void* obj, sidl_BaseInterface* _ex)

Cast method for interface and class type conversions

4.7.34

void*
bHYPRE_SStructParCSRMatrix__cast2 (void* obj, const char* type,
sidl_BaseInterface* _ex)

String cast method for interface and class type conversions

4.7.35

SIDL_C_INLINE_DECL void
bHYPRE_SStructParCSRMatrix__exec (bHYPRE_SStructParCSRMatrix self, const char* methodName, sidl_rmi_Call inArgs, sidl_rmi_Return outArgs, sidl_BaseInterface* _ex)

Select and execute a method by name

4.7.36

SIDL_C_INLINE_DECL char*
bHYPRE_SStructParCSRMatrix__getURL (
bHYPRE_SStructParCSRMatrix self, sidl_BaseInterface* _ex)

Get the URL of the Implementation of this object (for RMI)

4.7.37

SIDL_C_INLINE_DECL void
bHYPRE_SStructParCSRMatrix__raddRef (
bHYPRE_SStructParCSRMatrix self, sidl_BaseInterface* _ex)

On a remote object, addrefs the remote instance

4.7.38

SIDL_C_INLINE_DECL sidl_bool
bHYPRE_SStructParCSRMatrix__isRemote (
bHYPRE_SStructParCSRMatrix self, sidl_BaseInterface* _ex)

TRUE if this object is remote, false if local

4.7.39

sidl_bool
bHYPRE_SStructParCSRMatrix__isLocal (bHYPRE_SStructParCSRMatrix self, sidl_BaseInterface* _ex)

TRUE if this object is remote, false if local

4.7.40

struct bHYPRE_SStructParCSRMatrix__object*
bHYPRE_SStructParCSRMatrix__rmicast (void* obj, struct sidl_BaseInterface__object** _ex)

Cast method for interface and class type conversions

4.7.41

struct bHYPRE_SStructParCSRMatrix__object*
bHYPRE_SStructParCSRMatrix__connectI (const char* url, sidl_bool ar, struct sidl_BaseInterface__object** _ex)

RMI connector function for the class. (no addref)

4.8

SemiStructured ParCSR Vector

Names

4.8.1

struct **bHYPRE_SStructParCSRVector__object**

Symbol "bHYPRESStructParCSRVector" (version 100)

The SStructParCSR vector class.

Objects of this type can be cast to SStructVectorView or Vector objects using the `__cast` methods.

4.8.2

struct bHYPRE_SStructParCSRVector__object*
bHYPRE_SStructParCSRVector__create (sidl_BaseInterface* _ex)

Constructor function for the class

4.8.3

bHYPRE_SStructParCSRVector
bHYPRE_SStructParCSRVector__createRemote (const char* url, sidl_BaseInterface* _ex)

RMI constructor function for the class

4.8.4

bHYPRE_SStructParCSRVector
bHYPRE_SStructParCSRVector__wrapObj (void* data, sidl_BaseInterface* _ex)

Wraps up the private data struct pointer (struct bHYPRE_SStructParCSRVector__data) passed in rather than running the constructor

4.8.5

bHYPRE_SStructParCSRVector
bHYPRE_SStructParCSRVector__connect (const char* , sidl_BaseInterface* _ex)

RMI connector function for the class(addrefs)

4.8.6

bHYPRE_SStructParCSRVector
bHYPRE_SStructParCSRVector_Create (bHYPRE_MPICommunicator mpi_comm, bHYPRE_SStructGrid grid, sidl_BaseInterface* _ex)

This function is the preferred way to create a SStruct ParCSR Vector.

4.8.7

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructParCSRVector_SetGrid (bHYPRE_SStructParCSRVector self, bHYPRE_SStructGrid grid, sidl_BaseInterface* _ex)

Set the vector grid

4.8.8

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructParCSRVector_SetValues (bHYPRE_SStructParCSRVector self, int32_t part, int32_t* index, int32_t dim, int32_t var, double value, sidl_BaseInterface* _ex)

Set vector coefficients index by index.

NOTE: Users are required to set values on all processes that own the associated variables. This means that some data will be multiply defined.

If the vector is complex, then `value` consists of a pair of doubles representing the real and imaginary parts of the complex value.

4.8.9

int32_t
bHYPRE_SStructParCSRVector_SetBoxValues (
bHYPRE_SStructParCSRVector self, int32_t part, int32_t* ilower, int32_t* iupper, int32_t dim, int32_t var, double* values, int32_t nvalues, sidl_BaseInterface* _ex)

Set vector coefficients a box at a time.

NOTE: Users are required to set values on all processes that own the associated variables. This means that some data will be multiply defined.

If the vector is complex, then `values` consists of pairs of doubles representing the real and imaginary parts of each complex value.

4.8.10

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructParCSRVector_AddToValues (
bHYPRE_SStructParCSRVector self, int32_t part, int32_t* index, int32_t dim, int32_t var, double value, sidl_BaseInterface* _ex)

Set vector coefficients index by index.

NOTE: Users are required to set values on all processes that own the associated variables. This means that some data will be multiply defined.

If the vector is complex, then `value` consists of a pair of doubles representing the real and imaginary parts of the complex value.

4.8.11

int32_t
bHYPRE_SStructParCSRVector_AddToBoxValues (
bHYPRE_SStructParCSRVector self, int32_t part, int32_t* ilower, int32_t* iupper, int32_t dim, int32_t var, double* values, int32_t nvalues, sidl_BaseInterface* _ex)

Set vector coefficients a box at a time.

NOTE: Users are required to set values on all processes that own the associated variables. This means that some data will be multiply defined.

If the vector is complex, then `values` consists of pairs of doubles representing the real and imaginary parts of each complex value.

4.8.12

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructParCSRVector_Gather (bHYPRE_SStructParCSRVector self, sidl_BaseInterface* _ex)

Gather vector data before calling `GetValues`

4.8.13

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructParCSRVector_GetValues (
bHYPRE_SStructParCSRVector self, int32_t part, int32_t* index, int32_t dim, int32_t var, double* value, sidl_BaseInterface* _ex)

Get vector coefficients index by index.

NOTE: Users may only get values on processes that own the associated variables.

If the vector is complex, then `value` consists of a pair of doubles representing the real and imaginary parts of the complex value.

4.8.14

int32_t
bHYPRE_SStructParCSRVector_GetBoxValues (
bHYPRE_SStructParCSRVector self, int32_t part, int32_t* ilower, int32_t* iupper, int32_t dim, int32_t var, double* values, int32_t nvalues, sidl_BaseInterface* _ex)

Get vector coefficients a box at a time.

NOTE: Users may only get values on processes that own the associated variables.

If the vector is complex, then `values` consists of pairs of doubles representing the real and imaginary parts of each complex value.

4.8.15

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructParCSRVector_SetComplex (
bHYPRE_SStructParCSRVector self, sidl_BaseInterface* _ex)

Set the vector to be complex

4.8.16

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructParCSRVector_Print (bHYPRE_SStructParCSRVector self, const char* filename, int32_t all, sidl_BaseInterface* _ex)

Print the vector to file. This is mainly for debugging purposes.

4.8.17

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructParCSRVector_GetObject (
bHYPRE_SStructParCSRVector self, sidl_BaseInterface* A, sidl_BaseInterface* _ex)

A semi-structured matrix or vector contains a Struct or IJ matrix or vector. GetObject returns it. The returned type is a sidl.BaseInterface. A cast must be used on the returned object to convert it into a known type.

4.8.18

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructParCSRVector_SetCommunicator (
bHYPRE_SStructParCSRVector self, bHYPRE_MPICommunicator mpi_comm,
sidl_BaseInterface* _ex)

Set the MPI Communicator. DEPRECATED, Use Create()

4.8.19

SIDL_C_INLINE_DECL void
bHYPRE_SStructParCSRVector_Destroy (bHYPRE_SStructParCSRVector
self, sidl_BaseInterface* _ex)

The Destroy function doesn't necessarily destroy anything. It is just another name for deleteRef. Thus it decrements the object's reference count. The Babel memory management system will destroy the object if the reference count goes to zero.

4.8.20

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructParCSRVector_Initialize (bHYPRE_SStructParCSRVector
self, sidl_BaseInterface* _ex)

Prepare an object for setting coefficient values, whether for the first time or subsequently

4.8.21

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructParCSRVector_Assemble (bHYPRE_SStructParCSRVector self, sidl_BaseInterface* _ex)

Finalize the construction of an object before using, either for the first time or on subsequent uses. `Initialize` and `Assemble` always appear in a matched set, with Initialize preceding Assemble. Values can only be set in between a call to Initialize and Assemble.

4.8.22

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructParCSRVector_Clear (bHYPRE_SStructParCSRVector self, sidl_BaseInterface* _ex)

Set `self` to 0

4.8.23

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructParCSRVector_Copy (bHYPRE_SStructParCSRVector self, bHYPRE_Vector x, sidl_BaseInterface* _ex)

Copy data from x into `self`

4.8.24

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructParCSRVector_Clone (bHYPRE_SStructParCSRVector self, bHYPRE_Vector* x, sidl_BaseInterface* _ex)

Create an `x` compatible with `self`. The new vector's data is not specified.

NOTE: When this method is used in an inherited class, the cloned `Vector` object can be cast to an object with the inherited class type.

4.8.25

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructParCSRVector_Scale (bHYPRE_SStructParCSRVector self, double a, sidl_BaseInterface* _ex)

Scale `self` by `a`

4.8.26

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructParCSRVector_Dot (bHYPRE_SStructParCSRVector self, bHYPRE_Vector x, double* d, sidl_BaseInterface* _ex)

Compute `d`, the inner-product of `self` and `x`

4.8.27

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructParCSRVector_Axpy (bHYPRE_SStructParCSRVector self, double a, bHYPRE_Vector x, sidl_BaseInterface* _ex)

Add `ax` to `self`

4.8.28

struct bHYPRE_SStructParCSRVector__object*
bHYPRE_SStructParCSRVector__cast (void* obj, sidl_BaseInterface* _ex)

Cast method for interface and class type conversions

4.8.29

void*
bHYPRE_SStructParCSRVector__cast2 (void* obj, const char* type, sidl_BaseInterface* _ex)

String cast method for interface and class type conversions

4.8.30

SIDL_C_INLINE_DECL void
bHYPRE_SStructParCSRVector__exec (bHYPRE_SStructParCSRVector self, const char* methodName, sidl_rmi_Call inArgs, sidl_rmi_Return outArgs, sidl_BaseInterface* _ex)

Select and execute a method by name

4.8.31

SIDL_C_INLINE_DECL char*
bHYPRE_SStructParCSRVector__getURL (bHYPRE_SStructParCSRVector self, sidl_BaseInterface* _ex)

Get the URL of the Implementation of this object (for RMI)

4.8.32

SIDL_C_INLINE_DECL void
bHYPRE_SStructParCSRVector__raddRef (bHYPRE_SStructParCSRVector self, sidl_BaseInterface* _ex)

On a remote object, addrefs the remote instance

4.8.33

SIDL_C_INLINE_DECL sidl_bool
bHYPRE_SStructParCSRVector__isRemote (
bHYPRE_SStructParCSRVector self, sidl_BaseInterface* _ex)

TRUE if this object is remote, false if local

4.8.34

sidl_bool
bHYPRE_SStructParCSRVector__isLocal (bHYPRE_SStructParCSRVector
self, sidl_BaseInterface* _ex)

TRUE if this object is remote, false if local

4.8.35

struct bHYPRE_SStructParCSRVector__object*
bHYPRE_SStructParCSRVector__rmicast (void* obj, struct
sidl_BaseInterface__object** _ex)

Cast method for interface and class type conversions

4.8.36

struct bHYPRE_SStructParCSRVector__object*
bHYPRE_SStructParCSRVector__connectI (const char* url, sidl_bool ar,
struct sidl_BaseInterface__object** _ex)

RMI connector function for the class. (no addref)

5

Solver Interface

Names

5.1

struct **bHYPRE_Solver__object**

Symbol "bHYPRESolver" (version 100)

5.2

bHYPRE_Solver
bHYPRE_Solver__connect (const char* , sidl_BaseInterface* _ex)

RMI connector function for the class(addrefs)

5.3

SIDL_C_INLINE_DECL int32_t
bHYPRE_Solver_SetOperator (bHYPRE_Solver self, bHYPRE_Operator A, sidl_BaseInterface* _ex)

Set the operator for the linear system being solved. DEPRECATED. use Create

5.4

SIDL_C_INLINE_DECL int32_t
bHYPRE_Solver_SetTolerance (bHYPRE_Solver self, double tolerance, sidl_BaseInterface* _ex)

(Optional) Set the convergence tolerance. DEPRECATED. use SetDoubleParameter

5.5

SIDL_C_INLINE_DECL int32_t
bHYPRE_Solver_SetMaxIterations (bHYPRE_Solver self, int32_t max_iterations, sidl_BaseInterface* _ex)

(Optional) Set maximum number of iterations. DEPRECATED use SetIntParameter

5.6

SIDL_C_INLINE_DECL int32_t
bHYPRE_Solver_SetLogging (bHYPRE_Solver self, int32_t level, sidl_BaseInterface* _ex)

(Optional) Set the *logging level*, specifying the degree of additional informational data to be accumulated. Does nothing by default (level = 0). Other levels (if any) are implementation-specific. Must be called before `Setup` and `Apply`. DEPRECATED use SetIntParameter

5.7

SIDL_C_INLINE_DECL int32_t
bHYPRE_Solver_SetPrintLevel (bHYPRE_Solver self, int32_t level, sidl_BaseInterface* _ex)

(Optional) Set the *print level*, specifying the degree of informational data to be printed either to the screen or to a file. Does nothing by default (level=0). Other levels (if any) are implementation-specific. Must be called before `Setup` and `Apply`. DEPRECATED use SetIntParameter

5.8

SIDL_C_INLINE_DECL int32_t
bHYPRE_Solver_GetNumIterations (bHYPRE_Solver self, int32_t* num_iterations, sidl_BaseInterface* _ex)

(Optional) Return the number of iterations taken

5.9

SIDL_C_INLINE_DECL int32_t
bHYPRE_Solver_GetRelResidualNorm (bHYPRE_Solver self, double* norm, sidl_BaseInterface* _ex)

(Optional) Return the norm of the relative residual

5.10

struct bHYPRE_Solver__object*
bHYPRE_Solver__cast (void* obj, sidl_BaseInterface* _ex)

Cast method for interface and class type conversions

5.11

void*
bHYPRE_Solver__cast2 (void* obj, const char* type, sidl_BaseInterface* _ex)

String cast method for interface and class type conversions

5.12

SIDL_C_INLINE_DECL void
bHYPRE_Solver__exec (bHYPRE_Solver self, const char* methodName,
sidl_rmi_Call inArgs, sidl_rmi_Return outArgs, sidl_BaseInterface* _ex)

Select and execute a method by name

5.13

SIDL_C_INLINE_DECL char*
bHYPRE_Solver__getURL (bHYPRE_Solver self, sidl_BaseInterface* _ex)

Get the URL of the Implementation of this object (for RMI)

5.14

SIDL_C_INLINE_DECL void
bHYPRE_Solver__raddRef (bHYPRE_Solver self, sidl_BaseInterface* _ex)

On a remote object, addrefs the remote instance

5.15

SIDL_C_INLINE_DECL sidl_bool
bHYPRE_Solver__isRemote (bHYPRE_Solver self, sidl_BaseInterface* _ex)

TRUE if this object is remote, false if local

5.16

sidl_bool
bHYPRE_Solver__isLocal (bHYPRE_Solver self, sidl_BaseInterface* _ex)

TRUE if this object is remote, false if local

5.17

struct bHYPRE_Solver__object*
bHYPRE_Solver__rmicast (void* obj, struct sidl_BaseInterface__object** eex)

Cast method for interface and class type conversions

5.18

struct bHYPRE_Solver__object*
bHYPRE_Solver__connectI (const char* url, sidl_bool ar, struct sidl_BaseInterface__object** _ex)

RMI connector function for the class. (no addref)

5.19

Identity Solver (does nothing)

Names

5.19.1

struct **bHYPRE_IdentitySolver__object**

Symbol "bHYPREIdentitySolver" (version 100)

Identity solver, just solves an identity matrix, for when you don't really want a preconditioner

Objects of this type can be cast to Solver objects using the `__cast` methods.

5.19.2

struct bHYPRE_IdentitySolver__object*
bHYPRE_IdentitySolver__create (sidl_BaseInterface* _ex)

Constructor function for the class

5.19.3

bHYPRE_IdentitySolver
bHYPRE_IdentitySolver__createRemote (const char* url, sidl_BaseInterface* _ex)

RMI constructor function for the class

5.19.4

bHYPRE_IdentitySolver
bHYPRE_IdentitySolver__wrapObj (void* data, sidl_BaseInterface* _ex)

Wraps up the private data struct pointer (struct bHYPRE_IdentitySolver__data) passed in rather than running the constructor

5.19.5

bHYPRE_IdentitySolver
bHYPRE_IdentitySolver__connect (const char* , sidl_BaseInterface* _ex)

RMI connector function for the class(addrefs)

5.19.6

bHYPRE_IdentitySolver
bHYPRE_IdentitySolver_Create (bHYPRE_MPICommunicator mpi_comm, sidl_BaseInterface* _ex)

This function is the preferred way to create an Identity (null) solver.

5.19.7

SIDL_C_INLINE_DECL int32_t
bHYPRE_IdentitySolver_SetOperator (bHYPRE_IdentitySolver self, bHYPRE_Operator A, sidl_BaseInterface* _ex)

Set the operator for the linear system being solved. DEPRECATED. use Create

5.19.8

SIDL_C_INLINE_DECL int32_t
bHYPRE_IdentitySolver_SetTolerance (bHYPRE_IdentitySolver self, double tolerance, sidl_BaseInterface* _ex)

(Optional) Set the convergence tolerance. DEPRECATED. use SetDoubleParameter

5.19.9

SIDL_C_INLINE_DECL int32_t
bHYPRE_IdentitySolver_SetMaxIterations (bHYPRE_IdentitySolver self, int32_t max_iterations, sidl_BaseInterface* _ex)

(Optional) Set maximum number of iterations. DEPRECATED use SetIntParameter

5.19.10

SIDL_C_INLINE_DECL int32_t
bHYPRE_IdentitySolver_SetLogging (bHYPRE_IdentitySolver self, int32_t level, sidl_BaseInterface* _ex)

(Optional) Set the *logging level*, specifying the degree of additional informational data to be accumulated. Does nothing by default (level = 0). Other levels (if any) are implementation-specific. Must be called before `Setup` and `Apply`. DEPRECATED use SetIntParameter

5.19.11

SIDL_C_INLINE_DECL int32_t
bHYPRE_IdentitySolver_SetPrintLevel (bHYPRE_IdentitySolver self, int32_t level, sidl_BaseInterface* _ex)

(Optional) Set the *print level*, specifying the degree of informational data to be printed either to the screen or to a file. Does nothing by default (level=0). Other levels (if any) are implementation-specific. Must be called before `Setup` and `Apply`. DEPRECATED use SetIntParameter

5.19.12

SIDL_C_INLINE_DECL int32_t
bHYPRE_IdentitySolver_GetNumIterations (bHYPRE_IdentitySolver self, int32_t* num_iterations, sidl_BaseInterface* _ex)

(Optional) Return the number of iterations taken

5.19.13

SIDL_C_INLINE_DECL int32_t
bHYPRE_IdentitySolver_GetRelResidualNorm (bHYPRE_IdentitySolver self, double* norm, sidl_BaseInterface* _ex)

(Optional) Return the norm of the relative residual

5.19.14

SIDL_C_INLINE_DECL int32_t
bHYPRE_IdentitySolver_SetCommunicator (bHYPRE_IdentitySolver self, bHYPRE_MPICommunicator mpi_comm, sidl_BaseInterface* _ex)

Set the MPI Communicator. DEPRECATED, use Create:

5.19.15

SIDL_C_INLINE_DECL void
bHYPRE_IdentitySolver_Destroy (bHYPRE_IdentitySolver self,
sidl_BaseInterface* _ex)

The Destroy function doesn't necessarily destroy anything. It is just another name for deleteRef. Thus it decrements the object's reference count. The Babel memory management system will destroy the object if the reference count goes to zero.

5.19.16

SIDL_C_INLINE_DECL int32_t
bHYPRE_IdentitySolver_SetIntParameter (bHYPRE_IdentitySolver self,
const char* name, int32_t value, sidl_BaseInterface* _ex)

Set the int parameter associated with `name`

5.19.17

SIDL_C_INLINE_DECL int32_t
bHYPRE_IdentitySolver_SetDoubleParameter (bHYPRE_IdentitySolver
self, const char* name, double value, sidl_BaseInterface* _ex)

Set the double parameter associated with `name`

5.19.18

SIDL_C_INLINE_DECL int32_t
bHYPRE_IdentitySolver_SetStringParameter (bHYPRE_IdentitySolver self,
const char* name, const char* value, sidl_BaseInterface* _ex)

Set the string parameter associated with `name`

5.19.19

SIDL_C_INLINE_DECL int32_t
bHYPRE_IdentitySolver_SetIntArray1Parameter (bHYPRE_IdentitySolver self, const char* name, int32_t* value, int32_t nvalues, sidl_BaseInterface* _ex)

Set the int 1-D array parameter associated with `name`

5.19.20

SIDL_C_INLINE_DECL int32_t
bHYPRE_IdentitySolver_SetIntArray2Parameter (bHYPRE_IdentitySolver self, const char* name, struct sidl_int__array* value, sidl_BaseInterface* _ex)

Set the int 2-D array parameter associated with `name`

5.19.21

SIDL_C_INLINE_DECL int32_t
bHYPRE_IdentitySolver_SetDoubleArray1Parameter (
bHYPRE_IdentitySolver self, const char* name, double* value, int32_t nvalues, sidl_BaseInterface* _ex)

Set the double 1-D array parameter associated with `name`

5.19.22

SIDL_C_INLINE_DECL int32_t
bHYPRE_IdentitySolver_SetDoubleArray2Parameter (
bHYPRE_IdentitySolver self, const char* name, struct sidl_double__array* value, sidl_BaseInterface* _ex)

Set the double 2-D array parameter associated with `name`

5.19.23

SIDL_C_INLINE_DECL int32_t
bHYPRE_IdentitySolver_GetIntValue (bHYPRE_IdentitySolver self, const char* name, int32_t* value, sidl_BaseInterface* _ex)

Set the int parameter associated with `name`

5.19.24

SIDL_C_INLINE_DECL int32_t
bHYPRE_IdentitySolver_GetDoubleValue (bHYPRE_IdentitySolver self, const char* name, double* value, sidl_BaseInterface* _ex)

Get the double parameter associated with `name`

5.19.25

SIDL_C_INLINE_DECL int32_t
bHYPRE_IdentitySolver_Setup (bHYPRE_IdentitySolver self, bHYPRE_Vector b, bHYPRE_Vector x, sidl_BaseInterface* _ex)

(Optional) Do any preprocessing that may be necessary in order to execute `Apply`

5.19.26

SIDL_C_INLINE_DECL int32_t
bHYPRE_IdentitySolver_Apply (bHYPRE_IdentitySolver self, bHYPRE_Vector b, bHYPRE_Vector* x, sidl_BaseInterface* _ex)

Apply the operator to `b`, returning `x`

5.19.27

SIDL_C_INLINE_DECL int32_t
bHYPRE_IdentitySolver_ApplyAdjoint (bHYPRE_IdentitySolver self, bHYPRE_Vector b, bHYPRE_Vector* x, sidl_BaseInterface* _ex)

Apply the adjoint of the operator to `b`, returning `x`

5.19.28

struct bHYPRE_IdentitySolver__object*
bHYPRE_IdentitySolver__cast (void* obj, sidl_BaseInterface* _ex)

Cast method for interface and class type conversions

5.19.29

void*
bHYPRE_IdentitySolver__cast2 (void* obj, const char* type, sidl_BaseInterface* _ex)

String cast method for interface and class type conversions

5.19.30

SIDL_C_INLINE_DECL void
bHYPRE_IdentitySolver__exec (bHYPRE_IdentitySolver self, const char* methodName, sidl_rmi_Call inArgs, sidl_rmi_Return outArgs, sidl_BaseInterface* _ex)

Select and execute a method by name

5.19.31

SIDL_C_INLINE_DECL char*
bHYPRE_IdentitySolver__getURL (bHYPRE_IdentitySolver self,
sidl_BaseInterface* _ex)

Get the URL of the Implementation of this object (for RMI)

5.19.32

SIDL_C_INLINE_DECL void
bHYPRE_IdentitySolver__raddRef (bHYPRE_IdentitySolver self,
sidl_BaseInterface* _ex)

On a remote object, addrefs the remote instance

5.19.33

SIDL_C_INLINE_DECL sidl_bool
bHYPRE_IdentitySolver__isRemote (bHYPRE_IdentitySolver self,
sidl_BaseInterface* _ex)

TRUE if this object is remote, false if local

5.19.34

sidl_bool
bHYPRE_IdentitySolver__isLocal (bHYPRE_IdentitySolver self,
sidl_BaseInterface* _ex)

TRUE if this object is remote, false if local

5.19.35

struct bHYPRE_IdentitySolver__object*
bHYPRE_IdentitySolver__rmicast (void* obj, struct sidl_BaseInterface__object** _ex)

Cast method for interface and class type conversions

5.19.36

struct bHYPRE_IdentitySolver__object*
bHYPRE_IdentitySolver__connectI (const char* url, sidl_bool ar, struct sidl_BaseInterface__object** _ex)

RMI connector function for the class. (no addref)

5.20

Hybrid Solver

Names

5.20.1

struct **bHYPRE_Hybrid__object**

Symbol "bHYPREHybrid" (version 100)

Hybrid solver first tries to solve with the specified Krylov solver, preconditioned by If that fails to converge, it will try again with the user-specified

Specify the preconditioner by calling SecondSolver's SetPreconditioner method. If no preconditioner is specified (equivalently, if the preconditioner for SecondSolver is IdentitySolver), the preconditioner for the second try will be one of the following defaults. StructMatrix: SMG. other matrix types: not implemented

The Hybrid solver's Setup method will call Setup on KrylovSolver, so the user should not call Setup on KrylovSolver.

5.20.2

struct bHYPRE_Hybrid__object*
bHYPRE_Hybrid__create (sidl_BaseInterface* _ex)

Constructor function for the class

5.20.3

bHYPRE_Hybrid
bHYPRE_Hybrid__createRemote (const char* url, sidl_BaseInterface* _ex)

RMI constructor function for the class

5.20.4

bHYPRE_Hybrid
bHYPRE_Hybrid__wrapObj (void* data, sidl_BaseInterface* _ex)

Wraps up the private data struct pointer (struct bHYPRE_Hybrid__data) passed in rather than running the constructor

5.20.5

bHYPRE_Hybrid
bHYPRE_Hybrid__connect (const char* , sidl_BaseInterface* _ex)

RMI connector function for the class(addrefs)

5.20.6

bHYPRE_Hybrid
bHYPRE_Hybrid_Create (bHYPRE_MPICommunicator mpi_comm,
bHYPRE_PreconditionedSolver SecondSolver, bHYPRE_Operator A,
sidl_BaseInterface* _ex)

This function is the preferred way to create a Hybrid solver.

