

Smi
trunk

Generated by Doxygen 1.8.1.2

Mon Mar 16 2015 20:13:30

Contents

1	Class Index	1
1.1	Class Hierarchy	1
2	Class Index	5
2.1	Class List	5
3	File Index	5
3.1	File List	5
4	Class Documentation	6
4.1	SmiCoreCombineAdd Class Reference	6
4.1.1	Detailed Description	7
4.2	SmiCoreCombineReplace Class Reference	7
4.2.1	Detailed Description	8
4.3	SmiCoreCombineRule Class Reference	8
4.3.1	Detailed Description	9
4.4	SmiCoreData Class Reference	9
4.4.1	Detailed Description	9
4.5	SmiDiscreteDistribution Class Reference	9
4.5.1	Detailed Description	10
4.6	SmiDiscreteEvent Class Reference	10
4.6.1	Detailed Description	11
4.7	SmiDiscreteRV Class Reference	11
4.7.1	Detailed Description	11
4.8	SmiLinearData Class Reference	12
4.8.1	Detailed Description	12
4.9	SmiMessage Class Reference	12
4.9.1	Detailed Description	13
4.10	SmiNodeData Class Reference	13
4.10.1	Detailed Description	13
4.11	SmiQuadraticData Class Reference	14
4.11.1	Detailed Description	14
4.12	SmiQuadraticDataDC Class Reference	14
4.12.1	Detailed Description	15
4.13	SmiScenarioTree< T > Class Template Reference	15
4.13.1	Detailed Description	16
4.13.2	Member Function Documentation	16

4.14 SmiScnModel Class Reference	17
4.14.1 Detailed Description	20
4.14.2 Member Function Documentation	20
4.15 SmiScnModelAddNode Class Reference	21
4.15.1 Detailed Description	21
4.16 SmiScnModelDeleteNode Class Reference	21
4.16.1 Detailed Description	21
4.17 SmiScnNode Class Reference	21
4.17.1 Detailed Description	22
4.18 SmiSmplsCardReader Class Reference	22
4.18.1 Detailed Description	22
4.19 SmiSmplsIO Class Reference	23
4.19.1 Detailed Description	23
4.20 SmiTreeNode< T > Class Template Reference	23
4.20.1 Detailed Description	24

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]
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_ostringstream< char >
std::basic_ostringstream< wchar_t >
std::basic_string< char >

```

```

std::basic_string< wchar_t >
std::basic_stringstream< char >
std::basic_stringstream< wchar_t >
BitVector128 [external]
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]

```

SmiMessage

12

