

DyLP
1.10

Generated by Doxygen 1.8.9.1

Thu Oct 8 2015 22:40:48

Contents

1	Hierarchical Index	1
1.1	Class Hierarchy	1
2	Class Index	9
2.1	Class List	9
3	File Index	11
3.1	File List	11
4	Class Documentation	13
4.1	attvhdr_struct_tag Struct Reference	13
4.1.1	Detailed Description	13
4.2	basis_struct Struct Reference	13
4.2.1	Detailed Description	13
4.3	basisel_struct Struct Reference	13
4.3.1	Detailed Description	13
4.4	bnfdef_any Union Reference	14
4.4.1	Detailed Description	14
4.5	bnfdef_struct Struct Reference	14
4.5.1	Detailed Description	14
4.6	bnfGdef_struct Struct Reference	14
4.6.1	Detailed Description	14
4.7	bnfIdef_struct Struct Reference	14
4.7.1	Detailed Description	14
4.8	bnfLBdef_struct Struct Reference	14
4.8.1	Detailed Description	15
4.9	bnfLdef_struct Struct Reference	15
4.9.1	Detailed Description	15
4.10	bnfNPdef_struct Struct Reference	15

4.10.1 Detailed Description	15
4.11 bnrfref_any Union Reference	15
4.11.1 Detailed Description	15
4.12 bnrfref_struct_tag Struct Reference	15
4.12.1 Detailed Description	15
4.13 bnrfref_type2 Struct Reference	16
4.13.1 Detailed Description	16
4.14 bnrfref_type3 Struct Reference	16
4.14.1 Detailed Description	16
4.15 bnfTdef_struct Struct Reference	16
4.15.1 Detailed Description	16
4.16 coeff_struct_tag Struct Reference	16
4.16.1 Detailed Description	16
4.17 colhdr_struct_tag Struct Reference	17
4.17.1 Detailed Description	17
4.18 conbnd_struct Struct Reference	17
4.18.1 Detailed Description	17
4.19 conmtx_struct Struct Reference	17
4.19.1 Detailed Description	17
4.20 consys_struct Struct Reference	17
4.20.1 Detailed Description	17
4.21 ENV Struct Reference	18
4.21.1 Detailed Description	18
4.22 hel_tag Struct Reference	18
4.22.1 Detailed Description	18
4.23 INV Struct Reference	18
4.23.1 Detailed Description	18
4.24 keytab_entry_internal Struct Reference	18
4.24.1 Detailed Description	18
4.25 lex_struct Struct Reference	19
4.25.1 Detailed Description	19
4.26 Ink_struct_tag Struct Reference	19
4.26.1 Detailed Description	19
4.27 lpopts_struct Struct Reference	19
4.27.1 Detailed Description	19
4.28 lpprob_struct Struct Reference	19
4.28.1 Detailed Description	19

4.29	lpstats_struct Struct Reference	19
4.29.1	Detailed Description	20
4.30	lptols_struct Struct Reference	20
4.30.1	Detailed Description	20
4.31	LUF Struct Reference	20
4.31.1	Detailed Description	20
4.32	LUF_WA Struct Reference	20
4.32.1	Detailed Description	20
4.33	MEM Struct Reference	20
4.33.1	Detailed Description	20
4.34	OsiDyLpSolverInterface Class Reference	21
4.34.1	Detailed Description	26
4.34.2	Member Function Documentation	27
4.34.2.1	readMps	27
4.34.2.2	writeMps	27
4.34.2.3	getNumIntegers	27
4.34.2.4	getObjSense	27
4.34.2.5	setRowName	28
4.34.2.6	setColName	28
4.34.2.7	setContinuous	28
4.34.2.8	setInteger	28
4.34.2.9	setObjective	28
4.34.2.10	setObjSense	28
4.34.2.11	getWarmStart	28
4.34.2.12	setWarmStart	28
4.34.2.13	resolve	29
4.34.2.14	markHotStart	29
4.34.2.15	solveFromHotStart	29
4.34.2.16	unmarkHotStart	29
4.34.2.17	isPrimalObjectiveLimitReached	29
4.34.2.18	isDualObjectiveLimitReached	29
4.34.2.19	setLanguage	29
4.34.2.20	getDualRays	29
4.34.2.21	canDoSimplexInterface	30
4.34.2.22	enableFactorization	30
4.34.2.23	disableFactorization	30
4.34.2.24	basisIsAvailable	30

4.34.2.25	getBasisStatus	30
4.34.2.26	setBasisStatus	30
4.34.2.27	getReducedGradient	31
4.34.2.28	activateRowCutDebugger	31
4.34.2.29	activateRowCutDebugger	31
4.34.2.30	dylp_printsoln	31
4.34.2.31	branchAndBound	31
4.34.3	Friends And Related Function Documentation	31
4.34.3.1	OsiDylpSolverInterfaceUnitTest	31
4.34.4	Member Data Documentation	31
4.34.4.1	initialSolveOptions	32
4.34.4.2	resolveOptions	32
4.34.4.3	tolerances	32
4.35	OsiDylpWarmStartBasis Class Reference	32
4.35.1	Detailed Description	33
4.35.2	Member Function Documentation	34
4.35.2.1	getConStatus	34
4.35.2.2	getConstraintStatus	34
4.35.2.3	getPhase	34
4.35.2.4	generateDiff	34
4.35.2.5	compressRows	34
4.35.2.6	deleteRows	34
4.35.2.7	mergeBasis	35
4.36	OsiDylpWarmStartBasisDiff Class Reference	35
4.36.1	Detailed Description	35
4.37	parse_any Union Reference	36
4.37.1	Detailed Description	36
4.38	pkcoeff_struct Struct Reference	36
4.38.1	Detailed Description	36
4.39	pkvec_struct Struct Reference	36
4.39.1	Detailed Description	36
4.40	POOL Struct Reference	36
4.40.1	Detailed Description	36
4.41	rowhdr_struct_tag Struct Reference	36
4.41.1	Detailed Description	37