5.20.7

SIDL_C_INLINE_DECL int32_t
bHYPRE_Hybrid_GetFirstSolver (bHYPRE_Hybrid self,
bHYPRE_PreconditionedSolver* FirstSolver, sidl_BaseInterface* _ex)

Method: GetFirstSolver[]

5.20.8

SIDL_C_INLINE_DECL int32_t
bHYPRE_Hybrid_GetSecondSolver (bHYPRE_Hybrid self,
bHYPRE_PreconditionedSolver* SecondSolver, sidl_BaseInterface* _ex)

Method: GetSecondSolver[]

5.20.9

SIDL_C_INLINE_DECL int32_t
bHYPRE_Hybrid_SetOperator (bHYPRE_Hybrid self, bHYPRE_Operator A,
sidl_BaseInterface* _ex)

Set the operator for the linear system being solved. DEPRECATED. use Create

5.20.10

SIDL_C_INLINE_DECL int32_t
bHYPRE_Hybrid_SetTolerance (bHYPRE_Hybrid self, double tolerance,
sidl_BaseInterface* _ex)

(Optional) Set the convergence tolerance. DEPRECATED. use SetDoubleParameter

5.20.11

SIDL_C_INLINE_DECL int32_t
bHYPRE_Hybrid_SetMaxIterations (bHYPRE_Hybrid self, int32_t
max_iterations, sidl_BaseInterface* _ex)

(Optional) Set maximum number of iterations. DEPRECATED use SetIntParameter

5.20.12

SIDL_C_INLINE_DECL int32_t
bHYPRE_Hybrid_SetLogging (bHYPRE_Hybrid self, int32_t level, sidl_BaseInterface* _ex)

(Optional) Set the *logging level*, specifying the degree of additional informational data to be accumulated. Does nothing by default (level = 0). Other levels (if any) are implementation-specific. Must be called before `Setup` and `Apply`. DEPRECATED use SetIntParameter

5.20.13

SIDL_C_INLINE_DECL int32_t
bHYPRE_Hybrid_SetPrintLevel (bHYPRE_Hybrid self, int32_t level, sidl_BaseInterface* _ex)

(Optional) Set the *print level*, specifying the degree of informational data to be printed either to the screen or to a file. Does nothing by default (level=0). Other levels (if any) are implementation-specific. Must be called before `Setup` and `Apply`. DEPRECATED use SetIntParameter

5.20.14

SIDL_C_INLINE_DECL int32_t
bHYPRE_Hybrid_GetNumIterations (bHYPRE_Hybrid self, int32_t* num_iterations, sidl_BaseInterface* _ex)

(Optional) Return the number of iterations taken

5.20.15

SIDL_C_INLINE_DECL int32_t
bHYPRE_Hybrid_GetRelResidualNorm (bHYPRE_Hybrid self, double* norm, sidl_BaseInterface* _ex)

(Optional) Return the norm of the relative residual

5.20.16

SIDL_C_INLINE_DECL int32_t
bHYPRE_Hybrid_SetCommunicator (bHYPRE_Hybrid self,
bHYPRE_MPICommunicator mpi_comm, sidl_BaseInterface* _ex)

Set the MPI Communicator. DEPRECATED, use Create:

5.20.17

SIDL_C_INLINE_DECL void
bHYPRE_Hybrid_Destroy (bHYPRE_Hybrid self, sidl_BaseInterface* _ex)

The Destroy function doesn't necessarily destroy anything. It is just another name for deleteRef. Thus it decrements the object's reference count. The Babel memory management system will destroy the object if the reference count goes to zero.

5.20.18

SIDL_C_INLINE_DECL int32_t
bHYPRE_Hybrid_SetIntParameter (bHYPRE_Hybrid self, const char*
name, int32_t value, sidl_BaseInterface* _ex)

Set the int parameter associated with `name`

5.20.19

SIDL_C_INLINE_DECL int32_t
bHYPRE_Hybrid_SetDoubleParameter (bHYPRE_Hybrid self, const char*
name, double value, sidl_BaseInterface* _ex)

Set the double parameter associated with `name`

5.20.20

SIDL_C_INLINE_DECL int32_t
bHYPRE_Hybrid_SetStringParameter (bHYPRE_Hybrid self, const char* name, const char* value, sidl_BaseInterface* _ex)

Set the string parameter associated with `name`

5.20.21

SIDL_C_INLINE_DECL int32_t
bHYPRE_Hybrid_SetIntArray1Parameter (bHYPRE_Hybrid self, const char* name, int32_t* value, int32_t nvalues, sidl_BaseInterface* _ex)

Set the int 1-D array parameter associated with `name`

5.20.22

SIDL_C_INLINE_DECL int32_t
bHYPRE_Hybrid_SetIntArray2Parameter (bHYPRE_Hybrid self, const char* name, struct sidl_int__array* value, sidl_BaseInterface* _ex)

Set the int 2-D array parameter associated with `name`

5.20.23

SIDL_C_INLINE_DECL int32_t
bHYPRE_Hybrid_SetDoubleArray1Parameter (bHYPRE_Hybrid self, const char* name, double* value, int32_t nvalues, sidl_BaseInterface* _ex)

Set the double 1-D array parameter associated with `name`

5.20.24

SIDL_C_INLINE_DECL int32_t
bHYPRE_Hybrid_SetDoubleArray2Parameter (bHYPRE_Hybrid self, const char* name, struct sidl_double__array* value, sidl_BaseInterface* _ex)

Set the double 2-D array parameter associated with `name`

5.20.25

SIDL_C_INLINE_DECL int32_t
bHYPRE_Hybrid_GetIntValue (bHYPRE_Hybrid self, const char* name, int32_t* value, sidl_BaseInterface* _ex)

Set the int parameter associated with `name`

5.20.26

SIDL_C_INLINE_DECL int32_t
bHYPRE_Hybrid_GetDoubleValue (bHYPRE_Hybrid self, const char* name, double* value, sidl_BaseInterface* _ex)

Get the double parameter associated with `name`

5.20.27

SIDL_C_INLINE_DECL int32_t
bHYPRE_Hybrid_Setup (bHYPRE_Hybrid self, bHYPRE_Vector b, bHYPRE_Vector x, sidl_BaseInterface* _ex)

(Optional) Do any preprocessing that may be necessary in order to execute `Apply`

5.20.28

SIDL_C_INLINE_DECL int32_t
bHYPRE_Hybrid_Apply (bHYPRE_Hybrid self, bHYPRE_Vector b, bHYPRE_Vector* x, sidl_BaseInterface* _ex)

Apply the operator to `b`, returning `x`

5.20.29

SIDL_C_INLINE_DECL int32_t
bHYPRE_Hybrid_ApplyAdjoint (bHYPRE_Hybrid self, bHYPRE_Vector b, bHYPRE_Vector* x, sidl_BaseInterface* _ex)

Apply the adjoint of the operator to `b`, returning `x`

5.20.30

struct bHYPRE_Hybrid__object*
bHYPRE_Hybrid__cast (void* obj, sidl_BaseInterface* _ex)

Cast method for interface and class type conversions

5.20.31

void*
bHYPRE_Hybrid__cast2 (void* obj, const char* type, sidl_BaseInterface* _ex)

String cast method for interface and class type conversions

5.20.32

SIDL_C_INLINE_DECL void
bHYPRE_Hybrid__exec (bHYPRE_Hybrid self, const char* methodName,
sidl_rmi_Call inArgs, sidl_rmi_Return outArgs, sidl_BaseInterface* _ex)

Select and execute a method by name

5.20.33

SIDL_C_INLINE_DECL char*
bHYPRE_Hybrid__getURL (bHYPRE_Hybrid self, sidl_BaseInterface* _ex)

Get the URL of the Implementation of this object (for RMI)

5.20.34

SIDL_C_INLINE_DECL void
bHYPRE_Hybrid__raddRef (bHYPRE_Hybrid self, sidl_BaseInterface* _ex)

On a remote object, addrefs the remote instance

5.20.35

SIDL_C_INLINE_DECL sidl_bool
bHYPRE_Hybrid__isRemote (bHYPRE_Hybrid self, sidl_BaseInterface* _ex)

TRUE if this object is remote, false if local

5.20.36

sidl_bool
bHYPRE_Hybrid__isLocal (bHYPRE_Hybrid self, sidl_BaseInterface* _ex)

TRUE if this object is remote, false if local

5.20.37

struct bHYPRE_Hybrid__object*
bHYPRE_Hybrid__rmicast (void* obj, struct sidl_BaseInterface__object** _ex)

Cast method for interface and class type conversions

5.20.38

struct bHYPRE_Hybrid__object*
bHYPRE_Hybrid__connectI (const char* url, sidl_bool ar, struct sidl_BaseInterface__object** _ex)

RMI connector function for the class. (no addref)

6

ParCSR Matrix Solvers

Names

These solvers use matrix/vector storage schemes that are tailored for general sparse matrix systems.

6.1

ParCSRDiagScale Solver

Names

6.1.1

struct **bHYPRE_ParCSRDiagScale__object**

Symbol "bHYPREParCSRDiagScale" (version 100)

Diagonal scaling preconditioner for ParCSR matrix class.

Objects of this type can be cast to Solver objects using the `__cast` methods.

6.1.2

struct bHYPRE_ParCSRDiagScale__object*
bHYPRE_ParCSRDiagScale__create (sidl_BaseInterface* _ex)

Constructor function for the class

6.1.3

bHYPRE_ParCSRDiagScale
bHYPRE_ParCSRDiagScale__createRemote (const char* url,
sidl_BaseInterface* _ex)

RMI constructor function for the class

6.1.4

bHYPRE_ParCSRDiagScale
bHYPRE_ParCSRDiagScale__wrapObj (void* data, sidl_BaseInterface* _ex)

Wraps up the private data struct pointer (struct bHYPRE_ParCSRDiagScale__data) passed in rather than running the constructor

6.1.5

bHYPRE_ParCSRDiagScale
bHYPRE_ParCSRDiagScale__connect (const char* , sidl_BaseInterface* _ex)

RMI connector function for the class(addrefs)

6.1.6

bHYPRE_ParCSRDiagScale
bHYPRE_ParCSRDiagScale_Create (bHYPRE_MPICommunicator mpi_comm, bHYPRE_IJParCSRMatrix A, sidl_BaseInterface* _ex)

This function is the preferred way to create a ParCSR DiagScale solver.

6.1.7

SIDL_C_INLINE_DECL int32_t
bHYPRE_ParCSRDiagScale_SetOperator (bHYPRE_ParCSRDiagScale self, bHYPRE_Operator A, sidl_BaseInterface* _ex)

Set the operator for the linear system being solved. DEPRECATED. use Create

6.1.8

SIDL_C_INLINE_DECL int32_t
bHYPRE_ParCSRDiagScale_SetTolerance (bHYPRE_ParCSRDiagScale self, double tolerance, sidl_BaseInterface* _ex)

(Optional) Set the convergence tolerance. DEPRECATED. use SetDoubleParameter

6.1.9

SIDL_C_INLINE_DECL int32_t
bHYPRE_ParCSRDiagScale_SetMaxIterations (
bHYPRE_ParCSRDiagScale self, int32_t max_iterations, sidl_BaseInterface* _ex)

(Optional) Set maximum number of iterations. DEPRECATED use SetIntParameter

6.1.10

SIDL_C_INLINE_DECL int32_t
bHYPRE_ParCSRDiagScale_SetLogging (bHYPRE_ParCSRDiagScale self, int32_t level, sidl_BaseInterface* _ex)

(Optional) Set the *logging level*, specifying the degree of additional informational data to be accumulated. Does nothing by default (level = 0). Other levels (if any) are implementation-specific. Must be called before `Setup` and `Apply`. DEPRECATED use SetIntParameter

6.1.11

SIDL_C_INLINE_DECL int32_t
bHYPRE_ParCSRDiagScale_SetPrintLevel (bHYPRE_ParCSRDiagScale self, int32_t level, sidl_BaseInterface* _ex)

(Optional) Set the *print level*, specifying the degree of informational data to be printed either to the screen or to a file. Does nothing by default (level=0). Other levels (if any) are implementation-specific. Must be called before `Setup` and `Apply`. DEPRECATED use SetIntParameter

6.1.12

SIDL_C_INLINE_DECL int32_t
bHYPRE_ParCSRDiagScale_GetNumIterations (
bHYPRE_ParCSRDiagScale self, int32_t* num_iterations, sidl_BaseInterface* _ex)

(Optional) Return the number of iterations taken

6.1.13

SIDL_C_INLINE_DECL int32_t
bHYPRE_ParCSRDiagScale_GetRelResidualNorm (
bHYPRE_ParCSRDiagScale self, double* norm, sidl_BaseInterface* _ex)

(Optional) Return the norm of the relative residual

6.1.14

SIDL_C_INLINE_DECL int32_t
bHYPRE_ParCSRDiagScale_SetCommunicator (
bHYPRE_ParCSRDiagScale self, bHYPRE_MPICommunicator mpi_comm,
sidl_BaseInterface* _ex)

Set the MPI Communicator. DEPRECATED, use Create:

6.1.15

SIDL_C_INLINE_DECL void
bHYPRE_ParCSRDiagScale_Destroy (bHYPRE_ParCSRDiagScale self,
sidl_BaseInterface* _ex)

The Destroy function doesn't necessarily destroy anything. It is just another name for deleteRef. Thus it decrements the object's reference count. The Babel memory management system will destroy the object if the reference count goes to zero.

6.1.16

SIDL_C_INLINE_DECL int32_t
bHYPRE_ParCSRDiagScale_SetIntParameter (bHYPRE_ParCSRDiagScale self, const char* name, int32_t value, sidl_BaseInterface* _ex)

Set the int parameter associated with `name`

6.1.17

SIDL_C_INLINE_DECL int32_t
bHYPRE_ParCSRDiagScale_SetDoubleParameter (
bHYPRE_ParCSRDiagScale self, const char* name, double value,
sidl_BaseInterface* _ex)

Set the double parameter associated with `name`

6.1.18

SIDL_C_INLINE_DECL int32_t
bHYPRE_ParCSRDiagScale_SetStringParameter (
bHYPRE_ParCSRDiagScale self, const char* name, const char* value,
sidl_BaseInterface* _ex)

Set the string parameter associated with `name`

6.1.19

SIDL_C_INLINE_DECL int32_t
bHYPRE_ParCSRDiagScale_SetIntArray1Parameter (
bHYPRE_ParCSRDiagScale self, const char* name, int32_t* value, int32_t
nvalues, sidl_BaseInterface* _ex)

Set the int 1-D array parameter associated with `name`

6.1.20

SIDL_C_INLINE_DECL int32_t
bHYPRE_ParCSRDiagScale_SetIntArray2Parameter (
bHYPRE_ParCSRDiagScale self, const char* name, struct sidl_int__array* value,
sidl_BaseInterface* _ex)

Set the int 2-D array parameter associated with `name`

6.1.21

SIDL_C_INLINE_DECL int32_t
bHYPRE_ParCSRDiagScale_SetDoubleArray1Parameter (
bHYPRE_ParCSRDiagScale self, const char* name, double* value, int32_t
nvalues, sidl_BaseInterface* _ex)

Set the double 1-D array parameter associated with `name`

6.1.22

SIDL_C_INLINE_DECL int32_t
bHYPRE_ParCSRDiagScale_SetDoubleArray2Parameter (
bHYPRE_ParCSRDiagScale self, const char* name, struct sidl_double__array*
value, sidl_BaseInterface* _ex)

Set the double 2-D array parameter associated with `name`

6.1.23

SIDL_C_INLINE_DECL int32_t
bHYPRE_ParCSRDiagScale_GetIntValue (bHYPRE_ParCSRDiagScale self,
const char* name, int32_t* value, sidl_BaseInterface* _ex)

Set the int parameter associated with `name`

6.1.24

SIDL_C_INLINE_DECL int32_t
bHYPRE_ParCSRDiagScale_GetDoubleValue (bHYPRE_ParCSRDiagScale self, const char* name, double* value, sidl_BaseInterface* _ex)

Get the double parameter associated with `name`

6.1.25

SIDL_C_INLINE_DECL int32_t
bHYPRE_ParCSRDiagScale_Setup (bHYPRE_ParCSRDiagScale self, bHYPRE_Vector b, bHYPRE_Vector x, sidl_BaseInterface* _ex)

(Optional) Do any preprocessing that may be necessary in order to execute `Apply`

6.1.26

SIDL_C_INLINE_DECL int32_t
bHYPRE_ParCSRDiagScale_Apply (bHYPRE_ParCSRDiagScale self, bHYPRE_Vector b, bHYPRE_Vector* x, sidl_BaseInterface* _ex)

Apply the operator to `b`, returning `x`

6.1.27

SIDL_C_INLINE_DECL int32_t
bHYPRE_ParCSRDiagScale_ApplyAdjoint (bHYPRE_ParCSRDiagScale self, bHYPRE_Vector b, bHYPRE_Vector* x, sidl_BaseInterface* _ex)

Apply the adjoint of the operator to `b`, returning `x`

6.1.28

struct bHYPRE_ParCSRDiagScale__object*
bHYPRE_ParCSRDiagScale__cast (void* obj, sidl_BaseInterface* _ex)

Cast method for interface and class type conversions

6.1.29

void*
bHYPRE_ParCSRDiagScale__cast2 (void* obj, const char* type,
sidl_BaseInterface* _ex)

String cast method for interface and class type conversions

6.1.30

SIDL_C_INLINE_DECL void
bHYPRE_ParCSRDiagScale__exec (bHYPRE_ParCSRDiagScale self, const char* methodName, sidl_rmi_Call inArgs, sidl_rmi_Return outArgs,
sidl_BaseInterface* _ex)

Select and execute a method by name

6.1.31

SIDL_C_INLINE_DECL char*
bHYPRE_ParCSRDiagScale__getURL (bHYPRE_ParCSRDiagScale self,
sidl_BaseInterface* _ex)

Get the URL of the Implementation of this object (for RMI)

6.1.32

SIDL_C_INLINE_DECL void
bHYPRE_ParCSRDiagScale__raddRef (bHYPRE_ParCSRDiagScale self,
sidl_BaseInterface* _ex)

On a remote object, addrefs the remote instance

6.1.33

SIDL_C_INLINE_DECL sidl_bool
bHYPRE_ParCSRDiagScale__isRemote (bHYPRE_ParCSRDiagScale self,
sidl_BaseInterface* _ex)

TRUE if this object is remote, false if local

6.1.34

sidl_bool
bHYPRE_ParCSRDiagScale__isLocal (bHYPRE_ParCSRDiagScale self,
sidl_BaseInterface* _ex)

TRUE if this object is remote, false if local

6.1.35

struct bHYPRE_ParCSRDiagScale__object*
bHYPRE_ParCSRDiagScale__rmicast (void* obj, struct
sidl_BaseInterface__object** _ex)

Cast method for interface and class type conversions

6.1.36

struct bHYPRE_ParCSRDiagScale__object*
bHYPRE_ParCSRDiagScale__connectI (const char* url, sidl_bool ar, struct sidl_BaseInterface__object** _ex)

RMI connector function for the class. (no addref)

6.2

ParCSR BoomerAMG Solver

Names

6.2.1

struct **bHYPRE_BoomerAMG__object**

Symbol "bHYPREBoomerAMG" (version 100)

Algebraic multigrid solver, based on classical Ruge-Stueben.

BoomerAMG requires an IJParCSR matrix

The following optional parameters are available and may be set using the appropriate `Parameter` function (as indicated in parentheses):

MaxLevels (`Int`) - maximum number of multigrid levels.

StrongThreshold (`Double`) - AMG strength threshold.

MaxRowSum (`Double`) -

CoarsenType (`Int`) - type of parallel coarsening algorithm used.

MeasureType (`Int`) - type of measure used; local or global.

CycleType (`Int`) - type of cycle used; a V-cycle (default) or a W-cycle.

NumGridSweeps (`IntArray 1D`) - number of sweeps for fine and coarse grid, up and down cycle. DEPRECATED: Use NumSweeps or Cycle?NumSweeps instead.

NumSweeps (`Int`) - number of sweeps for fine grid, up and down cycle.

Cycle1NumSweeps (`Int`) - number of sweeps for down cycle

Cycle2NumSweeps (`Int`) - number of sweeps for up cycle

Cycle3NumSweeps (`Int`) - number of sweeps for coarse grid

GridRelaxType (`IntArray 1D`) - type of smoother used on fine and coarse grid, up and down cycle. DEPRECATED: Use RelaxType or Cycle?RelaxType instead.

RelaxType (`Int`) - type of smoother for fine grid, up and down cycle.

Cycle1RelaxType (`Int`) - type of smoother for down cycle

Cycle2RelaxType (`Int`) - type of smoother for up cycle

Cycle3RelaxType (`Int`) - type of smoother for coarse grid

GridRelaxPoints (`IntArray 2D`) - point ordering used in relaxation. DEPRECATED.

RelaxWeight (`DoubleArray 1D`) - relaxation weight for smoothed Jacobi and hybrid SOR. DEPRECATED: Instead, use the RelaxWt parameter and the SetLevelRelaxWt function.

RelaxWt (`Int`) - relaxation weight for all levels for smoothed Jacobi and hybrid SOR.

TruncFactor (`Double`) - truncation factor for interpolation.

JacobiTruncThreshold (`Double`) - threshold for truncation of Jacobi interpolation.

SmoothType (`Int`) - more complex smoothers.

SmoothNumLevels (`Int`) - number of levels for more complex smoothers.

SmoothNumSweeps (`Int`) - number of sweeps for more complex smoothers.

PrintFileName (`String`) - name of file printed to in association with `SetPrintLevel`.

NumFunctions (`Int`) - size of the system of PDEs (when using the systems version).

DOFFunc (`IntArray 1D`) - mapping that assigns the function to each variable (when using the systems version).

Variant (`Int`) - variant of Schwarz used.

Overlap (`Int`) - overlap for Schwarz.

DomainType (`Int`) - type of domain used for Schwarz.

SchwarzRlxWeight (`Double`) - the smoothing parameter for additive Schwarz.

Tolerance (`Double`) - convergence tolerance, if this is used as a solver; ignored if this is used as a preconditioner

DebugFlag (`Int`) -

InterpType (`Int`) - Defines which parallel interpolation operator is used. There are the following options for interp_type:

0	classical modified interpolation
1	LS interpolation (for use with GSMG)
2	classical modified interpolation for hyperbolic PDEs
3	direct interpolation (with separation of weights)
4	multipass interpolation
5	multipass interpolation (with separation of weights)
6	extended classical modified interpolation
7	extended (if no common C neighbor) classical modified interpolation
8	standard interpolation
9	standard interpolation (with separation of weights)
10	classical block interpolation (for use with nodal systems version only)
11	classical block interpolation (for use with nodal systems version only) with diagonalized diagonal blocks
12	FF interpolation
13	FF1 interpolation

The default is 0.

NumSamples (`Int`) - Defines the number of sample vectors used in GSMG or LS interpolation.

MaxIterations (`Int`) - maximum number of iterations

Logging (`Int`) - Set the *logging level*, specifying the degree of additional informational data to be accumulated. Does nothing by default (level = 0). Other levels (if any) are implementation-specific. Must be called before `Setup` and `Apply`.

PrintLevel (`Int`) - Set the *print level*, specifying the degree of informational data to be printed either to the screen or to a file. Does nothing by default (level=0). Other levels (if any) are implementation-specific. Must be called before `Setup` and `Apply`.

The following function is specific to this class:

SetLevelRelxWeight (`Double , Int`) - relaxation weight for one specified level of smoothed Jacobi and hybrid SOR.

Objects of this type can be cast to Solver objects using the `__cast` methods.

6.2.2

struct bHYPRE_BoomerAMG__object*
bHYPRE_BoomerAMG__create (sidl_BaseInterface* _ex)

Constructor function for the class

6.2.3

bHYPRE_BoomerAMG
bHYPRE_BoomerAMG__createRemote (const char* url, sidl_BaseInterface* _ex)

RMI constructor function for the class

6.2.4

bHYPRE_BoomerAMG
bHYPRE_BoomerAMG__wrapObj (void* data, sidl_BaseInterface* _ex)

Wraps up the private data struct pointer (struct bHYPRE_BoomerAMG__data) passed in rather than running the constructor

6.2.5

bHYPRE_BoomerAMG
bHYPRE_BoomerAMG__connect (const char* , sidl_BaseInterface* _ex)

RMI connector function for the class(addrefs)

6.2.6

bHYPRE_BoomerAMG
bHYPRE_BoomerAMG_Create (bHYPRE_MPICommunicator mpi_comm, bHYPRE_IJParCSRMatrix A, sidl_BaseInterface* _ex)

This function is the preferred way to create a BoomerAMG solver.

6.2.7

SIDL_C_INLINE_DECL int32_t
bHYPRE_BoomerAMG_SetLevelRelaxWt (bHYPRE_BoomerAMG self, double relax_wt, int32_t level, sidl_BaseInterface* _ex)

Method: SetLevelRelaxWt[]

6.2.8

SIDL_C_INLINE_DECL int32_t
bHYPRE_BoomerAMG_InitGridRelaxation (bHYPRE_BoomerAMG self, struct sidl_int__array** num_grid_sweeps, struct sidl_int__array** grid_relax_type, struct sidl_int__array** grid_relax_points, int32_t coarsen_type, struct sidl_double__array** relax_weights, int32_t max_levels, sidl_BaseInterface* _ex)

Method: InitGridRelaxation[]

6.2.9

SIDL_C_INLINE_DECL int32_t
bHYPRE_BoomerAMG_SetOperator (bHYPRE_BoomerAMG self, bHYPRE_Operator A, sidl_BaseInterface* _ex)

Set the operator for the linear system being solved. DEPRECATED. use Create

6.2.10

SIDL_C_INLINE_DECL int32_t
bHYPRE_BoomerAMG_SetTolerance (bHYPRE_BoomerAMG self, double tolerance, sidl_BaseInterface* _ex)

(Optional) Set the convergence tolerance. DEPRECATED. use SetDoubleParameter

6.2.11

SIDL_C_INLINE_DECL int32_t
bHYPRE_BoomerAMG_SetMaxIterations (bHYPRE_BoomerAMG self, int32_t max_iterations, sidl_BaseInterface* _ex)

(Optional) Set maximum number of iterations. DEPRECATED use SetIntParameter

6.2.12

SIDL_C_INLINE_DECL int32_t
bHYPRE_BoomerAMG_SetLogging (bHYPRE_BoomerAMG self, int32_t level, sidl_BaseInterface* _ex)

(Optional) Set the *logging level*, specifying the degree of additional informational data to be accumulated. Does nothing by default (level = 0). Other levels (if any) are implementation-specific. Must be called before `Setup` and `Apply`. DEPRECATED use SetIntParameter

6.2.13

SIDL_C_INLINE_DECL int32_t
bHYPRE_BoomerAMG_SetPrintLevel (bHYPRE_BoomerAMG self, int32_t level, sidl_BaseInterface* _ex)

(Optional) Set the *print level*, specifying the degree of informational data to be printed either to the screen or to a file. Does nothing by default (level=0). Other levels (if any) are implementation-specific. Must be called before `Setup` and `Apply`. DEPRECATED use SetIntParameter

6.2.14

SIDL_C_INLINE_DECL int32_t
bHYPRE_BoomerAMG_GetNumIterations (bHYPRE_BoomerAMG self, int32_t* num_iterations, sidl_BaseInterface* _ex)

(Optional) Return the number of iterations taken

6.2.15

SIDL_C_INLINE_DECL int32_t
bHYPRE_BoomerAMG_GetRelResidualNorm (bHYPRE_BoomerAMG self, double* norm, sidl_BaseInterface* _ex)

(Optional) Return the norm of the relative residual

6.2.16

SIDL_C_INLINE_DECL int32_t
bHYPRE_BoomerAMG_SetCommunicator (bHYPRE_BoomerAMG self, bHYPRE_MPICommunicator mpi_comm, sidl_BaseInterface* _ex)

Set the MPI Communicator. DEPRECATED, use Create:

6.2.17

SIDL_C_INLINE_DECL void
bHYPRE_BoomerAMG_Destroy (bHYPRE_BoomerAMG self, sidl_BaseInterface* _ex)

The Destroy function doesn't necessarily destroy anything. It is just another name for deleteRef. Thus it decrements the object's reference count. The Babel memory management system will destroy the object if the reference count goes to zero.