```

CoinModelHash [external]
CoinModelHash2 [external]
CoinModelHashLink [external]
CoinModelInfo2 [external]
CoinModelLink [external]
CoinModelLinkedList [external]
CoinModelTriple [external]
CoinMpsCardReader [external]

```

SmiSmtpsCardReader	22
CoinMpsIO [external]	
SmiSmtpsIO	23
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]	
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]	
twoxtwo_action [external]	
useless_constraint_action [external]	
CoinPresolveMonitor [external]	
CoinRational [external]	
CoinRelFitEq [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]	
CoinWarmStartDual[external]	
CoinWarmStartPrimalDual[external]	
CoinWarmStartVector< T >[external]	
CoinWarmStartVectorPair< T, U >[external]	
CoinWarmStartDiff[external]	
CoinWarmStartBasisDiff[external]	
CoinWarmStartDualDiff[external]	
CoinWarmStartPrimalDualDiff[external]	
CoinWarmStartVectorDiff< T >[external]	
CoinWarmStartVectorPairDiff< T, U >[external]	
CoinYacc[external]	
dropped_zero[external]	
EKKHlink[external]	
FactorPointers[external]	
presolvehlink[external]	
ReferencedObject[external]	
SmartPtr< T >[external]	
SmiCoreCombineRule	8
SmiCoreCombineAdd	6
SmiCoreCombineReplace	7
SmiCoreData	9
SmiDiscreteDistribution	9
SmiDiscreteRV	11
SmiLinearData	12
SmiDiscreteEvent	10
SmiNodeData	13
SmiQuadraticData	14
SmiQuadraticDataDC	14
SmiScenarioTree< T >	15
SmiScnModel	17
SmiScnModelAddNode	21
SmiScnModelDeleteNode	21
SmiScnNode	21
SmiTreeNode< T >	23
symrec[external]	

2 Class Index

2.1 Class List

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

SmiCoreCombineAdd	6
SmiCoreCombineReplace	7
SmiCoreCombineRule	
This deals with combining Core and Stochastic data	8
SmiCoreData	9
SmiDiscreteDistribution	9
SmiDiscreteEvent	10
SmiDiscreteRV	11
SmiLinearData	12
SmiMessage	
This deals with Clp messages (as against Osi messages etc)	12
SmiNodeData	13
SmiQuadraticData	14
SmiQuadraticDataDC	14
SmiScenarioTree< T >	15
SmiScnModel	
SmiScnModel : COIN-SMI Scenario Model Class	17
SmiScnModelAddNode	21
SmiScnModelDeleteNode	21
SmiScnNode	21
SmiSmplsCardReader	22
SmiSmplsIO	23
SmiTreeNode< T >	
Scenario Tree	23

3 File Index

3.1 File List

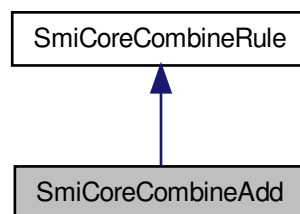
Here is a list of all documented files with brief descriptions:

<code>config_smi_default.h</code>	??
<code>SmiConfig.h</code>	??
<code>SmiCoreCombineRule.hpp</code>	??
<code>SmiDiscreteDistribution.hpp</code>	??
<code>SmiLinearData.hpp</code>	??
<code>SmiMessage.hpp</code>	??
<code>SmiQuadratic.hpp</code>	??
<code>SmiScenarioTree.hpp</code>	??
<code>SmiScnData.hpp</code>	??
<code>SmiScnModel.hpp</code>	??
<code>SmiSmplsIO.hpp</code>	??

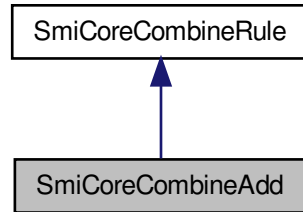
4 Class Documentation

4.1 SmiCoreCombineAdd Class Reference

Inheritance diagram for SmiCoreCombineAdd:



Collaboration diagram for SmiCoreCombineAdd:



Public Member Functions

- virtual void [Process](#) (double *d1, int o1, const **CoinPackedVector** &cpv2, char *type=0)
Process.

4.1.1 Detailed Description

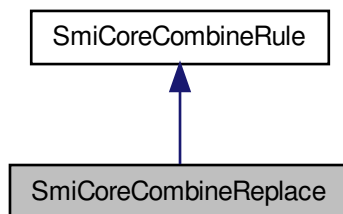
Definition at line 75 of file `SmiCoreCombineRule.hpp`.

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

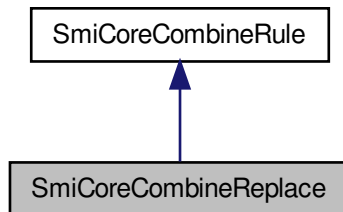
- `SmiCoreCombineRule.hpp`

4.2 SmiCoreCombineReplace Class Reference

Inheritance diagram for SmiCoreCombineReplace:



Collaboration diagram for SmiCoreCombineReplace:



Public Member Functions

- virtual void [Process](#) (double *d1, int o1, const **CoinPackedVector** &cpv2, char *type=0)
Process.

4.2.1 Detailed Description

Definition at line 54 of file SmiCoreCombineRule.hpp.

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

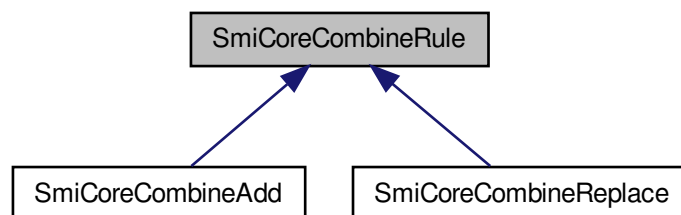
- SmiCoreCombineRule.hpp

4.3 SmiCoreCombineRule Class Reference

This deals with combining Core and Stochastic data.