CONTENTS	vii
5.1 OsiDyIpSolverInterface.hpp File Reference	39
5.2 OsiDyIpWarmStartBasis.hpp File Reference	39
5.2.1 Detailed Description	39
Index	41

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

<code>_EKKfactinfo</code>	
<code>forcing_constraint_action</code>	
<code>doubleton_action</code>	
<code>tripleton_action</code>	
<code>remove_fixed_action</code>	
<code>std::allocator</code>	
<code>OsiSolverInterface::ApplyCutsReturnCode</code>	
<code>std::array</code>	
<code>attvhdr_struct_tag</code>	13
<code>std::auto_ptr</code>	
<code>std::basic_string</code>	
<code>std::string</code>	
<code>std::wstring</code>	
<code>std::basic_string</code>	
<code>std::basic_string</code>	
<code>basis_struct</code>	13
<code>basisel_struct</code>	13
<code>std::bitset</code>	
<code>BitVector128</code>	
<code>bnfdef_any</code>	14
<code>bnfdef_struct</code>	14
<code>bnfGdef_struct</code>	14
<code>bnfIdef_struct</code>	14
<code>bnfLBdef_struct</code>	14
<code>bnfLdef_struct</code>	15
<code>bnfNPdef_struct</code>	15
<code>bnfref_any</code>	15
<code>bnfref_struct_tag</code>	15
<code>bnfref_type2</code>	16
<code>bnfref_type3</code>	16
<code>bnfTdef_struct</code>	16
<code>coeff_struct_tag</code>	16
<code>CoinAbsFitEq</code>	
<code>CoinArrayWithLength</code>	

- CoinArbitraryArrayWithLength [external]
- CoinBigIndexArrayWithLength [external]
- CoinDoubleArrayWithLength [external]
- CoinFactorizationDoubleArrayWithLength [external]
- CoinFactorizationLongDoubleArrayWithLength [external]
- CoinIntArrayWithLength [external]
- CoinUnsignedIntArrayWithLength [external]
- CoinVoidStarArrayWithLength [external]
- CoinBaseModel [external]
 - CoinModel [external]
 - CoinStructuredModel [external]
- CoinBuild [external]
- CoinDenseVector< T > [external]
- CoinError [external]
- CoinExternalVectorFirstGreater_2< class, class, class > [external]
- CoinExternalVectorFirstGreater_3< class, class, class, class > [external]
- CoinExternalVectorFirstLess_2< class, class, class > [external]
- CoinExternalVectorFirstLess_3< class, class, class, class > [external]
- CoinFactorization [external]
- CoinFileIOBase [external]
 - CoinFileInput [external]
 - CoinFileOutput [external]
- CoinFirstAbsGreater_2< class, class > [external]
- CoinFirstAbsGreater_3< class, class, class > [external]
- CoinFirstAbsLess_2< class, class > [external]
- CoinFirstAbsLess_3< class, class, class > [external]
- CoinFirstGreater_2< class, class > [external]
- CoinFirstGreater_3< class, class, class > [external]
- CoinFirstLess_2< class, class > [external]
- CoinFirstLess_3< class, class, class > [external]
- CoinLpIO::CoinHashLink [external]
- CoinMpsIO::CoinHashLink [external]
- CoinIndexedVector [external]
 - CoinPartitionedVector [external]
- CoinLpIO [external]
- CoinMessageHandler [external]
- CoinMessages [external]
 - CoinMessage [external]
- CoinModelHash [external]
- CoinModelHash2 [external]
- CoinModelHashLink [external]
- CoinModelInfo2 [external]
- CoinModelLink [external]
- CoinModelLinkedList [external]
- CoinModelTriple [external]
- CoinMpsCardReader [external]
- CoinMpsIO [external]
- CoinOneMessage [external]
- CoinOtherFactorization [external]
 - CoinDenseFactorization [external]
 - CoinOslFactorization [external]
 - CoinSimpFactorization [external]
- CoinPackedMatrix [external]
- CoinPackedVectorBase [external]

```

    CoinPackedVector [external]
    CoinShallowPackedVector [external]
CoinPair< S, T > [external]
CoinParam [external]
CoinPrePostsolveMatrix [external]
    CoinPostsolveMatrix [external]
    CoinPresolveMatrix [external]
CoinPresolveAction [external]
    do_tighten_action [external]
    doubleton_action [external]
    drop_empty_cols_action [external]
    drop_empty_rows_action [external]
    drop_zero_coefficients_action [external]
    dupcol_action [external]
    duprow3_action [external]
    duprow_action [external]
    forcing_constraint_action [external]
    gubrow_action [external]
    implied_free_action [external]
    isolated_constraint_action [external]
    make_fixed_action [external]
    remove_dual_action [external]
    remove_fixed_action [external]
    slack_doubleton_action [external]
    slack_singleton_action [external]
    subst_constraint_action [external]
    tripleton_action [external]
    twotwo_action [external]
    useless_constraint_action [external]
CoinPresolveMonitor [external]
CoinRational [external]
CoinRelFltEq [external]
CoinSearchTreeBase [external]
    CoinSearchTree< class > [external]
CoinSearchTreeCompareBest [external]
CoinSearchTreeCompareBreadth [external]
CoinSearchTreeCompareDepth [external]
CoinSearchTreeComparePreferred [external]
CoinSearchTreeManager [external]
CoinSet [external]
    CoinSosSet [external]
CoinSnapshot [external]
CoinThreadRandom [external]
CoinTimer [external]
CoinTreeNode [external]
CoinTreeSiblings [external]
CoinTriple< S, T, U > [external]
CoinWarmStart [external]
    CoinWarmStartBasis [external]

        OsiDyIpWarmStartBasis . . . . . 32
    CoinWarmStartDual [external]
    CoinWarmStartPrimalDual [external]
    CoinWarmStartVector< T > [external]
    CoinWarmStartVector< double > [external]