6.2.18

SIDL_C_INLINE_DECL int32_t
bHYPRE_BoomerAMG_SetIntParameter (bHYPRE_BoomerAMG self, const char* name, int32_t value, sidl_BaseInterface* _ex)

Set the int parameter associated with `name`

6.2.19

SIDL_C_INLINE_DECL int32_t
bHYPRE_BoomerAMG_SetDoubleParameter (bHYPRE_BoomerAMG self, const char* name, double value, sidl_BaseInterface* _ex)

Set the double parameter associated with `name`

6.2.20

SIDL_C_INLINE_DECL int32_t
bHYPRE_BoomerAMG_SetStringParameter (bHYPRE_BoomerAMG self, const char* name, const char* value, sidl_BaseInterface* _ex)

Set the string parameter associated with `name`

6.2.21

SIDL_C_INLINE_DECL int32_t
bHYPRE_BoomerAMG_SetIntArray1Parameter (bHYPRE_BoomerAMG self, const char* name, int32_t* value, int32_t nvalues, sidl_BaseInterface* _ex)

Set the int 1-D array parameter associated with `name`

6.2.22

SIDL_C_INLINE_DECL int32_t
bHYPRE_BoomerAMG_SetIntArray2Parameter (bHYPRE_BoomerAMG self, const char* name, struct sidl_int__array* value, sidl_BaseInterface* _ex)

Set the int 2-D array parameter associated with `name`

6.2.23

SIDL_C_INLINE_DECL int32_t
bHYPRE_BoomerAMG_SetDoubleArray1Parameter (
bHYPRE_BoomerAMG self, const char* name, double* value, int32_t nvalues, sidl_BaseInterface* _ex)

Set the double 1-D array parameter associated with `name`

6.2.24

SIDL_C_INLINE_DECL int32_t
bHYPRE_BoomerAMG_SetDoubleArray2Parameter (
bHYPRE_BoomerAMG self, const char* name, struct sidl_double__array* value, sidl_BaseInterface* _ex)

Set the double 2-D array parameter associated with `name`

6.2.25

SIDL_C_INLINE_DECL int32_t
bHYPRE_BoomerAMG_GetIntValue (bHYPRE_BoomerAMG self, const char* name, int32_t* value, sidl_BaseInterface* _ex)

Set the int parameter associated with `name`

6.2.26

SIDL_C_INLINE_DECL int32_t
bHYPRE_BoomerAMG_GetDoubleValue (bHYPRE_BoomerAMG self,
const char* name, double* value, sidl_BaseInterface* _ex)

Get the double parameter associated with `name`

6.2.27

SIDL_C_INLINE_DECL int32_t
bHYPRE_BoomerAMG_Setup (bHYPRE_BoomerAMG self,
bHYPRE_Vector b, bHYPRE_Vector x, sidl_BaseInterface* _ex)

(Optional) Do any preprocessing that may be necessary in order to execute `Apply`

6.2.28

SIDL_C_INLINE_DECL int32_t
bHYPRE_BoomerAMG_Apply (bHYPRE_BoomerAMG self,
bHYPRE_Vector b, bHYPRE_Vector* x, sidl_BaseInterface* _ex)

Apply the operator to `b`, returning `x`

6.2.29

SIDL_C_INLINE_DECL int32_t
bHYPRE_BoomerAMG_ApplyAdjoint (bHYPRE_BoomerAMG self,
bHYPRE_Vector b, bHYPRE_Vector* x, sidl_BaseInterface* _ex)

Apply the adjoint of the operator to `b`, returning `x`

6.2.30

struct bHYPRE_BoomerAMG__object*
bHYPRE_BoomerAMG__cast (void* obj, sidl_BaseInterface* _ex)

Cast method for interface and class type conversions

6.2.31

void*
bHYPRE_BoomerAMG__cast2 (void* obj, const char* type,
sidl_BaseInterface* _ex)

String cast method for interface and class type conversions

6.2.32

SIDL_C_INLINE_DECL void
bHYPRE_BoomerAMG__exec (bHYPRE_BoomerAMG self, const char*
methodName, sidl_rmi_Call inArgs, sidl_rmi_Return outArgs, sidl_BaseInterface*
_ex)

Select and execute a method by name

6.2.33

SIDL_C_INLINE_DECL char*
bHYPRE_BoomerAMG__getURL (bHYPRE_BoomerAMG self,
sidl_BaseInterface* _ex)

Get the URL of the Implementation of this object (for RMI)

6.2.34

SIDL_C_INLINE_DECL void
bHYPRE_BoomerAMG__raddRef (bHYPRE_BoomerAMG self,
sidl_BaseInterface* _ex)

On a remote object, addrefs the remote instance

6.2.35

SIDL_C_INLINE_DECL sidl_bool
bHYPRE_BoomerAMG__isRemote (bHYPRE_BoomerAMG self,
sidl_BaseInterface* _ex)

TRUE if this object is remote, false if local

6.2.36

sidl_bool
bHYPRE_BoomerAMG__isLocal (bHYPRE_BoomerAMG self,
sidl_BaseInterface* _ex)

TRUE if this object is remote, false if local

6.2.37

struct bHYPRE_BoomerAMG__object*
bHYPRE_BoomerAMG__rmicast (void* obj, struct
sidl_BaseInterface__object** _ex)

Cast method for interface and class type conversions

6.2.38

struct bHYPRE_BoomerAMG__object*
bHYPRE_BoomerAMG__connectI (const char* url, sidl_bool ar, struct sidl_BaseInterface__object** _ex)

RMI connector function for the class. (no addref)

6.3

ParCSR Euclid Solver

Names

6.3.1

struct **bHYPRE_Euclid__object**

Symbol "bHYPREEuclid" (version 100)

Objects of this type can be cast to Solver objects using the `__cast` methods.

Although the usual Solver SetParameter functions are available, a Euclid-stype parameter-setting function is also available, SetParameters.

6.3.2

struct bHYPRE_Euclid__object*
bHYPRE_Euclid__create (sidl_BaseInterface* _ex)

Constructor function for the class

6.3.3

bHYPRE_Euclid
bHYPRE_Euclid__createRemote (const char* url, sidl_BaseInterface* _ex)

RMI constructor function for the class

6.3.4

bHYPRE_Euclid
bHYPRE_Euclid__wrapObj (void* data, sidl_BaseInterface* _ex)

Wraps up the private data struct pointer (struct bHYPRE_Euclid__data) passed in rather than running the constructor

6.3.5

bHYPRE_Euclid
bHYPRE_Euclid__connect (const char* , sidl_BaseInterface* _ex)

RMI connector function for the class(addrefs)

6.3.6

bHYPRE_Euclid
bHYPRE_Euclid_Create (bHYPRE_MPICommunicator mpi_comm, bHYPRE_IJParCSRMatrix A, sidl_BaseInterface* _ex)

This function is the preferred way to create a Euclid solver.

6.3.7

SIDL_C_INLINE_DECL int32_t
bHYPRE_Euclid_SetParameters (bHYPRE_Euclid self, int32_t argc, char** argv, sidl_BaseInterface* _ex)

Method: SetParameters[]

6.3.8

SIDL_C_INLINE_DECL int32_t
bHYPRE_Euclid_SetOperator (bHYPRE_Euclid self, bHYPRE_Operator A, sidl_BaseInterface* _ex)

Set the operator for the linear system being solved. DEPRECATED. use Create

6.3.9

SIDL_C_INLINE_DECL int32_t
bHYPRE_Euclid_SetTolerance (bHYPRE_Euclid self, double tolerance, sidl_BaseInterface* _ex)

(Optional) Set the convergence tolerance. DEPRECATED. use SetDoubleParameter

6.3.10

SIDL_C_INLINE_DECL int32_t
bHYPRE_Euclid_SetMaxIterations (bHYPRE_Euclid self, int32_t max_iterations, sidl_BaseInterface* _ex)

(Optional) Set maximum number of iterations. DEPRECATED use SetIntParameter

6.3.11

SIDL_C_INLINE_DECL int32_t
bHYPRE_Euclid_SetLogging (bHYPRE_Euclid self, int32_t level, sidl_BaseInterface* _ex)

(Optional) Set the *logging level*, specifying the degree of additional informational data to be accumulated. Does nothing by default (level = 0). Other levels (if any) are implementation-specific. Must be called before `Setup` and `Apply`. DEPRECATED use SetIntParameter

6.3.12

SIDL_C_INLINE_DECL int32_t
bHYPRE_Euclid_SetPrintLevel (bHYPRE_Euclid self, int32_t level, sidl_BaseInterface* _ex)

(Optional) Set the *print level*, specifying the degree of informational data to be printed either to the screen or to a file. Does nothing by default (level=0). Other levels (if any) are implementation-specific. Must be called before `Setup` and `Apply`. DEPRECATED use SetIntParameter

6.3.13

SIDL_C_INLINE_DECL int32_t
bHYPRE_Euclid_GetNumIterations (bHYPRE_Euclid self, int32_t* num_iterations, sidl_BaseInterface* _ex)

(Optional) Return the number of iterations taken

6.3.14

SIDL_C_INLINE_DECL int32_t
bHYPRE_Euclid_GetRelResidualNorm (bHYPRE_Euclid self, double* norm, sidl_BaseInterface* _ex)

(Optional) Return the norm of the relative residual

6.3.15

SIDL_C_INLINE_DECL int32_t
bHYPRE_Euclid_SetCommunicator (bHYPRE_Euclid self, bHYPRE_MPICommunicator mpi_comm, sidl_BaseInterface* _ex)

Set the MPI Communicator. DEPRECATED, use Create:

6.3.16

SIDL_C_INLINE_DECL void
bHYPRE_Euclid_Destroy (bHYPRE_Euclid self, sidl_BaseInterface* _ex)

The Destroy function doesn't necessarily destroy anything. It is just another name for deleteRef. Thus it decrements the object's reference count. The Babel memory management system will destroy the object if the reference count goes to zero.

6.3.17

SIDL_C_INLINE_DECL int32_t
bHYPRE_Euclid_SetIntParameter (bHYPRE_Euclid self, const char* name, int32_t value, sidl_BaseInterface* _ex)

Set the int parameter associated with `name`

6.3.18

SIDL_C_INLINE_DECL int32_t
bHYPRE_Euclid_SetDoubleParameter (bHYPRE_Euclid self, const char* name, double value, sidl_BaseInterface* _ex)

Set the double parameter associated with `name`

6.3.19

SIDL_C_INLINE_DECL int32_t
bHYPRE_Euclid_SetStringParameter (bHYPRE_Euclid self, const char* name, const char* value, sidl_BaseInterface* _ex)

Set the string parameter associated with `name`

6.3.20

SIDL_C_INLINE_DECL int32_t
bHYPRE_Euclid_SetIntArray1Parameter (bHYPRE_Euclid self, const char* name, int32_t* value, int32_t nvalues, sidl_BaseInterface* _ex)

Set the int 1-D array parameter associated with `name`

6.3.21

SIDL_C_INLINE_DECL int32_t
bHYPRE_Euclid_SetIntArray2Parameter (bHYPRE_Euclid self, const char* name, struct sidl_int__array* value, sidl_BaseInterface* _ex)

Set the int 2-D array parameter associated with `name`

6.3.22

SIDL_C_INLINE_DECL int32_t
bHYPRE_Euclid_SetDoubleArray1Parameter (bHYPRE_Euclid self, const char* name, double* value, int32_t nvalues, sidl_BaseInterface* _ex)

Set the double 1-D array parameter associated with `name`

6.3.23

SIDL_C_INLINE_DECL int32_t
bHYPRE_Euclid_SetDoubleArray2Parameter (bHYPRE_Euclid self, const char* name, struct sidl_double__array* value, sidl_BaseInterface* _ex)

Set the double 2-D array parameter associated with `name`

6.3.24

SIDL_C_INLINE_DECL int32_t
bHYPRE_Euclid_GetIntValue (bHYPRE_Euclid self, const char* name, int32_t* value, sidl_BaseInterface* _ex)

Set the int parameter associated with `name`

6.3.25

SIDL_C_INLINE_DECL int32_t
bHYPRE_Euclid_GetDoubleValue (bHYPRE_Euclid self, const char* name, double* value, sidl_BaseInterface* _ex)

Get the double parameter associated with `name`

6.3.26

SIDL_C_INLINE_DECL int32_t
bHYPRE_Euclid_Setup (bHYPRE_Euclid self, bHYPRE_Vector b,
bHYPRE_Vector x, sidl_BaseInterface* _ex)

(Optional) Do any preprocessing that may be necessary in order to execute `Apply`

6.3.27

SIDL_C_INLINE_DECL int32_t
bHYPRE_Euclid_Apply (bHYPRE_Euclid self, bHYPRE_Vector b,
bHYPRE_Vector* x, sidl_BaseInterface* _ex)

Apply the operator to `b`, returning `x`

6.3.28

SIDL_C_INLINE_DECL int32_t
bHYPRE_Euclid_ApplyAdjoint (bHYPRE_Euclid self, bHYPRE_Vector b,
bHYPRE_Vector* x, sidl_BaseInterface* _ex)

Apply the adjoint of the operator to `b`, returning `x`

6.3.29

struct bHYPRE_Euclid__object*
bHYPRE_Euclid__cast (void* obj, sidl_BaseInterface* _ex)

Cast method for interface and class type conversions

6.3.30

void*
bHYPRE_Euclid__cast2 (void* obj, const char* type, sidl_BaseInterface* _ex)

String cast method for interface and class type conversions

6.3.31

SIDL_C_INLINE_DECL void
bHYPRE_Euclid__exec (bHYPRE_Euclid self, const char* methodName,
sidl_rmi_Call inArgs, sidl_rmi_Return outArgs, sidl_BaseInterface* _ex)

Select and execute a method by name

6.3.32

SIDL_C_INLINE_DECL char*
bHYPRE_Euclid__getURL (bHYPRE_Euclid self, sidl_BaseInterface* _ex)

Get the URL of the Implementation of this object (for RMI)

6.3.33

SIDL_C_INLINE_DECL void
bHYPRE_Euclid__raddRef (bHYPRE_Euclid self, sidl_BaseInterface* _ex)

On a remote object, addrefs the remote instance

6.3.34

SIDL_C_INLINE_DECL sidl_bool
bHYPRE_Euclid__isRemote (bHYPRE_Euclid self, sidl_BaseInterface* _ex)

TRUE if this object is remote, false if local

6.3.35

sidl_bool
bHYPRE_Euclid__isLocal (bHYPRE_Euclid self, sidl_BaseInterface* _ex)

TRUE if this object is remote, false if local

6.3.36

struct bHYPRE_Euclid__object*
bHYPRE_Euclid__rmicast (void* obj, struct sidl_BaseInterface__object** _ex)

Cast method for interface and class type conversions

6.3.37

struct bHYPRE_Euclid__object*
bHYPRE_Euclid__connectI (const char* url, sidl_bool ar, struct sidl_BaseInterface__object** _ex)

RMI connector function for the class. (no addref)

6.4

ParCSR Schwarz Solver

Names

6.4.1

struct **bHYPRE_Schwarz__object**

Symbol "bHYPRESchwarz" (version 100)

Objects of this type can be cast to Solver objects using the `__cast` methods.

Schwarz requires an IJParCSR matrix

6.4.2

struct bHYPRE_Schwarz__object*
bHYPRE_Schwarz__create (sidl_BaseInterface* _ex)

Constructor function for the class

6.4.3

bHYPRE_Schwarz
bHYPRE_Schwarz__createRemote (const char* url, sidl_BaseInterface* _ex)

RMI constructor function for the class

6.4.4

bHYPRE_Schwarz
bHYPRE_Schwarz__wrapObj (void* data, sidl_BaseInterface* _ex)

Wraps up the private data struct pointer (struct bHYPRE_Schwarz__data) passed in rather than running the constructor

6.4.5

bHYPRE_Schwarz
bHYPRE_Schwarz__connect (const char* , sidl_BaseInterface* _ex)

RMI connector function for the class(addrefs)

6.4.6

bHYPRE_Schwarz
bHYPRE_Schwarz_Create (bHYPRE_IJParCSRMatrix A, sidl_BaseInterface* _ex)

This function is the preferred way to create a Schwarz solver.

6.4.7

SIDL_C_INLINE_DECL int32_t
bHYPRE_Schwarz_SetOperator (bHYPRE_Schwarz self, bHYPRE_Operator A, sidl_BaseInterface* _ex)

Set the operator for the linear system being solved. DEPRECATED. use Create

6.4.8

SIDL_C_INLINE_DECL int32_t
bHYPRE_Schwarz_SetTolerance (bHYPRE_Schwarz self, double tolerance, sidl_BaseInterface* _ex)

(Optional) Set the convergence tolerance. DEPRECATED. use SetDoubleParameter

6.4.9

SIDL_C_INLINE_DECL int32_t
bHYPRE_Schwarz_SetMaxIterations (bHYPRE_Schwarz self, int32_t max_iterations, sidl_BaseInterface* _ex)

(Optional) Set maximum number of iterations. DEPRECATED use SetIntParameter

6.4.10

SIDL_C_INLINE_DECL int32_t
bHYPRE_Schwarz_SetLogging (bHYPRE_Schwarz self, int32_t level, sidl_BaseInterface* _ex)

(Optional) Set the *logging level*, specifying the degree of additional informational data to be accumulated. Does nothing by default (level = 0). Other levels (if any) are implementation-specific. Must be called before `Setup` and `Apply`. DEPRECATED use SetIntParameter

6.4.11

SIDL_C_INLINE_DECL int32_t
bHYPRE_Schwarz_SetPrintLevel (bHYPRE_Schwarz self, int32_t level, sidl_BaseInterface* _ex)

(Optional) Set the *print level*, specifying the degree of informational data to be printed either to the screen or to a file. Does nothing by default (level=0). Other levels (if any) are implementation-specific. Must be called before `Setup` and `Apply`. DEPRECATED use SetIntParameter

6.4.12

SIDL_C_INLINE_DECL int32_t
bHYPRE_Schwarz_GetNumIterations (bHYPRE_Schwarz self, int32_t* num_iterations, sidl_BaseInterface* _ex)

(Optional) Return the number of iterations taken

6.4.13

SIDL_C_INLINE_DECL int32_t
bHYPRE_Schwarz_GetRelResidualNorm (bHYPRE_Schwarz self, double* norm, sidl_BaseInterface* _ex)

(Optional) Return the norm of the relative residual

6.4.14

SIDL_C_INLINE_DECL int32_t
bHYPRE_Schwarz_SetCommunicator (bHYPRE_Schwarz self, bHYPRE_MPICommunicator mpi_comm, sidl_BaseInterface* _ex)

Set the MPI Communicator. DEPRECATED, use Create:

6.4.15

SIDL_C_INLINE_DECL void
bHYPRE_Schwarz_Destroy (bHYPRE_Schwarz self, sidl_BaseInterface* _ex)

The Destroy function doesn't necessarily destroy anything. It is just another name for deleteRef. Thus it decrements the object's reference count. The Babel memory management system will destroy the object if the reference count goes to zero.

6.4.16

SIDL_C_INLINE_DECL int32_t
bHYPRE_Schwarz_SetIntParameter (bHYPRE_Schwarz self, const char* name, int32_t value, sidl_BaseInterface* _ex)

Set the int parameter associated with `name`

6.4.17

SIDL_C_INLINE_DECL int32_t
bHYPRE_Schwarz_SetDoubleParameter (bHYPRE_Schwarz self, const char* name, double value, sidl_BaseInterface* _ex)

Set the double parameter associated with `name`

6.4.18

SIDL_C_INLINE_DECL int32_t
bHYPRE_Schwarz_SetStringParameter (bHYPRE_Schwarz self, const char* name, const char* value, sidl_BaseInterface* _ex)

Set the string parameter associated with `name`

6.4.19

SIDL_C_INLINE_DECL int32_t
bHYPRE_Schwarz_SetIntArray1Parameter (bHYPRE_Schwarz self, const char* name, int32_t* value, int32_t nvalues, sidl_BaseInterface* _ex)

Set the int 1-D array parameter associated with `name`

6.4.20

SIDL_C_INLINE_DECL int32_t
bHYPRE_Schwarz_SetIntArray2Parameter (bHYPRE_Schwarz self, const char* name, struct sidl_int__array* value, sidl_BaseInterface* _ex)

Set the int 2-D array parameter associated with `name`

6.4.21

SIDL_C_INLINE_DECL int32_t
bHYPRE_Schwarz_SetDoubleArray1Parameter (bHYPRE_Schwarz self, const char* name, double* value, int32_t nvalues, sidl_BaseInterface* _ex)

Set the double 1-D array parameter associated with `name`

6.4.22

SIDL_C_INLINE_DECL int32_t
bHYPRE_Schwarz_SetDoubleArray2Parameter (bHYPRE_Schwarz self, const char* name, struct sidl_double__array* value, sidl_BaseInterface* _ex)

Set the double 2-D array parameter associated with `name`

6.4.23

SIDL_C_INLINE_DECL int32_t
bHYPRE_Schwarz_GetIntValue (bHYPRE_Schwarz self, const char* name, int32_t* value, sidl_BaseInterface* _ex)

Set the int parameter associated with `name`

6.4.24

SIDL_C_INLINE_DECL int32_t
bHYPRE_Schwarz_GetDoubleValue (bHYPRE_Schwarz self, const char* name, double* value, sidl_BaseInterface* _ex)

Get the double parameter associated with `name`

6.4.25

SIDL_C_INLINE_DECL int32_t
bHYPRE_Schwarz_Setup (bHYPRE_Schwarz self, bHYPRE_Vector b, bHYPRE_Vector x, sidl_BaseInterface* _ex)

(Optional) Do any preprocessing that may be necessary in order to execute `Apply`

6.4.26

SIDL_C_INLINE_DECL int32_t
bHYPRE_Schwarz_Apply (bHYPRE_Schwarz self, bHYPRE_Vector b, bHYPRE_Vector* x, sidl_BaseInterface* _ex)

Apply the operator to `b`, returning `x`

6.4.27

SIDL_C_INLINE_DECL int32_t
bHYPRE_Schwarz_ApplyAdjoint (bHYPRE_Schwarz self, bHYPRE_Vector b, bHYPRE_Vector* x, sidl_BaseInterface* _ex)

Apply the adjoint of the operator to `b`, returning `x`

6.4.28

struct bHYPRE_Schwarz__object*
bHYPRE_Schwarz__cast (void* obj, sidl_BaseInterface* _ex)

Cast method for interface and class type conversions

6.4.29

void*
bHYPRE_Schwarz__cast2 (void* obj, const char* type, sidl_BaseInterface* _ex)

String cast method for interface and class type conversions

6.4.30

SIDL_C_INLINE_DECL void
bHYPRE_Schwarz__exec (bHYPRE_Schwarz self, const char* methodName,
sidl_rmi_Call inArgs, sidl_rmi_Return outArgs, sidl_BaseInterface* _ex)

Select and execute a method by name

6.4.31

SIDL_C_INLINE_DECL char*
bHYPRE_Schwarz__getURL (bHYPRE_Schwarz self, sidl_BaseInterface* _ex)

Get the URL of the Implementation of this object (for RMI)

6.4.32

SIDL_C_INLINE_DECL void
bHYPRE_Schwarz__raddRef (bHYPRE_Schwarz self, sidl_BaseInterface* _ex)

On a remote object, addrefs the remote instance

6.4.33

SIDL_C_INLINE_DECL sidl_bool
bHYPRE_Schwarz__isRemote (bHYPRE_Schwarz self, sidl_BaseInterface* _ex)

TRUE if this object is remote, false if local

6.4.34

sidl_bool
bHYPRE_Schwarz__isLocal (bHYPRE_Schwarz self, sidl_BaseInterface* _ex)

TRUE if this object is remote, false if local

6.4.35

struct bHYPRE_Schwarz__object*
bHYPRE_Schwarz__rmicast (void* obj, struct sidl_BaseInterface__object** _ex)

Cast method for interface and class type conversions

6.4.36

struct bHYPRE_Schwarz__object*
bHYPRE_Schwarz__connectI (const char* url, sidl_bool ar, struct sidl_BaseInterface__object** _ex)

RMI connector function for the class. (no addref)

6.5

ParCSR ParaSails Solver

Names

6.5.1

struct **bHYPRE_ParaSails__object**

Symbol "bHYPREParaSails" (version 100)

Objects of this type can be cast to Solver objects using the `__cast` methods.

ParaSails requires an IJParCSR matrix

6.5.2

struct bHYPRE_ParaSails__object*
bHYPRE_ParaSails__create (sidl_BaseInterface* _ex)

Constructor function for the class

6.5.3

bHYPRE_ParaSails
bHYPRE_ParaSails__createRemote (const char* url, sidl_BaseInterface* _ex)

RMI constructor function for the class

6.5.4

bHYPRE_ParaSails
bHYPRE_ParaSails__wrapObj (void* data, sidl_BaseInterface* _ex)

Wraps up the private data struct pointer (struct bHYPRE_ParaSails__data) passed in rather than running the constructor

6.5.5

bHYPRE_ParaSails
bHYPRE_ParaSails__connect (const char* , sidl_BaseInterface* _ex)

RMI connector function for the class(addrefs)

6.5.6

bHYPRE_ParaSails
bHYPRE_ParaSails_Create (bHYPRE_MPICommunicator mpi_comm,
bHYPRE_IJParCSRMatrix A, sidl_BaseInterface* _ex)

This function is the preferred way to create a ParaSails solver.

6.5.7

SIDL_C_INLINE_DECL int32_t
bHYPRE_ParaSails_SetOperator (bHYPRE_ParaSails self, bHYPRE_Operator A, sidl_BaseInterface* _ex)

Set the operator for the linear system being solved. DEPRECATED. use Create

6.5.8

SIDL_C_INLINE_DECL int32_t
bHYPRE_ParaSails_SetTolerance (bHYPRE_ParaSails self, double tolerance, sidl_BaseInterface* _ex)

(Optional) Set the convergence tolerance. DEPRECATED. use SetDoubleParameter

6.5.9

SIDL_C_INLINE_DECL int32_t
bHYPRE_ParaSails_SetMaxIterations (bHYPRE_ParaSails self, int32_t max_iterations, sidl_BaseInterface* _ex)

(Optional) Set maximum number of iterations. DEPRECATED use SetIntParameter

6.5.10

SIDL_C_INLINE_DECL int32_t
bHYPRE_ParaSails_SetLogging (bHYPRE_ParaSails self, int32_t level, sidl_BaseInterface* _ex)

(Optional) Set the *logging level*, specifying the degree of additional informational data to be accumulated. Does nothing by default (level = 0). Other levels (if any) are implementation-specific. Must be called before `Setup` and `Apply`. DEPRECATED use SetIntParameter

6.5.11

SIDL_C_INLINE_DECL int32_t
bHYPRE_ParaSails_SetPrintLevel (bHYPRE_ParaSails self, int32_t level, sidl_BaseInterface* _ex)

(Optional) Set the *print level*, specifying the degree of informational data to be printed either to the screen or to a file. Does nothing by default (level=0). Other levels (if any) are implementation-specific. Must be called before `Setup` and `Apply`. DEPRECATED use SetIntParameter

6.5.12

SIDL_C_INLINE_DECL int32_t
bHYPRE_ParaSails_GetNumIterations (bHYPRE_ParaSails self, int32_t* num_iterations, sidl_BaseInterface* _ex)

(Optional) Return the number of iterations taken

6.5.13

SIDL_C_INLINE_DECL int32_t
bHYPRE_ParaSails_GetRelResidualNorm (bHYPRE_ParaSails self, double* norm, sidl_BaseInterface* _ex)

(Optional) Return the norm of the relative residual

6.5.14

SIDL_C_INLINE_DECL int32_t
bHYPRE_ParaSails_SetCommunicator (bHYPRE_ParaSails self, bHYPRE_MPICommunicator mpi_comm, sidl_BaseInterface* _ex)

Set the MPI Communicator. DEPRECATED, use Create:

6.5.15

SIDL_C_INLINE_DECL void
bHYPRE_ParaSails_Destroy (bHYPRE_ParaSails self, sidl_BaseInterface* _ex)

The Destroy function doesn't necessarily destroy anything. It is just another name for deleteRef. Thus it decrements the object's reference count. The Babel memory management system will destroy the object if the reference count goes to zero.