```
#include <SmiCoreCombineRule.hpp>
```

Inheritance diagram for SmiCoreCombineRule:



Public Member Functions

Virtual Functions: Process and Diff

- virtual void [Process](#) (double *d1, int o1, const **CoinPackedVector** &cpv2, char *type=0)=0
Process.
- virtual void **Process** (double *d1, int o1, const int len, const int *inds, const double *dels, char *type=0)=0
- virtual **CoinPackedVector** * **Process** (**CoinPackedVector** *cpv1, **CoinPackedVector** *cpv2, char *type=0)=0
- virtual int **Process** (double *dr, const int dr_len, **CoinPackedVector** *cpv, double *dels, int *indx)=0
- virtual int **Process** (double *dr, const int dr_len, const int cpv_nels, const int *cpv_ind, const double *cpv_els, double *dels, int *indx)=0
- virtual ~**SmiCoreCombineRule** ()

4.3.1 Detailed Description

This deals with combining Core and Stochastic data.

In the Stochastic MPS standard, stochastic data updates the underlying core lp data. To specify a new scenario, one only has to identify those data that are different. So, in a sense, the stochastic data is really a "diff" between the scenario and the core data. This class specifies how to perform the "undiff", that is, how to combine core and stochastic data.

And of course, a complete implementation specifies the "diff" part as well. Now during a fit of original confusion in the birth of the SMPS standard, we decided to make default combine rule "replace", which has a rather special "diff", but we've learned to live with it.

There only needs to be one of these classes. so they're singletons.

Definition at line 36 of file SmiCoreCombineRule.hpp.

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

- SmiCoreCombineRule.hpp

4.4 SmiCoreData Class Reference

Public Member Functions

- void [addQuadraticObjectiveToCore](#) (int *starts, int *indx, double *dels)
Adds QP data after the constructor has been called.

4.4.1 Detailed Description

Definition at line 202 of file SmiScnData.hpp.

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

- SmiScnData.hpp

4.5 SmiDiscreteDistribution Class Reference

Public Member Functions

- void [addDiscreteRV](#) ([SmiDiscreteRV](#) *s)

- add discrete RV*
- [SmiDiscreteRV](#) * [getDiscreteRV](#) (int i)
- get discrete RV*
- int [getNumRV](#) ()
- get number of RV*
- [SmiCoreData](#) * [getCore](#) ()
- get core model*
- void [setCombineWithCoreRule](#) ([SmiCoreCombineRule](#) *r)
- set combine rule*
- [SmiCoreCombineRule](#) * [getCombineWithCoreRule](#) ()
- get combine rule*
- [SmiDiscreteDistribution](#) ([SmiCoreData](#) *c, [SmiCoreCombineRule](#) *r=[SmiCoreCombineReplace::Instance](#)())
- constructor requires core data and combine rule*

4.5.1 Detailed Description

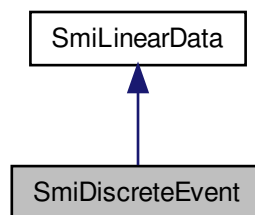
Definition at line 24 of file [SmiDiscreteDistribution.hpp](#).

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

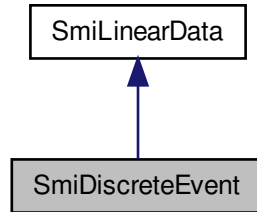
- [SmiDiscreteDistribution.hpp](#)

4.6 SmiDiscreteEvent Class Reference

Inheritance diagram for [SmiDiscreteEvent](#):



Collaboration diagram for SmiDiscreteEvent:



4.6.1 Detailed Description

Definition at line 69 of file SmiDiscreteDistribution.hpp.

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

- SmiDiscreteDistribution.hpp

4.7 SmiDiscreteRV Class Reference

4.7.1 Detailed Description

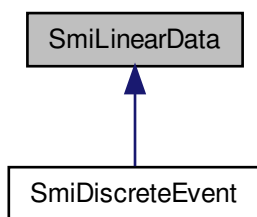
Definition at line 82 of file SmiDiscreteDistribution.hpp.

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

- SmiDiscreteDistribution.hpp

4.8 SmiLinearData Class Reference

Inheritance diagram for SmiLinearData:



4.8.1 Detailed Description

Definition at line 23 of file SmiLinearData.hpp.

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

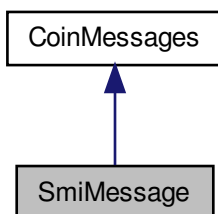
- SmiLinearData.hpp

4.9 SmiMessage Class Reference

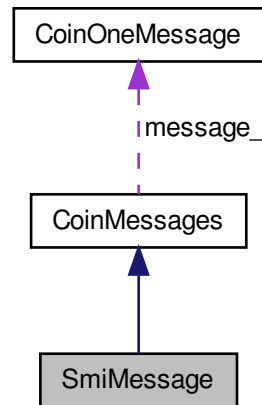
This deals with Clp messages (as against Osi messages etc)

