DyLP

1.9

Generated by Doxygen 1.7.1

Mon Nov 25 2013 17:37:32

Contents

1 Class Index

1.1 Class List

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

2 File Index

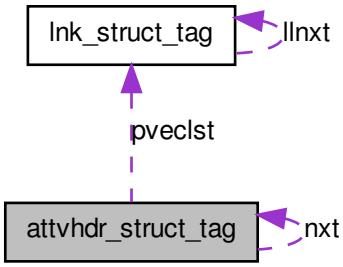
2.1 File List

Here is a list of all documented files with brief descriptions:

3 Class Documentation

3.1 attvhdr_struct_tag Struct Reference

Collaboration diagram for attvhdr_struct_tag:



3.1.1 Detailed Description

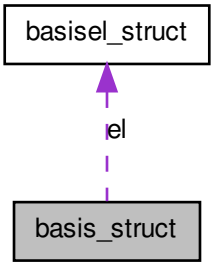
Definition at line 267 of file dy_consys.h.

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

- dy_consys.h

3.2 basis_struct Struct Reference

Collaboration diagram for basis_struct:



3.2.1 Detailed Description

Definition at line 453 of file dylp.h.

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

- dylp.h

3.3 basisel_struct Struct Reference

3.3.1 Detailed Description

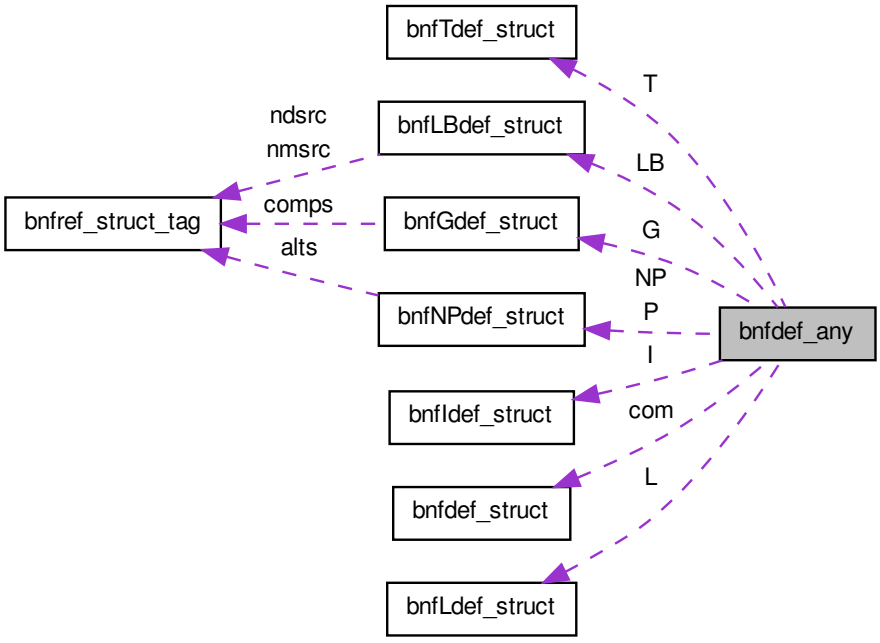
Definition at line 451 of file dylp.h.

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

- dylp.h

3.4 bnfdef_any Union Reference

Collaboration diagram for bnfdef_any:



3.4.1 Detailed Description

Definition at line 427 of file dylib_bnfrdr.h.

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

- dylib_bnfrdr.h

3.5 bnfdef_struct Struct Reference

3.5.1 Detailed Description

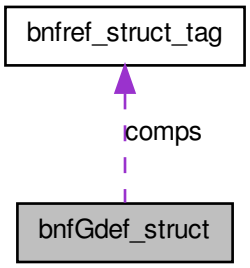
Definition at line 266 of file dylib_bnfrdr.h.

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

- dylib_bnfrdr.h

3.6 bnfGdef_struct Struct Reference

Collaboration diagram for bnfGdef_struct:



3.6.1 Detailed Description

Definition at line 285 of file dylib_bnfrdr.h.

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

- dylib_bnfrdr.h

3.7 bnfIdef_struct Struct Reference

3.7.1 Detailed Description

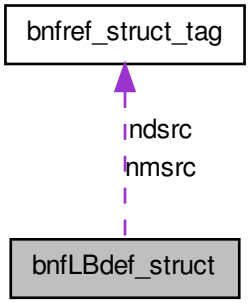
Definition at line 355 of file dylib_bnfrdr.h.

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

- dylib_bnfrdr.h

3.8 bnfLBdef_struct Struct Reference

Collaboration diagram for bnfLBdef_struct:



3.8.1 Detailed Description

Definition at line 406 of file dylib_bnfrdr.h.

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

- dylib_bnfrdr.h

3.9 bnfLdef_struct Struct Reference

3.9.1 Detailed Description

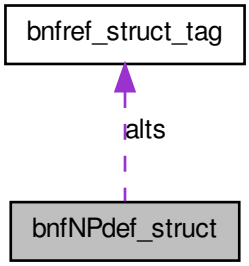
Definition at line 371 of file dylib_bnfrdr.h.

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

- dylib_bnfrdr.h

3.10 bnfNPdef_struct Struct Reference

Collaboration diagram for bnfNPdef_struct:



3.10.1 Detailed Description

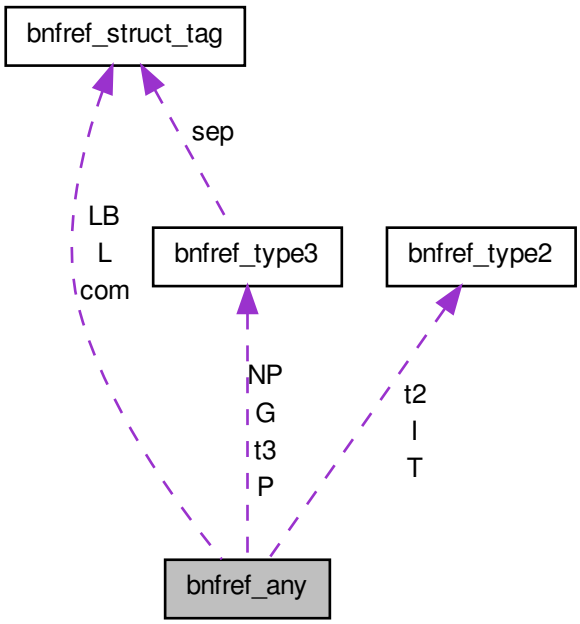
Definition at line 301 of file dylib_bnfrdr.h.

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

- dylib_bnfrdr.h

3.11 bnfref_any Union Reference

Collaboration diagram for bnfref_any:



3.11.1 Detailed Description

Definition at line 522 of file dylib_bnfrdr.h.

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

- dylib_bnfrdr.h

3.12 bnfref_struct_tag Struct Reference

3.12.1 Detailed Description

Definition at line 464 of file dylib_bnfrdr.h.

