

DyLP  
trunk

Generated by Doxygen 1.8.1.2

Mon Mar 16 2015 20:12:45

## Contents

<b>1</b>	<b>Class Index</b>	<b>1</b>
1.1	Class Hierarchy . . . . .	1
<b>2</b>	<b>Class Index</b>	<b>6</b>
2.1	Class List . . . . .	6
<b>3</b>	<b>File Index</b>	<b>7</b>
3.1	File List . . . . .	7
<b>4</b>	<b>Class Documentation</b>	<b>8</b>
4.1	attvhdr_struct_tag Struct Reference . . . . .	8
4.1.1	Detailed Description . . . . .	9
4.2	basis_struct Struct Reference . . . . .	9
4.2.1	Detailed Description . . . . .	9
4.3	basisel_struct Struct Reference . . . . .	9
4.3.1	Detailed Description . . . . .	9
4.4	bnfdef_any Union Reference . . . . .	10
4.4.1	Detailed Description . . . . .	10
4.5	bnfdef_struct Struct Reference . . . . .	10
4.5.1	Detailed Description . . . . .	10
4.6	bnfGdef_struct Struct Reference . . . . .	11
4.6.1	Detailed Description . . . . .	11
4.7	bnfIdef_struct Struct Reference . . . . .	11
4.7.1	Detailed Description . . . . .	11
4.8	bnfLBdef_struct Struct Reference . . . . .	12
4.8.1	Detailed Description . . . . .	12
4.9	bnfLdef_struct Struct Reference . . . . .	12
4.9.1	Detailed Description . . . . .	12
4.10	bnfNPdef_struct Struct Reference . . . . .	13
4.10.1	Detailed Description . . . . .	13
4.11	bnfref_any Union Reference . . . . .	14
4.11.1	Detailed Description . . . . .	14
4.12	bnfref_struct_tag Struct Reference . . . . .	14
4.12.1	Detailed Description . . . . .	14
4.13	bnfref_type2 Struct Reference . . . . .	14
4.13.1	Detailed Description . . . . .	14
4.14	bnfref_type3 Struct Reference . . . . .	15

4.14.1 Detailed Description . . . . .	15
4.15 bnfTdef_struct Struct Reference . . . . .	15
4.15.1 Detailed Description . . . . .	15
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 . . . . .	18
4.19.1 Detailed Description . . . . .	18
4.20 consys_struct Struct Reference . . . . .	19
4.20.1 Detailed Description . . . . .	19
4.21 ENV Struct Reference . . . . .	20
4.21.1 Detailed Description . . . . .	20
4.22 hel_tag Struct Reference . . . . .	20
4.22.1 Detailed Description . . . . .	20
4.23 INV Struct Reference . . . . .	21
4.23.1 Detailed Description . . . . .	21
4.24 keytab_entry_internal Struct Reference . . . . .	21
4.24.1 Detailed Description . . . . .	21
4.25 lex_struct Struct Reference . . . . .	21
4.25.1 Detailed Description . . . . .	21
4.26 Ink_struct_tag Struct Reference . . . . .	22
4.26.1 Detailed Description . . . . .	22
4.27 lpopts_struct Struct Reference . . . . .	22
4.27.1 Detailed Description . . . . .	22
4.28 lpprob_struct Struct Reference . . . . .	23
4.28.1 Detailed Description . . . . .	23
4.29 lpstats_struct Struct Reference . . . . .	23
4.29.1 Detailed Description . . . . .	23
4.30 lptols_struct Struct Reference . . . . .	23
4.30.1 Detailed Description . . . . .	24
4.31 LUF Struct Reference . . . . .	24
4.31.1 Detailed Description . . . . .	24
4.32 LUF_WA Struct Reference . . . . .	24
4.32.1 Detailed Description . . . . .	24

4.33	MEM Struct Reference	24
4.33.1	Detailed Description	24
4.34	OsiDyIpSolverInterface Class Reference	25
4.34.1	Detailed Description	31
4.34.2	Member Function Documentation	31
4.34.3	Friends And Related Function Documentation	34
4.34.4	Member Data Documentation	34
4.35	OsiDyIpWarmStartBasis Class Reference	35
4.35.1	Detailed Description	37
4.35.2	Member Function Documentation	37
4.36	OsiDyIpWarmStartBasisDiff Class Reference	39
4.36.1	Detailed Description	40
4.37	parse_any Union Reference	40
4.37.1	Detailed Description	40
4.38	pkcoeff_struct Struct Reference	40
4.38.1	Detailed Description	40
4.39	pkvec_struct Struct Reference	41
4.39.1	Detailed Description	41
4.40	POOL Struct Reference	41
4.40.1	Detailed Description	41
4.41	rowhdr_struct_tag Struct Reference	42
4.41.1	Detailed Description	42
<b>5</b>	<b>File Documentation</b>	<b>42</b>
5.1	OsiDyIpSolverInterface.hpp File Reference	42
5.1.1	Detailed Description	43
5.2	OsiDyIpWarmStartBasis.hpp File Reference	43
5.2.1	Detailed Description	44

## 1 Class Index

### 1.1 Class Hierarchy

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

```

_EKKfactinfo[external]
doubleton_action::action[external]
forcing_constraint_action::action[external]
remove_fixed_action::action[external]
tripleton_action::action[external]