```
#include <SmiMessage.hpp>
```

Inheritance diagram for SmiMessage:



Collaboration diagram for SmiMessage:



Public Member Functions

Constructors etc

- [SmiMessage](#) (**Language language=us_en**)
Constructor.

4.9.1 Detailed Description

This deals with Clp messages (as against Osi messages etc)

Definition at line 20 of file `SmiMessage.hpp`.

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

- `SmiMessage.hpp`

4.10 SmiNodeData Class Reference

4.10.1 Detailed Description

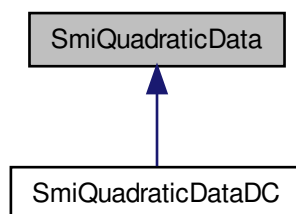
Definition at line 27 of file `SmiScnData.hpp`.

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

- `SmiScnData.hpp`

4.11 SmiQuadraticData Class Reference

Inheritance diagram for SmiQuadraticData:



4.11.1 Detailed Description

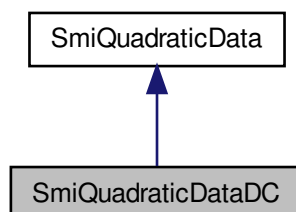
Definition at line 21 of file SmiQuadratic.hpp.

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

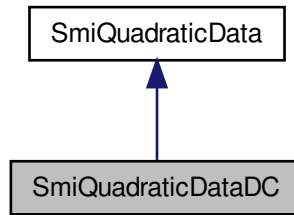
- SmiQuadratic.hpp

4.12 SmiQuadraticDataDC Class Reference

Inheritance diagram for SmiQuadraticDataDC:



Collaboration diagram for SmiQuadraticDataDC:



4.12.1 Detailed Description

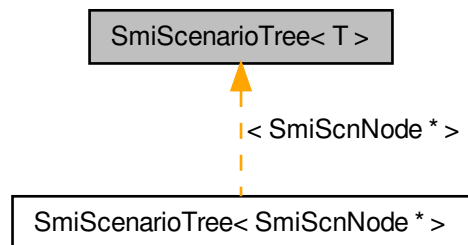
Definition at line 55 of file `SmiQuadratic.hpp`.

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

- `SmiQuadratic.hpp`

4.13 SmiScenarioTree< T > Class Template Reference

Inheritance diagram for `SmiScenarioTree< T >`:



Public Member Functions

Iterators

- `std::vector< T >::iterator` `treeBegin ()`
begin
- `std::vector< T >::iterator` `treeEnd ()`

- end*
- `std::vector< T > & wholeTree ()`
whole tree
- `std::vector< T >::iterator scenBegin (int s)`
scenario iterators TODO: native code for these iterators that does not depend on copying.
- `std::vector< T >::iterator scenEnd (int s)`

Query members

- `SmiTreeNode< T > * getRoot ()`
Get root node.
- `SmiTreeNode< T > * getLeaf (int scn)`
Get leaf node.
- `SmiTreeNode< T > * find (unsigned int scenario, int stage)`
Get node identified by scenario/stage.
- `int getNumScenarios ()`
get number of scenarios
- `SmiTreeNode< T > * find (std::vector< int > &label)`
Get node identified by longest match to array of labels.
- `SmiTreeNode< T > * find (int *label, unsigned int sz)`
Get node identified by longest match to array of labels.
- `std::vector< T > & getScenario (int scenario)`
Get vector of node data for given scenario.

Tree modification members

- `int addPathtoLeaf (int brscenario, int stage, std::vector< T > &pathdata, unsigned int start=0)`
Add new path from branching node to leaf.
- `void setChildLabels (SmiTreeNode< T > *n, std::vector< int > labels)`
Set child labels.
- `int addPathtoLeaf (std::vector< int > &labels, std::vector< T > &pathdata)`
Add new path using labels to find branching node.
- `SmiTreeNode< T > * addNodesToTree (SmiTreeNode< T > *parent, int scenario, std::vector< T > &pathdata, int start)`

Constructors, destructors and major modifying methods

- `SmiScenarioTree ()`
Default Constructor creates an empty scenario tree.
- `virtual ~SmiScenarioTree ()`
Destructor.

4.13.1 Detailed Description

`template<class T>class SmiScenarioTree< T >`

Definition at line 196 of file SmiScenarioTree.hpp.

4.13.2 Member Function Documentation

4.13.2.1 `template<class T> std::vector<T>::iterator SmiScenarioTree< T >::scenBegin (int s) [inline]`

scenario iterators TODO: native code for these iterators that does not depend on copying.

Definition at line 221 of file SmiScenarioTree.hpp.

4.13.2.2 `template<class T> SMITreeNode<T>* SMIScenarioTree< T >::getRoot () [inline]`

Get root node.

Definition at line 236 of file SMIScenarioTree.hpp.

4.13.2.3 `template<class T> SMITreeNode<T>* SMIScenarioTree< T >::getLeaf (int scn) [inline]`

Get leaf node.

Definition at line 241 of file SMIScenarioTree.hpp.

4.13.2.4 `template<class T> SMITreeNode<T>* SMIScenarioTree< T >::find (unsigned int scenario, int stage) [inline]`

Get node identified by scenario/stage.

Definition at line 246 of file SMIScenarioTree.hpp.

4.13.2.5 `template<class T> int SMIScenarioTree< T >::addPathToLeaf (int brscenario, int stage, std::vector< T > & pathdata, unsigned int start = 0) [inline]`

Add new path from branching node to leaf.

The branching node is the one belonging to "brscenario" at depth "stage". Length of incoming "pathdata" vector is leaf->depth() - stage. Responsibility for memory management of SMITreeNodeData elements is assigned to [SMIScenarioTree](#). SMITreeNodeData elements must be created with "new" operator.

Definition at line 322 of file SMIScenarioTree.hpp.

4.13.2.6 `template<class T> int SMIScenarioTree< T >::addPathToLeaf (std::vector< int > & labels, std::vector< T > & pathdata) [inline]`

Add new path using labels to find branching node.

The length of the incoming path is leaf.depth(). Responsibility for memory management of SMITreeNodeData elements is assigned to [SMIScenarioTree](#). SMITreeNodeData elements must be created with "new" operator.

Definition at line 353 of file SMIScenarioTree.hpp.

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

- SMIScenarioTree.hpp

4.14 SMI ScnModel Class Reference

[SMIScnModel](#): COIN-SMI Scenario Model Class.