```

CoinWarmStartVector< U >[external]	
CoinWarmStartVectorPair< T, U >[external]	
CoinWarmStartDiff[external]	
CoinWarmStartBasisDiff[external]	
OsiDyIpWarmStartBasisDiff	35
CoinWarmStartDualDiff[external]	
CoinWarmStartPrimalDualDiff[external]	
CoinWarmStartVectorDiff< T >[external]	
CoinWarmStartVectorDiff< double >[external]	
CoinWarmStartVectorDiff< U >[external]	
CoinWarmStartVectorPairDiff< T, U >[external]	
CoinYacc[external]	
colhdr_struct_tag	17
std::complex	
conbnd_struct	17
conmtx_struct	17
std::multimap< K, T >::const_iterator	
std::unordered_multimap< K, T >::const_iterator	
std::set< K >::const_iterator	
OsiCuts::const_iterator[external]	
std::unordered_set< K >::const_iterator	
std::multiset< K >::const_iterator	
std::unordered_multiset< K >::const_iterator	
std::vector< T >::const_iterator	
std::basic_string< Char >::const_iterator	
std::string::const_iterator	
std::wstring::const_iterator	
std::deque< T >::const_iterator	
std::list< T >::const_iterator	
std::forward_list< T >::const_iterator	
std::map< K, T >::const_iterator	
std::unordered_map< K, T >::const_iterator	
std::multimap< K, T >::const_reverse_iterator	
std::unordered_multimap< K, T >::const_reverse_iterator	
std::set< K >::const_reverse_iterator	
std::unordered_set< K >::const_reverse_iterator	
std::multiset< K >::const_reverse_iterator	
std::unordered_multiset< K >::const_reverse_iterator	
std::vector< T >::const_reverse_iterator	
std::basic_string< Char >::const_reverse_iterator	
std::string::const_reverse_iterator	
std::wstring::const_reverse_iterator	
std::deque< T >::const_reverse_iterator	
std::list< T >::const_reverse_iterator	
std::forward_list< T >::const_reverse_iterator	
std::map< K, T >::const_reverse_iterator	
std::unordered_map< K, T >::const_reverse_iterator	
consys_struct	17
std::deque< T >	
dropped_zero[external]	
EKKHlink[external]	
ENV	18
std::error_category	
std::error_code	

```

std::error_condition
std::exception
    std::bad_alloc
    std::bad_cast
    std::bad_exception
    std::bad_typeid
    std::ios_base::failure
    std::logic_error
        std::domain_error
        std::invalid_argument
        std::length_error
        std::out_of_range
    std::runtime_error
        std::overflow_error
        std::range_error
        std::underflow_error
FactorPointers[external]
std::forward_list< T >
glp_prob[external]
hel_tag . . . . . 18
INV . . . . . 18
std::ios_base
    basic_ios< char >
    basic_ios< wchar_t >
    std::basic_ios
        basic_istream< char >
        basic_istream< wchar_t >
        basic_ostream< char >
        basic_ostream< wchar_t >
        std::basic_istream
            basic_ifstream< char >
            basic_ifstream< wchar_t >
            basic_iostream< char >
            basic_iostream< wchar_t >
            basic_istreamstream< char >
            basic_istreamstream< wchar_t >
            std::basic_ifstream
                std::ifstream
                std::wifstream
            std::basic_iostream
                basic_fstream< char >
                basic_fstream< wchar_t >
                basic_stringstream< char >
                basic_stringstream< wchar_t >
                std::basic_fstream
                    std::fstream
                    std::wfstream
                std::basic_stringstream
                    std::stringstream
                    std::wstringstream
            std::basic_istreamstream
                std::istreamstream
                std::wistreamstream
        std::istream

```

std::wistream	
std::basic_ostream	
basic_iostream< char >	
basic_iostream< wchar_t >	
basic_ofstream< char >	
basic_ofstream< wchar_t >	
basic_ostringstream< char >	
basic_ostringstream< wchar_t >	
std::basic_iostream	
std::basic_ofstream	
std::ofstream	
std::wofstream	
std::basic_ostringstream	
std::ostringstream	
std::wostringstream	
std::ostream	
std::wostream	
std::ios	
std::wios	
std::map< K, T >::iterator	
std::multimap< K, T >::iterator	
std::unordered_multimap< K, T >::iterator	
std::set< K >::iterator	
std::unordered_set< K >::iterator	
std::multiset< K >::iterator	
std::unordered_multiset< K >::iterator	
std::forward_list< T >::iterator	
std::list< T >::iterator	
std::deque< T >::iterator	
OsiCuts::iterator[external]	
std::string::iterator	
std::basic_string< Char >::iterator	
std::wstring::iterator	
std::vector< T >::iterator	
std::unordered_map< K, T >::iterator	
keytab_entry_internal	18
lex_struct	19
std::list< T >	
Ink_struct_tag	19
lpopts_struct	19
lpprob_struct	19
lpstats_struct	19
lptols_struct	20
LUF	20
LUF_WA	20
std::map< K, T >	
MEM	20
std::multimap< K, T >	
std::multiset< K >	
OsiAuxInfo[external]	
OsiBabSolver[external]	
OsiBranchingInformation[external]	
OsiBranchingObject[external]	
OsiTwoWayBranchingObject[external]	