```

<b>attvhdr_struct_tag</b>	<b>8</b>
std::basic_fstream< char >	
std::basic_fstream< wchar_t >	
std::basic_ifstream< char >	
std::basic_ifstream< wchar_t >	
std::basic_ios< char >	
std::basic_ios< wchar_t >	
std::basic_iostream< char >	
std::basic_iostream< wchar_t >	
std::basic_istream< char >	
std::basic_istream< wchar_t >	
std::basic_istreamstream< char >	
std::basic_istreamstream< wchar_t >	
std::basic_ofstream< char >	
std::basic_ofstream< wchar_t >	
std::basic_ostream< char >	
std::basic_ostream< wchar_t >	
std::basic_ostreamstream< char >	
std::basic_ostreamstream< wchar_t >	
std::basic_string< char >	
std::basic_string< wchar_t >	
std::basic_stringstream< char >	
std::basic_stringstream< wchar_t >	
<b>basis_struct</b>	<b>9</b>
<b>basisel_struct</b>	<b>9</b>
BitVector128 [external]	
<b>bnfdef_any</b>	<b>10</b>
<b>bnfdef_struct</b>	<b>10</b>
<b>bnfGdef_struct</b>	<b>11</b>
<b>bnfldef_struct</b>	<b>11</b>
<b>bnfLBdef_struct</b>	<b>12</b>
<b>bnfLdef_struct</b>	<b>12</b>
<b>bnfNPdef_struct</b>	<b>13</b>
<b>bnfref_any</b>	<b>14</b>
<b>bnfref_struct_tag</b>	<b>14</b>
<b>bnfref_type2</b>	<b>14</b>
<b>bnfref_type3</b>	<b>15</b>
<b>bnfTdef_struct</b>	<b>15</b>
<b>coeff_struct_tag</b>	<b>16</b>
CoinAbsFltEq [external]	
CoinArrayWithLength [external]	
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]
  CoinOsIFactorization [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]
    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]