```
#include <SMIScnModel.hpp>
```

Public Member Functions

Read SMPS files.

There should be three files: {name}.

[core, time, stoch]. If you have different extension conventions, then you can hack the method yourself. The files can be compressed. The object that reads the files is derived from [CoinMpsIO](#).

The optional argument [SMICoreCombineRule](#) allows user to pass in a class to override the default methods to combine core and stochastic data.

- int **readSmps** (const char *name, [SmiCoreCombineRule](#) *r=NULL)

Writes SMPS files.

This method generates three files {name}.

[core, time, stoch] or {name}.[cor, tim, sto] (see second parameter).

Parameters

name	The name for the model and the written files
winFile- Extensions	optional; false by default, so extensions will be [core, time, stoch]. With this parameter set to true, it will be [cor, tim, sto].
strictFormat	optional, true by default. Set to false if SMPS files should be written in free format.

Returns

-1 in case of no existing SMI model, otherwise 0

- int **writeSmps** (const char *name, bool winFileExtensions=false, bool strictFormat=true)
- [SmiCoreData](#) * **getCore** ()

Direct methods.

Direct methods require the user to create instances of Core data and Scenario data.

Currently, the dimension of the core nodes determines the dimension of the scenario nodes, but this is something that could easily be changed.

- void **processDiscreteDistributionIntoScenarios** ([SmiDiscreteDistribution](#) *s, bool test=false)
generate scenarios from discrete distribution
- void **setModelProb** (double p)
- int **addNodeToSubmodel** ([SmiScnNode](#) *smiScnNode)
- SmiScenarioIndex **generateScenario** ([SmiCoreData](#) *core, **CoinPackedMatrix** *matrix, **CoinPackedVector** *dclo, **CoinPackedVector** *dcup, **CoinPackedVector** *dobj, **CoinPackedVector** *drlo, **CoinPackedVector** *drup, SmiStageIndex branch, SmiScenarioIndex anc, double prob, [SmiCoreCombineRule](#) *r=SmiCoreCombineReplace::Instance())
generate scenario with ancestor/branch node identification
- SmiScenarioIndex **generateScenario** ([SmiCoreData](#) *core, **CoinPackedMatrix** *matrix, **CoinPackedVector** *dclo, **CoinPackedVector** *dcup, **CoinPackedVector** *dobj, **CoinPackedVector** *drlo, **CoinPackedVector** *drup, std::vector< int >labels, double prob, [SmiCoreCombineRule](#) *r=SmiCoreCombineReplace::Instance())
generate scenario with labels information
- SmiScenarioIndex **generateScenario** (**CoinPackedMatrix** *matrix, **CoinPackedVector** *dclo, **CoinPackedVector** *dcup, **CoinPackedVector** *dobj, **CoinPackedVector** *drlo, **CoinPackedVector** *drup, SmiStageIndex branch, SmiScenarioIndex anc, double prob, [SmiCoreCombineRule](#) *r=SmiCoreCombineReplace::Instance())
generate scenario with ancestor/branch node identification
- SmiScenarioIndex **generateScenario** (**CoinPackedMatrix** *matrix, **CoinPackedVector** *dclo, **CoinPackedVector** *dcup, **CoinPackedVector** *dobj, **CoinPackedVector** *drlo, **CoinPackedVector** *drup, std::vector< int >labels, double prob, [SmiCoreCombineRule](#) *r=SmiCoreCombineReplace::Instance())
generate scenario with labels information
- SmiScenarioIndex **generateScenarioFromCore** ([SmiCoreData](#) *core, double prob, [SmiCoreCombineRule](#) *r=SmiCoreCombineReplace::Instance())