6.5.16

SIDL_C_INLINE_DECL int32_t
bHYPRE_ParaSails_SetIntParameter (bHYPRE_ParaSails self, const char* name, int32_t value, sidl_BaseInterface* _ex)

Set the int parameter associated with `name`

6.5.17

SIDL_C_INLINE_DECL int32_t
bHYPRE_ParaSails_SetDoubleParameter (bHYPRE_ParaSails self, const char* name, double value, sidl_BaseInterface* _ex)

Set the double parameter associated with `name`

6.5.18

SIDL_C_INLINE_DECL int32_t
bHYPRE_ParaSails_SetStringParameter (bHYPRE_ParaSails self, const char* name, const char* value, sidl_BaseInterface* _ex)

Set the string parameter associated with `name`

6.5.19

SIDL_C_INLINE_DECL int32_t
bHYPRE_ParaSails_SetIntArray1Parameter (bHYPRE_ParaSails self, const char* name, int32_t* value, int32_t nvalues, sidl_BaseInterface* _ex)

Set the int 1-D array parameter associated with `name`

6.5.20

SIDL_C_INLINE_DECL int32_t
bHYPRE_ParaSails_SetIntArray2Parameter (bHYPRE_ParaSails self, const char* name, struct sidl_int__array* value, sidl_BaseInterface* _ex)

Set the int 2-D array parameter associated with `name`

6.5.21

SIDL_C_INLINE_DECL int32_t
bHYPRE_ParaSails_SetDoubleArray1Parameter (bHYPRE_ParaSails self, const char* name, double* value, int32_t nvalues, sidl_BaseInterface* _ex)

Set the double 1-D array parameter associated with `name`

6.5.22

SIDL_C_INLINE_DECL int32_t
bHYPRE_ParaSails_SetDoubleArray2Parameter (bHYPRE_ParaSails self, const char* name, struct sidl_double__array* value, sidl_BaseInterface* _ex)

Set the double 2-D array parameter associated with `name`

6.5.23

SIDL_C_INLINE_DECL int32_t
bHYPRE_ParaSails_GetIntValue (bHYPRE_ParaSails self, const char* name, int32_t* value, sidl_BaseInterface* _ex)

Set the int parameter associated with `name`

6.5.24

SIDL_C_INLINE_DECL int32_t
bHYPRE_ParaSails_GetDoubleValue (bHYPRE_ParaSails self, const char* name, double* value, sidl_BaseInterface* _ex)

Get the double parameter associated with `name`

6.5.25

SIDL_C_INLINE_DECL int32_t
bHYPRE_ParaSails_Setup (bHYPRE_ParaSails self, bHYPRE_Vector b, bHYPRE_Vector x, sidl_BaseInterface* _ex)

(Optional) Do any preprocessing that may be necessary in order to execute `Apply`

6.5.26

SIDL_C_INLINE_DECL int32_t
bHYPRE_ParaSails_Apply (bHYPRE_ParaSails self, bHYPRE_Vector b, bHYPRE_Vector* x, sidl_BaseInterface* _ex)

Apply the operator to `b`, returning `x`

6.5.27

SIDL_C_INLINE_DECL int32_t
bHYPRE_ParaSails_ApplyAdjoint (bHYPRE_ParaSails self,
bHYPRE_Vector b, bHYPRE_Vector* x, sidl_BaseInterface* _ex)

Apply the adjoint of the operator to `b`, returning `x`

6.5.28

struct bHYPRE_ParaSails__object*
bHYPRE_ParaSails__cast (void* obj, sidl_BaseInterface* _ex)

Cast method for interface and class type conversions

6.5.29

void*
bHYPRE_ParaSails__cast2 (void* obj, const char* type, sidl_BaseInterface* _ex)

String cast method for interface and class type conversions

6.5.30

SIDL_C_INLINE_DECL void
bHYPRE_ParaSails__exec (bHYPRE_ParaSails self, const char* methodName,
sidl_rmi_Call inArgs, sidl_rmi_Return outArgs, sidl_BaseInterface* _ex)

Select and execute a method by name

6.5.31

SIDL_C_INLINE_DECL char*
bHYPRE_ParaSails__getURL (bHYPRE_ParaSails self, sidl_BaseInterface* _ex)

Get the URL of the Implementation of this object (for RMI)

6.5.32

SIDL_C_INLINE_DECL void
bHYPRE_ParaSails__raddRef (bHYPRE_ParaSails self, sidl_BaseInterface* _ex)

On a remote object, addrefs the remote instance

6.5.33

SIDL_C_INLINE_DECL sidl_bool
bHYPRE_ParaSails__isRemote (bHYPRE_ParaSails self, sidl_BaseInterface* _ex)

TRUE if this object is remote, false if local

6.5.34

sidl_bool
bHYPRE_ParaSails__isLocal (bHYPRE_ParaSails self, sidl_BaseInterface* _ex)

TRUE if this object is remote, false if local

6.5.35

struct bHYPRE_ParaSails__object*
bHYPRE_ParaSails__rmicast (void* obj, struct sidl_BaseInterface__object** _ex)

Cast method for interface and class type conversions

6.5.36

struct bHYPRE_ParaSails__object*
bHYPRE_ParaSails__connectI (const char* url, sidl_bool ar, struct sidl_BaseInterface__object** _ex)

RMI connector function for the class. (no addref)

6.6

ParCSR Pilut Solver

Names

6.6.1

struct **bHYPRE_Pilut__object**

Symbol "bHYPREPilut" (version 100)

Objects of this type can be cast to Solver objects using the __cast methods.

Pilut has not been implemented yet.

6.6.2

struct bHYPRE_Pilut__object* **bHYPRE_Pilut__create** (sidl_BaseInterface* _ex)

Constructor function for the class

6.6.3

bHYPRE_Pilut
bHYPRE_Pilut__createRemote (const char* url, sidl_BaseInterface* _ex)

RMI constructor function for the class

6.6.4

bHYPRE_Pilut **bHYPRE_Pilut__wrapObj** (void* data, sidl_BaseInterface* _ex)

Wraps up the private data struct pointer (struct bHYPRE_Pilut__data) passed in rather than running the constructor

6.6.5

bHYPRE_Pilut **bHYPRE_Pilut__connect** (const char* , sidl_BaseInterface* _ex)

RMI connector function for the class(addrefs)

6.6.6

bHYPRE_Pilut
bHYPRE_Pilut_Create (bHYPRE_MPICommunicator mpi_comm, bHYPRE_Operator A, sidl_BaseInterface* _ex)

This function is the preferred way to create a Pilut solver.

6.6.7

SIDL_C_INLINE_DECL int32_t
bHYPRE_Pilut_SetOperator (bHYPRE_Pilut self, bHYPRE_Operator A, sidl_BaseInterface* _ex)

Set the operator for the linear system being solved. DEPRECATED. use Create

6.6.8

SIDL_C_INLINE_DECL int32_t
bHYPRE_Pilut_SetTolerance (bHYPRE_Pilut self, double tolerance, sidl_BaseInterface* _ex)

(Optional) Set the convergence tolerance. DEPRECATED. use SetDoubleParameter

6.6.9

SIDL_C_INLINE_DECL int32_t
bHYPRE_Pilut_SetMaxIterations (bHYPRE_Pilut self, int32_t max_iterations, sidl_BaseInterface* _ex)

(Optional) Set maximum number of iterations. DEPRECATED use SetIntParameter

6.6.10

SIDL_C_INLINE_DECL int32_t
bHYPRE_Pilut_SetLogging (bHYPRE_Pilut self, int32_t level,
sidl_BaseInterface* _ex)

(Optional) Set the *logging level*, specifying the degree of additional informational data to be accumulated. Does nothing by default (level = 0). Other levels (if any) are implementation-specific. Must be called before `Setup` and `Apply`. DEPRECATED use SetIntParameter

6.6.11

SIDL_C_INLINE_DECL int32_t
bHYPRE_Pilut_SetPrintLevel (bHYPRE_Pilut self, int32_t level,
sidl_BaseInterface* _ex)

(Optional) Set the *print level*, specifying the degree of informational data to be printed either to the screen or to a file. Does nothing by default (level=0). Other levels (if any) are implementation-specific. Must be called before `Setup` and `Apply`. DEPRECATED use SetIntParameter

6.6.12

SIDL_C_INLINE_DECL int32_t
bHYPRE_Pilut_GetNumIterations (bHYPRE_Pilut self, int32_t*
num_iterations, sidl_BaseInterface* _ex)

(Optional) Return the number of iterations taken

6.6.13

SIDL_C_INLINE_DECL int32_t
bHYPRE_Pilut_GetRelResidualNorm (bHYPRE_Pilut self, double* norm,
sidl_BaseInterface* _ex)

(Optional) Return the norm of the relative residual

6.6.14

SIDL_C_INLINE_DECL int32_t
bHYPRE_Pilut_SetCommunicator (bHYPRE_Pilut self,
bHYPRE_MPICommunicator mpi_comm, sidl_BaseInterface* _ex)

Set the MPI Communicator. DEPRECATED, use Create:

6.6.15

SIDL_C_INLINE_DECL void
bHYPRE_Pilut_Destroy (bHYPRE_Pilut self, sidl_BaseInterface* _ex)

The Destroy function doesn't necessarily destroy anything. It is just another name for deleteRef. Thus it decrements the object's reference count. The Babel memory management system will destroy the object if the reference count goes to zero.

6.6.16

SIDL_C_INLINE_DECL int32_t
bHYPRE_Pilut_SetIntParameter (bHYPRE_Pilut self, const char* name,
int32_t value, sidl_BaseInterface* _ex)

Set the int parameter associated with `name`

6.6.17

SIDL_C_INLINE_DECL int32_t
bHYPRE_Pilut_SetDoubleParameter (bHYPRE_Pilut self, const char*
name, double value, sidl_BaseInterface* _ex)

Set the double parameter associated with `name`

6.6.18

SIDL_C_INLINE_DECL int32_t
bHYPRE_Pilut_SetStringParameter (bHYPRE_Pilut self, const char* name, const char* value, sidl_BaseInterface* _ex)

Set the string parameter associated with `name`

6.6.19

SIDL_C_INLINE_DECL int32_t
bHYPRE_Pilut_SetIntArray1Parameter (bHYPRE_Pilut self, const char* name, int32_t* value, int32_t nvalues, sidl_BaseInterface* _ex)

Set the int 1-D array parameter associated with `name`

6.6.20

SIDL_C_INLINE_DECL int32_t
bHYPRE_Pilut_SetIntArray2Parameter (bHYPRE_Pilut self, const char* name, struct sidl_int__array* value, sidl_BaseInterface* _ex)

Set the int 2-D array parameter associated with `name`

6.6.21

SIDL_C_INLINE_DECL int32_t
bHYPRE_Pilut_SetDoubleArray1Parameter (bHYPRE_Pilut self, const char* name, double* value, int32_t nvalues, sidl_BaseInterface* _ex)

Set the double 1-D array parameter associated with `name`

6.6.22

SIDL_C_INLINE_DECL int32_t
bHYPRE_Pilut_SetDoubleArray2Parameter (bHYPRE_Pilut self, const char* name, struct sidl_double__array* value, sidl_BaseInterface* _ex)

Set the double 2-D array parameter associated with `name`

6.6.23

SIDL_C_INLINE_DECL int32_t
bHYPRE_Pilut_GetIntValue (bHYPRE_Pilut self, const char* name, int32_t* value, sidl_BaseInterface* _ex)

Set the int parameter associated with `name`

6.6.24

SIDL_C_INLINE_DECL int32_t
bHYPRE_Pilut_GetDoubleValue (bHYPRE_Pilut self, const char* name, double* value, sidl_BaseInterface* _ex)

Get the double parameter associated with `name`

6.6.25

SIDL_C_INLINE_DECL int32_t
bHYPRE_Pilut_Setup (bHYPRE_Pilut self, bHYPRE_Vector b, bHYPRE_Vector x, sidl_BaseInterface* _ex)

(Optional) Do any preprocessing that may be necessary in order to execute `Apply`

6.6.26

SIDL_C_INLINE_DECL int32_t
bHYPRE_Pilut_Apply (bHYPRE_Pilut self, bHYPRE_Vector b, bHYPRE_Vector* x, sidl_BaseInterface* _ex)

Apply the operator to `b`, returning `x`

6.6.27

SIDL_C_INLINE_DECL int32_t
bHYPRE_Pilut_ApplyAdjoint (bHYPRE_Pilut self, bHYPRE_Vector b, bHYPRE_Vector* x, sidl_BaseInterface* _ex)

Apply the adjoint of the operator to `b`, returning `x`

6.6.28

struct bHYPRE_Pilut__object*
bHYPRE_Pilut__cast (void* obj, sidl_BaseInterface* _ex)

Cast method for interface and class type conversions

6.6.29

void*
bHYPRE_Pilut__cast2 (void* obj, const char* type, sidl_BaseInterface* _ex)

String cast method for interface and class type conversions

6.6.30

SIDL_C_INLINE_DECL void
bHYPRE_Pilut__exec (bHYPRE_Pilut self, const char* methodName,
sidl_rmi_Call inArgs, sidl_rmi_Return outArgs, sidl_BaseInterface* _ex)

Select and execute a method by name

6.6.31

SIDL_C_INLINE_DECL char*
bHYPRE_Pilut__getURL (bHYPRE_Pilut self, sidl_BaseInterface* _ex)

Get the URL of the Implementation of this object (for RMI)

6.6.32

SIDL_C_INLINE_DECL void
bHYPRE_Pilut__raddRef (bHYPRE_Pilut self, sidl_BaseInterface* _ex)

On a remote object, addrefs the remote instance

6.6.33

SIDL_C_INLINE_DECL sidl_bool
bHYPRE_Pilut__isRemote (bHYPRE_Pilut self, sidl_BaseInterface* _ex)

TRUE if this object is remote, false if local

6.6.34

sidl_bool
bHYPRE_Pilut__isLocal (bHYPRE_Pilut self, sidl_BaseInterface* _ex)

TRUE if this object is remote, false if local

6.6.35

struct bHYPRE_Pilut__object*
bHYPRE_Pilut__rmicast (void* obj, struct sidl_BaseInterface__object** _ex)

Cast method for interface and class type conversions

6.6.36

struct bHYPRE_Pilut__object*
bHYPRE_Pilut__connectI (const char* url, sidl_bool ar, struct sidl_BaseInterface__object** _ex)

RMI connector function for the class. (no addref)

7

Structured Matrix Solvers

Names

These solvers use structured matrix/vector storage schemes.

7.1

StructDiagScale Solver

Names

7.1.1

struct **bHYPRE_StructDiagScale__object**

Symbol "bHYPREStructDiagScale" (version 100)

Diagonal scaling preconditioner for STruct matrix class.

Objects of this type can be cast to Solver objects using the __cast methods.

7.1.2

struct bHYPRE_StructDiagScale__object*
bHYPRE_StructDiagScale__create (sidl_BaseInterface* _ex)

Constructor function for the class

7.1.3

bHYPRE_StructDiagScale
bHYPRE_StructDiagScale__createRemote (const char* url,
sidl_BaseInterface* _ex)

RMI constructor function for the class

7.1.4

bHYPRE_StructDiagScale
bHYPRE_StructDiagScale__wrapObj (void* data, sidl_BaseInterface* _ex)

Wraps up the private data struct pointer (struct bHYPRE_StructDiagScale__data) passed in rather than running the constructor

7.1.5

bHYPRE_StructDiagScale
bHYPRE_StructDiagScale__connect (const char* , sidl_BaseInterface* _ex)

RMI connector function for the class(addrefs)

7.1.6

bHYPRE_StructDiagScale
bHYPRE_StructDiagScale_Create (bHYPRE_MPICommunicator mpi_comm, bHYPRE_StructMatrix A, sidl_BaseInterface* eex)

This function is the preferred way to create a Struct DiagScale solver.

7.1.7

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructDiagScale_SetOperator (bHYPRE_StructDiagScale self, bHYPRE_Operator A, sidl_BaseInterface* _ex)

Set the operator for the linear system being solved. DEPRECATED. use Create

7.1.8

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructDiagScale_SetTolerance (bHYPRE_StructDiagScale self, double tolerance, sidl_BaseInterface* _ex)

(Optional) Set the convergence tolerance. DEPRECATED. use SetDoubleParameter

7.1.9

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructDiagScale_SetMaxIterations (bHYPRE_StructDiagScale self, int32_t max_iterations, sidl_BaseInterface* _ex)

(Optional) Set maximum number of iterations. DEPRECATED use SetIntParameter

7.1.10

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructDiagScale_SetLogging (bHYPRE_StructDiagScale self, int32_t level, sidl_BaseInterface* _ex)

(Optional) Set the *logging level*, specifying the degree of additional informational data to be accumulated. Does nothing by default (level = 0). Other levels (if any) are implementation-specific. Must be called before `Setup` and `Apply`. DEPRECATED use SetIntParameter

7.1.11

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructDiagScale_SetPrintLevel (bHYPRE_StructDiagScale self, int32_t level, sidl_BaseInterface* _ex)

(Optional) Set the *print level*, specifying the degree of informational data to be printed either to the screen or to a file. Does nothing by default (level=0). Other levels (if any) are implementation-specific. Must be called before `Setup` and `Apply`. DEPRECATED use SetIntParameter

7.1.12

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructDiagScale_GetNumIterations (bHYPRE_StructDiagScale self, int32_t* num_iterations, sidl_BaseInterface* _ex)

(Optional) Return the number of iterations taken

7.1.13

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructDiagScale_GetRelResidualNorm (bHYPRE_StructDiagScale self, double* norm, sidl_BaseInterface* _ex)

(Optional) Return the norm of the relative residual

7.1.14

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructDiagScale_SetCommunicator (bHYPRE_StructDiagScale self, bHYPRE_MPICommunicator mpi_comm, sidl_BaseInterface* _ex)

Set the MPI Communicator. DEPRECATED, use Create:

7.1.15

SIDL_C_INLINE_DECL void
bHYPRE_StructDiagScale_Destroy (bHYPRE_StructDiagScale self, sidl_BaseInterface* _ex)

The Destroy function doesn't necessarily destroy anything. It is just another name for deleteRef. Thus it decrements the object's reference count. The Babel memory management system will destroy the object if the reference count goes to zero.

7.1.16

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructDiagScale_SetIntParameter (bHYPRE_StructDiagScale self, const char* name, int32_t value, sidl_BaseInterface* _ex)

Set the int parameter associated with `name`

7.1.17

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructDiagScale_SetDoubleParameter (bHYPRE_StructDiagScale self, const char* name, double value, sidl_BaseInterface* _ex)

Set the double parameter associated with `name`

7.1.18

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructDiagScale_SetStringParameter (bHYPRE_StructDiagScale self, const char* name, const char* value, sidl_BaseInterface* _ex)

Set the string parameter associated with `name`

7.1.19

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructDiagScale_SetIntArray1Parameter (
bHYPRE_StructDiagScale self, const char* name, int32_t* value, int32_t nvalues, sidl_BaseInterface* _ex)

Set the int 1-D array parameter associated with `name`

7.1.20

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructDiagScale_SetIntArray2Parameter (
bHYPRE_StructDiagScale self, const char* name, struct sidl_int__array* value, sidl_BaseInterface* _ex)

Set the int 2-D array parameter associated with `name`

7.1.21

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructDiagScale_SetDoubleArray1Parameter (
bHYPRE_StructDiagScale self, const char* name, double* value, int32_t nvalues, sidl_BaseInterface* _ex)

Set the double 1-D array parameter associated with `name`

7.1.22

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructDiagScale_SetDoubleArray2Parameter (
bHYPRE_StructDiagScale self, const char* name, struct sidl_double__array* value,
sidl_BaseInterface* _ex)

Set the double 2-D array parameter associated with `name`

7.1.23

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructDiagScale_GetIntValue (bHYPRE_StructDiagScale self,
const char* name, int32_t* value, sidl_BaseInterface* _ex)

Set the int parameter associated with `name`

7.1.24

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructDiagScale_GetDoubleValue (bHYPRE_StructDiagScale self,
const char* name, double* value, sidl_BaseInterface* _ex)

Get the double parameter associated with `name`

7.1.25

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructDiagScale_Setup (bHYPRE_StructDiagScale self,
bHYPRE_Vector b, bHYPRE_Vector x, sidl_BaseInterface* _ex)

(Optional) Do any preprocessing that may be necessary in order to execute `Apply`

7.1.26

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructDiagScale_Apply (bHYPRE_StructDiagScale self,
bHYPRE_Vector b, bHYPRE_Vector* x, sidl_BaseInterface* _ex)

Apply the operator to `b`, returning `x`

7.1.27

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructDiagScale_ApplyAdjoint (bHYPRE_StructDiagScale self,
bHYPRE_Vector b, bHYPRE_Vector* x, sidl_BaseInterface* _ex)

Apply the adjoint of the operator to `b`, returning `x`

7.1.28

struct bHYPRE_StructDiagScale__object*
bHYPRE_StructDiagScale__cast (void* obj, sidl_BaseInterface* _ex)

Cast method for interface and class type conversions

7.1.29

void*
bHYPRE_StructDiagScale__cast2 (void* obj, const char* type,
sidl_BaseInterface* _ex)

String cast method for interface and class type conversions

7.1.30

SIDL_C_INLINE_DECL void
bHYPRE_StructDiagScale__exec (bHYPRE_StructDiagScale self, const char* methodName, sidl_rmi_Call inArgs, sidl_rmi_Return outArgs, sidl_BaseInterface* _ex)

Select and execute a method by name

7.1.31

SIDL_C_INLINE_DECL char*
bHYPRE_StructDiagScale__getURL (bHYPRE_StructDiagScale self, sidl_BaseInterface* _ex)

Get the URL of the Implementation of this object (for RMI)

7.1.32

SIDL_C_INLINE_DECL void
bHYPRE_StructDiagScale__raddRef (bHYPRE_StructDiagScale self, sidl_BaseInterface* _ex)

On a remote object, addrefs the remote instance

7.1.33

SIDL_C_INLINE_DECL sidl_bool
bHYPRE_StructDiagScale__isRemote (bHYPRE_StructDiagScale self, sidl_BaseInterface* _ex)

TRUE if this object is remote, false if local

7.1.34

sidl_bool
bHYPRE_StructDiagScale__isLocal (bHYPRE_StructDiagScale self,
sidl_BaseInterface* _ex)

TRUE if this object is remote, false if local

7.1.35

struct bHYPRE_StructDiagScale__object*
bHYPRE_StructDiagScale__rmicast (void* obj, struct
sidl_BaseInterface__object** _ex)

Cast method for interface and class type conversions

7.1.36

struct bHYPRE_StructDiagScale__object*
bHYPRE_StructDiagScale__connectI (const char* url, sidl_bool ar, struct
sidl_BaseInterface__object** _ex)

RMI connector function for the class. (no addref)

7.2

Struct Jacobi Solver

Names

7.2.1

struct **bHYPRE_StructJacobi__object**

Symbol "bHYPREStructJacobi" (version 100)

Objects of this type can be cast to Solver objects using the __cast methods.

The StructJacobi solver requires a Struct matrix.

7.2.2

struct bHYPRE_StructJacobi__object*
bHYPRE_StructJacobi__create (sidl_BaseInterface* _ex)

Constructor function for the class

7.2.3

bHYPRE_StructJacobi
bHYPRE_StructJacobi__createRemote (const char* url, sidl_BaseInterface* _ex)

RMI constructor function for the class

7.2.4

bHYPRE_StructJacobi
bHYPRE_StructJacobi__wrapObj (void* data, sidl_BaseInterface* _ex)

Wraps up the private data struct pointer (struct bHYPRE_StructJacobi__data) passed in rather than running the constructor

7.2.5

bHYPRE_StructJacobi
bHYPRE_StructJacobi__connect (const char* , sidl_BaseInterface* _ex)

RMI connector function for the class(addrefs)

7.2.6

bHYPRE_StructJacobi
bHYPRE_StructJacobi_Create (bHYPRE_MPICommunicator mpi_comm, bHYPRE_StructMatrix A, sidl_BaseInterface* _ex)

This function is the preferred way to create a Struct Jacobi solver.

7.2.7

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructJacobi_SetOperator (bHYPRE_StructJacobi self, bHYPRE_Operator A, sidl_BaseInterface* _ex)

Set the operator for the linear system being solved. DEPRECATED. use Create

7.2.8

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructJacobi_SetTolerance (bHYPRE_StructJacobi self, double tolerance, sidl_BaseInterface* _ex)

(Optional) Set the convergence tolerance. DEPRECATED. use SetDoubleParameter

7.2.9

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructJacobi_SetMaxIterations (bHYPRE_StructJacobi self, int32_t max_iterations, sidl_BaseInterface* _ex)

(Optional) Set maximum number of iterations. DEPRECATED use SetIntParameter

7.2.10

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructJacobi_SetLogging (bHYPRE_StructJacobi self, int32_t level, sidl_BaseInterface* _ex)

(Optional) Set the *logging level*, specifying the degree of additional informational data to be accumulated. Does nothing by default (level = 0). Other levels (if any) are implementation-specific. Must be called before `Setup` and `Apply`. DEPRECATED use SetIntParameter

7.2.11

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructJacobi_SetPrintLevel (bHYPRE_StructJacobi self, int32_t level, sidl_BaseInterface* _ex)

(Optional) Set the *print level*, specifying the degree of informational data to be printed either to the screen or to a file. Does nothing by default (level=0). Other levels (if any) are implementation-specific. Must be called before `Setup` and `Apply`. DEPRECATED use SetIntParameter

7.2.12

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructJacobi_GetNumIterations (bHYPRE_StructJacobi self, int32_t* num_iterations, sidl_BaseInterface* _ex)

(Optional) Return the number of iterations taken

7.2.13

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructJacobi_GetRelResidualNorm (bHYPRE_StructJacobi self, double* norm, sidl_BaseInterface* _ex)

(Optional) Return the norm of the relative residual

7.2.14

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructJacobi_SetCommunicator (bHYPRE_StructJacobi self, bHYPRE_MPICommunicator mpi_comm, sidl_BaseInterface* _ex)

Set the MPI Communicator. DEPRECATED, use Create:

7.2.15

SIDL_C_INLINE_DECL void
bHYPRE_StructJacobi_Destroy (bHYPRE_StructJacobi self, sidl_BaseInterface* _ex)

The Destroy function doesn't necessarily destroy anything. It is just another name for deleteRef. Thus it decrements the object's reference count. The Babel memory management system will destroy the object if the reference count goes to zero.