    OsiDyLPWarmStartBasis
    CoinWarmStartDual [external]
    CoinWarmStartPrimalDual [external]
    CoinWarmStartVector< T > [external]
    CoinWarmStartVectorPair< T, U > [external]
CoinWarmStartDiff [external]

```

CoinWarmStartBasisDiff [external]	
<b>OsiDylpWarmStartBasisDiff</b>	<b>39</b>
CoinWarmStartDualDiff [external]	
CoinWarmStartPrimalDualDiff [external]	
CoinWarmStartVectorDiff< T > [external]	
CoinWarmStartVectorPairDiff< T, U > [external]	
CoinYacc [external]	
<b>colhdr_struct_tag</b>	<b>17</b>
<b>conbnd_struct</b>	<b>17</b>
<b>conmtx_struct</b>	<b>18</b>
<b>consys_struct</b>	<b>19</b>
dropped_zero [external]	
EKKHlink [external]	
<b>ENV</b>	<b>20</b>
FactorPointers [external]	
<b>hel_tag</b>	<b>20</b>
<b>INV</b>	<b>21</b>
<b>keytab_entry_internal</b>	<b>21</b>
<b>lex_struct</b>	<b>21</b>
<b>lnk_struct_tag</b>	<b>22</b>
<b>lpopts_struct</b>	<b>22</b>
<b>lpprob_struct</b>	<b>23</b>
<b>lpstats_struct</b>	<b>23</b>
<b>lptols_struct</b>	<b>23</b>
<b>LUF</b>	<b>24</b>
<b>LUF_WA</b>	<b>24</b>
<b>MEM</b>	<b>24</b>
<b>OsiDylpSolverInterface</b>	<b>25</b>
<b>parse_any</b>	<b>40</b>
<b>pkcoeff_struct</b>	<b>40</b>
<b>pkvec_struct</b>	<b>41</b>
<b>POOL</b>	<b>41</b>
presolvehlink [external]	
ReferencedObject [external]	



<b>rowhdr_struct_tag</b>	<b>42</b>
SmartPtr< T >[external]	
symrec[external]	

## 2 Class Index

### 2.1 Class List

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

<b>attvhdr_struct_tag</b>	<b>8</b>
<b>basis_struct</b>	<b>9</b>
<b>basisel_struct</b>	<b>9</b>
<b>bnfdef_any</b>	<b>10</b>
<b>bnfdef_struct</b>	<b>10</b>
<b>bnfGdef_struct</b>	<b>11</b>
<b>bnfIdef_struct</b>	<b>11</b>
<b>bnfLBdef_struct</b>	<b>12</b>
<b>bnfLdef_struct</b>	<b>12</b>
<b>bnfNPdef_struct</b>	<b>13</b>
<b>bnfref_any</b>	<b>14</b>
<b>bnfref_struct_tag</b>	<b>14</b>
<b>bnfref_type2</b>	<b>14</b>
<b>bnfref_type3</b>	<b>15</b>
<b>bnfTdef_struct</b>	<b>15</b>
<b>coeff_struct_tag</b>	<b>16</b>
<b>colhdr_struct_tag</b>	<b>17</b>
<b>conbnd_struct</b>	<b>17</b>
<b>conmtx_struct</b>	<b>18</b>
<b>consys_struct</b>	<b>19</b>
<b>ENV</b>	<b>20</b>
<b>hel_tag</b>	<b>20</b>
<b>INV</b>	<b>21</b>

<a href="#">keytab_entry_internal</a>	21
<a href="#">lex_struct</a>	21
<a href="#">lnk_struct_tag</a>	22
<a href="#">lpopts_struct</a>	22
<a href="#">lpprob_struct</a>	23
<a href="#">lpstats_struct</a>	23
<a href="#">lptols_struct</a>	23
<a href="#">LUF</a>	24
<a href="#">LUF_WA</a>	24
<a href="#">MEM</a>	24