- SmiScenarioIndex **generateScenarioFromCore** (double prob, [SmiCoreCombineRule](#) *r=SmiCoreCombineReplace::Instance())

loadOsiSolverData

Loads deterministic equivalent model into internal osi data structures and return handle.

Note: this uses a callback class [SmiCoreCombineRule](#) to decide how to combine the core and stochastic data. The user can override the default methods when the scenario is generated (see [SmiScnModel::generateScenario](#)) or when SMPS files are processed (see [SmiScnModel::readSmgs](#)).

- OsiSolverInterface * **loadOsiSolverData** ()
- OsiSolverInterface * **loadOsiSolverDataForSubproblem** (int stage, int scenStart)
- std::vector< std::pair< double, double > > **solveWS** (OsiSolverInterface *osiSolver, double objSense)
- std::pair< double, double * > **solveEV** (OsiSolverInterface *osiSolver, double objSense)
- double **solveEEV** (OsiSolverInterface *osiSolver, double objSense)
- double **getWSValue** (OsiSolverInterface *osiSolver, double objSense)
- double **getEVValue** (OsiSolverInterface *osiSolver, double objSense)
- double **getEEVValue** (OsiSolverInterface *osiSolver, double objSense)
- void **setCore** ([SmiCoreData](#) *val)
- SmiScenarioIndex **getNumScenarios** ()
- double **getScenarioProb** (SmiScenarioIndex ns)
- [SmiScnNode](#) * **getLeafNode** (SmiScenarioIndex i)
- [SmiScnNode](#) * **getRootNode** ()
- int * **getIntegerInd** ()
- int **getIntegerLen** ()
- int * **getBinaryInd** ()
- int **getBinaryLen** ()
- std::vector< int > **getIntIndices** ()
- void **addIntIndice** (int indice)
- double **getObjectiveValue** (SmiScenarioIndex ns)
- double * **getColSolution** (SmiScenarioIndex ns, int *length)
- double * **getRowSolution** (SmiScenarioIndex ns, int *length)
- double * **getRowDuals** (SmiScenarioIndex ns, int *length)
- double **getColSolution** (SmiScenarioIndex ns, int stage, int colIndex)
- double **getRowSolution** (SmiScenarioIndex ns, int stage, int rowIndex)
- double **getRowDuals** (SmiScenarioIndex ns, int stage, int rowIndex)
- double * **getColValue** (const double *d, SmiScenarioIndex ns, int *length)
- double **getColValue** (const double *d, SmiScenarioIndex ns, int stage, int rowIndex)
- double * **getRowValue** (const double *d, SmiScenarioIndex ns, int *length, bool isDual)
- double **getRowValue** (const double *d, SmiScenarioIndex ns, int stage, int rowIndex, bool isDual)
- void **setOsiSolverHandle** (OsiSolverInterface &osi)
- void **setOsiSolverHandle** (OsiSolverInterface *osi)
- OsiSolverInterface * **getOsiSolverInterface** ()
- void **releaseSolver** ()
- void **releaseCore** ()
- void **setQuadraticSolver** (ClpModel *clp)
- ClpModel * **getQuadraticSolver** ()
- ClpModel * **loadQuadraticSolverData** ()
- **SmiScnModel** ()
- **~SmiScnModel** ()
- void **addNode** ([SmiScnNode](#) *node, bool notDetEq=false)
- void **deleteNode** ([SmiScnNode](#) *tnode)
- void **addNode** ([SmiNodeData](#) *node)
- [SmiScenarioTree](#)< [SmiScnNode](#) * > * **getSmiTree** ()

4.14.1 Detailed Description

SmiScnModel: COIN-SMI Scenario Model Class.

Concrete class for generating scenario stochastic linear programs.

This class implements the Scenarios format of the Stochastic MPS modeling system (TODO: web pointer). Core data and Scenarios data can be passed using COIN/OSI structures, or can be read from SMPS formatted files.

Typical driver fragment looks like this