7.2.16

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructJacobi_SetIntParameter (bHYPRE_StructJacobi self, const char* name, int32_t value, sidl_BaseInterface* _ex)

Set the int parameter associated with `name`

7.2.17

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructJacobi_SetDoubleParameter (bHYPRE_StructJacobi self, const char* name, double value, sidl_BaseInterface* _ex)

Set the double parameter associated with `name`

7.2.18

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructJacobi_SetStringParameter (bHYPRE_StructJacobi self, const char* name, const char* value, sidl_BaseInterface* _ex)

Set the string parameter associated with `name`

7.2.19

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructJacobi_SetIntArray1Parameter (bHYPRE_StructJacobi self, const char* name, int32_t* value, int32_t nvalues, sidl_BaseInterface* _ex)

Set the int 1-D array parameter associated with `name`

7.2.20

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructJacobi_SetIntArray2Parameter (bHYPRE_StructJacobi self, const char* name, struct sidl_int__array* value, sidl_BaseInterface* _ex)

Set the int 2-D array parameter associated with `name`

7.2.21

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructJacobi_SetDoubleArray1Parameter (bHYPRE_StructJacobi self, const char* name, double* value, int32_t nvalues, sidl_BaseInterface* _ex)

Set the double 1-D array parameter associated with `name`

7.2.22

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructJacobi_SetDoubleArray2Parameter (
bHYPRE_StructJacobi self, const char* name, struct sidl_double__array* value,
sidl_BaseInterface* _ex)

Set the double 2-D array parameter associated with `name`

7.2.23

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructJacobi_GetIntValue (bHYPRE_StructJacobi self, const char* name, int32_t* value, sidl_BaseInterface* _ex)

Set the int parameter associated with `name`

7.2.24

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructJacobi_GetDoubleValue (bHYPRE_StructJacobi self, const char* name, double* value, sidl_BaseInterface* _ex)

Get the double parameter associated with `name`

7.2.25

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructJacobi_Setup (bHYPRE_StructJacobi self, bHYPRE_Vector b, bHYPRE_Vector x, sidl_BaseInterface* _ex)

(Optional) Do any preprocessing that may be necessary in order to execute `Apply`

7.2.26

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructJacobi_Apply (bHYPRE_StructJacobi self, bHYPRE_Vector b, bHYPRE_Vector* x, sidl_BaseInterface* _ex)

Apply the operator to `b`, returning `x`

7.2.27

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructJacobi_ApplyAdjoint (bHYPRE_StructJacobi self, bHYPRE_Vector b, bHYPRE_Vector* x, sidl_BaseInterface* _ex)

Apply the adjoint of the operator to `b`, returning `x`

7.2.28

struct bHYPRE_StructJacobi__object*
bHYPRE_StructJacobi__cast (void* obj, sidl_BaseInterface* _ex)

Cast method for interface and class type conversions

7.2.29

void*
bHYPRE_StructJacobi__cast2 (void* obj, const char* type, sidl_BaseInterface* _ex)

String cast method for interface and class type conversions

7.2.30

SIDL_C_INLINE_DECL void
bHYPRE_StructJacobi__exec (bHYPRE_StructJacobi self, const char* methodName, sidl_rmi_Call inArgs, sidl_rmi_Return outArgs, sidl_BaseInterface* _ex)

Select and execute a method by name

7.2.31

SIDL_C_INLINE_DECL char*
bHYPRE_StructJacobi__getURL (bHYPRE_StructJacobi self, sidl_BaseInterface* _ex)

Get the URL of the Implementation of this object (for RMI)

7.2.32

SIDL_C_INLINE_DECL void
bHYPRE_StructJacobi__raddRef (bHYPRE_StructJacobi self, sidl_BaseInterface* _ex)

On a remote object, addrefs the remote instance

7.2.33

SIDL_C_INLINE_DECL sidl_bool
bHYPRE_StructJacobi__isRemote (bHYPRE_StructJacobi self, sidl_BaseInterface* _ex)

TRUE if this object is remote, false if local

7.2.34

sidl_bool
bHYPRE_StructJacobi__isLocal (bHYPRE_StructJacobi self,
sidl_BaseInterface* _ex)

TRUE if this object is remote, false if local

7.2.35

struct bHYPRE_StructJacobi__object*
bHYPRE_StructJacobi__rmicast (void* obj, struct sidl_BaseInterface__object**
_ex)

Cast method for interface and class type conversions

7.2.36

struct bHYPRE_StructJacobi__object*
bHYPRE_StructJacobi__connectI (const char* url, sidl_bool ar, struct
sidl_BaseInterface__object** _ex)

RMI connector function for the class. (no addref)

7.3

Struct PFMG Solver

Names

7.3.1

struct **bHYPRE_StructPFMG__object**

Symbol "bHYPREStructPFMG" (version 100)

Objects of this type can be cast to Solver objects using the __cast methods.

The StructPFMG solver requires a Struct matrix.

7.3.2

struct bHYPRE_StructPFMG__object*
bHYPRE_StructPFMG__create (sidl_BaseInterface* _ex)

Constructor function for the class

7.3.3

bHYPRE_StructPFMG
bHYPRE_StructPFMG__createRemote (const char* url, sidl_BaseInterface* _ex)

RMI constructor function for the class

7.3.4

bHYPRE_StructPFMG
bHYPRE_StructPFMG__wrapObj (void* data, sidl_BaseInterface* _ex)

Wraps up the private data struct pointer (struct bHYPRE_StructPFMG__data) passed in rather than running the constructor

7.3.5

bHYPRE_StructPFMG
bHYPRE_StructPFMG__connect (const char* , sidl_BaseInterface* _ex)

RMI connector function for the class(addrefs)

7.3.6

bHYPRE_StructPFMG
bHYPRE_StructPFMG_Create (bHYPRE_MPICommunicator mpi_comm, bHYPRE_StructMatrix A, sidl_BaseInterface* _ex)

This function is the preferred way to create a Struct PFMG solver.

7.3.7

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructPFMG_SetOperator (bHYPRE_StructPFMG self, bHYPRE_Operator A, sidl_BaseInterface* _ex)

Set the operator for the linear system being solved. DEPRECATED. use Create

7.3.8

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructPFMG_SetTolerance (bHYPRE_StructPFMG self, double tolerance, sidl_BaseInterface* _ex)

(Optional) Set the convergence tolerance. DEPRECATED. use SetDoubleParameter

7.3.9

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructPFMG_SetMaxIterations (bHYPRE_StructPFMG self, int32_t max_iterations, sidl_BaseInterface* _ex)

(Optional) Set maximum number of iterations. DEPRECATED use SetIntParameter

7.3.10

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructPFMG_SetLogging (bHYPRE_StructPFMG self, int32_t level, sidl_BaseInterface* _ex)

(Optional) Set the *logging level*, specifying the degree of additional informational data to be accumulated. Does nothing by default (level = 0). Other levels (if any) are implementation-specific. Must be called before `Setup` and `Apply`. DEPRECATED use SetIntParameter

7.3.11

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructPFMG_SetPrintLevel (bHYPRE_StructPFMG self, int32_t level, sidl_BaseInterface* _ex)

(Optional) Set the *print level*, specifying the degree of informational data to be printed either to the screen or to a file. Does nothing by default (level=0). Other levels (if any) are implementation-specific. Must be called before `Setup` and `Apply`. DEPRECATED use SetIntParameter

7.3.12

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructPFMG_GetNumIterations (bHYPRE_StructPFMG self, int32_t* num_iterations, sidl_BaseInterface* _ex)

(Optional) Return the number of iterations taken

7.3.13

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructPFMG_GetRelResidualNorm (bHYPRE_StructPFMG self, double* norm, sidl_BaseInterface* _ex)

(Optional) Return the norm of the relative residual

7.3.14

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructPFMG_SetCommunicator (bHYPRE_StructPFMG self,
bHYPRE_MPICommunicator mpi_comm, sidl_BaseInterface* _ex)

Set the MPI Communicator. DEPRECATED, use Create:

7.3.15

SIDL_C_INLINE_DECL void
bHYPRE_StructPFMG_Destroy (bHYPRE_StructPFMG self,
sidl_BaseInterface* _ex)

The Destroy function doesn't necessarily destroy anything. It is just another name for deleteRef. Thus it decrements the object's reference count. The Babel memory management system will destroy the object if the reference count goes to zero.

7.3.16

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructPFMG_SetIntParameter (bHYPRE_StructPFMG self,
const char* name, int32_t value, sidl_BaseInterface* _ex)

Set the int parameter associated with `name`

7.3.17

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructPFMG_SetDoubleParameter (bHYPRE_StructPFMG self,
const char* name, double value, sidl_BaseInterface* _ex)

Set the double parameter associated with `name`

7.3.18

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructPFMG_SetStringParameter (bHYPRE_StructPFMG self, const char* name, const char* value, sidl_BaseInterface* _ex)

Set the string parameter associated with `name`

7.3.19

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructPFMG_SetIntArray1Parameter (bHYPRE_StructPFMG self, const char* name, int32_t* value, int32_t nvalues, sidl_BaseInterface* _ex)

Set the int 1-D array parameter associated with `name`

7.3.20

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructPFMG_SetIntArray2Parameter (bHYPRE_StructPFMG self, const char* name, struct sidl_int__array* value, sidl_BaseInterface* _ex)

Set the int 2-D array parameter associated with `name`

7.3.21

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructPFMG_SetDoubleArray1Parameter (bHYPRE_StructPFMG self, const char* name, double* value, int32_t nvalues, sidl_BaseInterface* _ex)

Set the double 1-D array parameter associated with `name`

7.3.22

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructPFMG_SetDoubleArray2Parameter (
bHYPRE_StructPFMG self, const char* name, struct sidl_double__array* value,
sidl_BaseInterface* _ex)

Set the double 2-D array parameter associated with `name`

7.3.23

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructPFMG_GetIntValue (bHYPRE_StructPFMG self, const
char* name, int32_t* value, sidl_BaseInterface* _ex)

Set the int parameter associated with `name`

7.3.24

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructPFMG_GetDoubleValue (bHYPRE_StructPFMG self,
const char* name, double* value, sidl_BaseInterface* _ex)

Get the double parameter associated with `name`

7.3.25

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructPFMG_Setup (bHYPRE_StructPFMG self, bHYPRE_Vector
b, bHYPRE_Vector x, sidl_BaseInterface* _ex)

(Optional) Do any preprocessing that may be necessary in order to execute `Apply`

7.3.26

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructPFMG_Apply (bHYPRE_StructPFMG self, bHYPRE_Vector b, bHYPRE_Vector* x, sidl_BaseInterface* _ex)

Apply the operator to `b`, returning `x`

7.3.27

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructPFMG_ApplyAdjoint (bHYPRE_StructPFMG self, bHYPRE_Vector b, bHYPRE_Vector* x, sidl_BaseInterface* _ex)

Apply the adjoint of the operator to `b`, returning `x`

7.3.28

struct bHYPRE_StructPFMG__object*
bHYPRE_StructPFMG__cast (void* obj, sidl_BaseInterface* _ex)

Cast method for interface and class type conversions

7.3.29

void*
bHYPRE_StructPFMG__cast2 (void* obj, const char* type, sidl_BaseInterface* _ex)

String cast method for interface and class type conversions

7.3.30

SIDL_C_INLINE_DECL void
bHYPRE_StructPFMG__exec (bHYPRE_StructPFMG self, const char* methodName, sidl_rmi_Call inArgs, sidl_rmi_Return outArgs, sidl_BaseInterface* _ex)

Select and execute a method by name

7.3.31

SIDL_C_INLINE_DECL char*
bHYPRE_StructPFMG__getURL (bHYPRE_StructPFMG self, sidl_BaseInterface* _ex)

Get the URL of the Implementation of this object (for RMI)

7.3.32

SIDL_C_INLINE_DECL void
bHYPRE_StructPFMG__raddRef (bHYPRE_StructPFMG self, sidl_BaseInterface* _ex)

On a remote object, addrefs the remote instance

7.3.33

SIDL_C_INLINE_DECL sidl_bool
bHYPRE_StructPFMG__isRemote (bHYPRE_StructPFMG self, sidl_BaseInterface* _ex)

TRUE if this object is remote, false if local

7.3.34

sidl_bool
bHYPRE_StructPFMG__isLocal (bHYPRE_StructPFMG self,
sidl_BaseInterface* _ex)

TRUE if this object is remote, false if local

7.3.35

struct bHYPRE_StructPFMG__object*
bHYPRE_StructPFMG__rmicast (void* obj, struct
sidl_BaseInterface__object** _ex)

Cast method for interface and class type conversions

7.3.36

struct bHYPRE_StructPFMG__object*
bHYPRE_StructPFMG__connectI (const char* url, sidl_bool ar, struct
sidl_BaseInterface__object** _ex)

RMI connector function for the class. (no addref)

7.4

Struct SMG Solver

Names

7.4.1

struct **bHYPRE_StructSMG__object**

Symbol "bHYPREStructSMG" (version 100)

Objects of this type can be cast to Solver objects using the __cast methods.

The StructSMG solver requires a Struct matrix.

7.4.2

struct bHYPRE_StructSMG__object*
bHYPRE_StructSMG__create (sidl_BaseInterface* _ex)

Constructor function for the class

7.4.3

bHYPRE_StructSMG
bHYPRE_StructSMG__createRemote (const char* url, sidl_BaseInterface* _ex)

RMI constructor function for the class

7.4.4

bHYPRE_StructSMG
bHYPRE_StructSMG__wrapObj (void* data, sidl_BaseInterface* _ex)

Wraps up the private data struct pointer (struct bHYPRE_StructSMG__data) passed in rather than running the constructor

7.4.5

bHYPRE_StructSMG
bHYPRE_StructSMG__connect (const char* , sidl_BaseInterface* _ex)

RMI connector function for the class(addrefs)

7.4.6

bHYPRE_StructSMG
bHYPRE_StructSMG_Create (bHYPRE_MPICommunicator mpi_comm, bHYPRE_StructMatrix A, sidl_BaseInterface* _ex)

This function is the preferred way to create a Struct SMG solver.

7.4.7

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructSMG_SetOperator (bHYPRE_StructSMG self, bHYPRE_Operator A, sidl_BaseInterface* _ex)

Set the operator for the linear system being solved. DEPRECATED. use Create

7.4.8

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructSMG_SetTolerance (bHYPRE_StructSMG self, double tolerance, sidl_BaseInterface* _ex)

(Optional) Set the convergence tolerance. DEPRECATED. use SetDoubleParameter

7.4.9

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructSMG_SetMaxIterations (bHYPRE_StructSMG self, int32_t max_iterations, sidl_BaseInterface* _ex)

(Optional) Set maximum number of iterations. DEPRECATED use SetIntParameter

7.4.10

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructSMG_SetLogging (bHYPRE_StructSMG self, int32_t level, sidl_BaseInterface* _ex)

(Optional) Set the *logging level*, specifying the degree of additional informational data to be accumulated. Does nothing by default (level = 0). Other levels (if any) are implementation-specific. Must be called before `Setup` and `Apply`. DEPRECATED use SetIntParameter

7.4.11

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructSMG_SetPrintLevel (bHYPRE_StructSMG self, int32_t level, sidl_BaseInterface* _ex)

(Optional) Set the *print level*, specifying the degree of informational data to be printed either to the screen or to a file. Does nothing by default (level=0). Other levels (if any) are implementation-specific. Must be called before `Setup` and `Apply`. DEPRECATED use SetIntParameter

7.4.12

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructSMG_GetNumIterations (bHYPRE_StructSMG self, int32_t* num_iterations, sidl_BaseInterface* _ex)

(Optional) Return the number of iterations taken

7.4.13

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructSMG_GetRelResidualNorm (bHYPRE_StructSMG self, double* norm, sidl_BaseInterface* _ex)

(Optional) Return the norm of the relative residual

7.4.14

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructSMG_SetCommunicator (bHYPRE_StructSMG self, bHYPRE_MPICommunicator mpi_comm, sidl_BaseInterface* _ex)

Set the MPI Communicator. DEPRECATED, use Create:

7.4.15

SIDL_C_INLINE_DECL void
bHYPRE_StructSMG_Destroy (bHYPRE_StructSMG self, sidl_BaseInterface* _ex)

The Destroy function doesn't necessarily destroy anything. It is just another name for deleteRef. Thus it decrements the object's reference count. The Babel memory management system will destroy the object if the reference count goes to zero.

7.4.16

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructSMG_SetIntParameter (bHYPRE_StructSMG self, const char* name, int32_t value, sidl_BaseInterface* _ex)

Set the int parameter associated with `name`

7.4.17

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructSMG_SetDoubleParameter (bHYPRE_StructSMG self, const char* name, double value, sidl_BaseInterface* _ex)

Set the double parameter associated with `name`

7.4.18

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructSMG_SetStringParameter (bHYPRE_StructSMG self, const char* name, const char* value, sidl_BaseInterface* _ex)

Set the string parameter associated with `name`

7.4.19

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructSMG_SetIntArray1Parameter (bHYPRE_StructSMG self, const char* name, int32_t* value, int32_t nvalues, sidl_BaseInterface* _ex)

Set the int 1-D array parameter associated with `name`

7.4.20

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructSMG_SetIntArray2Parameter (bHYPRE_StructSMG self, const char* name, struct sidl_int__array* value, sidl_BaseInterface* _ex)

Set the int 2-D array parameter associated with `name`

7.4.21

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructSMG_SetDoubleArray1Parameter (bHYPRE_StructSMG self, const char* name, double* value, int32_t nvalues, sidl_BaseInterface* _ex)

Set the double 1-D array parameter associated with `name`

7.4.22

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructSMG_SetDoubleArray2Parameter (bHYPRE_StructSMG self, const char* name, struct sidl_double__array* value, sidl_BaseInterface* _ex)

Set the double 2-D array parameter associated with `name`

7.4.23

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructSMG_GetIntValue (bHYPRE_StructSMG self, const char* name, int32_t* value, sidl_BaseInterface* _ex)

Set the int parameter associated with `name`

7.4.24

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructSMG_GetDoubleValue (bHYPRE_StructSMG self, const char* name, double* value, sidl_BaseInterface* _ex)

Get the double parameter associated with `name`

7.4.25

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructSMG_Setup (bHYPRE_StructSMG self, bHYPRE_Vector b, bHYPRE_Vector x, sidl_BaseInterface* _ex)

(Optional) Do any preprocessing that may be necessary in order to execute `Apply`

7.4.26

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructSMG_Apply (bHYPRE_StructSMG self, bHYPRE_Vector b, bHYPRE_Vector* x, sidl_BaseInterface* _ex)

Apply the operator to `b`, returning `x`

7.4.27

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructSMG_ApplyAdjoint (bHYPRE_StructSMG self, bHYPRE_Vector b, bHYPRE_Vector* x, sidl_BaseInterface* _ex)

Apply the adjoint of the operator to `b`, returning `x`

7.4.28

struct bHYPRE_StructSMG__object*
bHYPRE_StructSMG__cast (void* obj, sidl_BaseInterface* _ex)

Cast method for interface and class type conversions

7.4.29

void*
bHYPRE_StructSMG__cast2 (void* obj, const char* type, sidl_BaseInterface* _ex)

String cast method for interface and class type conversions

7.4.30

SIDL_C_INLINE_DECL void
bHYPRE_StructSMG__exec (bHYPRE_StructSMG self, const char* methodName, sidl_rmi_Call inArgs, sidl_rmi_Return outArgs, sidl_BaseInterface* _ex)

Select and execute a method by name

7.4.31

SIDL_C_INLINE_DECL char*
bHYPRE_StructSMG__getURL (bHYPRE_StructSMG self, sidl_BaseInterface* _ex)

Get the URL of the Implementation of this object (for RMI)

7.4.32

SIDL_C_INLINE_DECL void
bHYPRE_StructSMG__raddRef (bHYPRE_StructSMG self, sidl_BaseInterface* _ex)

On a remote object, addrefs the remote instance

7.4.33

SIDL_C_INLINE_DECL sidl_bool
bHYPRE_StructSMG__isRemote (bHYPRE_StructSMG self, sidl_BaseInterface* _ex)

TRUE if this object is remote, false if local

7.4.34

sidl_bool
bHYPRE_StructSMG__isLocal (bHYPRE_StructSMG self, sidl_BaseInterface* _ex)

TRUE if this object is remote, false if local

7.4.35

struct bHYPRE_StructSMG__object*
bHYPRE_StructSMG__rmicast (void* obj, struct sidl_BaseInterface__object** _ex)

Cast method for interface and class type conversions

7.4.36

struct bHYPRE_StructSMG__object*
bHYPRE_StructSMG__connectI (const char* url, sidl_bool ar, struct sidl_BaseInterface__object** _ex)

RMI connector function for the class. (no addref)

8

SemiStructured Matrix Solvers

Names

These solvers use semi-structured matrix/vector storage schemes.

8.1

SemiStruct DiagScale Solver

Names

8.1.1

struct **bHYPRE_SStructDiagScale__object**

Symbol "bHYPRESStructDiagScale" (version 100)

8.1.2

struct bHYPRE_SStructDiagScale__object*
bHYPRE_SStructDiagScale__create (sidl_BaseInterface* _ex)

Constructor function for the class

8.1.3

bHYPRE_SStructDiagScale
bHYPRE_SStructDiagScale__createRemote (const char* url,
sidl_BaseInterface* _ex)

RMI constructor function for the class

8.1.4

bHYPRE_SStructDiagScale
bHYPRE_SStructDiagScale__wrapObj (void* data, sidl_BaseInterface* _ex)

Wraps up the private data struct pointer (struct bHYPRE_SStructDiagScale__data) passed in rather than running the constructor

8.1.5

bHYPRE_SStructDiagScale
bHYPRE_SStructDiagScale__connect (const char* , sidl_BaseInterface* _ex)

RMI connector function for the class(addrefs)

8.1.6

bHYPRE_SStructDiagScale
bHYPRE_SStructDiagScale_Create (bHYPRE_MPICommunicator mpi_comm, bHYPRE_Operator A, sidl_BaseInterface* _ex)

This function is the preferred way to create a SStruct DiagScale solver.

8.1.7

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructDiagScale_SetOperator (bHYPRE_SStructDiagScale self, bHYPRE_Operator A, sidl_BaseInterface* _ex)

Set the operator for the linear system being solved. DEPRECATED. use Create

8.1.8

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructDiagScale_SetTolerance (bHYPRE_SStructDiagScale self, double tolerance, sidl_BaseInterface* eex)

(Optional) Set the convergence tolerance. DEPRECATED. use SetDoubleParameter

8.1.9

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructDiagScale_SetMaxIterations (bHYPRE_SStructDiagScale self, int32_t max_iterations, sidl_BaseInterface* _ex)

(Optional) Set maximum number of iterations. DEPRECATED use SetIntParameter

8.1.10

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructDiagScale_SetLogging (bHYPRE_SStructDiagScale self, int32_t level, sidl_BaseInterface* _ex)

(Optional) Set the *logging level*, specifying the degree of additional informational data to be accumulated. Does nothing by default (level = 0). Other levels (if any) are implementation-specific. Must be called before `Setup` and `Apply`. DEPRECATED use SetIntParameter

8.1.11

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructDiagScale_SetPrintLevel (bHYPRE_SStructDiagScale self, int32_t level, sidl_BaseInterface* _ex)

(Optional) Set the *print level*, specifying the degree of informational data to be printed either to the screen or to a file. Does nothing by default (level=0). Other levels (if any) are implementation-specific. Must be called before `Setup` and `Apply`. DEPRECATED use SetIntParameter

8.1.12

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructDiagScale_GetNumIterations (bHYPRE_SStructDiagScale self, int32_t* num_iterations, sidl_BaseInterface* _ex)

(Optional) Return the number of iterations taken

8.1.13

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructDiagScale_GetRelResidualNorm (
bHYPRE_SStructDiagScale self, double* norm, sidl_BaseInterface* _ex)

(Optional) Return the norm of the relative residual

8.1.14

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructDiagScale_SetCommunicator (bHYPRE_SStructDiagScale self, bHYPRE_MPICommunicator mpi_comm, sidl_BaseInterface* _ex)

Set the MPI Communicator. DEPRECATED, use Create:

8.1.15

SIDL_C_INLINE_DECL void
bHYPRE_SStructDiagScale_Destroy (bHYPRE_SStructDiagScale self, sidl_BaseInterface* _ex)

The Destroy function doesn't necessarily destroy anything. It is just another name for deleteRef. Thus it decrements the object's reference count. The Babel memory management system will destroy the object if the reference count goes to zero.

8.1.16

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructDiagScale_SetIntParameter (bHYPRE_SStructDiagScale self, const char* name, int32_t value, sidl_BaseInterface* _ex)

Set the int parameter associated with `name`

8.1.17

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructDiagScale_SetDoubleParameter (bHYPRE_SStructDiagScale self, const char* name, double value, sidl_BaseInterface* _ex)

Set the double parameter associated with `name`

8.1.18

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructDiagScale_SetStringParameter (
bHYPRE_SStructDiagScale self, const char* name, const char* value,
sidl_BaseInterface* _ex)

Set the string parameter associated with `name`

8.1.19

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructDiagScale_SetIntArray1Parameter (
bHYPRE_SStructDiagScale self, const char* name, int32_t* value, int32_t nvalues,
sidl_BaseInterface* _ex)

Set the int 1-D array parameter associated with `name`

8.1.20

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructDiagScale_SetIntArray2Parameter (
bHYPRE_SStructDiagScale self, const char* name, struct sidl_int__array* value,
sidl_BaseInterface* _ex)

Set the int 2-D array parameter associated with `name`

8.1.21

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructDiagScale_SetDoubleArray1Parameter (
bHYPRE_SStructDiagScale self, const char* name, double* value, int32_t nvalues,
sidl_BaseInterface* _ex)

Set the double 1-D array parameter associated with `name`

8.1.22

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructDiagScale_SetDoubleArray2Parameter (
bHYPRE_SStructDiagScale self, const char* name, struct sidl_double__array*
value, sidl_BaseInterface* _ex)

Set the double 2-D array parameter associated with `name`

8.1.23

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructDiagScale_GetIntValue (bHYPRE_SStructDiagScale self,
const char* name, int32_t* value, sidl_BaseInterface* _ex)

Set the int parameter associated with `name`

8.1.24

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructDiagScale_GetDoubleValue (bHYPRE_SStructDiagScale
self, const char* name, double* value, sidl_BaseInterface* _ex)

Get the double parameter associated with `name`

8.1.25

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructDiagScale_Setup (bHYPRE_SStructDiagScale self,
bHYPRE_Vector b, bHYPRE_Vector x, sidl_BaseInterface* _ex)

(Optional) Do any preprocessing that may be necessary in order to execute `Apply`

8.1.26

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructDiagScale_Apply (bHYPRE_SStructDiagScale self,
bHYPRE_Vector b, bHYPRE_Vector* x, sidl_BaseInterface* _ex)

Apply the operator to `b`, returning `x`

8.1.27

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructDiagScale_ApplyAdjoint (bHYPRE_SStructDiagScale self,
bHYPRE_Vector b, bHYPRE_Vector* x, sidl_BaseInterface* _ex)

Apply the adjoint of the operator to `b`, returning `x`

8.1.28

struct bHYPRE_SStructDiagScale__object*
bHYPRE_SStructDiagScale__cast (void* obj, sidl_BaseInterface* _ex)

Cast method for interface and class type conversions

8.1.29

void*
bHYPRE_SStructDiagScale__cast2 (void* obj, const char* type,
sidl_BaseInterface* _ex)

String cast method for interface and class type conversions

8.1.30

SIDL_C_INLINE_DECL void
bHYPRE_SStructDiagScale__exec (bHYPRE_SStructDiagScale self, const char* methodName, sidl_rmi_Call inArgs, sidl_rmi_Return outArgs, sidl_BaseInterface* _ex)

Select and execute a method by name

8.1.31

SIDL_C_INLINE_DECL char*
bHYPRE_SStructDiagScale__getURL (bHYPRE_SStructDiagScale self, sidl_BaseInterface* _ex)

Get the URL of the Implementation of this object (for RMI)

8.1.32

SIDL_C_INLINE_DECL void
bHYPRE_SStructDiagScale__raddRef (bHYPRE_SStructDiagScale self, sidl_BaseInterface* _ex)

On a remote object, addrefs the remote instance

8.1.33

SIDL_C_INLINE_DECL sidl_bool
bHYPRE_SStructDiagScale__isRemote (bHYPRE_SStructDiagScale self, sidl_BaseInterface* _ex)

TRUE if this object is remote, false if local

8.1.34

sidl_bool
bHYPRE_SStructDiagScale__isLocal (bHYPRE_SStructDiagScale self, sidl_BaseInterface* _ex)

TRUE if this object is remote, false if local

8.1.35

struct bHYPRE_SStructDiagScale__object*
bHYPRE_SStructDiagScale__rmicast (void* obj, struct sidl_BaseInterface__object** _ex)

Cast method for interface and class type conversions

8.1.36

struct bHYPRE_SStructDiagScale__object*
bHYPRE_SStructDiagScale__connectI (const char* url, sidl_bool ar, struct sidl_BaseInterface__object** _ex)

RMI connector function for the class. (no addref)

8.2

Struct Split Solver

Names

8.2.1

struct **bHYPRE_SStructSplit__object**

Symbol "bHYPRESStructSplit" (version 100)

The SStructSplit solver requires a SStruct matrix.