<a href="#">OsiDylpSolverInterface</a> COIN OSI API for dylp	25
<a href="#">OsiDylpWarmStartBasis</a> The dylp warm start class	35
<a href="#">OsiDylpWarmStartBasisDiff</a> A 'diff' between two <a href="#">OsiDylpWarmStartBasis</a> objects	39
<a href="#">parse_any</a>	40
<a href="#">pkcoeff_struct</a>	40
<a href="#">pkvec_struct</a>	41
<a href="#">POOL</a>	41
<a href="#">rowhdr_struct_tag</a>	42

## 3 File Index

### 3.1 File List

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

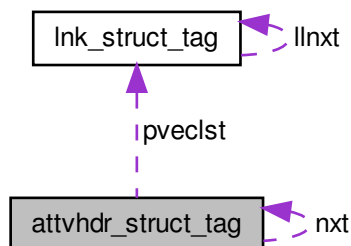
<a href="#">config_default.h</a>	??
<a href="#">config_dylp_default.h</a>	??
<a href="#">dy_cmdint.h</a>	??
<a href="#">dy_consys.h</a>	??
<a href="#">dy_vector.h</a>	??
<a href="#">dylib_bnfrdr.h</a>	??

<code>dylib_errs.h</code>	??
<code>dylib_fortran.h</code>	??
<code>dylib_hash.h</code>	??
<code>dylib_io.h</code>	??
<code>dylib_keytab.h</code>	??
<code>dylib_std.h</code>	??
<code>dylib_strtns.h</code>	??
<code>dylp.h</code>	??
<code>DylpConfig.h</code>	??
<code>glpinv.h</code>	??
<code>glplib.h</code>	??
<code>glpluf.h</code>	??
<code>OsiDylpMessages.hpp</code>	??
<a href="#">OsiDylpSolverInterface.hpp</a>	
Declarations of the COIN OSI API for the dylp solver	42
<a href="#">OsiDylpWarmStartBasis.hpp</a>	
Copyright (C) 2003 – 2007 Lou Hafer, International Business Machines Corporation and others	43

## 4 Class Documentation

### 4.1 `attvhdr_struct_tag` Struct Reference

Collaboration diagram for `attvhdr_struct_tag`:



#### 4.1.1 Detailed Description

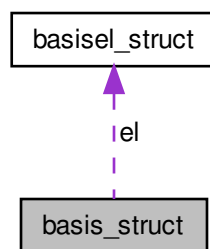
Definition at line 267 of file dy\_consys.h.

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

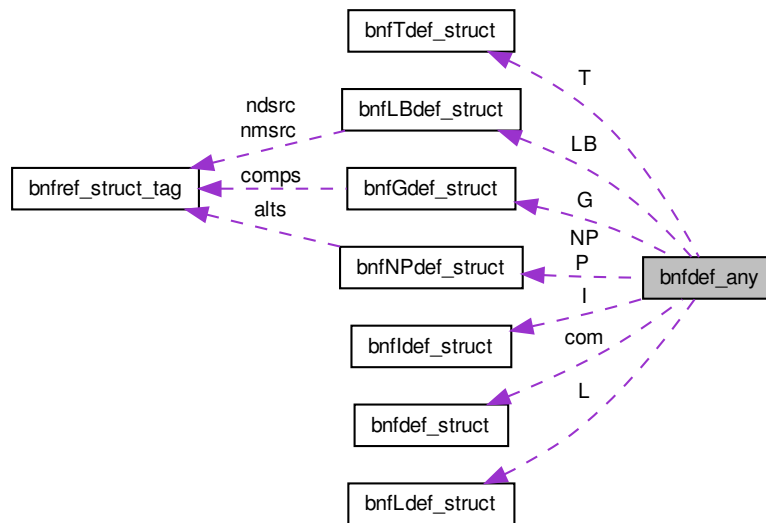
Definition at line 451 of file dylp.h.

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

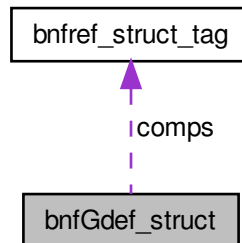
Definition at line 266 of file `dylib_bnfrdr.h`.

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

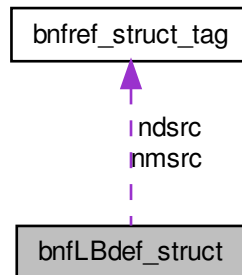
Definition at line 355 of file `dylib_bnfrdr.h`.

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_bnfrdr.h`.

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

- `dylib_bnfrdr.h`

## 4.9 bnfLdef\_struct Struct Reference

### 4.9.1 Detailed Description

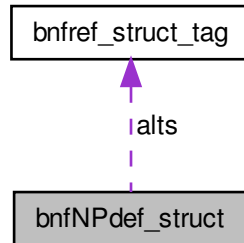
Definition at line 371 of file `dylib_bnfrdr.h`.

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

- `dylib_bnfrdr.h`

## 4.10 bnfNPdef\_struct Struct Reference

Collaboration diagram for bnfNPdef\_struct:



### 4.10.1 Detailed Description

Definition at line 301 of file `dylib_bnfrdr.h`.

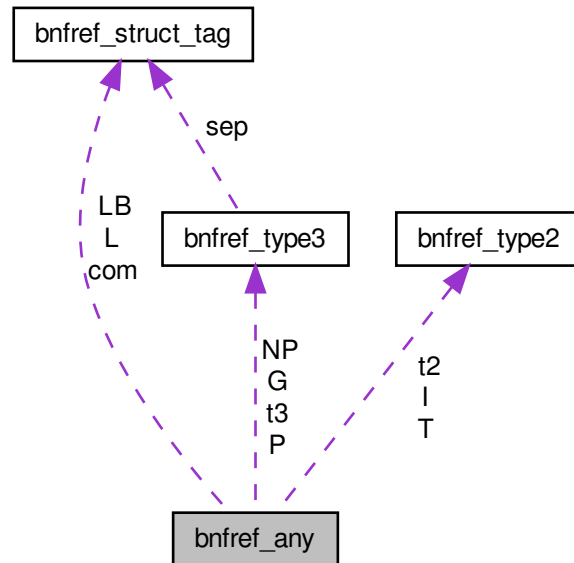
The documentation for this struct was generated from the following file:

- `dylib_bnfrdr.h`



## 4.11 bnref\_any Union Reference

Collaboration diagram for bnref\_any:



### 4.11.1 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`

## 4.12 bnref\_struct\_tag Struct Reference

### 4.12.1 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`

## 4.13 bnref\_type2 Struct Reference

### 4.13.1 Detailed Description

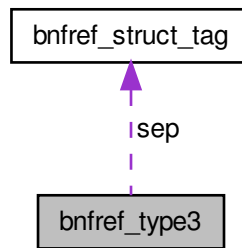
Definition at line 487 of file `dylib_bnfrdr.h`.

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

- `dylib_bnfdr.h`

#### 4.14 bnfref\_type3 Struct Reference

Collaboration diagram for `bnfref_type3`:



##### 4.14.1 Detailed Description

Definition at line 508 of file `dylib_bnfdr.h`.

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

- `dylib_bnfdr.h`

#### 4.15 bnfTdef\_struct Struct Reference

##### 4.15.1 Detailed Description

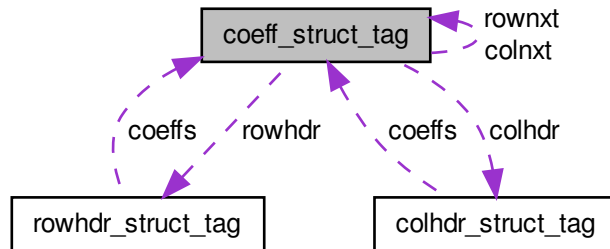
Definition at line 337 of file `dylib_bnfdr.h`.

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

- `dylib_bnfdr.h`

## 4.16 `coeff_struct_tag` Struct Reference

Collaboration diagram for `coeff_struct_tag`:



### 4.16.1 Detailed Description

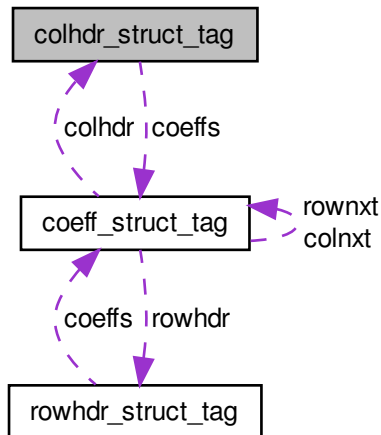
Definition at line 102 of file `dy_consys.h`.

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

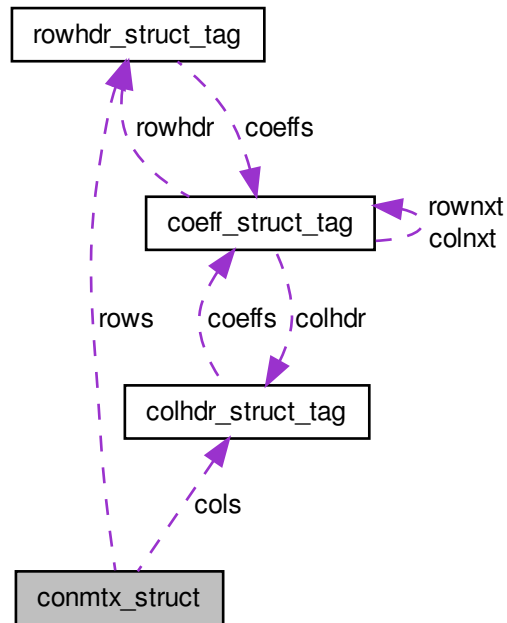
Definition at line 308 of file `dy_consys.h`.

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

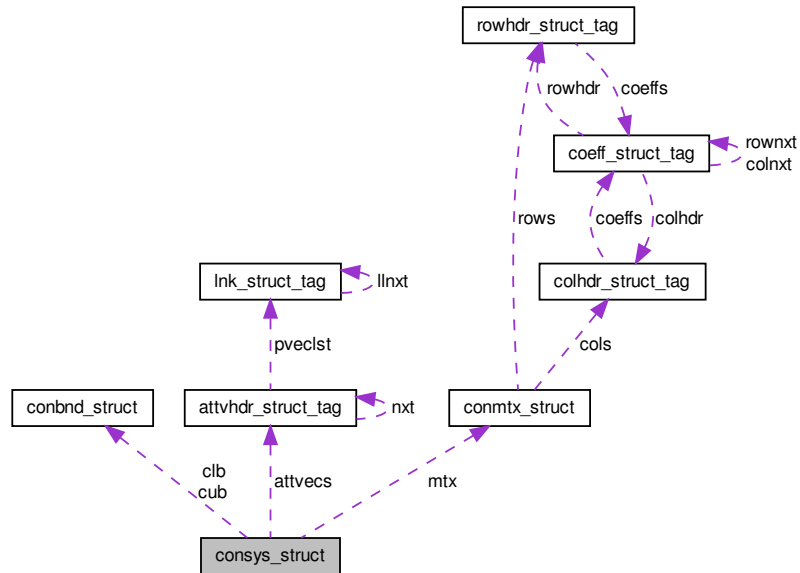
Definition at line 153 of file `dy_consys.h`.

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

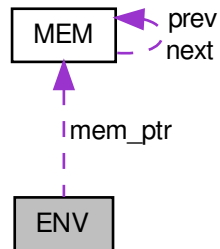
Definition at line 460 of file `dy_consys.h`.

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 glpinv.h.

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

- glpinv.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.

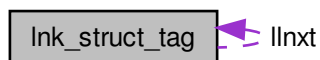
The documentation for this struct was generated from the following file:

- 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

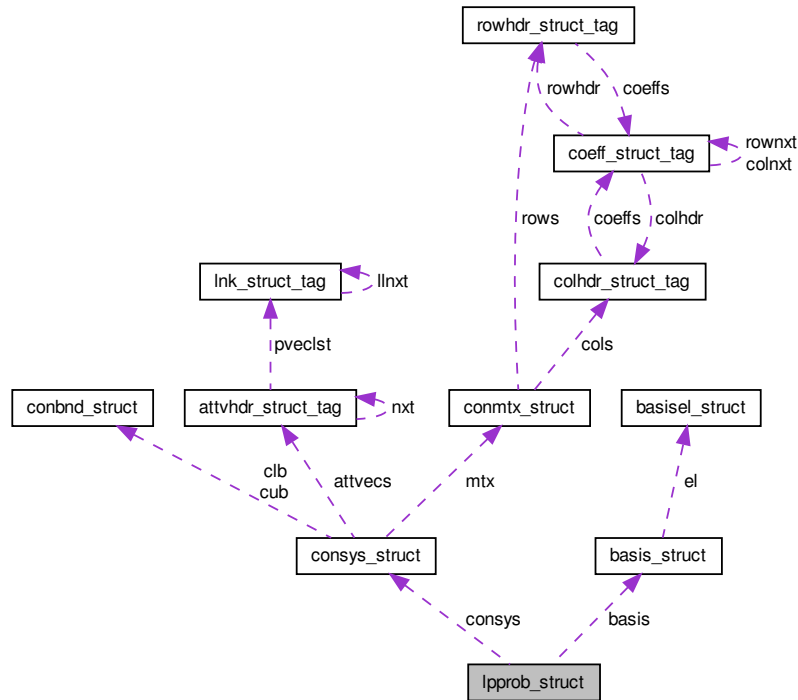
Definition at line 1114 of file dylp.h.

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:

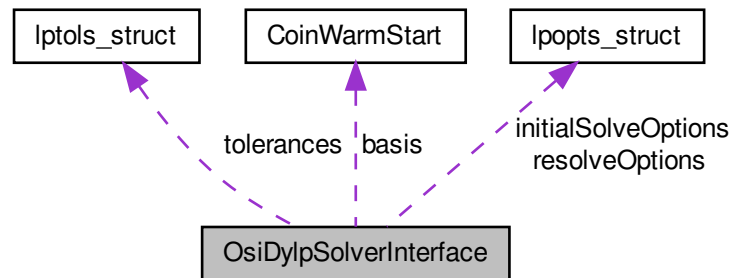
- glplib.h

## 4.34 OsiDyIpSolverInterface Class Reference

COIN OSI API for dylp.