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

- dylib_bnfrdr.h

3.13 bnfref_type2 Struct Reference

3.13.1 Detailed Description

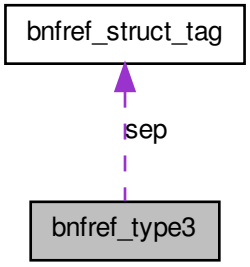
Definition at line 487 of file dylib_bnfrdr.h.

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

- dylib_bnfrdr.h

3.14 bnfref_type3 Struct Reference

Collaboration diagram for bnfref_type3:



3.14.1 Detailed Description

Definition at line 508 of file dylib_bnfrdr.h.

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

- dylib_bnfrdr.h

3.15 bnfTdef_struct Struct Reference

3.15.1 Detailed Description

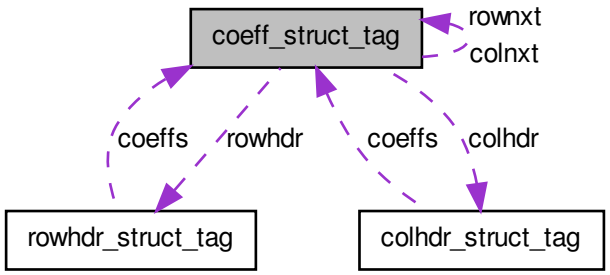
Definition at line 337 of file dylib_bnfrdr.h.

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

- dylib_bnfrdr.h

3.16 coeff_struct_tag Struct Reference

Collaboration diagram for coeff_struct_tag:



3.16.1 Detailed Description

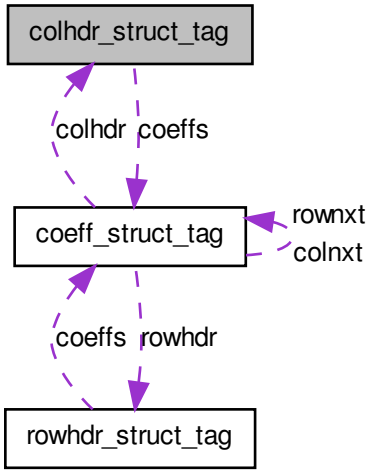
Definition at line 102 of file dy_consys.h.

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

- dy_consys.h

3.17 colhdr_struct_tag Struct Reference

Collaboration diagram for colhdr_struct_tag:



3.17.1 Detailed Description

Definition at line 120 of file dy_consys.h.

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

- dy_consys.h

3.18 conbnd_struct Struct Reference

3.18.1 Detailed Description

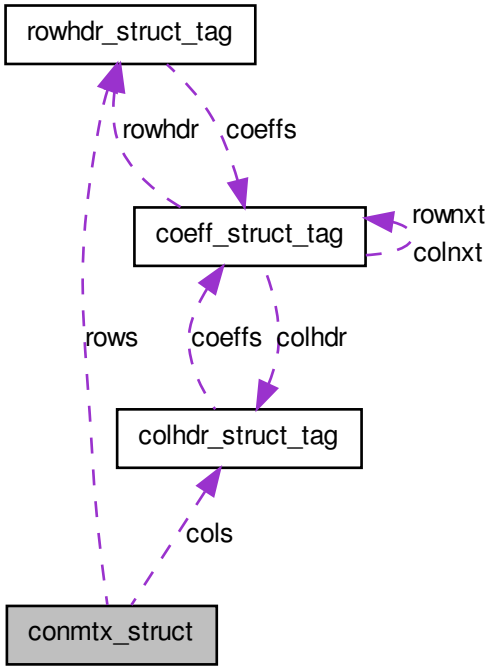
Definition at line 308 of file dy_consys.h.

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

- dy_consys.h

3.19 conmtx_struct Struct Reference

Collaboration diagram for conmtx_struct:



3.19.1 Detailed Description

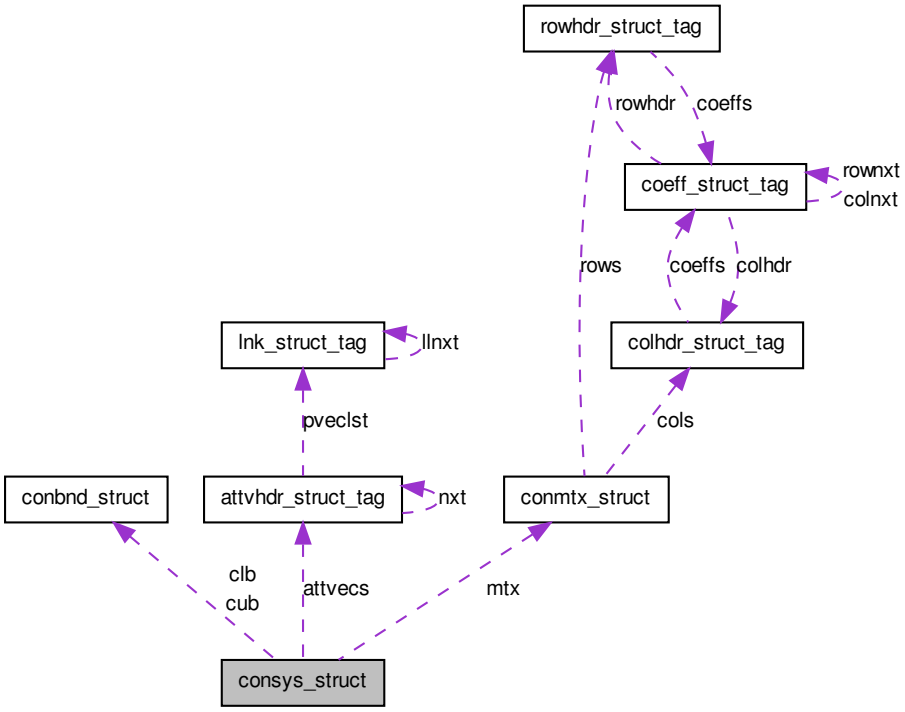
Definition at line 153 of file dy_consys.h.

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

- dy_consys.h

3.20 consys_struct Struct Reference

Collaboration diagram for consys_struct:



3.20.1 Detailed Description

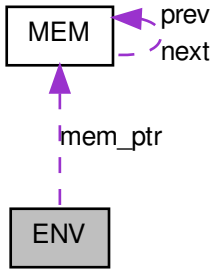
Definition at line 460 of file dy_consys.h.

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

- dy_consys.h

3.21 ENV Struct Reference

Collaboration diagram for ENV:



3.21.1 Detailed Description

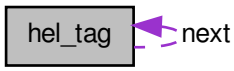
Definition at line 53 of file glplib.h.

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

- glplib.h

3.22 hel_tag Struct Reference

Collaboration diagram for hel_tag:



3.22.1 Detailed Description

Definition at line 37 of file dylib_hash.h.