OsiIntegerBranchingObject [external]	
OsiLotsizeBranchingObject [external]	
OsiSOSBranchingObject [external]	
OsiChooseVariable [external]	
OsiChooseStrong [external]	
OsiCut [external]	
OsiColCut [external]	
OsiRowCut [external]	
OsiRowCut2 [external]	
OsiCuts [external]	
OsiHotInfo [external]	
OsiObject [external]	
OsiObject2 [external]	
OsiLotsize [external]	
OsiSimpleInteger [external]	
OsiSOS [external]	
OsiPresolve [external]	
OsiPseudoCosts [external]	
OsiRowCutDebugger [external]	
OsiSolverBranch [external]	
OsiSolverInterface [external]	
OsiCpxSolverInterface [external]	
OsiDyIpSolverInterface	21
OsiGlpkSolverInterface [external]	
OsiGrbSolverInterface [external]	
OsiMskSolverInterface [external]	
OsiSpxSolverInterface [external]	
OsiXprSolverInterface [external]	
OsiSolverResult [external]	
parse_any	36
pkcoeff_struct	36
pkvec_struct	36
POOL	36
presolvehlink [external]	
std::priority_queue< T >	
std::queue< T >	
Coin::ReferencedObject [external]	
std::set< K >::reverse_iterator	
std::vector< T >::reverse_iterator	
std::unordered_set< K >::reverse_iterator	
std::map< K, T >::reverse_iterator	
std::multimap< K, T >::reverse_iterator	
std::deque< T >::reverse_iterator	
std::list< T >::reverse_iterator	
std::unordered_multiset< K >::reverse_iterator	
std::forward_list< T >::reverse_iterator	
std::basic_string< Char >::reverse_iterator	
std::wstring::reverse_iterator	
std::unordered_multimap< K, T >::reverse_iterator	
std::multiset< K >::reverse_iterator	
std::string::reverse_iterator	
std::unordered_map< K, T >::reverse_iterator	
rowhdr_struct_tag	36
std::set< K >	

`std::smart_ptr< T >`
`Coin::SmartPtr< T >[external]`
`std::stack< T >`
`symrec[external]`
`std::system_error`
`OsiUnitTest::TestOutcome[external]`
`OsiUnitTest::TestOutcomes[external]`
`std::thread`
`std::unique_ptr< T >`
`std::unordered_map< K, T >`
`std::unordered_multimap< K, T >`
`std::unordered_multiset< K >`
`std::unordered_set< K >`
`std::valarray< T >`
`std::vector< T >`
`std::vector< double >`
`std::weak_ptr< T >`
K
S
T
U

Chapter 2

Class Index

2.1 Class List

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

attvhdr_struct_tag	13
basis_struct	13
basisel_struct	13
bnfdef_any	14
bnfdef_struct	14
bnfGdef_struct	14
bnfIdef_struct	14
bnfLBdef_struct	14
bnfLdef_struct	15
bnfNPdef_struct	15
bnfref_any	15
bnfref_struct_tag	15
bnfref_type2	16
bnfref_type3	16
bnfTdef_struct	16
coeff_struct_tag	16
colhdr_struct_tag	17
conbnd_struct	17
conmtx_struct	17
consys_struct	17
ENV	18
hel_tag	18
INV	18
keytab_entry_internal	18
lex_struct	19
lnk_struct_tag	19
lpopts_struct	19
lpprob_struct	19
lpstats_struct	19
lptols_struct	20
LUF	20
LUF_WA	20
MEM	20

OsiDyIpSolverInterface	
COIN OSI API for dylp	21
OsiDyIpWarmStartBasis	
The dylp warm start class	32
OsiDyIpWarmStartBasisDiff	
A 'diff' between two OsiDyIpWarmStartBasis objects	35
parse_any	36
pkcoeff_struct	36
pkvec_struct	36
POOL	36
rowhdr_struct_tag	36

Chapter 3

File Index

3.1 File List

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

config_default.h	??
config_dylp_default.h	??
dy_cmdint.h	??
dy_consys.h	??
dy_vector.h	??
dylib_bnfdr.h	??
dylib_errs.h	??
dylib_fortran.h	??
dylib_hash.h	??
dylib_io.h	??
dylib_keytab.h	??
dylib_std.h	??
dylib_strtns.h	??
dylp.h	??
DylpConfig.h	??
glpinv.h	??
glplib.h	??
glpluf.h	??
OsiDylpMessages.hpp	??
OsiDylpSolverInterface.hpp	
Declarations of the COIN OSI API for the dylp solver	39
OsiDylpWarmStartBasis.hpp	
Copyright (C) 2003 – 2007 Lou Hafer, International Business Machines Corporation and others	39

Chapter 4

Class Documentation

4.1 attvhdr_struct_tag Struct Reference

Collaboration diagram for attvhdr_struct_tag:

4.1.1 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

4.2 basis_struct Struct Reference

Collaboration diagram for basis_struct:

4.2.1 Detailed Description

Definition at line 453 of file dylp.h.

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

- dylp.h

4.3 basisel_struct Struct Reference

4.3.1 Detailed Description

Definition at line 451 of file dylp.h.