```
SmiScnModel smi;
smi.readSmps("app0110R");
smi.setOsiSolverHandle(new OsiClpSolverInterface());
OsiSolverInterface *osiStoch = smi.loadOsiSolverData();
osiStoch->initialSolve();
```

The setOsiSolverHandle method allows the user to pass in any OSI compatible solver.

Definition at line 49 of file SmiScnModel.hpp.

4.14.2 Member Function Documentation

4.14.2.1 `SmiScenarioIndex SmiScnModel::generateScenario (SmiCoreData * core, CoinPackedMatrix * matrix, CoinPackedVector * dcl, CoinPackedVector * dcp, CoinPackedVector * dobj, CoinPackedVector * drlo, CoinPackedVector * drup, SmiStageIndex branch, SmiScenarioIndex anc, double prob, SmiCoreCombineRule * r = SmiCoreCombineReplace::Instance())`

generate scenario with ancestor/branch node identification

Core argument must be supplied. Data values combine with corresponding core values, if found, or creates them if not.

Scenario nodes need to have same dimensions as core nodes.

Data field arguments can be NULL, or empty.

branch, anc, arguments must be supplied. These identify the branching node according to the Stochastic MPS standard.

prob is unconditional probability of scenario

4.14.2.2 `SmiScenarioIndex SmiScnModel::generateScenario (SmiCoreData * core, CoinPackedMatrix * matrix, CoinPackedVector * dcl, CoinPackedVector * dcp, CoinPackedVector * dobj, CoinPackedVector * drlo, CoinPackedVector * drup, std::vector<int> labels, double prob, SmiCoreCombineRule * r = SmiCoreCombineReplace::Instance())`

generate scenario with labels information

Core argument must be supplied. Data values combine with corresponding core values, if found, or creates them if not.

Scenario nodes need to have same dimensions as core nodes.

Data field arguments can be NULL, or empty.

Labels are passed as vector<int> array. Adds new path using labels to find branching node. The depth (root to leaf) of new path is labels.size().

4.14.2.3 `SmiScenarioIndex SmiScnModel::generateScenario (CoinPackedMatrix * matrix, CoinPackedVector * dcl, CoinPackedVector * dcp, CoinPackedVector * dobj, CoinPackedVector * drlo, CoinPackedVector * drup, SmiStageIndex branch, SmiScenarioIndex anc, double prob, SmiCoreCombineRule * r = SmiCoreCombineReplace::Instance())`

generate scenario with ancestor/branch node identification

Core argument must be supplied. Data values combine with corresponding core values, if found, or creates them if not. Scenario nodes need to have same dimensions as core nodes.

Data field arguments can be NULL, or empty.

branch, anc, arguments must be supplied. These identify the branching node according to the Stochastic MPS standard. prob is unconditional probability of scenario