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

- dylib_hash.h

3.23 INV Struct Reference

Collaboration diagram for INV:

3.23.1 Detailed Description

Definition at line 78 of file glpinv.h.

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

- glpinv.h

3.24 keytab_entry_internal Struct Reference

3.24.1 Detailed Description

Definition at line 33 of file dylib_keytab.h.

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

- dylib_keytab.h

3.25 lex_struct Struct Reference

3.25.1 Detailed Description

Definition at line 74 of file dylib_io.h.

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

- dylib_io.h

3.26 lnk_struct_tag Struct Reference

Collaboration diagram for lnk_struct_tag:

3.26.1 Detailed Description

Definition at line 115 of file dylib_std.h.

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

- dylib_std.h

3.27 lpopts_struct Struct Reference

3.27.1 Detailed Description

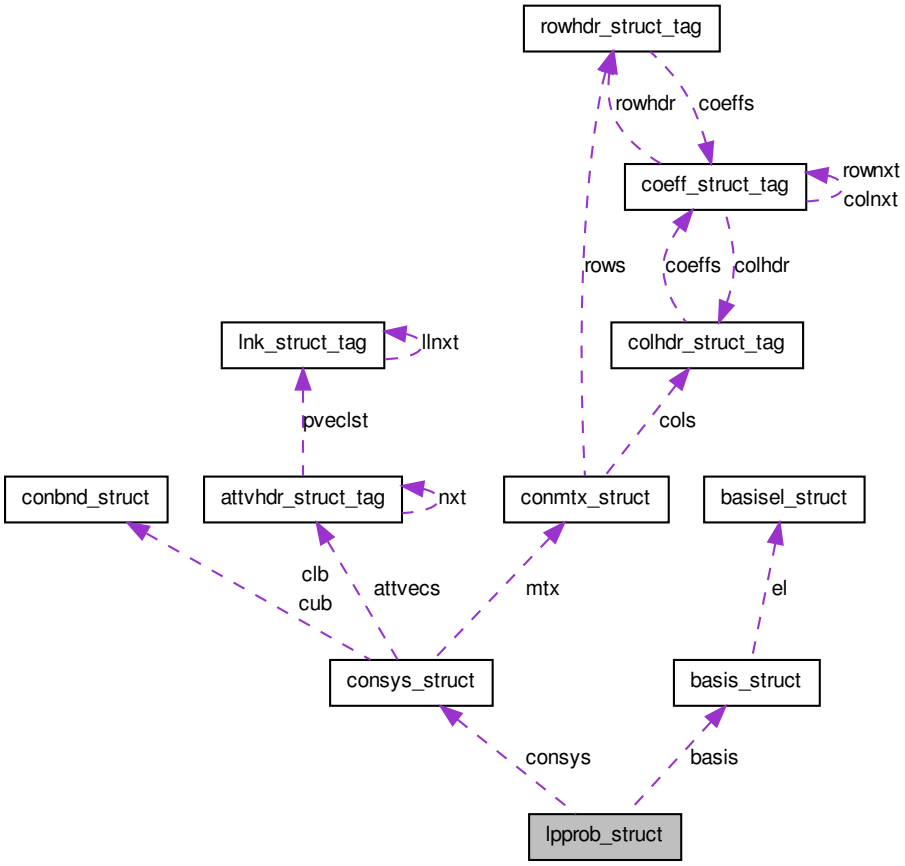
Definition at line 1114 of file dylp.h.

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

- dylp.h

3.28 lpprob_struct Struct Reference

Collaboration diagram for lpprob_struct:



3.28.1 Detailed Description

Definition at line 586 of file dylp.h.

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

- dylp.h

3.29 lpstats_struct Struct Reference

3.29.1 Detailed Description

Definition at line 1303 of file dylp.h.

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

- dylp.h

3.30 lptols_struct Struct Reference

3.30.1 Detailed Description

Definition at line 666 of file dylp.h.

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

- dylp.h

3.31 LUF Struct Reference

3.31.1 Detailed Description

Definition at line 83 of file glpluf.h.

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

- glpluf.h

3.32 LUF_WA Struct Reference

3.32.1 Detailed Description

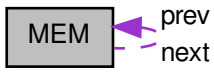
Definition at line 270 of file glpluf.h.

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

- glpluf.h

3.33 MEM Struct Reference

Collaboration diagram for MEM:



3.33.1 Detailed Description

Definition at line 105 of file glplib.h.

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

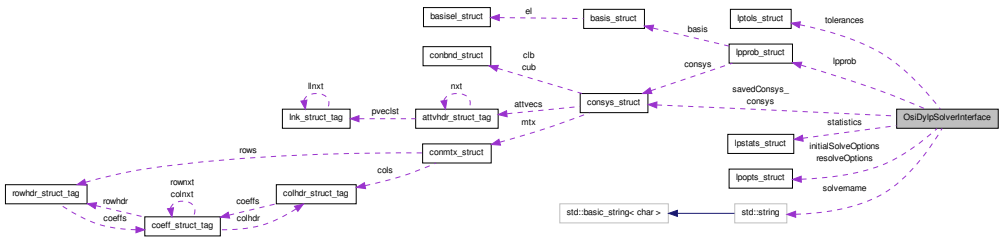
- glplib.h

3.34 OsiDylpSolverInterface Class Reference

COIN OSI API for dylp.

`#include <OsiDylpSolverInterface.hpp>`

Collaboration diagram for OsiDylpSolverInterface:



Public Member Functions

Constructors and Destructors

- OsiDylpSolverInterface ()
  Default constructor.
- OsiDylpSolverInterface (const OsiDylpSolverInterface &src)
  Copy constructor.
- OsiSolverInterface * clone (bool copyData=true) const
  Clone the solver object.
- OsiDylpSolverInterface & operator= (const OsiDylpSolverInterface &rhs)
  Assignment.
- ~OsiDylpSolverInterface ()
  Destructor.
- void reset ()
  Reset the solver object to the state produced by the default constructor.

Methods to load a problem

- int readMps (const char *filename, const char *extension="mps")
  Read a problem description in MPS format from a file.
- int readMps (const char *filename, const char *extension, int &numberSets, CoinSet **&sets)
  Read a problem description in MPS format from a file, including SOS information.
- void writeMps (const char *basename, const char *extension="mps", double objsense=0.0) const
  Write the problem into the specified file in MPS format.
- void loadProblem (const CoinPackedMatrix &matrix, const double *collb, const double *colub, const double *obj, const char *rowsen, const double *rowrhs, const double *rowrng)
  Load a problem description (OSI packed matrix, row sense, parameters unaffected).
- void loadProblem (const CoinPackedMatrix &matrix, const double *collb, const double *colub, const double *obj, const double *rowlb, const double *rowub)
  Load a problem description (OSI packed matrix, row bounds, parameters unaffected).
- void loadProblem (const int colcnt, const int rowcnt, const int *start, const int *index, const double *value, const double *collb, const double *colub, const double *obj, const char *sense, const double *rhsin, const double *range)

Load a problem description (standard column-major packed matrix, row sense, parameters unaffected).