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

- dylp.h

4.4 bnfdef_any Union Reference

Collaboration diagram for bnfdef_any:

4.4.1 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

4.5 bnfdef_struct Struct Reference

4.5.1 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

4.6 bnfGdef_struct Struct Reference

Collaboration diagram for bnfGdef_struct:

4.6.1 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

4.7 bnfldef_struct Struct Reference

4.7.1 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

4.8 bnflBdef_struct Struct Reference

Collaboration diagram for bnflBdef_struct:

4.8.1 Detailed Description

Definition at line 406 of file dylib_bnfdr.h.

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

- dylib_bnfdr.h

4.9 bnfLdef_struct Struct Reference

4.9.1 Detailed Description

Definition at line 371 of file dylib_bnfdr.h.

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

- dylib_bnfdr.h

4.10 bnfNPdef_struct Struct Reference

Collaboration diagram for bnfNPdef_struct:

4.10.1 Detailed Description

Definition at line 301 of file dylib_bnfdr.h.

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

- dylib_bnfdr.h

4.11 bnfref_any Union Reference

Collaboration diagram for bnfref_any:

4.11.1 Detailed Description

Definition at line 522 of file dylib_bnfdr.h.

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

- dylib_bnfdr.h

4.12 bnfref_struct_tag Struct Reference

4.12.1 Detailed Description

Definition at line 464 of file dylib_bnfdr.h.

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

- `dylib_bnfrdr.h`

4.13 `bnfref_type2` Struct Reference

4.13.1 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`

4.14 `bnfref_type3` Struct Reference

Collaboration diagram for `bnfref_type3`:

4.14.1 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`

4.15 `bnfTdef_struct` Struct Reference

4.15.1 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`

4.16 `coeff_struct_tag` Struct Reference

Collaboration diagram for `coeff_struct_tag`:

4.16.1 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`

4.17 colhdr_struct_tag Struct Reference

Collaboration diagram for colhdr_struct_tag:

4.17.1 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

4.18 conbnd_struct Struct Reference

4.18.1 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

4.19 conmtx_struct Struct Reference

Collaboration diagram for conmtx_struct:

4.19.1 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

4.20 consys_struct Struct Reference

Collaboration diagram for consys_struct:

4.20.1 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

4.21 ENV Struct Reference

Collaboration diagram for ENV:

4.21.1 Detailed Description

Definition at line 53 of file `gplib.h`.

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

- `gplib.h`

4.22 hel_tag Struct Reference

Collaboration diagram for `hel_tag`:

4.22.1 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`

4.23 INV Struct Reference

Collaboration diagram for INV:

4.23.1 Detailed Description

Definition at line 78 of file `gpinv.h`.

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

- `gpinv.h`

4.24 keytab_entry_internal Struct Reference

4.24.1 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`

4.25 lex_struct Struct Reference

4.25.1 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

4.26 Ink_struct_tag Struct Reference

Collaboration diagram for Ink_struct_tag:

4.26.1 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

4.27 lpopts_struct Struct Reference

4.27.1 Detailed Description

Definition at line 1114 of file dylp.h.

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

- dylp.h

4.28 lpprob_struct Struct Reference

Collaboration diagram for lpprob_struct:

4.28.1 Detailed Description

Definition at line 586 of file dylp.h.

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

- dylp.h

4.29 lpstats_struct Struct Reference

4.29.1 Detailed Description

Definition at line 1303 of file dylp.h.

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

- dylp.h

4.30 Iptols_struct Struct Reference

4.30.1 Detailed Description

Definition at line 666 of file dylp.h.

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

- dylp.h

4.31 LUF Struct Reference

4.31.1 Detailed Description

Definition at line 83 of file glpluf.h.

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

- glpluf.h

4.32 LUF_WA Struct Reference

4.32.1 Detailed Description

Definition at line 270 of file glpluf.h.

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

- glpluf.h

4.33 MEM Struct Reference

Collaboration diagram for MEM:

4.33.1 Detailed Description

Definition at line 105 of file glplib.h.

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

- glplib.h

4.34 OsiDyIpsolverInterface Class Reference

COIN OSI API for dyIps.

```
#include <OsiDyIpsolverInterface.hpp>
```

Inheritance diagram for OsiDyIpsolverInterface:

Collaboration diagram for OsiDyIpsolverInterface:

Public Member Functions

Constructors and Destructors

- [OsiDyIpsolverInterface](#) ()
Default constructor.
- [OsiDyIpsolverInterface](#) (const [OsiDyIpsolverInterface](#) &src)
Copy constructor.
- [OsiSolverInterface](#) * [clone](#) (bool copyData=true) const
Clone the solver object.
- [OsiDyIpsolverInterface](#) & [operator=](#) (const [OsiDyIpsolverInterface](#) &rhs)
Assignment.
- [~OsiDyIpsolverInterface](#) ()
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 [OsiDyIpWarmStartBasis](#) 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 [setDbiParam](#) (OsiDbiParam 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 [getDbiParam](#) (OsiDbiParam 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.

DyIp-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.

4.34.1 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`.

4.34.2 Member Function Documentation

4.34.2.1 `int OsiDyLpSolverInterface::readMps (const char * filename, const char * extension = "mps") [virtual]`

Read a problem description in MPS format from a file.

Reimplemented from `OsiSolverInterface`.

4.34.2.2 `void OsiDyLpSolverInterface::writeMps (const char * basename, const char * extension = "mps", double objsense = 0.0) const [virtual]`

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.

Implements `OsiSolverInterface`.

4.34.2.3 `int OsiDyLpSolverInterface::getNumIntegers () const [virtual]`

Get the number of integer variables.

Counts both binary and general integer variables.

Reimplemented from `OsiSolverInterface`.