8.2.2

struct bHYPRE_SStructSplit__object*
bHYPRE_SStructSplit__create (sidl_BaseInterface* _ex)

Constructor function for the class

8.2.3

bHYPRE_SStructSplit
bHYPRE_SStructSplit__createRemote (const char* url, sidl_BaseInterface* _ex)

RMI constructor function for the class

8.2.4

bHYPRE_SStructSplit
bHYPRE_SStructSplit__wrapObj (void* data, sidl_BaseInterface* _ex)

Wraps up the private data struct pointer (struct bHYPRE_SStructSplit__data) passed in rather than running the constructor

8.2.5

bHYPRE_SStructSplit
bHYPRE_SStructSplit__connect (const char* , sidl_BaseInterface* _ex)

RMI connector function for the class(addrefs)

8.2.6

bHYPRE_SStructSplit
bHYPRE_SStructSplit_Create (bHYPRE_MPICommunicator mpi_comm, bHYPRE_Operator A, sidl_BaseInterface* _ex)

This function is the preferred way to create a SStruct Split solver.

8.2.7

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructSplit_SetOperator (bHYPRE_SStructSplit self, bHYPRE_Operator A, sidl_BaseInterface* _ex)

Set the operator for the linear system being solved. DEPRECATED. use Create

8.2.8

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructSplit_SetTolerance (bHYPRE_SStructSplit self, double tolerance, sidl_BaseInterface* _ex)

(Optional) Set the convergence tolerance. DEPRECATED. use SetDoubleParameter

8.2.9

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructSplit_SetMaxIterations (bHYPRE_SStructSplit self, int32_t max_iterations, sidl_BaseInterface* _ex)

(Optional) Set maximum number of iterations. DEPRECATED use SetIntParameter

8.2.10

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructSplit_SetLogging (bHYPRE_SStructSplit self, int32_t level, sidl_BaseInterface* _ex)

(Optional) Set the *logging level*, specifying the degree of additional informational data to be accumulated. Does nothing by default (level = 0). Other levels (if any) are implementation-specific. Must be called before `Setup` and `Apply`. DEPRECATED use SetIntParameter

8.2.11

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructSplit_SetPrintLevel (bHYPRE_SStructSplit self, int32_t level, sidl_BaseInterface* _ex)

(Optional) Set the *print level*, specifying the degree of informational data to be printed either to the screen or to a file. Does nothing by default (level=0). Other levels (if any) are implementation-specific. Must be called before `Setup` and `Apply`. DEPRECATED use SetIntParameter

8.2.12

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructSplit_GetNumIterations (bHYPRE_SStructSplit self, int32_t* num_iterations, sidl_BaseInterface* _ex)

(Optional) Return the number of iterations taken

8.2.13

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructSplit_GetRelResidualNorm (bHYPRE_SStructSplit self, double* norm, sidl_BaseInterface* _ex)

(Optional) Return the norm of the relative residual

8.2.14

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructSplit_SetCommunicator (bHYPRE_SStructSplit self, bHYPRE_MPICommunicator mpi_comm, sidl_BaseInterface* _ex)

Set the MPI Communicator. DEPRECATED, use Create:

8.2.15

SIDL_C_INLINE_DECL void
bHYPRE_SStructSplit_Destroy (bHYPRE_SStructSplit self, sidl_BaseInterface* _ex)

The Destroy function doesn't necessarily destroy anything. It is just another name for deleteRef. Thus it decrements the object's reference count. The Babel memory management system will destroy the object if the reference count goes to zero.

8.2.16

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructSplit_SetIntParameter (bHYPRE_SStructSplit self, const char* name, int32_t value, sidl_BaseInterface* _ex)

Set the int parameter associated with `name`

8.2.17

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructSplit_SetDoubleParameter (bHYPRE_SStructSplit self, const char* name, double value, sidl_BaseInterface* _ex)

Set the double parameter associated with `name`

8.2.18

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructSplit_SetStringParameter (bHYPRE_SStructSplit self, const char* name, const char* value, sidl_BaseInterface* eex)

Set the string parameter associated with `name`

8.2.19

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructSplit_SetIntArray1Parameter (bHYPRE_SStructSplit self, const char* name, int32_t* value, int32_t nvalues, sidl_BaseInterface* _ex)

Set the int 1-D array parameter associated with `name`

8.2.20

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructSplit_SetIntArray2Parameter (bHYPRE_SStructSplit self, const char* name, struct sidl_int__array* value, sidl_BaseInterface* _ex)

Set the int 2-D array parameter associated with `name`

8.2.21

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructSplit_SetDoubleArray1Parameter (bHYPRE_SStructSplit self, const char* name, double* value, int32_t nvalues, sidl_BaseInterface* _ex)

Set the double 1-D array parameter associated with `name`

8.2.22

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructSplit_SetDoubleArray2Parameter (bHYPRE_SStructSplit self, const char* name, struct sidl_double__array* value, sidl_BaseInterface* _ex)

Set the double 2-D array parameter associated with `name`

8.2.23

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructSplit_GetIntValue (bHYPRE_SStructSplit self, const char* name, int32_t* value, sidl_BaseInterface* _ex)

Set the int parameter associated with `name`

8.2.24

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructSplit_GetDoubleValue (bHYPRE_SStructSplit self, const char* name, double* value, sidl_BaseInterface* _ex)

Get the double parameter associated with `name`

8.2.25

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructSplit_Setup (bHYPRE_SStructSplit self, bHYPRE_Vector b, bHYPRE_Vector x, sidl_BaseInterface* _ex)

(Optional) Do any preprocessing that may be necessary in order to execute `Apply`

8.2.26

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructSplit_Apply (bHYPRE_SStructSplit self, bHYPRE_Vector b, bHYPRE_Vector* x, sidl_BaseInterface* _ex)

Apply the operator to `b`, returning `x`

8.2.27

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructSplit_ApplyAdjoint (bHYPRE_SStructSplit self, bHYPRE_Vector b, bHYPRE_Vector* x, sidl_BaseInterface* _ex)

Apply the adjoint of the operator to `b`, returning `x`

8.2.28

struct bHYPRE_SStructSplit__object*
bHYPRE_SStructSplit__cast (void* obj, sidl_BaseInterface* _ex)

Cast method for interface and class type conversions

8.2.29

void*
bHYPRE_SStructSplit__cast2 (void* obj, const char* type, sidl_BaseInterface* _ex)

String cast method for interface and class type conversions

8.2.30

SIDL_C_INLINE_DECL void
bHYPRE_SStructSplit__exec (bHYPRE_SStructSplit self, const char* methodName, sidl_rmi_Call inArgs, sidl_rmi_Return outArgs, sidl_BaseInterface* _ex)

Select and execute a method by name

8.2.31

SIDL_C_INLINE_DECL char*
bHYPRE_SStructSplit__getURL (bHYPRE_SStructSplit self, sidl_BaseInterface* _ex)

Get the URL of the Implementation of this object (for RMI)

8.2.32

SIDL_C_INLINE_DECL void
bHYPRE_SStructSplit__raddRef (bHYPRE_SStructSplit self, sidl_BaseInterface* _ex)

On a remote object, addrefs the remote instance

8.2.33

SIDL_C_INLINE_DECL sidl_bool
bHYPRE_SStructSplit__isRemote (bHYPRE_SStructSplit self, sidl_BaseInterface* _ex)

TRUE if this object is remote, false if local

8.2.34

sidl_bool
bHYPRE_SStructSplit__isLocal (bHYPRE_SStructSplit self,
sidl_BaseInterface* _ex)

TRUE if this object is remote, false if local

8.2.35

struct bHYPRE_SStructSplit__object*
bHYPRE_SStructSplit__rmicast (void* obj, struct sidl_BaseInterface__object**
_ex)

Cast method for interface and class type conversions

8.2.36

struct bHYPRE_SStructSplit__object*
bHYPRE_SStructSplit__connectI (const char* url, sidl_bool ar, struct
sidl_BaseInterface__object** _ex)

RMI connector function for the class. (no addref)

9

PreconditionedSolver Interface

Names

9.1

struct **bHYPRE_PreconditionedSolver__object**

Symbol "bHYPREPreconditionedSolver" (version 100)

9.2

bHYPRE_PreconditionedSolver **bHYPRE_PreconditionedSolver__connect** (const char* , sidl_BaseInterface* _ex)

RMI connector function for the class(addrefs)

9.3

SIDL_C_INLINE_DECL int32_t
bHYPRE_PreconditionedSolver_SetPreconditioner (
bHYPRE_PreconditionedSolver self, bHYPRE_Solver s, sidl_BaseInterface* _ex)

Set the preconditioner

9.4

SIDL_C_INLINE_DECL int32_t
bHYPRE_PreconditionedSolver_GetPreconditioner (
bHYPRE_PreconditionedSolver self, bHYPRE_Solver* s, sidl_BaseInterface* _ex)

Method: GetPreconditioner[]

9.5

SIDL_C_INLINE_DECL int32_t
bHYPRE_PreconditionedSolver_Clone (bHYPRE_PreconditionedSolver self,
bHYPRE_PreconditionedSolver* x, sidl_BaseInterface* _ex)

Method: Clone[]

9.6

struct bHYPRE_PreconditionedSolver__object*
bHYPRE_PreconditionedSolver__cast (void* obj, sidl_BaseInterface* _ex)

Cast method for interface and class type conversions

9.7

void*
bHYPRE_PreconditionedSolver__cast2 (void* obj, const char* type, sidl_BaseInterface* _ex)

String cast method for interface and class type conversions

9.8

SIDL_C_INLINE_DECL void
bHYPRE_PreconditionedSolver__exec (bHYPRE_PreconditionedSolver self, const char* methodName, sidl_rmi_Call inArgs, sidl_rmi_Return outArgs, sidl_BaseInterface* _ex)

Select and execute a method by name

9.9

SIDL_C_INLINE_DECL char*
bHYPRE_PreconditionedSolver__getURL (bHYPRE_PreconditionedSolver self, sidl_BaseInterface* _ex)

Get the URL of the Implementation of this object (for RMI)

9.10

SIDL_C_INLINE_DECL void
bHYPRE_PreconditionedSolver__raddRef (bHYPRE_PreconditionedSolver self, sidl_BaseInterface* _ex)

On a remote object, addrefs the remote instance

9.11

SIDL_C_INLINE_DECL sidl_bool
bHYPRE_PreconditionedSolver__isRemote (bHYPRE_PreconditionedSolver self, sidl_BaseInterface* _ex)

TRUE if this object is remote, false if local

9.12

sidl_bool
bHYPRE_PreconditionedSolver__isLocal (bHYPRE_PreconditionedSolver self, sidl_BaseInterface* _ex)

TRUE if this object is remote, false if local

9.13

struct bHYPRE_PreconditionedSolver__object*
bHYPRE_PreconditionedSolver__rmicast (void* obj, struct sidl_BaseInterface__object** _ex)

Cast method for interface and class type conversions

9.14

struct bHYPRE_PreconditionedSolver__object*
bHYPRE_PreconditionedSolver__connectI (const char* url, sidl_bool ar, struct sidl_BaseInterface__object** _ex)

RMI connector function for the class. (no addref)

10

Preconditioned Solvers

Names

10.1

PCG Preconditioned Solver

Names

10.1.1

struct **bHYPRE_PCG__object**

Symbol "bHYPREPCG" (version 100)

PCG solver. This calls Babel-interface matrix and vector functions, so it will work with any consistent matrix, vector, and preconditioner classes.

10.1.2

struct bHYPRE_PCG__object* **bHYPRE_PCG__create** (sidl_BaseInterface* _ex)

Constructor function for the class

10.1.3

bHYPRE_PCG
bHYPRE_PCG__createRemote (const char* url, sidl_BaseInterface* _ex)

RMI constructor function for the class

10.1.4

bHYPRE_PCG **bHYPRE_PCG__wrapObj** (void* data, sidl_BaseInterface* _ex)

Wraps up the private data struct pointer (struct bHYPRE_PCG__data) passed in rather than running the constructor

10.1.5

bHYPRE_PCG **bHYPRE_PCG__connect** (const char* , sidl_BaseInterface* _ex)

RMI connector function for the class(addrefs)

10.1.6

bHYPRE_PCG
bHYPRE_PCG_Create (bHYPRE_MPICommunicator mpi_comm,
bHYPRE_Operator A, sidl_BaseInterface* _ex)

This function is the preferred way to create a PCG solver.

10.1.7

SIDL_C_INLINE_DECL int32_t
bHYPRE_PCG_SetPreconditioner (bHYPRE_PCG self, bHYPRE_Solver s,
sidl_BaseInterface* _ex)

Set the preconditioner

10.1.8

SIDL_C_INLINE_DECL int32_t
bHYPRE_PCG_GetPreconditioner (bHYPRE_PCG self, bHYPRE_Solver* s,
sidl_BaseInterface* _ex)

Method: GetPreconditioner[]

10.1.9

SIDL_C_INLINE_DECL int32_t
bHYPRE_PCG_Clone (bHYPRE_PCG self, bHYPRE_PreconditionedSolver* x, sidl_BaseInterface* _ex)

Method: Clone[]

10.1.10

SIDL_C_INLINE_DECL int32_t
bHYPRE_PCG_SetOperator (bHYPRE_PCG self, bHYPRE_Operator A, sidl_BaseInterface* _ex)

Set the operator for the linear system being solved. DEPRECATED. use Create

10.1.11

SIDL_C_INLINE_DECL int32_t
bHYPRE_PCG_SetTolerance (bHYPRE_PCG self, double tolerance, sidl_BaseInterface* _ex)

(Optional) Set the convergence tolerance. DEPRECATED. use SetDoubleParameter

10.1.12

SIDL_C_INLINE_DECL int32_t
bHYPRE_PCG_SetMaxIterations (bHYPRE_PCG self, int32_t max_iterations, sidl_BaseInterface* _ex)

(Optional) Set maximum number of iterations. DEPRECATED use SetIntParameter

10.1.13

SIDL_C_INLINE_DECL int32_t
bHYPRE_PCG_SetLogging (bHYPRE_PCG self, int32_t level,
sidl_BaseInterface* _ex)

(Optional) Set the *logging level*, specifying the degree of additional informational data to be accumulated. Does nothing by default (level = 0). Other levels (if any) are implementation-specific. Must be called before `Setup` and `Apply`. DEPRECATED use SetIntParameter

10.1.14

SIDL_C_INLINE_DECL int32_t
bHYPRE_PCG_SetPrintLevel (bHYPRE_PCG self, int32_t level,
sidl_BaseInterface* _ex)

(Optional) Set the *print level*, specifying the degree of informational data to be printed either to the screen or to a file. Does nothing by default (level=0). Other levels (if any) are implementation-specific. Must be called before `Setup` and `Apply`. DEPRECATED use SetIntParameter

10.1.15

SIDL_C_INLINE_DECL int32_t
bHYPRE_PCG_GetNumIterations (bHYPRE_PCG self, int32_t*
num_iterations, sidl_BaseInterface* _ex)

(Optional) Return the number of iterations taken

10.1.16

SIDL_C_INLINE_DECL int32_t
bHYPRE_PCG_GetRelResidualNorm (bHYPRE_PCG self, double* norm,
sidl_BaseInterface* _ex)

(Optional) Return the norm of the relative residual

10.1.17

SIDL_C_INLINE_DECL int32_t
bHYPRE_PCG_SetCommunicator (bHYPRE_PCG self,
bHYPRE_MPICommunicator mpi_comm, sidl_BaseInterface* _ex)

Set the MPI Communicator. DEPRECATED, use Create:

10.1.18

SIDL_C_INLINE_DECL void
bHYPRE_PCG_Destroy (bHYPRE_PCG self, sidl_BaseInterface* _ex)

The Destroy function doesn't necessarily destroy anything. It is just another name for deleteRef. Thus it decrements the object's reference count. The Babel memory management system will destroy the object if the reference count goes to zero.

10.1.19

SIDL_C_INLINE_DECL int32_t
bHYPRE_PCG_SetIntParameter (bHYPRE_PCG self, const char* name,
int32_t value, sidl_BaseInterface* _ex)

Set the int parameter associated with `name`

10.1.20

SIDL_C_INLINE_DECL int32_t
bHYPRE_PCG_SetDoubleParameter (bHYPRE_PCG self, const char*
name, double value, sidl_BaseInterface* _ex)

Set the double parameter associated with `name`

10.1.21

SIDL_C_INLINE_DECL int32_t
bHYPRE_PCG_SetStringParameter (bHYPRE_PCG self, const char* name, const char* value, sidl_BaseInterface* _ex)

Set the string parameter associated with `name`

10.1.22

SIDL_C_INLINE_DECL int32_t
bHYPRE_PCG_SetIntArray1Parameter (bHYPRE_PCG self, const char* name, int32_t* value, int32_t nvalues, sidl_BaseInterface* _ex)

Set the int 1-D array parameter associated with `name`

10.1.23

SIDL_C_INLINE_DECL int32_t
bHYPRE_PCG_SetIntArray2Parameter (bHYPRE_PCG self, const char* name, struct sidl_int__array* value, sidl_BaseInterface* _ex)

Set the int 2-D array parameter associated with `name`

10.1.24

SIDL_C_INLINE_DECL int32_t
bHYPRE_PCG_SetDoubleArray1Parameter (bHYPRE_PCG self, const char* name, double* value, int32_t nvalues, sidl_BaseInterface* _ex)

Set the double 1-D array parameter associated with `name`

10.1.25

SIDL_C_INLINE_DECL int32_t
bHYPRE_PCG_SetDoubleArray2Parameter (bHYPRE_PCG self, const char* name, struct sidl_double__array* value, sidl_BaseInterface* _ex)

Set the double 2-D array parameter associated with `name`

10.1.26

SIDL_C_INLINE_DECL int32_t
bHYPRE_PCG_GetIntValue (bHYPRE_PCG self, const char* name, int32_t* value, sidl_BaseInterface* _ex)

Set the int parameter associated with `name`

10.1.27

SIDL_C_INLINE_DECL int32_t
bHYPRE_PCG_GetDoubleValue (bHYPRE_PCG self, const char* name, double* value, sidl_BaseInterface* _ex)

Get the double parameter associated with `name`

10.1.28

SIDL_C_INLINE_DECL int32_t
bHYPRE_PCG_Setup (bHYPRE_PCG self, bHYPRE_Vector b, bHYPRE_Vector x, sidl_BaseInterface* _ex)

(Optional) Do any preprocessing that may be necessary in order to execute `Apply`

10.1.29

SIDL_C_INLINE_DECL int32_t
bHYPRE_PCG_Apply (bHYPRE_PCG self, bHYPRE_Vector b,
bHYPRE_Vector* x, sidl_BaseInterface* _ex)

Apply the operator to `b`, returning `x`

10.1.30

SIDL_C_INLINE_DECL int32_t
bHYPRE_PCG_ApplyAdjoint (bHYPRE_PCG self, bHYPRE_Vector b,
bHYPRE_Vector* x, sidl_BaseInterface* _ex)

Apply the adjoint of the operator to `b`, returning `x`

10.1.31

struct bHYPRE_PCG__object*
bHYPRE_PCG__cast (void* obj, sidl_BaseInterface* _ex)

Cast method for interface and class type conversions

10.1.32

void*
bHYPRE_PCG__cast2 (void* obj, const char* type, sidl_BaseInterface* _ex)

String cast method for interface and class type conversions

10.1.33

SIDL_C_INLINE_DECL void
bHYPRE_PCG__exec (bHYPRE_PCG self, const char* methodName,
sidl_rmi_Call inArgs, sidl_rmi_Return outArgs, sidl_BaseInterface* _ex)

Select and execute a method by name

10.1.34

SIDL_C_INLINE_DECL char*
bHYPRE_PCG__getURL (bHYPRE_PCG self, sidl_BaseInterface* _ex)

Get the URL of the Implementation of this object (for RMI)

10.1.35

SIDL_C_INLINE_DECL void
bHYPRE_PCG__raddRef (bHYPRE_PCG self, sidl_BaseInterface* _ex)

On a remote object, addrefs the remote instance

10.1.36

SIDL_C_INLINE_DECL sidl_bool
bHYPRE_PCG__isRemote (bHYPRE_PCG self, sidl_BaseInterface* _ex)

TRUE if this object is remote, false if local

10.1.37

sidl_bool
bHYPRE_PCG__isLocal (bHYPRE_PCG self, sidl_BaseInterface* _ex)

TRUE if this object is remote, false if local

10.1.38

struct bHYPRE_PCG__object*
bHYPRE_PCG__rmicast (void* obj, struct sidl_BaseInterface__object** _ex)

Cast method for interface and class type conversions

10.1.39

struct bHYPRE_PCG__object*
bHYPRE_PCG__connectI (const char* url, sidl_bool ar, struct sidl_BaseInterface__object** _ex)

RMI connector function for the class. (no addref)

10.2

GMRES Preconditioned Solver

Names

10.2.1

struct **bHYPRE_GMRES__object**

Symbol "bHYPREGMRES" (version 100)

GMRES solver. This calls Babel-interface matrix and vector functions, so it will work with any consistent matrix, vector, and preconditioner classes.

10.2.2

struct bHYPRE_GMRES__object*
bHYPRE_GMRES__create (sidl_BaseInterface* _ex)

Constructor function for the class

10.2.3

bHYPRE_GMRES
bHYPRE_GMRES__createRemote (const char* url, sidl_BaseInterface* _ex)

RMI constructor function for the class

10.2.4

bHYPRE_GMRES
bHYPRE_GMRES__wrapObj (void* data, sidl_BaseInterface* _ex)

Wraps up the private data struct pointer (struct bHYPRE_GMRES__data) passed in rather than running the constructor

10.2.5

bHYPRE_GMRES
bHYPRE_GMRES__connect (const char* , sidl_BaseInterface* _ex)

RMI connector function for the class(addrefs)

10.2.6

bHYPRE_GMRES
bHYPRE_GMRES_Create (bHYPRE_MPICommunicator mpi_comm,
bHYPRE_Operator A, sidl_BaseInterface* _ex)

This function is the preferred way to create a GMRES solver.

10.2.7

SIDL_C_INLINE_DECL int32_t
bHYPRE_GMRES_SetPreconditioner (bHYPRE_GMRES self,
bHYPRE_Solver s, sidl_BaseInterface* _ex)

Set the preconditioner

10.2.8

SIDL_C_INLINE_DECL int32_t
bHYPRE_GMRES_GetPreconditioner (bHYPRE_GMRES self,
bHYPRE_Solver* s, sidl_BaseInterface* _ex)

Method: GetPreconditioner[]

10.2.9

SIDL_C_INLINE_DECL int32_t
bHYPRE_GMRES_Clone (bHYPRE_GMRES self,
bHYPRE_PreconditionedSolver* x, sidl_BaseInterface* _ex)

Method: Clone[]

10.2.10

SIDL_C_INLINE_DECL int32_t
bHYPRE_GMRES_SetOperator (bHYPRE_GMRES self, bHYPRE_Operator A, sidl_BaseInterface* _ex)

Set the operator for the linear system being solved. DEPRECATED. use Create

10.2.11

SIDL_C_INLINE_DECL int32_t
bHYPRE_GMRES_SetTolerance (bHYPRE_GMRES self, double tolerance, sidl_BaseInterface* _ex)

(Optional) Set the convergence tolerance. DEPRECATED. use SetDoubleParameter

10.2.12

SIDL_C_INLINE_DECL int32_t
bHYPRE_GMRES_SetMaxIterations (bHYPRE_GMRES self, int32_t max_iterations, sidl_BaseInterface* _ex)

(Optional) Set maximum number of iterations. DEPRECATED use SetIntParameter

10.2.13

SIDL_C_INLINE_DECL int32_t
bHYPRE_GMRES_SetLogging (bHYPRE_GMRES self, int32_t level, sidl_BaseInterface* _ex)

(Optional) Set the *logging level*, specifying the degree of additional informational data to be accumulated. Does nothing by default (level = 0). Other levels (if any) are implementation-specific. Must be called before `Setup` and `Apply`. DEPRECATED use SetIntParameter

10.2.14

SIDL_C_INLINE_DECL int32_t
bHYPRE_GMRES_SetPrintLevel (bHYPRE_GMRES self, int32_t level, sidl_BaseInterface* _ex)

(Optional) Set the *print level*, specifying the degree of informational data to be printed either to the screen or to a file. Does nothing by default (level=0). Other levels (if any) are implementation-specific. Must be called before `Setup` and `Apply`. DEPRECATED use SetIntParameter

10.2.15

SIDL_C_INLINE_DECL int32_t
bHYPRE_GMRES_GetNumIterations (bHYPRE_GMRES self, int32_t* num_iterations, sidl_BaseInterface* _ex)

(Optional) Return the number of iterations taken

10.2.16

SIDL_C_INLINE_DECL int32_t
bHYPRE_GMRES_GetRelResidualNorm (bHYPRE_GMRES self, double* norm, sidl_BaseInterface* _ex)

(Optional) Return the norm of the relative residual

10.2.17

SIDL_C_INLINE_DECL int32_t
bHYPRE_GMRES_SetCommunicator (bHYPRE_GMRES self, bHYPRE_MPICommunicator mpi_comm, sidl_BaseInterface* _ex)

Set the MPI Communicator. DEPRECATED, use Create:

10.2.18

SIDL_C_INLINE_DECL void
bHYPRE_GMRES_Destroy (bHYPRE_GMRES self, sidl_BaseInterface* _ex)

The Destroy function doesn't necessarily destroy anything. It is just another name for deleteRef. Thus it decrements the object's reference count. The Babel memory management system will destroy the object if the reference count goes to zero.