```
#include <OsiDyIpSolverInterface.hpp>
```

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 \*rowlen, 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 OsiDyIp 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 [getDbParam](#) (OsiDbParam 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 OsiDyIp messages.*
- void [setLanguage](#) (CoinMessages::Language language)
- *An alias for [OsiDyIpSolverInterface::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 [getBlvACol](#) (int col, double \*abarj) const  
*Get a column of the tableau.*
- virtual void [getBlvRow](#) (int row, double \*betai) const  
*Get a row of the basis inverse.*
- virtual void [getBlvARow](#) (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.*

### DyIps-specific methods

- void [dyIps\\_controlfile](#) (const char \*name, const bool silent, const bool mustexist=true)  
*Process an options (.spc) file.*
- void [dyIps\\_logfile](#) (const char \*name, bool echo=false)  
*Establish a log file.*
- void [dyIps\\_outfile](#) (const char \*name)  
*Establish an output (solution and/or statistics) file.*
- void [dyIps\\_printsoln](#) (bool wantSoln, bool wantStats)  
*Print the solution and/or statistics to the output file.*
- void [setOsiDyIpsMessages](#) (**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 [OsiDyIpsolverInterfaceUnitTest](#) (const std::string &mpsDir, const std::string &netLibDir)  
*Unit test for [OsiDyIpsolverInterface](#).*

### DyIps data structures

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

- [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 [OsiDyIpSolverInterface](#) (ODSI) implements the public functions defined for the COIN OsiSolverInterface (OSI) API.

[OsiDyIpSolverInterface](#) 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 [OsiDyIpWarmStartBasis](#) 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, OsiDyIp 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 OsiDyIpSolverInterface.cpp, which is not normally scanned when generating COIN OSI API documentation.

Definition at line 107 of file OsiDyIpSolverInterface.hpp.

## 4.34.2 Member Function Documentation

4.34.2.1 `int OsiDyIpSolverInterface::readMps ( const char * filename, const char * extension = "mps" )`

Read a problem description in MPS format from a file.

4.34.2.2 `void OsiDyIpSolverInterface::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.

4.34.2.3 `int OsiDyIpSolverInterface::getNumIntegers ( ) const`

Get the number of integer variables.

Counts both binary and general integer variables.

4.34.2.4 `double OsiDyLpSolverInterface::getObjSense ( ) const`

Get the objective function sense (min/max)

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

4.34.2.5 `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.

4.34.2.6 `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.

4.34.2.7 `void OsiDyLpSolverInterface::setContinuous ( int index )`

Set a single variable to be continuous.

4.34.2.8 `void OsiDyLpSolverInterface::setInteger ( int index )`

Set a single variable to be integer.

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

Set the objective coefficients for all columns.

4.34.2.10 `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.)

4.34.2.11 `CoinWarmStart* OsiDyLpSolverInterface::getWarmStart ( ) const`

Build a warm start object for the current lp solution.

4.34.2.12 `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.

4.34.2.13 `void OsiDyLpSolverInterface::resolve ( )`

Call dylp to reoptimize (warm start).

4.34.2.14 `void OsiDyLpSolverInterface::markHotStart ( )`

Create a hot start snapshot.

4.34.2.15 `void OsiDyLpSolverInterface::solveFromHotStart ( )`

Call dylp to reoptimize (hot start).

4.34.2.16 void OsiDyIpSolverInterface::unmarkHotStart ( )

Delete the hot start snapshot.

4.34.2.17 bool OsiDyIpSolverInterface::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.

4.34.2.18 bool OsiDyIpSolverInterface::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.

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

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.

4.34.2.21 int OsiDyIpSolverInterface::canDoSimplexInterface ( ) const

Return the simplex implementation level.

4.34.2.22 void OsiDyIpSolverInterface::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.

4.34.2.23 void OsiDyIpSolverInterface::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.

4.34.2.24 bool OsiDyIpSolverInterface::basisIsAvailable ( ) const

Check if an optimal basis is available.

4.34.2.25 void OsiDyIpSolverInterface::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.

#### 4.34.2.26 int OsiDyIpSolverInterface::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.

#### 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](#))

#### 4.34.2.28 void OsiDyIpSolverInterface::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.

#### 4.34.2.29 void OsiDyIpSolverInterface::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.

#### 4.34.2.30 void OsiDyIpSolverInterface::dyIp\_printsoln ( bool *wantSoln*, bool *wantStats* )

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

#### 4.34.2.31 void OsiDyIpSolverInterface::branchAndBound ( )

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

### 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:

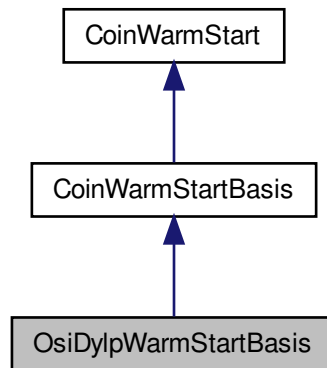
- [OsiDyIpSolverInterface.hpp](#)

### 4.35 OsiDyIpWarmStartBasis Class Reference

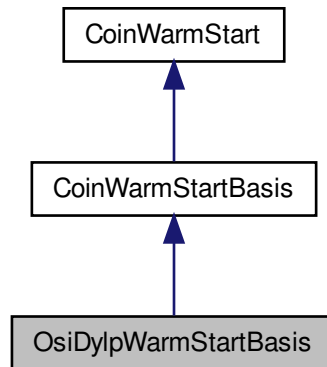
The dylp warm start class.