4.34.2.4 `double OsiDyLpSolverInterface::getObjSense () const [virtual]`

Get the objective function sense (min/max)

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

Implements `OsiSolverInterface`.

4.34.2.5 `void OsiDyLpSolverInterface::setRowName (int ndx, std::string name) [virtual]`

Set a row name.

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

Reimplemented from **OsiSolverInterface**.

4.34.2.6 `void OsiDyLpSolverInterface::setColName (int ndx, std::string name) [virtual]`

Set a column name.

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

Reimplemented from **OsiSolverInterface**.

4.34.2.7 `void OsiDyLpSolverInterface::setContinuous (int index) [virtual]`

Set a single variable to be continuous.

Implements **OsiSolverInterface**.

4.34.2.8 `void OsiDyLpSolverInterface::setInteger (int index) [virtual]`

Set a single variable to be integer.

Implements **OsiSolverInterface**.

4.34.2.9 `void OsiDyLpSolverInterface::setObjective (const double * array) [virtual]`

Set the objective coefficients for all columns.

Reimplemented from **OsiSolverInterface**.

4.34.2.10 `void OsiDyLpSolverInterface::setObjSense (double sense) [virtual]`

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.)

Implements **OsiSolverInterface**.

4.34.2.11 `CoinWarmStart* OsiDyLpSolverInterface::getWarmStart () const [virtual]`

Build a warm start object for the current lp solution.

Implements **OsiSolverInterface**.

4.34.2.12 `bool OsiDyLpSolverInterface::setWarmStart (const CoinWarmStart * warmStart) [virtual]`

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.

Implements **OsiSolverInterface**.

4.34.2.13 `void OsiDyIpSolverInterface::resolve () [virtual]`

Call dylp to reoptimize (warm start).

Implements **OsiSolverInterface**.

4.34.2.14 `void OsiDyIpSolverInterface::markHotStart () [virtual]`

Create a hot start snapshot.

Reimplemented from **OsiSolverInterface**.

4.34.2.15 `void OsiDyIpSolverInterface::solveFromHotStart () [virtual]`

Call dylp to reoptimize (hot start).

Reimplemented from **OsiSolverInterface**.

4.34.2.16 `void OsiDyIpSolverInterface::unmarkHotStart () [virtual]`

Delete the hot start snapshot.

Reimplemented from **OsiSolverInterface**.

4.34.2.17 `bool OsiDyIpSolverInterface::isPrimalObjectiveLimitReached () const [virtual]`

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.

Reimplemented from **OsiSolverInterface**.

4.34.2.18 `bool OsiDyIpSolverInterface::isDualObjectiveLimitReached () const [virtual]`

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.

Reimplemented from **OsiSolverInterface**.

4.34.2.19 `void OsiDyIpSolverInterface::setLanguage (CoinMessages::Language language) [inline]`

An alias for [OsiDyIpSolverInterface::newLanguage](#).

Definition at line 574 of file OsiDyIpSolverInterface.hpp.

4.34.2.20 `std::vector<double*> OsiDyIpSolverInterface::getDualRays (int maxNumRays, bool fullRay) const [virtual]`

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.

Implements **OsiSolverInterface**.

4.34.2.21 `int OsiDyIpSolverInterface::canDoSimplexInterface () const [virtual]`

Return the simplex implementation level.

Reimplemented from **OsiSolverInterface**.

4.34.2.22 `void OsiDyIpSolverInterface::enableFactorization () const [virtual]`

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.

Reimplemented from **OsiSolverInterface**.

4.34.2.23 `void OsiDyIpSolverInterface::disableFactorization () const [virtual]`

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.

Reimplemented from **OsiSolverInterface**.

4.34.2.24 `bool OsiDyIpSolverInterface::basisIsAvailable () const [virtual]`

Check if an optimal basis is available.

Reimplemented from **OsiSolverInterface**.

4.34.2.25 `void OsiDyIpSolverInterface::getBasisStatus (int * archStatus, int * logStatus) const [virtual]`

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.

Reimplemented from **OsiSolverInterface**.

4.34.2.26 `int OsiDyIpSolverInterface::setBasisStatus (const int * archStatus, const int * logStatus) [virtual]`

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.

Reimplemented from **OsiSolverInterface**.

4.34.2.27 `virtual void OsiDyIpSolverInterface::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`)

Reimplemented from **OsiSolverInterface**.

4.34.2.28 `void OsiDyIpSolverInterface::activateRowCutDebugger (const char * modelName)` [virtual]

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.

Reimplemented from **OsiSolverInterface**.

4.34.2.29 `void OsiDyIpSolverInterface::activateRowCutDebugger (const double * solution, bool keepContinuous = false)` [virtual]

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.

Reimplemented from **OsiSolverInterface**.

4.34.2.30 `void OsiDyIpSolverInterface::dylp_printsoln (bool wantSoln, bool wantStats)`

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

4.34.2.31 `void OsiDyIpSolverInterface::branchAndBound ()` [virtual]

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

Implements **OsiSolverInterface**.

4.34.3 Friends And Related Function Documentation

4.34.3.1 `void OsiDyIpSolverInterfaceUnitTest (const std::string & mpsDir, const std::string & netLibDir)` [friend]

Unit test for [OsiDyIpSolverInterface](#).

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

4.34.4 Member Data Documentation

4.34.4.1 `Ipopts_struct*` `OsiDyIpSolverInterface::initialSolveOptions`

Solver options for an initial solve.

Definition at line 778 of file `OsiDyIpSolverInterface.hpp`.

4.34.4.2 `Ipopts_struct*` `OsiDyIpSolverInterface::resolveOptions`

Solver options for a resolve.