- void loadProblem (const int colcnt, const int rowcnt, const int ∗start, const int ∗index, const double ∗value, const double ∗collb, const double ∗colub, const double ∗obj, const double ∗row_lower, const double ∗row_upper)

  Load a problem description (standard column-major packed matrix, row bounds, parameters unaffected).

- void assignProblem (CoinPackedMatrix ∗&matrix, double ∗&collb, double ∗&colub, double ∗&obj, char ∗&rowsen, double ∗&rowrhs, double ∗&rowrng)

  Load a problem description (OSI packed matrix, row sense, parameters destroyed).

- void assignProblem (CoinPackedMatrix ∗&matrix, double ∗&collb, double ∗&colub, double ∗&obj, double ∗&rowlb, double ∗&rowub)

  Load a problem description (OSI packed matrix, row bounds, parameters destroyed).

Methods to obtain problem information

- int getNumCols () const

  Get the number of columns (variables).

- int getNumRows () const

  Get the number of rows (constraints).

- int getNumElements () const

  Get the number of non-zero coefficients.

- int getNumIntegers () const

  Get the number of integer variables.

- const double ∗ getColLower () const

  Get the column (variable) lower bound vector.

- const double ∗ getColUpper () const

  Get the column (variable) upper bound vector.

- bool isContinuous (int colIndex) const

  Return true if the variable is continuous.

- bool isBinary (int colIndex) const

  Return true if the variable is binary.

- bool isIntegerNonBinary (int colIndex) const
  Return true if the variable is general integer.
- bool isInteger (int colIndex) const
  Return true if the variable is integer (general or binary).
- const char * getRowSense () const
  Get the row sense (constraint type) vector.
- const double * getRightHandSide () const
  Get the row (constraint) right-hand-side vector.
- const double * getRowRange () const
  Get the row (constraint) range vector.
- const double * getRowLower () const
  Get the row (constraint) lower bound vector.
- const double * getRowUpper () const
  Get the row (constraint) upper bound vector.
- const double * getObjCoefficients () const
  Get the objective function coefficient vector.
- double getObjSense () const
  Get the objective function sense (min/max).
- const CoinPackedMatrix * getMatrixByRow () const
  Get a pointer to a row-major copy of the constraint matrix.
- const CoinPackedMatrix * getMatrixByCol () const
  Get a pointer to a column-major copy of the constraint matrix.

Methods for row and column names.

Only the set methods need to be overridden to ensure consistent names between OsiDylp and the OSI base class.

- void setObjName (std::string name)
  Set the objective function name.
- void setRowName (int ndx, std::string name)
  Set a row name.
- void setColName (int ndx, std::string name)

Set a column name.

Methods to modify the problem

- void setContinuous (int index)
  Set a single variable to be continuous.
- void setInteger (int index)
  Set a single variable to be integer.
- void setColLower (int index, double value)
  Set the lower bound on a column (variable).
- void setColUpper (int index, double value)
  Set the upper bound on a column (variable).
- void setRowLower (int index, double value)
  Set the lower bound on a row (constraint).
- void setRowUpper (int index, double value)
  Set the upper bound on a row (constraint).
- void setRowType (int index, char rowsen, double rowrhs, double rowrng)
  Set the type of a row (constraint).
- void setObjCoeff (int index, double value)
  Set an objective function coefficient.
- void setObjective (const double *array)
  Set the objective coefficients for all columns.
- void setObjSense (double sense)
  Set the sense (min/max) of the objective.
- void setColSolution (const double *colsol)
  Set the value of the primal variables in the problem solution.
- void setRowPrice (const double *)
  Set the value of the dual variables in the problem solution.
- void addCol (const CoinPackedVectorBase &vec, const double collb, const double colub, const double obj)
  Add a column (variable) to the problem.
- void deleteCols (const int num, const int *colIndices)

Remove column(s) (variable(s)) from the problem.

- void addRow (const CoinPackedVectorBase &row, const double rowlb, const double rowub)

  Add a row (constraint) to the problem.

- void addRow (const CoinPackedVectorBase &row, const char rowsen, const double rowrhs, const double rowrng)

  Add a row (constraint) to the problem.

- void deleteRows (const int num, const int *rowIndices)

  Delete row(s) (constraint(s)) from the problem.

- void applyRowCut (const OsiRowCut &cut)

  Apply a row (constraint) cut (add one constraint).

- void applyColCut (const OsiColCut &cut)

  Apply a column (variable) cut (adjust one or more bounds).

Solve methods

- void initialSolve ()

  Solve an lp from scratch.

- CoinWarmStart * getEmptyWarmStart () const

  Get an empty OsiDylpWarmStartBasis object.

- CoinWarmStart * getWarmStart () const

  Build a warm start object for the current lp solution.

- bool setWarmStart (const CoinWarmStart *warmStart)

  Apply a warm start object.

- void resolve ()

  Call dylp to reoptimize (warm start).

- void markHotStart ()

  Create a hot start snapshot.

- void solveFromHotStart ()

  Call dylp to reoptimize (hot start).

- void unmarkHotStart ()

  Delete the hot start snapshot.

Methods returning solver termination status

- bool isAbandoned () const
  True if dylp abandoned the problem.
- bool isProvenOptimal () const
  True if dylp reported an optimal solution.
- bool isProvenPrimalInfeasible () const
  True if dylp reported the problem to be primal infeasible.
- bool isProvenDualInfeasible () const
  True if dylp reported the problem to be dual infeasible (primal unbounded).
- bool isIterationLimitReached () const
  True if dylp reached the iteration limit.
- int getIterationCount () const
  Get the number of iterations for the last lp.
- bool isPrimalObjectiveLimitReached () const
  Is the primal objective limit reached?
- bool isDualObjectiveLimitReached () const
  Is the dual objective limit reached?

Methods to set/get solver parameters

- double getInfinity () const
  Get dylp's value for infinity.
- bool setIntParam (OsiIntParam key, int value)
  Set an OSI integer parameter.
- bool setDblParam (OsiDblParam key, double value)
  Set an OSI double parameter.
- bool setStrParam (OsiStrParam key, const std::string &value)
  Set an OSI string parameter.
- bool setHintParam (OsiHintParam key, bool sense=true, OsiHintStrength strength=OsiHintTry, void *info=0)
  Set an OSI hint.
- bool getIntParam (OsiIntParam key, int &value) const

Get an OSI integer parameter.

- bool getDblParam (OsiDblParam key, double &value) const
  Get an OSI double parameter.
- bool getStrParam (OsiStrParam key, std::string &value) const
  Get an OSI string parameter.
- bool getHintParam (OsiHintParam key, bool &sense, OsiHintStrength &strength, void *&info) const
  Get an OSI hint.
- void newLanguage (CoinMessages::Language language)
  Change the language for OsiDylp messages.
- void setLanguage (CoinMessages::Language language)
  An alias for OsiDylpSolverInterface::newLanguage.