```
#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 *atLower-Bound*, 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 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 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:

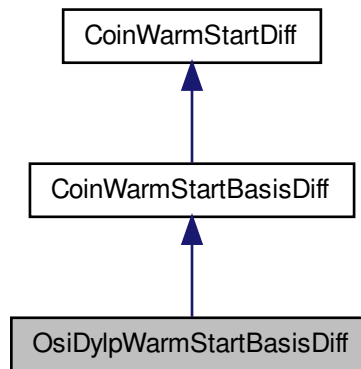
- [OsiDyIpWarmStartBasis.hpp](#)

### 4.36 OsiDyIpWarmStartBasisDiff Class Reference

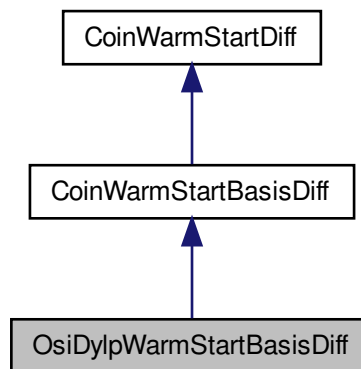
A 'diff' between two [OsiDyIpWarmStartBasis](#) objects.

```
#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_bnfdr.h`.

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

- `dylib_bnfdr.h`

## 4.38 `pkcoeff_struct` Struct Reference

#### 4.38.1 Detailed Description

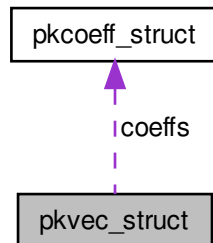
Definition at line 238 of file `dy_vector.h`.

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

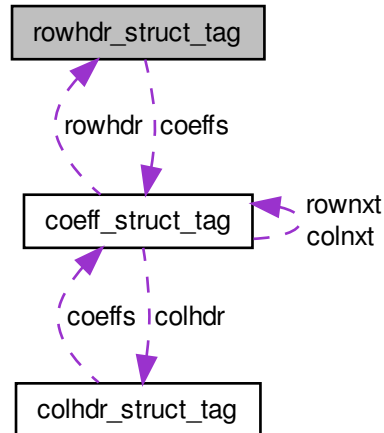
Definition at line 130 of file gplib.h.

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`

## 5 File Documentation

### 5.1 OsiDyIpsolverInterface.hpp File Reference

Declarations of the COIN OSI API for the dylp solver.

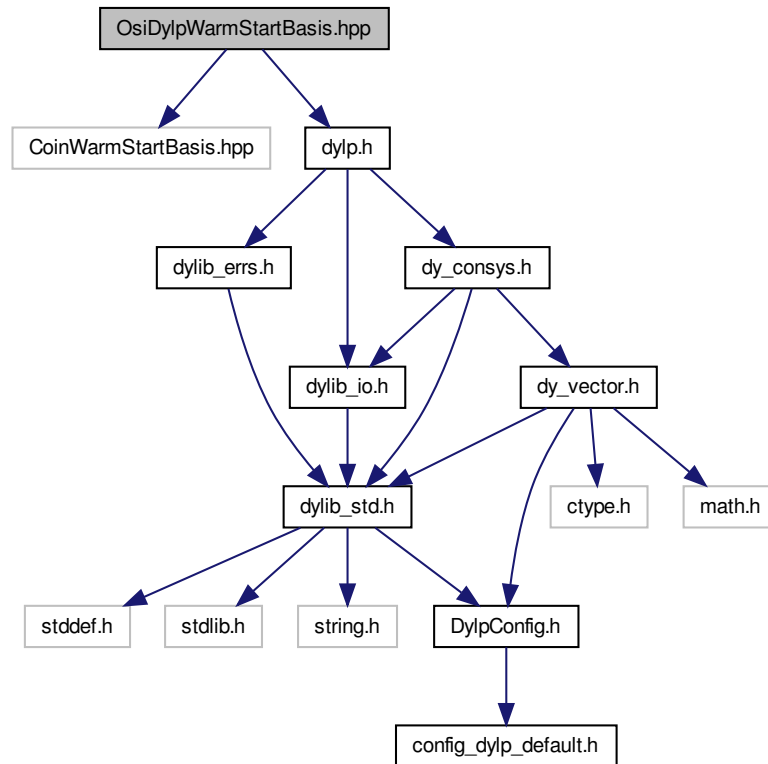
```

#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 OsiDyLpWarmStartBasis.hpp:



### Classes

- class [OsiDyLpWarmStartBasis](#)  
The *dylp* warm start class.
- class [OsiDyLpWarmStartBasisDiff](#)  
A 'diff' between two [OsiDyLpWarmStartBasis](#) 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.

Definition in file [OsiDyLpWarmStartBasis.hpp](#).