Definition at line 781 of file `OsiDyIpSolverInterface.hpp`.

4.34.4.3 `Iptols_struct*` `OsiDyIpSolverInterface::tolerances`

Solver numeric tolerances.

Definition at line 784 of file `OsiDyIpSolverInterface.hpp`.

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

- [OsiDyIpSolverInterface.hpp](#)

4.35 OsiDyIpWarmStartBasis Class Reference

The dylp warm start class.

```
#include <OsiDyIpWarmStartBasis.hpp>
```

Inheritance diagram for `OsiDyIpWarmStartBasis`:

Collaboration diagram for `OsiDyIpWarmStartBasis`:

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

- [OsiDyIpWarmStartBasis](#) ()
Default constructor (empty object)
- [OsiDyIpWarmStartBasis](#) (int ns, int na, const char *sStat, const char *aStat, const char *cStat=0)
Constructs a warm start object with the specified status arrays.
- [OsiDyIpWarmStartBasis](#) (const **CoinWarmStartBasis** &cwsb)
*Construct an [OsiDyIpWarmStartBasis](#) from a **CoinWarmStartBasis**.*
- [OsiDyIpWarmStartBasis](#) (const [OsiDyIpWarmStartBasis](#) &ws)
Copy constructor.
- **CoinWarmStart** * [clone](#) () const
'Virtual constructor'
- [~OsiDyIpWarmStartBasis](#) ()
Destructor.
- [OsiDyIpWarmStartBasis](#) & [operator=](#) (const [OsiDyIpWarmStartBasis](#) &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)

4.35.1 Detailed Description

The dyIp warm start class.

This derived class is necessary because dyIp 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 OsiDyIpWarmStartBasis.hpp.

4.35.2 Member Function Documentation

4.35.2.1 Status OsiDyIpWarmStartBasis::getConStatus (int *i*) const [inline]

Return the status of the specified constraint.

Definition at line 64 of file OsiDyIpWarmStartBasis.hpp.

4.35.2.2 char* OsiDyIpWarmStartBasis::getConstraintStatus () [inline]

Return the status array for constraints.

Definition at line 81 of file OsiDyIpWarmStartBasis.hpp.

4.35.2.3 dyphase_enum OsiDyIpWarmStartBasis::getPhase () const [inline]

Get the lp phase for this basis.

Definition at line 100 of file OsiDyIpWarmStartBasis.hpp.

4.35.2.4 CoinWarmStartDiff* OsiDyIpWarmStartBasis::generateDiff (const CoinWarmStart *const *oldCWS*) const [virtual]

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

Reimplemented from **CoinWarmStartBasis**.

4.35.2.5 void OsiDyIpWarmStartBasis::compressRows (int *tgtCnt*, const int * *tgts*) [virtual]

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).

Reimplemented from **CoinWarmStartBasis**.

4.35.2.6 void OsiDyIpWarmStartBasis::deleteRows (int *number*, const int * *which*) [virtual]

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).

Reimplemented from **CoinWarmStartBasis**.

4.35.2.7 `virtual void OsiDyIpWarmStartBasis::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.

Reimplemented from **CoinWarmStartBasis**.

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

- [OsiDyIpWarmStartBasis.hpp](#)

4.36 OsiDyIpWarmStartBasisDiff Class Reference

A 'diff' between two [OsiDyIpWarmStartBasis](#) objects.

```
#include <OsiDyIpWarmStartBasis.hpp>
```

Inheritance diagram for [OsiDyIpWarmStartBasisDiff](#):

Collaboration diagram for [OsiDyIpWarmStartBasisDiff](#):

Public Member Functions

- virtual **CoinWarmStartDiff** * `clone` () const
'Virtual constructor'
- virtual [OsiDyIpWarmStartBasisDiff](#) & `operator=` (const [OsiDyIpWarmStartBasisDiff](#) &rhs)
Assignment.
- virtual `~OsiDyIpWarmStartBasisDiff` ()
Destructor.

4.36.1 Detailed Description

A 'diff' between two [OsiDyIpWarmStartBasis](#) objects.

This class exists in order to hide from the world the details of calculating and representing a 'diff' between two [OsiDyIpWarmStartBasis](#) 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 [OsiDyIpWarmStartBasis::generateDiff\(\)](#) and [OsiDyIpWarmStartBasis::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 [OsiDyIpWarmStartBasis.hpp](#).

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

- [OsiDyIpWarmStartBasis.hpp](#)

4.37 `parse_any` Union Reference

4.37.1 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`

4.38 `pkcoeff_struct` Struct Reference

4.38.1 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`

4.39 `pkvec_struct` Struct Reference

Collaboration diagram for `pkvec_struct`:

4.39.1 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`

4.40 `POOL` Struct Reference

4.40.1 Detailed Description

Definition at line 130 of file `gplib.h`.

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

- `gplib.h`

4.41 `rowhdr_struct_tag` Struct Reference

Collaboration diagram for `rowhdr_struct_tag`:

4.41.1 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

Chapter 5

File Documentation

5.1 OsiDyIpSolverInterface.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 OsiDyIpSolverInterface.hpp:

5.2 OsiDyIpWarmStartBasis.hpp File Reference

Copyright (C) 2003 – 2007 Lou Hafer, International Business Machines Corporation and others.

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

Include dependency graph for OsiDyIpWarmStartBasis.hpp:

Classes

- class [OsiDyIpWarmStartBasis](#)
The dylp warm start class.
- class [OsiDyIpWarmStartBasisDiff](#)
A 'diff' between two [OsiDyIpWarmStartBasis](#) objects.