Methods to obtain solution information

- double getObjValue () const
  Get the objective function value for the solution.
- const double * getColSolution () const
  Return the vector of primal variables for the solution.
- const double * getRowPrice () const
  Return the vector of dual variables for the solution.
- const double * getReducedCost () const
  Return the vector of reduced costs for the solution.
- const double * getRowActivity () const
  Return the vector of row activity for the solution.
- std::vector< double * > getDualRays (int maxNumRays, bool fullRay) const
  Get as many dual rays as the solver can provide.
- std::vector< double * > getPrimalRays (int maxNumRays) const
  Get as many primal rays as the solver can provide.

Simplex API methods

- int canDoSimplexInterface () const
  Return the simplex implementation level.
- void enableFactorization () const
  Prepare the solver for the use of tableau access methods.
- void disableFactorization () const
  Undo the effects of enableFactorization.
- bool basisIsAvailable () const
  Check if an optimal basis is available.
- void getBasisStatus (int *archStatus, int *logStatus) const
  Retrieve status information for architectural and logical variables.
- int setBasisStatus (const int *archStatus, const int *logStatus)
  Set a basis and update the factorization and solution.
- virtual void getReducedGradient (double *columnReducedCosts, double *duals, const double *c) const
  Calculate duals and reduced costs for the given objective coefficients.
- virtual void getBasics (int *index) const
  Get indices of basic variables.
- virtual void getBInvCol (int col, double *betak) const
  Get a column of the basis inverse.
- virtual void getBInvACol (int col, double *abarj) const
  Get a column of the tableau.
- virtual void getBInvRow (int row, double *betai) const
  Get a row of the basis inverse.
- virtual void getBInvARow (int row, double *abari, double *betai=0) const
  Get a row of the tableau.

Debugging Methods

- void activateRowCutDebugger (const char *modelName)
  Activate the row cut debugger.
- void activateRowCutDebugger (const double *solution, bool keepContinuous=false)
  Activate the row cut debugger.

Dylp-specific methods

- void dylp_controlfile (const char *name, const bool silent, const bool mustexist=true)
  Process an options (.spc) file.
- void dylp_logfile (const char *name, bool echo=false)
  Establish a log file.
- void dylp_outfile (const char *name)
  Establish an output (solution and/or statistics) file.
- void dylp_printsoln (bool wantSoln, bool wantStats)
  Print the solution and/or statistics to the output file.
- void setOsiDylpMessages (CoinMessages::Language local_language)
  Set the language for messages.

Unsupported functions

- void branchAndBound ()
  Invoke the solver's built-in branch-and-bound algorithm.

Friends

- void OsiDylpSolverInterfaceUnitTest (const std::string &mpsDir, const std::string &netLibDir)
  Unit test for OsiDylpSolverInterface.

Dylp data structures

These fields hold pointers to the data structures which are used to pass an lp problem to dylp.

- lpopts_struct * initialSolveOptions
  Solver options for an initial solve.
- lpopts_struct * resolveOptions
  Solver options for a resolve.

- lptols_struct * tolerances

  Solver numeric tolerances.

3.34.1 Detailed Description

COIN OSI API for dylp. The class OsiDylpSolverInterface (ODSI) implements the public functions defined for the COIN OsiSolverInterface (OSI) API.

OsiDylpSolverInterface Principles for Users In addition to the principles outlined for the OsiSolverInterface class, ODSI maintains the following:

Construction of a Constraint System: A constraint system can be batch loaded from a file (MPS format) or from a data structure, or it can be built incrementally. When building a constraint system incrementally, keep in mind that you must create a row or column (addRow or addCol, respectively) before you can adjust other properties (row or column bounds, objective, variable values, *etc*.)

Existence of a Solution: For proper operation, OSI requires that a SI maintain a basic primal solution at all times after a problem has been loaded.

When a problem is loaded, ODSI generates a basic primal solution (primal variable values and a matching basis). The solution is not necessarily primal or dual feasible. In terms of the objective function, this solution is pessimistic, but not necessarily worst-case. ODSI does not generate matching values for the dual variables (row prices).

Any successful call to dylp (*i.e.*, a call that results in an optimal, infeasible, or unbounded result, or that terminates on iteration limit) will replace the existing solution with the result of the call to dylp.

It is possible to specify initial values for the primal and dual variables using setColSolution() and setRowPrice(). To specify an initial basis, see the documentation for the CoinWarmStartBasis and OsiDylpWarmStartBasis classes. When these functions are used, it is the responsibility of the client to ensure validity and consistency.

Maintenance of an LP Basis Skirting the edges of the principle that changing the problem invalidates the solution, OsiDylp will maintain a valid basis across two common operations used in branch-and-cut: deletion of a loose constraint and deletion of a nonbasic variable. Arguably the set of allowable modifications could be increased.

Assignment Assignment (operator=()) works pretty much as you'd expect, with one exception. Only one ODSI object can control the dylp solver at a time, so hot start information is not copied on assignment.

Detailed implementation comments are contained in OsiDylpSolverInterface.cpp, which is not normally scanned when generating COIN OSI API documentation.

Definition at line 107 of file OsiDylpSolverInterface.hpp.

3.34.2 Member Function Documentation

3.34.2.1 int OsiDylpSolverInterface::readMps (const char * *filename,* const char * *extension* = `"mps"`)

Read a problem description in MPS format from a file.

3.34.2.2 void OsiDylpSolverInterface::writeMps (const char * *basename,* const char * *extension* = `"mps"`, double *objsense* = `0.0`) const

Write the problem into the specified file in MPS format.

`objsense` == 1 forces the file to be written as a maximisation problem, while -1 forces a minimisation problem. The default of 0 writes the file as maximisation or minimisation using the solver's current setting.

3.34.2.3 int OsiDylpSolverInterface::getNumIntegers ( ) const

Get the number of integer variables.

Counts both binary and general integer variables.

3.34.2.4 double OsiDylpSolverInterface::getObjSense ( ) const

Get the objective function sense (min/max).

A value of 1 indicates minimisation; -1 indicates maximisation.

3.34.2.5 void OsiDylpSolverInterface::setRowName (int *ndx,* std::string *name*)

Set a row name.

Quietly does nothing if the name discipline (OsiNameDiscipline) is auto. Quietly fails if the row index is invalid.

3.34.2.6 void OsiDylpSolverInterface::setColName (int *ndx*, std::string *name*)

Set a column name.

Quietly does nothing if the name discipline (OsiNameDiscipline) is auto. Quietly fails if the column index is invalid.

3.34.2.7 void OsiDylpSolverInterface::setContinuous (int *index*)

Set a single variable to be continuous.