## Index

activateRowCutDebugger  
    OsiDyIpSolverInterface, 33  
attvhdr\_struct\_tag, 8  
  
basis\_struct, 8  
basisIsAvailable  
    OsiDyIpSolverInterface, 32  
basisel\_struct, 9  
bnfGdef\_struct, 10  
bnfIdef\_struct, 10  
bnfLBdef\_struct, 11  
bnfLdef\_struct, 11  
bnfNPdef\_struct, 12  
bnfTdef\_struct, 14  
bnfdef\_any, 9  
bnfdef\_struct, 9  
bnfref\_any, 13  
bnfref\_struct\_tag, 13  
bnfref\_type2, 13  
bnfref\_type3, 14  
branchAndBound  
    OsiDyIpSolverInterface, 33  
  
canDoSimplexInterface  
    OsiDyIpSolverInterface, 32  
coeff\_struct\_tag, 15  
colhdr\_struct\_tag, 16  
compressRows  
    OsiDyIpWarmStartBasis, 37  
conbnd\_struct, 16  
conmtx\_struct, 17  
consys\_struct, 18  
  
deleteRows  
    OsiDyIpWarmStartBasis, 37  
disableFactorization  
    OsiDyIpSolverInterface, 32  
dylp\_printsoln  
    OsiDyIpSolverInterface, 33  
  
ENV, 19  
enableFactorization  
    OsiDyIpSolverInterface, 32  
  
generateDiff  
    OsiDyIpWarmStartBasis, 37  
getBasisStatus  
    OsiDyIpSolverInterface, 32  
getConStatus  
    OsiDyIpWarmStartBasis, 36  
getConstraintStatus  
    OsiDyIpWarmStartBasis, 36  
  
getDualRays  
    OsiDyIpSolverInterface, 32  
getNumIntegers  
    OsiDyIpSolverInterface, 30  
getObjSense  
    OsiDyIpSolverInterface, 30  
getPhase  
    OsiDyIpWarmStartBasis, 37  
getReducedGradient  
    OsiDyIpSolverInterface, 33  
getWarmStart  
    OsiDyIpSolverInterface, 31  
  
hel\_tag, 19  
  
INV, 20  
initialSolveOptions  
    OsiDyIpSolverInterface, 33  
isDualObjectiveLimitReached  
    OsiDyIpSolverInterface, 32  
isPrimalObjectiveLimitReached  
    OsiDyIpSolverInterface, 32  
  
keytab\_entry\_internal, 20  
  
LUF, 23  
LUF\_WA, 23  
lex\_struct, 20  
lnk\_struct\_tag, 21  
lpopts\_struct, 21  
lpprob\_struct, 22  
lpstats\_struct, 22  
lptols\_struct, 22  
  
MEM, 23  
markHotStart  
    OsiDyIpSolverInterface, 31  
mergeBasis  
    OsiDyIpWarmStartBasis, 37  
  
OsiDyIpSolverInterface, 24  
    activateRowCutDebugger, 33  
    basisIsAvailable, 32  
    branchAndBound, 33  
    canDoSimplexInterface, 32  
    disableFactorization, 32  
    dylp\_printsoln, 33  
    enableFactorization, 32  
    getBasisStatus, 32  
    getDualRays, 32  
    getNumIntegers, 30  
    getObjSense, 30



- getReducedGradient, 33
- getWarmStart, 31
- initialSolveOptions, 33
- isDualObjectiveLimitReached, 32
- isPrimalObjectiveLimitReached, 32
- markHotStart, 31
- OsiDyLpSolverInterfaceUnitTest, 33
- readMps, 30
- resolve, 31
- resolveOptions, 33
- setBasisStatus, 32
- setColName, 31
- setContinuous, 31
- setInteger, 31
- setLanguage, 32
- setObjSense, 31
- setObjective, 31
- setRowName, 31
- setWarmStart, 31
- solveFromHotStart, 31
- tolerances, 34
- unmarkHotStart, 31
- writeMps, 30
- OsiDyLpSolverInterface.hpp, 41
- OsiDyLpSolverInterfaceUnitTest
  - OsiDyLpSolverInterface, 33
- OsiDyLpWarmStartBasis, 34
  - compressRows, 37
  - deleteRows, 37
  - generateDiff, 37
  - getConStatus, 36
  - getConstraintStatus, 36
  - getPhase, 37
  - mergeBasis, 37
- OsiDyLpWarmStartBasis.hpp, 42
- OsiDyLpWarmStartBasisDiff, 38
- POOL, 40
- parse\_any, 39
- pkcoeff\_struct, 39
- pkvec\_struct, 40
- readMps
  - OsiDyLpSolverInterface, 30
- resolve
  - OsiDyLpSolverInterface, 31
- resolveOptions
  - OsiDyLpSolverInterface, 33
- rowhdr\_struct\_tag, 41
- setBasisStatus
  - OsiDyLpSolverInterface, 32
- setColName
  - OsiDyLpSolverInterface, 31
- setContinuous
  - OsiDyLpSolverInterface, 31
- setInteger
  - OsiDyLpSolverInterface, 31
- setLanguage
  - OsiDyLpSolverInterface, 32
- setObjSense
  - OsiDyLpSolverInterface, 31
- setObjective
  - OsiDyLpSolverInterface, 31
- setRowName
  - OsiDyLpSolverInterface, 31
- setWarmStart
  - OsiDyLpSolverInterface, 31
- solveFromHotStart
  - OsiDyLpSolverInterface, 31
- tolerances
  - OsiDyLpSolverInterface, 34
- unmarkHotStart
  - OsiDyLpSolverInterface, 31
- writeMps
  - OsiDyLpSolverInterface, 30