5.2.1 Detailed Description

Copyright (C) 2003 – 2007 Lou Hafer, International Business Machines Corporation and others.

All Rights Reserved.

This file is a portion of the COIN/OSI interface for dylp and is licensed under the terms of the Eclipse Public License (EPL)

Declaration of the warm start class for dylp.

Index

activateRowCutDebugger
 OsiDyIpSolverInterface, 31
attvhdr_struct_tag, 13

basis_struct, 13
basisIsAvailable
 OsiDyIpSolverInterface, 30
basisel_struct, 13
bnfGdef_struct, 14
bnfIdef_struct, 14
bnfLBdef_struct, 14
bnfLdef_struct, 15
bnfNPdef_struct, 15
bnfTdef_struct, 16
bnfdef_any, 14
bnfdef_struct, 14
bnfref_any, 15
bnfref_struct_tag, 15
bnfref_type2, 16
bnfref_type3, 16
branchAndBound
 OsiDyIpSolverInterface, 31

canDoSimplexInterface
 OsiDyIpSolverInterface, 30
coeff_struct_tag, 16
colhdr_struct_tag, 17
compressRows
 OsiDyIpWarmStartBasis, 34
conbnd_struct, 17
conmtx_struct, 17
consys_struct, 17

deleteRows
 OsiDyIpWarmStartBasis, 34
disableFactorization
 OsiDyIpSolverInterface, 30
dylp_printsoln
 OsiDyIpSolverInterface, 31

ENV, 18
enableFactorization
 OsiDyIpSolverInterface, 30

generateDiff
 OsiDyIpWarmStartBasis, 34

getBasisStatus
 OsiDyIpSolverInterface, 30
getConStatus
 OsiDyIpWarmStartBasis, 34
getConstraintStatus
 OsiDyIpWarmStartBasis, 34
getDualRays
 OsiDyIpSolverInterface, 29
getNumIntegers
 OsiDyIpSolverInterface, 27
getObjSense
 OsiDyIpSolverInterface, 27
getPhase
 OsiDyIpWarmStartBasis, 34
getReducedGradient
 OsiDyIpSolverInterface, 30
getWarmStart
 OsiDyIpSolverInterface, 28

hel_tag, 18

INV, 18
initialSolveOptions
 OsiDyIpSolverInterface, 31
isDualObjectiveLimitReached
 OsiDyIpSolverInterface, 29
isPrimalObjectiveLimitReached
 OsiDyIpSolverInterface, 29

keytab_entry_internal, 18

LUF, 20
LUF_WA, 20
lex_struct, 19
lnk_struct_tag, 19
lpopts_struct, 19
lpprob_struct, 19
lpstats_struct, 19
lptols_struct, 20

MEM, 20
markHotStart
 OsiDyIpSolverInterface, 29
mergeBasis
 OsiDyIpWarmStartBasis, 34
OsiDyIpSolverInterface, 21

- activateRowCutDebugger, 31
- basisIsAvailable, 30
- branchAndBound, 31
- canDoSimplexInterface, 30
- disableFactorization, 30
- dylp_printsoln, 31
- enableFactorization, 30
- getBasisStatus, 30
- getDualRays, 29
- getNumIntegers, 27
- getObjSense, 27
- getReducedGradient, 30
- getWarmStart, 28
- initialSolveOptions, 31
- isDualObjectiveLimitReached, 29
- isPrimalObjectiveLimitReached, 29
- markHotStart, 29
- OsiDylpSolverInterfaceUnitTest, 31
- readMps, 27
- resolve, 29
 - resolveOptions, 32
 - setBasisStatus, 30
 - setColName, 28
 - setContinuous, 28
 - setInteger, 28
 - setLanguage, 29
 - setObjSense, 28
 - setObjective, 28
 - setRowName, 27
 - setWarmStart, 28
 - solveFromHotStart, 29
 - tolerances, 32
 - unmarkHotStart, 29
 - writeMps, 27
- OsiDylpSolverInterface.hpp, 39
- OsiDylpSolverInterfaceUnitTest
 - OsiDylpSolverInterface, 31
- OsiDylpWarmStartBasis, 32
 - compressRows, 34
 - deleteRows, 34
 - generateDiff, 34
 - getConStatus, 34
 - getConstraintStatus, 34
 - getPhase, 34
 - mergeBasis, 34
- OsiDylpWarmStartBasis.hpp, 39
- OsiDylpWarmStartBasisDiff, 35
- POOL, 36
- parse_any, 36
- pkcoeff_struct, 36
- pkvec_struct, 36
- readMps
 - OsiDylpSolverInterface, 27
- resolve
 - OsiDylpSolverInterface, 29
- resolveOptions
 - OsiDylpSolverInterface, 32
- rowhdr_struct_tag, 36
- setBasisStatus
 - OsiDylpSolverInterface, 30
- setColName
 - OsiDylpSolverInterface, 28
- setContinuous
 - OsiDylpSolverInterface, 28
- setInteger
 - OsiDylpSolverInterface, 28
- setLanguage
 - OsiDylpSolverInterface, 29
- setObjSense
 - OsiDylpSolverInterface, 28
- setObjective
 - OsiDylpSolverInterface, 28
- setRowName
 - OsiDylpSolverInterface, 27
- setWarmStart
 - OsiDylpSolverInterface, 28
- solveFromHotStart
 - OsiDylpSolverInterface, 29
- tolerances
 - OsiDylpSolverInterface, 32
- unmarkHotStart
 - OsiDylpSolverInterface, 29
- writeMps
 - OsiDylpSolverInterface, 27