3.34.2.8 void OsiDylpSolverInterface::setInteger (int *index*)

Set a single variable to be integer.

3.34.2.9 void OsiDylpSolverInterface::setObjective (const double * *array*)

Set the objective coefficients for all columns.

3.34.2.10 void OsiDylpSolverInterface::setObjSense (double *sense*)

Set the sense (min/max) of the objective.

Use 1 for minimisation, -1 for maximisation. (The default is minimisation; the objective is multiplied by -1 to maximise.)

3.34.2.11 CoinWarmStart* OsiDylpSolverInterface::getWarmStart ( ) const

Build a warm start object for the current lp solution.

3.34.2.12 bool OsiDylpSolverInterface::setWarmStart (const CoinWarmStart * *warmStart*)

Apply a warm start object.

By definition, a null parameter is a request to synch the warm start basis with the solver. ODSI interprets a 0x0 basis as a request to remove warm start information.

3.34.2.13 void OsiDylpSolverInterface::resolve ( )

Call dylp to reoptimize (warm start).

3.34.2.14 void OsiDylpSolverInterface::markHotStart ( )

Create a hot start snapshot.

3.34.2.15 void OsiDylpSolverInterface::solveFromHotStart ( )

Call dylp to reoptimize (hot start).

3.34.2.16 void OsiDylpSolverInterface::unmarkHotStart ( )

Delete the hot start snapshot.

3.34.2.17 bool OsiDylpSolverInterface::isPrimalObjectiveLimitReached ( ) const

Is the primal objective limit reached?

Put in different terms, quit when the objective value becomes better than the given limit for an acceptable value.

3.34.2.18 bool OsiDylpSolverInterface::isDualObjectiveLimitReached ( ) const

Is the dual objective limit reached?

Put in different terms, quit when the objective value becomes worse than the given limit for an acceptable value.

3.34.2.19 void OsiDylpSolverInterface::setLanguage (CoinMessages::Language *language*) [inline]

An alias for OsiDylpSolverInterface::newLanguage.

Definition at line 574 of file OsiDylpSolverInterface.hpp.

3.34.2.20 std::vector<double *> OsiDylpSolverInterface::getDualRays (int *maxNumRays,* bool *fullRay*) const

Get as many dual rays as the solver can provide.

If `fullRay` is false (the default), the ray will contain only the components associated with the row duals. If `fullRay` is set to `true`, the ray will also contain the components associated with nonbasic variables.

3.34.2.21 int OsiDylpSolverInterface::canDoSimplexInterface ( ) const

Return the simplex implementation level.

3.34.2.22 void OsiDylpSolverInterface::enableFactorization ( ) const

Prepare the solver for the use of tableau access methods.

In order for the tableau methods to work, the ODSI object invoking them must own the solver; the most recent call to optimise the problem must have resulted in an optimal solution; and the solver must be holding retained data structures for that optimal solution. It's much more efficient if the solver is using the full system, but it's not mandatory.

Because this is a const method, we can't force any of this; we can only check.

3.34.2.23 void OsiDylpSolverInterface::disableFactorization ( ) const

Undo the effects of enableFactorization.

Even if resolve was invoked by enableFactorization, little needs to be done here. Ownership of the solver is transferred by invocation, so there's no need to explicitly give it back.

3.34.2.24 bool OsiDylpSolverInterface::basisIsAvailable ( ) const

Check if an optimal basis is available.

3.34.2.25 void OsiDylpSolverInterface::getBasisStatus (int * *archStatus*, int * *logStatus*) const

Retrieve status information for architectural and logical variables.

Retrieve status vectors for architectural (also called structural or column) and logical (also called artificial or row) variables. Returns the same information as getWarmStart, but in a different format.

3.34.2.26 int OsiDylpSolverInterface::setBasisStatus (const int * *archStatus*, const int * *logStatus*)

Set a basis and update the factorization and solution.

Provides the combined functionality of setWarmStart followed by resolve. As with getBasisStatus, the status vectors are coded as integers.

3.34.2.27 virtual void OsiDylpSolverInterface::getReducedGradient (double * *columnReducedCosts*, double * *duals*, const double * *c*) const [virtual]

Calculate duals and reduced costs for the given objective coefficients.

The solver's objective coefficient vector is not changed (cf. setObjectiveAndRefresh)

3.34.2.28 void OsiDylpSolverInterface::activateRowCutDebugger (const char * *modelName*)

Activate the row cut debugger.

Activate the debugger for a model known to the debugger. The debugger will consult an internal database for an optimal solution vector.

3.34.2.29 void OsiDylpSolverInterface::activateRowCutDebugger (const double * *solution*, bool *keepContinuous* = `false`)

Activate the row cut debugger.

Activate the debugger for a model not included in the debugger's internal database. `solution` must be a full solution vector, but only the integer variables need to be correct. The debugger will fill in the continuous variables by solving an lp relaxation with the integer variables fixed as specified. If the given values for the continuous variables should be preserved, set `keepContinuous` to true.

3.34.2.30 void OsiDylpSolverInterface::dylp_printsoln (bool *wantSoln*, bool *wantStats*)

Print the solution and/or statistics to the output file.

3.34.2.31 void OsiDylpSolverInterface::branchAndBound ( )

Invoke the solver's built-in branch-and-bound algorithm.

3.34.3 Friends And Related Function Documentation

3.34.3.1 void OsiDylpSolverInterfaceUnitTest (const std::string & *mpsDir*, const std::string & *netLibDir*) `[friend]`

Unit test for OsiDylpSolverInterface.

Performs various tests to see if ODSI is functioning correctly. Not an exhaustive test, but it'll (usually) catch gross problems.

3.34.4 Member Data Documentation

3.34.4.1 lpopts_struct* OsiDylpSolverInterface::initialSolveOptions

Solver options for an initial solve.

Definition at line 778 of file OsiDylpSolverInterface.hpp.

3.34.4.2 lpopts_struct* OsiDylpSolverInterface::resolveOptions

Solver options for a resolve.

Definition at line 781 of file OsiDylpSolverInterface.hpp.

3.34.4.3 lptols_struct* OsiDylpSolverInterface::tolerances

Solver numeric tolerances.

Definition at line 784 of file OsiDylpSolverInterface.hpp.

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

- OsiDylpSolverInterface.hpp

3.35 OsiDylpWarmStartBasis Class Reference

The dylp warm start class.

`#include <OsiDylpWarmStartBasis.hpp>`

Public Member Functions

Methods to get and set basis information.

Methods for structural and artificial variables are inherited from CoinWarmStartBasis.

Constraint status is coded using the CoinWarmStartBasis::Status codes. Active constraints are coded as atLowerBound, inactive as isFree.