10.2.19

SIDL_C_INLINE_DECL int32_t
bHYPRE_GMRES_SetIntParameter (bHYPRE_GMRES self, const char* name, int32_t value, sidl_BaseInterface* _ex)

Set the int parameter associated with `name`

10.2.20

SIDL_C_INLINE_DECL int32_t
bHYPRE_GMRES_SetDoubleParameter (bHYPRE_GMRES self, const char* name, double value, sidl_BaseInterface* _ex)

Set the double parameter associated with `name`

10.2.21

SIDL_C_INLINE_DECL int32_t
bHYPRE_GMRES_SetStringParameter (bHYPRE_GMRES self, const char* name, const char* value, sidl_BaseInterface* _ex)

Set the string parameter associated with `name`

10.2.22

SIDL_C_INLINE_DECL int32_t
bHYPRE_GMRES_SetIntArray1Parameter (bHYPRE_GMRES self, const char* name, int32_t* value, int32_t nvalues, sidl_BaseInterface* _ex)

Set the int 1-D array parameter associated with `name`

10.2.23

SIDL_C_INLINE_DECL int32_t
bHYPRE_GMRES_SetIntArray2Parameter (bHYPRE_GMRES self, const char* name, struct sidl_int__array* value, sidl_BaseInterface* _ex)

Set the int 2-D array parameter associated with `name`

10.2.24

SIDL_C_INLINE_DECL int32_t
bHYPRE_GMRES_SetDoubleArray1Parameter (bHYPRE_GMRES self, const char* name, double* value, int32_t nvalues, sidl_BaseInterface* _ex)

Set the double 1-D array parameter associated with `name`

10.2.25

SIDL_C_INLINE_DECL int32_t
bHYPRE_GMRES_SetDoubleArray2Parameter (bHYPRE_GMRES self, const char* name, struct sidl_double__array* value, sidl_BaseInterface* _ex)

Set the double 2-D array parameter associated with `name`

10.2.26

SIDL_C_INLINE_DECL int32_t
bHYPRE_GMRES_GetIntValue (bHYPRE_GMRES self, const char* name, int32_t* value, sidl_BaseInterface* _ex)

Set the int parameter associated with `name`

10.2.27

SIDL_C_INLINE_DECL int32_t
bHYPRE_GMRES_GetDoubleValue (bHYPRE_GMRES self, const char* name, double* value, sidl_BaseInterface* _ex)

Get the double parameter associated with `name`

10.2.28

SIDL_C_INLINE_DECL int32_t
bHYPRE_GMRES_Setup (bHYPRE_GMRES self, bHYPRE_Vector b, bHYPRE_Vector x, sidl_BaseInterface* _ex)

(Optional) Do any preprocessing that may be necessary in order to execute `Apply`

10.2.29

SIDL_C_INLINE_DECL int32_t
bHYPRE_GMRES_Apply (bHYPRE_GMRES self, bHYPRE_Vector b, bHYPRE_Vector* x, sidl_BaseInterface* _ex)

Apply the operator to `b`, returning `x`

10.2.30

SIDL_C_INLINE_DECL int32_t
bHYPRE_GMRES_ApplyAdjoint (bHYPRE_GMRES self, bHYPRE_Vector b, bHYPRE_Vector* x, sidl_BaseInterface* _ex)

Apply the adjoint of the operator to `b`, returning `x`

10.2.31

struct bHYPRE_GMRES__object*
bHYPRE_GMRES__cast (void* obj, sidl_BaseInterface* _ex)

Cast method for interface and class type conversions

10.2.32

void*
bHYPRE_GMRES__cast2 (void* obj, const char* type, sidl_BaseInterface* _ex)

String cast method for interface and class type conversions

10.2.33

SIDL_C_INLINE_DECL void
bHYPRE_GMRES__exec (bHYPRE_GMRES self, const char* methodName, sidl_rmi_Call inArgs, sidl_rmi_Return outArgs, sidl_BaseInterface* _ex)

Select and execute a method by name

10.2.34

SIDL_C_INLINE_DECL char*
bHYPRE_GMRES__getURL (bHYPRE_GMRES self, sidl_BaseInterface* _ex)

Get the URL of the Implementation of this object (for RMI)

10.2.35

SIDL_C_INLINE_DECL void
bHYPRE_GMRES__raddRef (bHYPRE_GMRES self, sidl_BaseInterface* _ex)

On a remote object, addrefs the remote instance

10.2.36

SIDL_C_INLINE_DECL sidl_bool
bHYPRE_GMRES__isRemote (bHYPRE_GMRES self, sidl_BaseInterface* _ex)

TRUE if this object is remote, false if local

10.2.37

sidl_bool
bHYPRE_GMRES__isLocal (bHYPRE_GMRES self, sidl_BaseInterface* _ex)

TRUE if this object is remote, false if local

10.2.38

struct bHYPRE_GMRES__object*
bHYPRE_GMRES__rmicast (void* obj, struct sidl_BaseInterface__object** _ex)

Cast method for interface and class type conversions

10.2.39

struct bHYPRE_GMRES__object*
bHYPRE_GMRES__connectI (const char* url, sidl_bool ar, struct sidl_BaseInterface__object** _ex)

RMI connector function for the class. (no addref)

10.3

BiCGSTAB Preconditioned Solver

Names

10.3.1

struct **bHYPRE_BiCGSTAB__object**

Symbol "bHYPREBiCGSTAB" (version 100)

BiCGSTAB solver. This calls Babel-interface matrix and vector functions, so it will work with any consistent matrix, vector, and preconditioner classes.

10.3.2

struct bHYPRE_BiCGSTAB__object*
bHYPRE_BiCGSTAB__create (sidl_BaseInterface* _ex)

Constructor function for the class

10.3.3

bHYPRE_BiCGSTAB
bHYPRE_BiCGSTAB__createRemote (const char* url, sidl_BaseInterface* _ex)

RMI constructor function for the class

10.3.4

bHYPRE_BiCGSTAB
bHYPRE_BiCGSTAB__wrapObj (void* data, sidl_BaseInterface* _ex)

Wraps up the private data struct pointer (struct bHYPRE_BiCGSTAB__data) passed in rather than running the constructor

10.3.5

bHYPRE_BiCGSTAB
bHYPRE_BiCGSTAB__connect (const char* , sidl_BaseInterface* _ex)

RMI connector function for the class(addrefs)

10.3.6

bHYPRE_BiCGSTAB
bHYPRE_BiCGSTAB_Create (bHYPRE_MPICommunicator mpi_comm,
bHYPRE_Operator A, sidl_BaseInterface* _ex)

This function is the preferred way to create a BiCGSTAB solver.

10.3.7

SIDL_C_INLINE_DECL int32_t
bHYPRE_BiCGSTAB_SetPreconditioner (bHYPRE_BiCGSTAB self,
bHYPRE_Solver s, sidl_BaseInterface* _ex)

Set the preconditioner

10.3.8

SIDL_C_INLINE_DECL int32_t
bHYPRE_BiCGSTAB_GetPreconditioner (bHYPRE_BiCGSTAB self,
bHYPRE_Solver* s, sidl_BaseInterface* _ex)

Method: GetPreconditioner[]

10.3.9

SIDL_C_INLINE_DECL int32_t
bHYPRE_BiCGSTAB_Clone (bHYPRE_BiCGSTAB self,
bHYPRE_PreconditionedSolver* x, sidl_BaseInterface* _ex)

Method: Clone[]

10.3.10

SIDL_C_INLINE_DECL int32_t
bHYPRE_BiCGSTAB_SetOperator (bHYPRE_BiCGSTAB self,
bHYPRE_Operator A, sidl_BaseInterface* _ex)

Set the operator for the linear system being solved. DEPRECATED. use Create

10.3.11

SIDL_C_INLINE_DECL int32_t
bHYPRE_BiCGSTAB_SetTolerance (bHYPRE_BiCGSTAB self, double
tolerance, sidl_BaseInterface* _ex)

(Optional) Set the convergence tolerance. DEPRECATED. use SetDoubleParameter

10.3.12

SIDL_C_INLINE_DECL int32_t
bHYPRE_BiCGSTAB_SetMaxIterations (bHYPRE_BiCGSTAB self, int32_t
max_iterations, sidl_BaseInterface* _ex)

(Optional) Set maximum number of iterations. DEPRECATED use SetIntParameter

10.3.13

SIDL_C_INLINE_DECL int32_t
bHYPRE_BiCGSTAB_SetLogging (bHYPRE_BiCGSTAB self, int32_t level, sidl_BaseInterface* _ex)

(Optional) Set the *logging level*, specifying the degree of additional informational data to be accumulated. Does nothing by default (level = 0). Other levels (if any) are implementation-specific. Must be called before `Setup` and `Apply`. DEPRECATED use SetIntParameter

10.3.14

SIDL_C_INLINE_DECL int32_t
bHYPRE_BiCGSTAB_SetPrintLevel (bHYPRE_BiCGSTAB self, int32_t level, sidl_BaseInterface* _ex)

(Optional) Set the *print level*, specifying the degree of informational data to be printed either to the screen or to a file. Does nothing by default (level=0). Other levels (if any) are implementation-specific. Must be called before `Setup` and `Apply`. DEPRECATED use SetIntParameter

10.3.15

SIDL_C_INLINE_DECL int32_t
bHYPRE_BiCGSTAB_GetNumIterations (bHYPRE_BiCGSTAB self, int32_t* num_iterations, sidl_BaseInterface* _ex)

(Optional) Return the number of iterations taken

10.3.16

SIDL_C_INLINE_DECL int32_t
bHYPRE_BiCGSTAB_GetRelResidualNorm (bHYPRE_BiCGSTAB self, double* norm, sidl_BaseInterface* _ex)

(Optional) Return the norm of the relative residual

10.3.17

SIDL_C_INLINE_DECL int32_t
bHYPRE_BiCGSTAB_SetCommunicator (bHYPRE_BiCGSTAB self, bHYPRE_MPICommunicator mpi_comm, sidl_BaseInterface* _ex)

Set the MPI Communicator. DEPRECATED, use Create:

10.3.18

SIDL_C_INLINE_DECL void
bHYPRE_BiCGSTAB_Destroy (bHYPRE_BiCGSTAB self, sidl_BaseInterface* _ex)

The Destroy function doesn't necessarily destroy anything. It is just another name for deleteRef. Thus it decrements the object's reference count. The Babel memory management system will destroy the object if the reference count goes to zero.

10.3.19

SIDL_C_INLINE_DECL int32_t
bHYPRE_BiCGSTAB_SetIntParameter (bHYPRE_BiCGSTAB self, const char* name, int32_t value, sidl_BaseInterface* _ex)

Set the int parameter associated with `name`

10.3.20

SIDL_C_INLINE_DECL int32_t
bHYPRE_BiCGSTAB_SetDoubleParameter (bHYPRE_BiCGSTAB self, const char* name, double value, sidl_BaseInterface* _ex)

Set the double parameter associated with `name`

10.3.21

SIDL_C_INLINE_DECL int32_t
bHYPRE_BiCGSTAB_SetStringParameter (bHYPRE_BiCGSTAB self, const char* name, const char* value, sidl_BaseInterface* _ex)

Set the string parameter associated with `name`

10.3.22

SIDL_C_INLINE_DECL int32_t
bHYPRE_BiCGSTAB_SetIntArray1Parameter (bHYPRE_BiCGSTAB self, const char* name, int32_t* value, int32_t nvalues, sidl_BaseInterface* _ex)

Set the int 1-D array parameter associated with `name`

10.3.23

SIDL_C_INLINE_DECL int32_t
bHYPRE_BiCGSTAB_SetIntArray2Parameter (bHYPRE_BiCGSTAB self, const char* name, struct sidl_int__array* value, sidl_BaseInterface* _ex)

Set the int 2-D array parameter associated with `name`

10.3.24

SIDL_C_INLINE_DECL int32_t
bHYPRE_BiCGSTAB_SetDoubleArray1Parameter (bHYPRE_BiCGSTAB self, const char* name, double* value, int32_t nvalues, sidl_BaseInterface* _ex)

Set the double 1-D array parameter associated with `name`

10.3.25

SIDL_C_INLINE_DECL int32_t
bHYPRE_BiCGSTAB_SetDoubleArray2Parameter (bHYPRE_BiCGSTAB self, const char* name, struct sidl_double__array* value, sidl_BaseInterface* _ex)

Set the double 2-D array parameter associated with `name`

10.3.26

SIDL_C_INLINE_DECL int32_t
bHYPRE_BiCGSTAB_GetIntValue (bHYPRE_BiCGSTAB self, const char* name, int32_t* value, sidl_BaseInterface* _ex)

Set the int parameter associated with `name`

10.3.27

SIDL_C_INLINE_DECL int32_t
bHYPRE_BiCGSTAB_GetDoubleValue (bHYPRE_BiCGSTAB self, const char* name, double* value, sidl_BaseInterface* _ex)

Get the double parameter associated with `name`

10.3.28

SIDL_C_INLINE_DECL int32_t
bHYPRE_BiCGSTAB_Setup (bHYPRE_BiCGSTAB self, bHYPRE_Vector b, bHYPRE_Vector x, sidl_BaseInterface* _ex)

(Optional) Do any preprocessing that may be necessary in order to execute `Apply`

10.3.29

SIDL_C_INLINE_DECL int32_t
bHYPRE_BiCGSTAB_Apply (bHYPRE_BiCGSTAB self, bHYPRE_Vector b, bHYPRE_Vector* x, sidl_BaseInterface* _ex)

Apply the operator to `b`, returning `x`

10.3.30

SIDL_C_INLINE_DECL int32_t
bHYPRE_BiCGSTAB_ApplyAdjoint (bHYPRE_BiCGSTAB self, bHYPRE_Vector b, bHYPRE_Vector* x, sidl_BaseInterface* _ex)

Apply the adjoint of the operator to `b`, returning `x`

10.3.31

struct bHYPRE_BiCGSTAB__object*
bHYPRE_BiCGSTAB__cast (void* obj, sidl_BaseInterface* _ex)

Cast method for interface and class type conversions

10.3.32

void*
bHYPRE_BiCGSTAB__cast2 (void* obj, const char* type, sidl_BaseInterface* _ex)

String cast method for interface and class type conversions

10.3.33

SIDL_C_INLINE_DECL void
bHYPRE_BiCGSTAB__exec (bHYPRE_BiCGSTAB self, const char* methodName, sidl_rmi_Call inArgs, sidl_rmi_Return outArgs, sidl_BaseInterface* _ex)

Select and execute a method by name

10.3.34

SIDL_C_INLINE_DECL char*
bHYPRE_BiCGSTAB__getURL (bHYPRE_BiCGSTAB self, sidl_BaseInterface* _ex)

Get the URL of the Implementation of this object (for RMI)

10.3.35

SIDL_C_INLINE_DECL void
bHYPRE_BiCGSTAB__raddRef (bHYPRE_BiCGSTAB self, sidl_BaseInterface* _ex)

On a remote object, addrefs the remote instance

10.3.36

SIDL_C_INLINE_DECL sidl_bool
bHYPRE_BiCGSTAB__isRemote (bHYPRE_BiCGSTAB self, sidl_BaseInterface* _ex)

TRUE if this object is remote, false if local

10.3.37

sidl_bool
bHYPRE_BiCGSTAB__isLocal (bHYPRE_BiCGSTAB self,
sidl_BaseInterface* _ex)

TRUE if this object is remote, false if local

10.3.38

struct bHYPRE_BiCGSTAB__object*
bHYPRE_BiCGSTAB__rmicast (void* obj, struct sidl_BaseInterface__object**
_ex)

Cast method for interface and class type conversions

10.3.39

struct bHYPRE_BiCGSTAB__object*
bHYPRE_BiCGSTAB__connectI (const char* url, sidl_bool ar, struct
sidl_BaseInterface__object** _ex)

RMI connector function for the class. (no addref)

10.4

CGNR Preconditioned Solver

Names

10.4.1

struct **bHYPRE_CGNR__object**

Symbol "bHYPRECGNR" (version 100)

CGNR solver. This calls Babel-interface matrix and vector functions, so it will work with any consistent matrix, vector, and preconditioner classes.

10.4.2

struct bHYPRE_CGNR__object*
bHYPRE_CGNR__create (sidl_BaseInterface* _ex)

Constructor function for the class

10.4.3

bHYPRE_CGNR
bHYPRE_CGNR__createRemote (const char* url, sidl_BaseInterface* _ex)

RMI constructor function for the class

10.4.4

bHYPRE_CGNR
bHYPRE_CGNR__wrapObj (void* data, sidl_BaseInterface* _ex)

Wraps up the private data struct pointer (struct bHYPRE_CGNR__data) passed in rather than running the constructor

10.4.5

bHYPRE_CGNR
bHYPRE_CGNR__connect (const char* , sidl_BaseInterface* _ex)

RMI connector function for the class(addrefs)

10.4.6

bHYPRE_CGNR
bHYPRE_CGNR_Create (bHYPRE_MPICommunicator mpi_comm,
bHYPRE_Operator A, sidl_BaseInterface* _ex)

This function is the preferred way to create a CGNR solver.

10.4.7

SIDL_C_INLINE_DECL int32_t
bHYPRE_CGNR_SetPreconditioner (bHYPRE_CGNR self, bHYPRE_Solver s, sidl_BaseInterface* _ex)

Set the preconditioner

10.4.8

SIDL_C_INLINE_DECL int32_t
bHYPRE_CGNR_GetPreconditioner (bHYPRE_CGNR self, bHYPRE_Solver* s, sidl_BaseInterface* _ex)

Method: GetPreconditioner[]

10.4.9

SIDL_C_INLINE_DECL int32_t
bHYPRE_CGNR_Clone (bHYPRE_CGNR self, bHYPRE_PreconditionedSolver* x, sidl_BaseInterface* _ex)

Method: Clone[]

10.4.10

SIDL_C_INLINE_DECL int32_t
bHYPRE_CGNR_SetOperator (bHYPRE_CGNR self, bHYPRE_Operator A, sidl_BaseInterface* _ex)

Set the operator for the linear system being solved. DEPRECATED. use Create

10.4.11

SIDL_C_INLINE_DECL int32_t
bHYPRE_CGNR_SetTolerance (bHYPRE_CGNR self, double tolerance, sidl_BaseInterface* _ex)

(Optional) Set the convergence tolerance. DEPRECATED. use SetDoubleParameter

10.4.12

SIDL_C_INLINE_DECL int32_t
bHYPRE_CGNR_SetMaxIterations (bHYPRE_CGNR self, int32_t max_iterations, sidl_BaseInterface* _ex)

(Optional) Set maximum number of iterations. DEPRECATED use SetIntParameter

10.4.13

SIDL_C_INLINE_DECL int32_t
bHYPRE_CGNR_SetLogging (bHYPRE_CGNR self, int32_t level, sidl_BaseInterface* _ex)

(Optional) Set the *logging level*, specifying the degree of additional informational data to be accumulated. Does nothing by default (level = 0). Other levels (if any) are implementation-specific. Must be called before `Setup` and `Apply`. DEPRECATED use SetIntParameter

10.4.14

SIDL_C_INLINE_DECL int32_t
bHYPRE_CGNR_SetPrintLevel (bHYPRE_CGNR self, int32_t level, sidl_BaseInterface* _ex)

(Optional) Set the *print level*, specifying the degree of informational data to be printed either to the screen or to a file. Does nothing by default (level=0). Other levels (if any) are implementation-specific. Must be called before `Setup` and `Apply`. DEPRECATED use SetIntParameter

10.4.15

SIDL_C_INLINE_DECL int32_t
bHYPRE_CGNR_GetNumIterations (bHYPRE_CGNR self, int32_t* num_iterations, sidl_BaseInterface* _ex)

(Optional) Return the number of iterations taken

10.4.16

SIDL_C_INLINE_DECL int32_t
bHYPRE_CGNR_GetRelResidualNorm (bHYPRE_CGNR self, double* norm, sidl_BaseInterface* _ex)

(Optional) Return the norm of the relative residual

10.4.17

SIDL_C_INLINE_DECL int32_t
bHYPRE_CGNR_SetCommunicator (bHYPRE_CGNR self, bHYPRE_MPICommunicator mpi_comm, sidl_BaseInterface* _ex)

Set the MPI Communicator. DEPRECATED, use Create:

10.4.18

SIDL_C_INLINE_DECL void
bHYPRE_CGNR_Destroy (bHYPRE_CGNR self, sidl_BaseInterface* _ex)

The Destroy function doesn't necessarily destroy anything. It is just another name for deleteRef. Thus it decrements the object's reference count. The Babel memory management system will destroy the object if the reference count goes to zero.

10.4.19

SIDL_C_INLINE_DECL int32_t
bHYPRE_CGNR_SetIntParameter (bHYPRE_CGNR self, const char* name, int32_t value, sidl_BaseInterface* _ex)

Set the int parameter associated with `name`

10.4.20

SIDL_C_INLINE_DECL int32_t
bHYPRE_CGNR_SetDoubleParameter (bHYPRE_CGNR self, const char* name, double value, sidl_BaseInterface* _ex)

Set the double parameter associated with `name`

10.4.21

SIDL_C_INLINE_DECL int32_t
bHYPRE_CGNR_SetStringParameter (bHYPRE_CGNR self, const char* name, const char* value, sidl_BaseInterface* eex)

Set the string parameter associated with `name`

10.4.22

SIDL_C_INLINE_DECL int32_t
bHYPRE_CGNR_SetIntArray1Parameter (bHYPRE_CGNR self, const char* name, int32_t* value, int32_t nvalues, sidl_BaseInterface* _ex)

Set the int 1-D array parameter associated with `name`

10.4.23

SIDL_C_INLINE_DECL int32_t
bHYPRE_CGNR_SetIntArray2Parameter (bHYPRE_CGNR self, const char* name, struct sidl_int__array* value, sidl_BaseInterface* _ex)

Set the int 2-D array parameter associated with `name`

10.4.24

SIDL_C_INLINE_DECL int32_t
bHYPRE_CGNR_SetDoubleArray1Parameter (bHYPRE_CGNR self, const char* name, double* value, int32_t nvalues, sidl_BaseInterface* _ex)

Set the double 1-D array parameter associated with `name`

10.4.25

SIDL_C_INLINE_DECL int32_t
bHYPRE_CGNR_SetDoubleArray2Parameter (bHYPRE_CGNR self, const char* name, struct sidl_double__array* value, sidl_BaseInterface* _ex)

Set the double 2-D array parameter associated with `name`

10.4.26

SIDL_C_INLINE_DECL int32_t
bHYPRE_CGNR_GetIntValue (bHYPRE_CGNR self, const char* name, int32_t* value, sidl_BaseInterface* _ex)

Set the int parameter associated with `name`

10.4.27

SIDL_C_INLINE_DECL int32_t
bHYPRE_CGNR_GetDoubleValue (bHYPRE_CGNR self, const char* name, double* value, sidl_BaseInterface* _ex)

Get the double parameter associated with `name`

10.4.28

SIDL_C_INLINE_DECL int32_t
bHYPRE_CGNR_Setup (bHYPRE_CGNR self, bHYPRE_Vector b, bHYPRE_Vector x, sidl_BaseInterface* _ex)

(Optional) Do any preprocessing that may be necessary in order to execute `Apply`

10.4.29

SIDL_C_INLINE_DECL int32_t
bHYPRE_CGNR_Apply (bHYPRE_CGNR self, bHYPRE_Vector b, bHYPRE_Vector* x, sidl_BaseInterface* _ex)

Apply the operator to `b`, returning `x`

10.4.30

SIDL_C_INLINE_DECL int32_t
bHYPRE_CGNR_ApplyAdjoint (bHYPRE_CGNR self, bHYPRE_Vector b, bHYPRE_Vector* x, sidl_BaseInterface* _ex)

Apply the adjoint of the operator to `b`, returning `x`

10.4.31

struct bHYPRE_CGNR__object*
bHYPRE_CGNR__cast (void* obj, sidl_BaseInterface* _ex)

Cast method for interface and class type conversions

10.4.32

void*
bHYPRE_CGNR__cast2 (void* obj, const char* type, sidl_BaseInterface* _ex)

String cast method for interface and class type conversions

10.4.33

SIDL_C_INLINE_DECL void
bHYPRE_CGNR__exec (bHYPRE_CGNR self, const char* methodName,
sidl_rmi_Call inArgs, sidl_rmi_Return outArgs, sidl_BaseInterface* _ex)

Select and execute a method by name

10.4.34

SIDL_C_INLINE_DECL char*
bHYPRE_CGNR__getURL (bHYPRE_CGNR self, sidl_BaseInterface* _ex)

Get the URL of the Implementation of this object (for RMI)

10.4.35

SIDL_C_INLINE_DECL void
bHYPRE_CGNR__raddRef (bHYPRE_CGNR self, sidl_BaseInterface* _ex)

On a remote object, addrefs the remote instance

10.4.36

SIDL_C_INLINE_DECL sidl_bool
bHYPRE_CGNR__isRemote (bHYPRE_CGNR self, sidl_BaseInterface* _ex)

TRUE if this object is remote, false if local

10.4.37

sidl_bool
bHYPRE_CGNR__isLocal (bHYPRE_CGNR self, sidl_BaseInterface* _ex)

TRUE if this object is remote, false if local

10.4.38

struct bHYPRE_CGNR__object*
bHYPRE_CGNR__rmicast (void* obj, struct sidl_BaseInterface__object** _ex)

Cast method for interface and class type conversions

10.4.39

struct bHYPRE_CGNR__object*
bHYPRE_CGNR__connectI (const char* url, sidl_bool ar, struct sidl_BaseInterface__object** _ex)

RMI connector function for the class. (no addref)

11

Other

Names

11.1

MPI Communicator

Names

11.1.1

struct **bHYPRE_MPICommunicator__object**

Symbol "bHYPREMPICommunicator" (version 100)

MPICommunicator class - two general Create functions: use CreateC if called from C code, CreateF if called from Fortran code. - Create_MPICommWorld will create a MPICommunicator to represent MPI_Comm_World, and can be called from any language.

11.1.2

struct bHYPRE_MPICommunicator__object*
bHYPRE_MPICommunicator__create (sidl_BaseInterface* _ex)

Constructor function for the class

11.1.3

bHYPRE_MPICommunicator
bHYPRE_MPICommunicator__createRemote (const char* url, sidl_BaseInterface* _ex)

RMI constructor function for the class

11.1.4

bHYPRE_MPICommunicator
bHYPRE_MPICommunicator__wrapObj (void* data, sidl_BaseInterface* _ex)

Wraps up the private data struct pointer (struct bHYPRE_MPICommunicator__data) passed in rather than running the constructor

11.1.5

bHYPRE_MPICommunicator
bHYPRE_MPICommunicator__connect (const char* , sidl_BaseInterface* _ex)

RMI connector function for the class(addrefs)

11.1.6

bHYPRE_MPICommunicator
bHYPRE_MPICommunicator_CreateC (void* mpi_comm,
sidl_BaseInterface* _ex)

Create an MPICommunicator object from C code.

11.1.7

bHYPRE_MPICommunicator
bHYPRE_MPICommunicator_CreateF (void* mpi_comm,
sidl_BaseInterface* _ex)

Create an MPICommunicator object from Fortran code.

11.1.8

bHYPRE_MPICommunicator
bHYPRE_MPICommunicator_Create_MPICommWorld (
sidl_BaseInterface* _ex)

Create an MPICommunicator object which represents MPI_Comm_World.

11.1.9

SIDL_C_INLINE_DECL void
bHYPRE_MPICommunicator_Destroy (bHYPRE_MPICommunicator self,
sidl_BaseInterface* _ex)

The Destroy function doesn't necessarily destroy anything. It is just another name for deleteRef. Thus it decrements the object's reference count. The Babel memory management system will destroy the object if the reference count goes to zero.

11.1.10

void **bHYPRE_MPICommunicator_Init** (sidl_BaseInterface* _ex)

Init and Finalize are to help debug MPI interfaces; you should normally use the MPI library more directly:

11.1.11

void **bHYPRE_MPICommunicator_Finalize** (sidl_BaseInterface* _ex)

Method: Finalize[]

11.1.12

struct bHYPRE_MPICommunicator__object*
bHYPRE_MPICommunicator__cast (void* obj, sidl_BaseInterface* _ex)

Cast method for interface and class type conversions

11.1.13

void*
bHYPRE_MPICommunicator__cast2 (void* obj, const char* type,
sidl_BaseInterface* _ex)

String cast method for interface and class type conversions

11.1.14

SIDL_C_INLINE_DECL void
bHYPRE_MPICommunicator__exec (bHYPRE_MPICommunicator self,
const char* methodName, sidl_rmi_Call inArgs, sidl_rmi_Return outArgs,
sidl_BaseInterface* _ex)

Select and execute a method by name

11.1.15

SIDL_C_INLINE_DECL char*
bHYPRE_MPICommunicator__getURL (bHYPRE_MPICommunicator self,
sidl_BaseInterface* _ex)

Get the URL of the Implementation of this object (for RMI)

11.1.16

SIDL_C_INLINE_DECL void
bHYPRE_MPICommunicator__raddRef (bHYPRE_MPICommunicator self, sidl_BaseInterface* _ex)

On a remote object, addrefs the remote instance

11.1.17

SIDL_C_INLINE_DECL sidl_bool
bHYPRE_MPICommunicator__isRemote (bHYPRE_MPICommunicator self, sidl_BaseInterface* _ex)

TRUE if this object is remote, false if local

11.1.18

sidl_bool
bHYPRE_MPICommunicator__isLocal (bHYPRE_MPICommunicator self, sidl_BaseInterface* _ex)

TRUE if this object is remote, false if local

11.1.19

struct bHYPRE_MPICommunicator__object*
bHYPRE_MPICommunicator__rmicast (void* obj, struct sidl_BaseInterface__object** _ex)

Cast method for interface and class type conversions

11.1.20

struct bHYPRE_MPICommunicator__object*
bHYPRE_MPICommunicator__connectI (const char* url, sidl_bool ar, struct sidl_BaseInterface__object** _ex)

RMI connector function for the class. (no addref)

12

Struct Grid, etc.