- int numberActiveConstraints () const
  Return the number of active constraints.
- Status getConStatus (int i) const
  Return the status of the specified constraint.
- void setConStatus (int i, Status st)
  Set the status of the specified constraint.
- char * getConstraintStatus ()
  Return the status array for constraints.
- const char * getConstraintStatus () const
  `const` *overload for getConstraintStatus()*
- void setPhase (dyphase_enum phase)
  Set the lp phase for this basis.
- dyphase_enum getPhase () const
  Get the lp phase for this basis.

Basis 'diff' methods

- CoinWarmStartDiff * generateDiff (const CoinWarmStart *const oldCWS) const
  Generate a 'diff' that can convert oldBasis to this basis.
- void applyDiff (const CoinWarmStartDiff *const cwsdDiff)
  Apply `diff` *to this basis.*

Methods to modify the warm start object

- void setSize (int ns, int na)
  Set basis capacity; existing basis is discarded.
- void resize (int numRows, int numCols)
  Set basis capacity; existing basis is maintained.
- void compressRows (int tgtCnt, const int *tgts)
  Delete a set of rows from the basis.
- void deleteRows (int number, const int *which)
  Delete a set of rows from the basis.

- virtual void mergeBasis (const CoinWarmStartBasis ∗src, const XferVec ∗xferRows, const XferVec ∗xferCols)

  Merge entries from a source basis into this basis.

Constructors, destructors, and related functions

- OsiDylpWarmStartBasis ()

  Default constructor (empty object).

- OsiDylpWarmStartBasis (int ns, int na, const char ∗sStat, const char ∗aStat, const char ∗cStat=0)

  Constructs a warm start object with the specified status arrays.

- OsiDylpWarmStartBasis (const CoinWarmStartBasis &cwsb)

  Construct an OsiDylpWarmStartBasis from a CoinWarmStartBasis.

- OsiDylpWarmStartBasis (const OsiDylpWarmStartBasis &ws)

  Copy constructor.

- CoinWarmStart ∗ clone () const

  'Virtual constructor'

- ∼OsiDylpWarmStartBasis ()

  Destructor.

- OsiDylpWarmStartBasis & operator= (const OsiDylpWarmStartBasis &rhs)

  Assignment.

- void assignBasisStatus (int ns, int na, char ∗&sStat, char ∗&aStat, char ∗&cStat)

  Assign the status vectors to be the warm start information.

- void assignBasisStatus (int ns, int na, char ∗&sStat, char ∗&aStat)

  Assign the status vectors to be the warm start information.

Miscellaneous methods

- void print () const

  Prints in readable format (for debug).

- void checkBasis (CoinMessageHandler ∗msghandler=NULL) const

  Performs basis consistency checks (for debug).

3.35.1 Detailed Description

The dylp warm start class. This derived class is necessary because dylp by default works with a subset of the full constraint system. The warm start object needs to contain a list of the active constraints in addition to the status information included in CoinWarmStartBasis. It is also convenient to include the solver phase in the warm start object.

Definition at line 44 of file OsiDylpWarmStartBasis.hpp.

3.35.2 Member Function Documentation

3.35.2.1 Status OsiDylpWarmStartBasis::getConStatus (int *i*) const [inline]

Return the status of the specified constraint.

Definition at line 64 of file OsiDylpWarmStartBasis.hpp.

3.35.2.2 char* OsiDylpWarmStartBasis::getConstraintStatus ( ) [inline]

Return the status array for constraints.

Definition at line 81 of file OsiDylpWarmStartBasis.hpp.

3.35.2.3 dyphase_enum OsiDylpWarmStartBasis::getPhase ( ) const [inline]

Get the lp phase for this basis.

Definition at line 100 of file OsiDylpWarmStartBasis.hpp.

3.35.2.4 CoinWarmStartDiff* OsiDylpWarmStartBasis::generateDiff (const CoinWarmStart *const *oldCWS*) const

Generate a 'diff' that can convert oldBasis to this basis.

3.35.2.5 void OsiDylpWarmStartBasis::compressRows (int *tgtCnt,* const int * *tgts*)

Delete a set of rows from the basis.

Warning

This routine assumes that the set of indices to be deleted is sorted in ascending order and is free from duplicates. Use deleteRows if this is not guaranteed.
The resulting basis is guaranteed valid only if all deleted constraints are slack (hence the associated logicals are basic).

3.35.2.6 void OsiDylpWarmStartBasis::deleteRows (int *number,* const int * *which*)

Delete a set of rows from the basis.

Warning

The resulting basis is guaranteed valid only if all deleted constraints are slack (hence the associated logicals are basic).

3.35.2.7 virtual void OsiDylpWarmStartBasis::mergeBasis (const CoinWarmStartBasis * *src,* const XferVec * *xferRows,* const XferVec * *xferCols*) [virtual]

Merge entries from a source basis into this basis.

Warning

It's the client's responsibility to ensure validity of the merged basis, if that's important to the application.

The vector xferCols (xferRows) specifies runs of entries to be taken from the source basis and placed in this basis. Each entry is a CoinTriple, with first specifying the starting source index of a run, second specifying the starting destination index, and third specifying the run length.

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

- OsiDylpWarmStartBasis.hpp

3.36 OsiDylpWarmStartBasisDiff Class Reference

A 'diff' between two OsiDylpWarmStartBasis objects.

`#include <OsiDylpWarmStartBasis.hpp>`

Public Member Functions

- virtual CoinWarmStartDiff * clone () const

  'Virtual constructor'

- virtual OsiDylpWarmStartBasisDiff & operator= (const OsiDylpWarmStartBasisDiff &rhs)

  Assignment.

- virtual ~OsiDylpWarmStartBasisDiff ()

  Destructor.

3.36.1 Detailed Description

A 'diff' between two OsiDylpWarmStartBasis objects. This class exists in order to hide from the world the details of calculating and representing a 'diff' between two OsiDylpWarmStartBasis objects. For convenience, assignment, cloning, and deletion are visible to the world, and default and copy constructors are visible to derived classes. Knowledge of the rest of this structure, and of generating and applying diffs, is restricted to the functions OsiDylpWarmStartBasis::generateDiff() and OsiDylpWarmStartBasis::applyDiff().

The actual data structure is a pair of unsigned int vectors, diffNdxs_ and diffVals_, and a CoinWarmStartBasisDiff object.

Definition at line 266 of file OsiDylpWarmStartBasis.hpp.

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

- OsiDylpWarmStartBasis.hpp

3.37 parse_any Union Reference

3.37.1 Detailed Description

Definition at line 718 of file dylib_bnfrdr.h.

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

- dylib_bnfrdr.h

3.38 pkcoeff_struct Struct Reference

3.38.1 Detailed Description

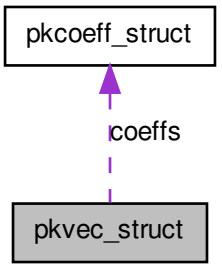
Definition at line 238 of file dy_vector.h.