Names

12.1

Struct Grid

Names

12.1.1

struct **bHYPRE_StructGrid__object**

Symbol "bHYPREStructGrid" (version 100)

Define a structured grid class.

12.1.2

struct bHYPRE_StructGrid__object*
bHYPRE_StructGrid__create (sidl_BaseInterface* _ex)

Constructor function for the class

12.1.3

bHYPRE_StructGrid
bHYPRE_StructGrid__createRemote (const char* url, sidl_BaseInterface* _ex)

RMI constructor function for the class

12.1.4

bHYPRE_StructGrid
bHYPRE_StructGrid__wrapObj (void* data, sidl_BaseInterface* _ex)

Wraps up the private data struct pointer (struct bHYPRE_StructGrid__data) passed in rather than running the constructor

12.1.5

bHYPRE_StructGrid
bHYPRE_StructGrid__connect (const char* , sidl_BaseInterface* _ex)

RMI connector function for the class(addrefs)

12.1.6

bHYPRE_StructGrid
bHYPRE_StructGrid_Create (bHYPRE_MPICommunicator mpi_comm,
int32_t dim, sidl_BaseInterface* _ex)

This function is the preferred way to create a Struct Grid.

12.1.7

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructGrid_SetCommunicator (bHYPRE_StructGrid self,
bHYPRE_MPICommunicator mpi_comm, sidl_BaseInterface* _ex)

Set the MPI Communicator. DEPRECATED, use Create:

12.1.8

SIDL_C_INLINE_DECL void
bHYPRE_StructGrid_Destroy (bHYPRE_StructGrid self, sidl_BaseInterface* _ex)

The Destroy function doesn't necessarily destroy anything. It is just another name for deleteRef. Thus it decrements the object's reference count. The Babel memory management system will destroy the object if the reference count goes to zero.

12.1.9

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructGrid_SetDimension (bHYPRE_StructGrid self, int32_t dim, sidl_BaseInterface* _ex)

Method: SetDimension[]

12.1.10

int32_t
bHYPRE_StructGrid_SetExtents (bHYPRE_StructGrid self, int32_t* ilower, int32_t* iupper, int32_t dim, sidl_BaseInterface* _ex)

Define the lower and upper corners of a box of the grid. "ilower" and "iupper" are arrays of size "dim", the number of spatial dimensions.

12.1.11

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructGrid_SetPeriodic (bHYPRE_StructGrid self, int32_t* periodic, int32_t dim, sidl_BaseInterface* _ex)

Set the periodicity for the grid. Default is no periodicity.

The argument `periodic` is an `dim`-dimensional integer array that contains the periodicity for each dimension. A zero value for a dimension means non-periodic, while a nonzero value means periodic and contains the

actual period. For example, periodicity in the first and third dimensions for a 10x11x12 grid is indicated by the array [10,0,12].

NOTE: Some of the solvers in hypre have power-of-two restrictions on the size of the periodic dimensions.

12.1.12

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructGrid_SetNumGhost (bHYPRE_StructGrid self, int32_t* num_ghost, int32_t dim2, sidl_BaseInterface* _ex)

Set the number of ghost zones, separately on the lower and upper sides for each dimension. "num_ghost" is an array of size "dim2", twice the number of dimensions.

12.1.13

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructGrid_Assemble (bHYPRE_StructGrid self, sidl_BaseInterface* _ex)

final construction of the object before its use

12.1.14

struct bHYPRE_StructGrid__object*
bHYPRE_StructGrid__cast (void* obj, sidl_BaseInterface* _ex)

Cast method for interface and class type conversions

12.1.15

void*
bHYPRE_StructGrid__cast2 (void* obj, const char* type, sidl_BaseInterface* _ex)

String cast method for interface and class type conversions

12.1.16

SIDL_C_INLINE_DECL void
bHYPRE_StructGrid__exec (bHYPRE_StructGrid self, const char* methodName, sidl_rmi_Call inArgs, sidl_rmi_Return outArgs, sidl_BaseInterface* _ex)

Select and execute a method by name

12.1.17

SIDL_C_INLINE_DECL char*
bHYPRE_StructGrid__getURL (bHYPRE_StructGrid self, sidl_BaseInterface* _ex)

Get the URL of the Implementation of this object (for RMI)

12.1.18

SIDL_C_INLINE_DECL void
bHYPRE_StructGrid__raddRef (bHYPRE_StructGrid self, sidl_BaseInterface* _ex)

On a remote object, addrefs the remote instance

12.1.19

SIDL_C_INLINE_DECL sidl_bool
bHYPRE_StructGrid__isRemote (bHYPRE_StructGrid self,
sidl_BaseInterface* _ex)

TRUE if this object is remote, false if local

12.1.20

sidl_bool
bHYPRE_StructGrid__isLocal (bHYPRE_StructGrid self, sidl_BaseInterface*
_ex)

TRUE if this object is remote, false if local

12.1.21

struct bHYPRE_StructGrid__object*
bHYPRE_StructGrid__rmicast (void* obj, struct sidl_BaseInterface__object**
_ex)

Cast method for interface and class type conversions

12.1.22

struct bHYPRE_StructGrid__object*
bHYPRE_StructGrid__connectI (const char* url, sidl_bool ar, struct
sidl_BaseInterface__object** _ex)

RMI connector function for the class. (no addref)

12.2

Struct Stencil

Names

12.2.1

struct **bHYPRE_StructStencil__object**

Symbol "bHYPREStructStencil" (version 100)

Define a structured stencil for a structured problem description. More than one implementation is not envisioned, thus the decision has been made to make this a class rather than an interface.

12.2.2

struct bHYPRE_StructStencil__object*
bHYPRE_StructStencil__create (sidl_BaseInterface* _ex)

Constructor function for the class

12.2.3

bHYPRE_StructStencil
bHYPRE_StructStencil__createRemote (const char* url, sidl_BaseInterface* _ex)

RMI constructor function for the class

12.2.4

bHYPRE_StructStencil
bHYPRE_StructStencil__wrapObj (void* data, sidl_BaseInterface* _ex)

Wraps up the private data struct pointer (struct bHYPRE_StructStencil__data) passed in rather than running the constructor

12.2.5

bHYPRE_StructStencil
bHYPRE_StructStencil__connect (const char* , sidl_BaseInterface* _ex)

RMI connector function for the class(addrefs)

12.2.6

bHYPRE_StructStencil
bHYPRE_StructStencil_Create (int32_t ndim, int32_t size,
sidl_BaseInterface* _ex)

This function is the preferred way to create a Struct Stencil. You provide the number of spatial dimensions and the number of stencil entries.

12.2.7

SIDL_C_INLINE_DECL void
bHYPRE_StructStencil_Destroy (bHYPRE_StructStencil self,
sidl_BaseInterface* _ex)

The Destroy function doesn't necessarily destroy anything. It is just another name for deleteRef. Thus it decrements the object's reference count. The Babel memory management system will destroy the object if the reference count goes to zero.

12.2.8

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructStencil_SetDimension (bHYPRE_StructStencil self, int32_t dim, sidl_BaseInterface* _ex)

Set the number of dimensions. DEPRECATED, use StructStencilCreate

12.2.9

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructStencil_SetSize (bHYPRE_StructStencil self, int32_t size,
sidl_BaseInterface* _ex)

Set the number of stencil entries. DEPRECATED, use StructStencilCreate

12.2.10

SIDL_C_INLINE_DECL int32_t
bHYPRE_StructStencil_SetElement (bHYPRE_StructStencil self, int32_t index, int32_t* offset, int32_t dim, sidl_BaseInterface* _ex)

Set a stencil element. Specify the stencil index, and an array of offsets. "offset" is an array of length "dim", the number of spatial dimensions.

12.2.11

struct bHYPRE_StructStencil__object*
bHYPRE_StructStencil__cast (void* obj, sidl_BaseInterface* _ex)

Cast method for interface and class type conversions

12.2.12

void*
bHYPRE_StructStencil__cast2 (void* obj, const char* type, sidl_BaseInterface* _ex)

String cast method for interface and class type conversions

12.2.13

SIDL_C_INLINE_DECL void
bHYPRE_StructStencil__exec (bHYPRE_StructStencil self, const char* methodName, sidl_rmi_Call inArgs, sidl_rmi_Return outArgs, sidl_BaseInterface* _ex)

Select and execute a method by name

12.2.14

SIDL_C_INLINE_DECL char*
bHYPRE_StructStencil__getURL (bHYPRE_StructStencil self,
sidl_BaseInterface* _ex)

Get the URL of the Implementation of this object (for RMI)

12.2.15

SIDL_C_INLINE_DECL void
bHYPRE_StructStencil__raddRef (bHYPRE_StructStencil self,
sidl_BaseInterface* _ex)

On a remote object, addrefs the remote instance

12.2.16

SIDL_C_INLINE_DECL sidl_bool
bHYPRE_StructStencil__isRemote (bHYPRE_StructStencil self,
sidl_BaseInterface* _ex)

TRUE if this object is remote, false if local

12.2.17

sidl_bool
bHYPRE_StructStencil__isLocal (bHYPRE_StructStencil self,
sidl_BaseInterface* _ex)

TRUE if this object is remote, false if local

12.2.18

struct bHYPRE_StructStencil__object*
bHYPRE_StructStencil__rmicast (void* obj, struct
sidl_BaseInterface__object** _ex)

Cast method for interface and class type conversions

12.2.19

struct bHYPRE_StructStencil__object*
bHYPRE_StructStencil__connectI (const char* url, sidl_bool ar, struct
sidl_BaseInterface__object** _ex)

RMI connector function for the class. (no addref)

13

Semi-Structured Grid, etc.

Names

13.1

Semi-Structured Graph

Names

13.1.1

struct **bHYPRE_SStructGraph__object**

Symbol "bHYPRESStructGraph" (version 100)

The semi-structured grid graph class.

13.1.2

struct bHYPRE_SStructGraph__object*
bHYPRE_SStructGraph__create (sidl_BaseInterface* _ex)

Constructor function for the class

13.1.3

bHYPRE_SStructGraph
bHYPRE_SStructGraph__createRemote (const char* url, sidl_BaseInterface* _ex)

RMI constructor function for the class

13.1.4

bHYPRE_SStructGraph
bHYPRE_SStructGraph__wrapObj (void* data, sidl_BaseInterface* _ex)

Wraps up the private data struct pointer (struct bHYPRE_SStructGraph__data) passed in rather than running the constructor

13.1.5

bHYPRE_SStructGraph
bHYPRE_SStructGraph__connect (const char* , sidl_BaseInterface* _ex)

RMI connector function for the class(addrefs)

13.1.6

bHYPRE_SStructGraph
bHYPRE_SStructGraph_Create (bHYPRE_MPICommunicator mpi_comm, bHYPRE_SStructGrid grid, sidl_BaseInterface* _ex)

This function is the preferred way to create a SStruct Graph.

13.1.7

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructGraph_SetCommGrid (bHYPRE_SStructGraph self, bHYPRE_MPICommunicator mpi_comm, bHYPRE_SStructGrid grid, sidl_BaseInterface* _ex)

Set the grid and communicator. DEPRECATED, use Create:

13.1.8

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructGraph_SetStencil (bHYPRE_SStructGraph self, int32_t part, int32_t var, bHYPRE_SStructStencil stencil, sidl_BaseInterface* _ex)

Set the stencil for a variable on a structured part of the grid

13.1.9

int32_t
bHYPRE_SStructGraph_AddEntries (bHYPRE_SStructGraph self, int32_t part, int32_t* index, int32_t dim, int32_t var, int32_t to_part, int32_t* to_index, int32_t to_var, sidl_BaseInterface* _ex)

Add a non-stencil graph entry at a particular index. This graph entry is appended to the existing graph entries, and is referenced as such.

NOTE: Users are required to set graph entries on all processes that own the associated variables. This means that some data will be multiply defined.

13.1.10

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructGraph_SetObjectType (bHYPRE_SStructGraph self, int32_t type, sidl_BaseInterface* _ex)

Method: SetObjectType[]

13.1.11

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructGraph_SetCommunicator (bHYPRE_SStructGraph self, bHYPRE_MPICommunicator mpi_comm, sidl_BaseInterface* _ex)

Set the MPI Communicator. DEPRECATED, Use Create()

13.1.12

SIDL_C_INLINE_DECL void
bHYPRE_SStructGraph_Destroy (bHYPRE_SStructGraph self, sidl_BaseInterface* _ex)

The Destroy function doesn't necessarily destroy anything. It is just another name for deleteRef. Thus it decrements the object's reference count. The Babel memory management system will destroy the object if the reference count goes to zero.

13.1.13

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructGraph_Initialize (bHYPRE_SStructGraph self,
sidl_BaseInterface* _ex)

Prepare an object for setting coefficient values, whether for the first time or subsequently

13.1.14

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructGraph_Assemble (bHYPRE_SStructGraph self,
sidl_BaseInterface* _ex)

Finalize the construction of an object before using, either for the first time or on subsequent uses. `Initialize` and `Assemble` always appear in a matched set, with Initialize preceding Assemble. Values can only be set in between a call to Initialize and Assemble.

13.1.15

struct bHYPRE_SStructGraph__object*
bHYPRE_SStructGraph__cast (void* obj, sidl_BaseInterface* _ex)

Cast method for interface and class type conversions

13.1.16

void*
bHYPRE_SStructGraph__cast2 (void* obj, const char* type,
sidl_BaseInterface* _ex)

String cast method for interface and class type conversions

13.1.17

SIDL_C_INLINE_DECL void
bHYPRE_SStructGraph__exec (bHYPRE_SStructGraph self, const char* methodName, sidl_rmi_Call inArgs, sidl_rmi_Return outArgs, sidl_BaseInterface* _ex)

Select and execute a method by name

13.1.18

SIDL_C_INLINE_DECL char*
bHYPRE_SStructGraph__getURL (bHYPRE_SStructGraph self, sidl_BaseInterface* _ex)

Get the URL of the Implementation of this object (for RMI)

13.1.19

SIDL_C_INLINE_DECL void
bHYPRE_SStructGraph__raddRef (bHYPRE_SStructGraph self, sidl_BaseInterface* _ex)

On a remote object, addrefs the remote instance

13.1.20

SIDL_C_INLINE_DECL sidl_bool
bHYPRE_SStructGraph__isRemote (bHYPRE_SStructGraph self, sidl_BaseInterface* _ex)

TRUE if this object is remote, false if local

13.1.21

sidl_bool
bHYPRE_SStructGraph__isLocal (bHYPRE_SStructGraph self,
sidl_BaseInterface* _ex)

TRUE if this object is remote, false if local

13.1.22

struct bHYPRE_SStructGraph__object*
bHYPRE_SStructGraph__rmicast (void* obj, struct
sidl_BaseInterface__object** _ex)

Cast method for interface and class type conversions

13.1.23

struct bHYPRE_SStructGraph__object*
bHYPRE_SStructGraph__connectI (const char* url, sidl_bool ar, struct
sidl_BaseInterface__object** _ex)

RMI connector function for the class. (no addref)

13.2

Semi-Structured Grid

Names

13.2.1

struct **bHYPRE_SStructGrid__object**

Symbol "bHYPRESStructGrid" (version 100)

The semi-structured grid class.

13.2.2

struct bHYPRE_SStructGrid__object*
bHYPRE_SStructGrid__create (sidl_BaseInterface* _ex)

Constructor function for the class

13.2.3

bHYPRE_SStructGrid
bHYPRE_SStructGrid__createRemote (const char* url, sidl_BaseInterface* _ex)

RMI constructor function for the class

13.2.4

bHYPRE_SStructGrid
bHYPRE_SStructGrid__wrapObj (void* data, sidl_BaseInterface* _ex)

Wraps up the private data struct pointer (struct bHYPRE_SStructGrid__data) passed in rather than running the constructor

13.2.5

bHYPRE_SStructGrid
bHYPRE_SStructGrid__connect (const char* , sidl_BaseInterface* _ex)

RMI connector function for the class(addrefs)

13.2.6

bHYPRE_SStructGrid
bHYPRE_SStructGrid_Create (bHYPRE_MPICommunicator mpi_comm, int32_t ndim, int32_t nparts, sidl_BaseInterface* _ex)

This function is the preferred way to create a SStruct Grid.

13.2.7

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructGrid_SetNumDimParts (bHYPRE_SStructGrid self, int32_t ndim, int32_t nparts, sidl_BaseInterface* _ex)

Method: SetNumDimParts[]

13.2.8

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructGrid_SetCommunicator (bHYPRE_SStructGrid self, bHYPRE_MPICommunicator mpi_comm, sidl_BaseInterface* _ex)

Method: SetCommunicator[]

13.2.9

SIDL_C_INLINE_DECL void
bHYPRE_SStructGrid_Destroy (bHYPRE_SStructGrid self, sidl_BaseInterface* _ex)

The Destroy function doesn't necessarily destroy anything. It is just another name for deleteRef. Thus it decrements the object's reference count. The Babel memory management system will destroy the object if the reference count goes to zero.

13.2.10

int32_t
bHYPRE_SStructGrid_SetExtents (bHYPRE_SStructGrid self, int32_t part, int32_t* ilower, int32_t* iupper, int32_t dim, sidl_BaseInterface* _ex)

Set the extents for a box on a structured part of the grid

13.2.11

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructGrid_SetVariable (bHYPRE_SStructGrid self, int32_t part, int32_t var, int32_t nvars, enum bHYPRE_SStructVariable__enum vartype, sidl_BaseInterface* _ex)

Describe the variables that live on a structured part of the grid. Input: part number, variable number, total number of variables on that part (needed for memory allocation), variable type.

13.2.12

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructGrid_AddVariable (bHYPRE_SStructGrid self, int32_t part, int32_t* index, int32_t dim, int32_t var, enum bHYPRE_SStructVariable__enum vartype, sidl_BaseInterface* _ex)

Describe additional variables that live at a particular index. These variables are appended to the array of variables set in `SetVariables`, and are referenced as such.

13.2.13

int32_t
bHYPRE_SStructGrid_SetNeighborBox (bHYPRE_SStructGrid self, int32_t part, int32_t* ilower, int32_t* iupper, int32_t nbor_part, int32_t* nbor_ilower, int32_t* nbor_iupper, int32_t* index_map, int32_t dim, sidl_BaseInterface* _ex)

Describe how regions just outside of a part relate to other parts. This is done a box at a time.

The indexes `ilower` and `iupper` map directly to the indexes `nbor_ilower` and `nbor_iupper`. Although, it is required that indexes increase from `ilower` to `iupper`, indexes may increase and/or decrease from `nbor_ilower` to `nbor_iupper`.

The `index_map` describes the mapping of indexes 0, 1, and 2 on part `part` to the corresponding indexes on part `nbor_part`. For example, triple (1, 2, 0) means that indexes 0, 1, and 2 on part `part` map to indexes 1, 2, and 0 on part `nbor_part`, respectively.

NOTE: All parts related to each other via this routine must have an identical list of variables and variable types. For example, if part 0 has only two variables on it, a cell centered variable and a node centered variable, and we declare part 1 to be a neighbor of part 0, then part 1 must also have only two variables on it, and they must be of type cell and node.

13.2.14

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructGrid_AddUnstructuredPart (bHYPRE_SStructGrid self, int32_t ilower, int32_t iupper, sidl_BaseInterface* _ex)

Add an unstructured part to the grid. The variables in the unstructured part of the grid are referenced by a global rank between 0 and the total number of unstructured variables minus one. Each process owns some unique consecutive range of variables, defined by `ilower` and `iupper`.

NOTE: This is just a placeholder. This part of the interface is not finished.

13.2.15

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructGrid_SetPeriodic (bHYPRE_SStructGrid self, int32_t part, int32_t* periodic, int32_t dim, sidl_BaseInterface* _ex)

(Optional) Set periodic for a particular part

13.2.16

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructGrid_SetNumGhost (bHYPRE_SStructGrid self, int32_t* num_ghost, int32_t dim2, sidl_BaseInterface* _ex)

Setting ghost in the sgrids

13.2.17

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructGrid_Assemble (bHYPRE_SStructGrid self, sidl_BaseInterface* _ex)

final construction of the object before its use

13.2.18

struct bHYPRE_SStructGrid__object*
bHYPRE_SStructGrid__cast (void* obj, sidl_BaseInterface* _ex)

Cast method for interface and class type conversions

13.2.19

void*
bHYPRE_SStructGrid__cast2 (void* obj, const char* type, sidl_BaseInterface* _ex)

String cast method for interface and class type conversions

13.2.20

SIDL_C_INLINE_DECL void
bHYPRE_SStructGrid__exec (bHYPRE_SStructGrid self, const char* methodName, sidl_rmi_Call inArgs, sidl_rmi_Return outArgs, sidl_BaseInterface* _ex)

Select and execute a method by name

13.2.21

SIDL_C_INLINE_DECL char*
bHYPRE_SStructGrid__getURL (bHYPRE_SStructGrid self, sidl_BaseInterface* _ex)

Get the URL of the Implementation of this object (for RMI)

13.2.22

SIDL_C_INLINE_DECL void
bHYPRE_SStructGrid__raddRef (bHYPRE_SStructGrid self,
sidl_BaseInterface* _ex)

On a remote object, addrefs the remote instance

13.2.23

SIDL_C_INLINE_DECL sidl_bool
bHYPRE_SStructGrid__isRemote (bHYPRE_SStructGrid self,
sidl_BaseInterface* _ex)

TRUE if this object is remote, false if local

13.2.24

sidl_bool
bHYPRE_SStructGrid__isLocal (bHYPRE_SStructGrid self,
sidl_BaseInterface* _ex)

TRUE if this object is remote, false if local

13.2.25

struct bHYPRE_SStructGrid__object*
bHYPRE_SStructGrid__rmicast (void* obj, struct sidl_BaseInterface__object**
_ex)

Cast method for interface and class type conversions

13.2.26

struct bHYPRE_SStructGrid__object*
bHYPRE_SStructGrid__connectI (const char* url, sidl_bool ar, struct sidl_BaseInterface__object** _ex)

RMI connector function for the class. (no addref)

13.3

Semi-Structured Stencil

Names

13.3.1

struct **bHYPRE_SStructStencil__object**

Symbol "bHYPRESStructStencil" (version 100)

The semi-structured grid stencil class.

13.3.2

struct bHYPRE_SStructStencil__object*
bHYPRE_SStructStencil__create (sidl_BaseInterface* _ex)

Constructor function for the class

13.3.3

bHYPRE_SStructStencil
bHYPRE_SStructStencil__createRemote (const char* url, sidl_BaseInterface* _ex)

RMI constructor function for the class

13.3.4

bHYPRE_SStructStencil
bHYPRE_SStructStencil__wrapObj (void* data, sidl_BaseInterface* _ex)

Wraps up the private data struct pointer (struct bHYPRE_SStructStencil__data) passed in rather than running the constructor

13.3.5

bHYPRE_SStructStencil
bHYPRE_SStructStencil__connect (const char* , sidl_BaseInterface* _ex)

RMI connector function for the class(addrefs)

13.3.6

bHYPRE_SStructStencil
bHYPRE_SStructStencil_Create (int32_t ndim, int32_t size,
sidl_BaseInterface* _ex)

This function is the preferred way to create a SStruct Stencil.

13.3.7

SIDL_C_INLINE_DECL void
bHYPRE_SStructStencil_Destroy (bHYPRE_SStructStencil self,
sidl_BaseInterface* _ex)

The Destroy function doesn't necessarily destroy anything. It is just another name for deleteRef. Thus it decrements the object's reference count. The Babel memory management system will destroy the object if the reference count goes to zero.

13.3.8

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructStencil_SetNumDimSize (bHYPRE_SStructStencil self,
int32_t ndim, int32_t size, sidl_BaseInterface* _ex)

Set the number of spatial dimensions and stencil entries. DEPRECATED, use Create:

13.3.9

SIDL_C_INLINE_DECL int32_t
bHYPRE_SStructStencil_SetEntry (bHYPRE_SStructStencil self, int32_t entry, int32_t* offset, int32_t dim, int32_t var, sidl_BaseInterface* _ex)

Set a stencil entry

13.3.10

struct bHYPRE_SStructStencil__object*
bHYPRE_SStructStencil__cast (void* obj, sidl_BaseInterface* _ex)

Cast method for interface and class type conversions

13.3.11

void*
bHYPRE_SStructStencil__cast2 (void* obj, const char* type, sidl_BaseInterface* _ex)

String cast method for interface and class type conversions

13.3.12

SIDL_C_INLINE_DECL void
bHYPRE_SStructStencil__exec (bHYPRE_SStructStencil self, const char* methodName, sidl_rmi_Call inArgs, sidl_rmi_Return outArgs, sidl_BaseInterface* _ex)

Select and execute a method by name

13.3.13

SIDL_C_INLINE_DECL char*
bHYPRE_SStructStencil__getURL (bHYPRE_SStructStencil self,
sidl_BaseInterface* _ex)

Get the URL of the Implementation of this object (for RMI)

13.3.14

SIDL_C_INLINE_DECL void
bHYPRE_SStructStencil__raddRef (bHYPRE_SStructStencil self,
sidl_BaseInterface* _ex)

On a remote object, addrefs the remote instance

13.3.15

SIDL_C_INLINE_DECL sidl_bool
bHYPRE_SStructStencil__isRemote (bHYPRE_SStructStencil self,
sidl_BaseInterface* _ex)

TRUE if this object is remote, false if local

13.3.16

sidl_bool
bHYPRE_SStructStencil__isLocal (bHYPRE_SStructStencil self,
sidl_BaseInterface* _ex)

TRUE if this object is remote, false if local

13.3.17

struct bHYPRE_SStructStencil__object*
bHYPRE_SStructStencil__rmicast (void* obj, struct sidl_BaseInterface__object** _ex)

Cast method for interface and class type conversions

13.3.18

struct bHYPRE_SStructStencil__object*
bHYPRE_SStructStencil__connectI (const char* url, sidl_bool ar, struct sidl_BaseInterface__object** _ex)

RMI connector function for the class. (no addref)

13.4

Semi-Structured Variable

Names

13.4.1

enum **bHYPRE_SStructVariable__enum**

Symbol "bHYPRESStructVariable" (version 100)

The SStructVariable enumerated type.

An enumerated type that supports cell centered, node centered, face centered, and edge centered variables. Face centered variables are split into x-face, y-face, and z-face variables, and edge centered variables are split

into x-edge, y-edge, and z-edge variables. The edge centered variable types are only used in 3D. In 2D, edge centered variables are handled by the face centered types.

Variables are referenced relative to an abstract (cell centered) index in the following way:

- cell centered variables are aligned with the index;
- node centered variables are aligned with the cell corner at relative index (1/2, 1/2, 1/2);
- x-face, y-face, and z-face centered variables are aligned with the faces at relative indexes (1/2, 0, 0), (0, 1/2, 0), and (0, 0, 1/2), respectively;
- x-edge, y-edge, and z-edge centered variables are aligned with the edges at relative indexes (0, 1/2, 1/2), (1/2, 0, 1/2), and (1/2, 1/2, 0), respectively.

The supported identifiers are:

- `HYPRE_SSTRUCT_VARIABLE_CELL`
- `HYPRE_SSTRUCT_VARIABLE_NODE`
- `HYPRE_SSTRUCT_VARIABLE_XFACE`
- `HYPRE_SSTRUCT_VARIABLE_YFACE`
- `HYPRE_SSTRUCT_VARIABLE_ZFACE`
- `HYPRE_SSTRUCT_VARIABLE_XEDGE`
- `HYPRE_SSTRUCT_VARIABLE_YEDGE`
- `HYPRE_SSTRUCT_VARIABLE_ZEDGE`

NOTE: Although variables are referenced relative to a unique abstract cell-centered index, some variables are associated with multiple grid cells. For example, node centered variables in 3D are associated with 8 cells (away from boundaries). Although grid cells are distributed uniquely to different processes, variables may be owned by multiple processes because they may be associated with multiple cells.