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

- dy_vector.h

3.39 pkvec_struct Struct Reference

Collaboration diagram for pkvec_struct:



3.39.1 Detailed Description

Definition at line 241 of file dy_vector.h.

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

- dy_vector.h

3.40 POOL Struct Reference

3.40.1 Detailed Description

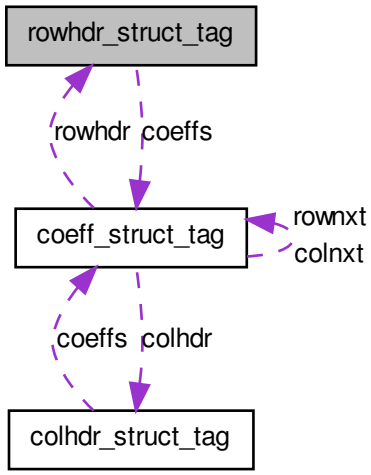
Definition at line 130 of file glplib.h.

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

- glplib.h

3.41 rowhdr_struct_tag Struct Reference

Collaboration diagram for rowhdr_struct_tag:



3.41.1 Detailed Description

Definition at line 137 of file dy_consys.h.

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

- dy_consys.h

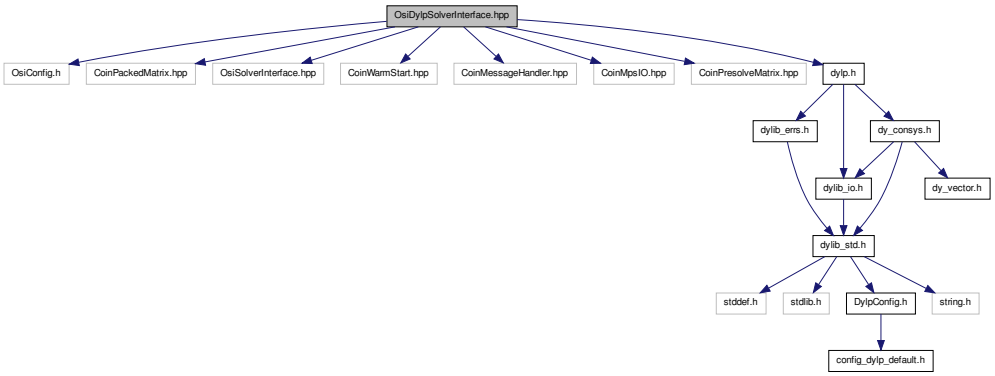
4 File Documentation

4.1 OsiDylpSolverInterface.hpp File Reference

Declarations of the COIN OSI API for the dylp solver.

```
#include "OsiConfig.h"
#include <CoinPackedMatrix.hpp>
#include <OsiSolverInterface.hpp>
#include <CoinWarmStart.hpp>
#include <CoinMessageHandler.hpp>
#include <CoinMpsIO.hpp>
#include <CoinPresolveMatrix.hpp>
#include "dylp.h"
```

Include dependency graph for OsiDylpSolverInterface.hpp:



Classes

- class OsiDylpSolverInterface

  COIN OSI API for dylp.

Enumerations

- enum ODSI_start_enum

Enum to specify cold/warm/hot start.

4.1.1 Detailed Description

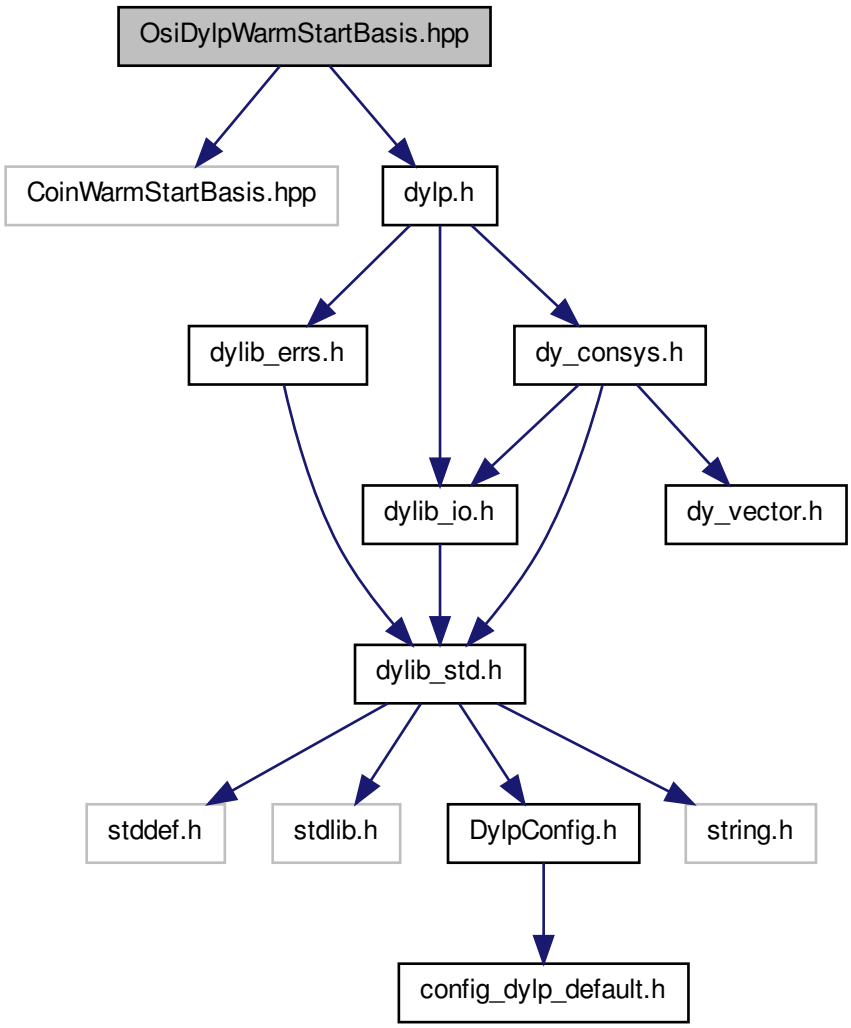
Declarations of the COIN OSI API for the dylp solver. This file contains the declaration of the class OsiDylpSolverInterface (ODSI), an implementation of the COIN OSI API for the dylp LP solver. The documentation here most often provides only brief descriptions of methods. See the OsiSolverInterface documentation for additional details.

Definition in file OsiDylpSolverInterface.hpp.

4.2 OsiDylpWarmStartBasis.hpp File Reference


```
#include "CoinWarmStartBasis.hpp"
#include "dylp.h"
```

Include dependency graph for OsiDylpWarmStartBasis.hpp:



Classes

- class OsiDylpWarmStartBasis

  The dylp warm start class.

- class OsiDylpWarmStartBasisDiff

  A 'diff' between two OsiDylpWarmStartBasis objects.

4.2.1 Detailed Description

Declaration of the warm start class for dylp.

Definition in file OsiDylpWarmStartBasis.hpp.

Index