```
4.14.2.4 SmiScenarioIndex SmiScnModel::generateScenario ( CoinPackedMatrix * matrix, CoinPackedVector
* dclo, CoinPackedVector * dcup, CoinPackedVector * dobj, CoinPackedVector * drlo,
CoinPackedVector * drup, std::vector< int > labels, double prob, SmiCoreCombineRule * r =
SmiCoreCombineReplace::Instance() )
```

generate scenario with labels information

Core argument must be supplied. Data values combine with corresponding core values, if found, or creates them if not. Scenario nodes need to have same dimensions as core nodes.

Data field arguments can be NULL, or empty.

Labels are passed as vector<int> array. Adds new path using labels to find branching node. The depth (root to leaf) of new path is labels.size().

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

- SmiScnModel.hpp

4.15 SmiScnModelAddNode Class Reference

4.15.1 Detailed Description

Definition at line 440 of file SmiScnModel.hpp.

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

- SmiScnModel.hpp

4.16 SmiScnModelDeleteNode Class Reference

4.16.1 Detailed Description

Definition at line 455 of file SmiScnModel.hpp.

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

- SmiScnModel.hpp

4.17 SmiScnNode Class Reference

Friends

- class **SmiScnModel**

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

- ## 4.18 SmiSmpsCardReader Class Reference

```

classDiagram
    class CoinMpsCardReader
    class SmiSmpsCardReader
    SmiSmpsCardReader --|> CoinMpsCardReader

```

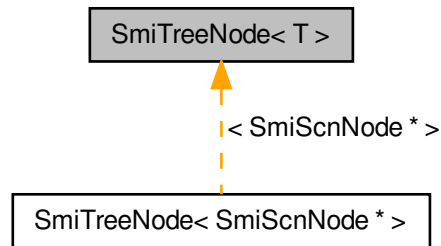
[illegible]

- **SmiSmpsCardReader** (**CoinFileInput** *input, **CoinMpsIO** *reader)
Constructor expects file to be open This one takes gzFile if fp null.

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

- SmiSmpsIO.hpp

Inheritance diagram for SmiTreeNode< T >:



Public Member Functions

Constructors, destructors and major modifying methods

- [SmiTreeNode](#) ()
Default Constructor creates an empty node.
- [SmiTreeNode](#) (T p)
Constructor from P.
- [~SmiTreeNode](#) ()
Destructor.

4.20.1 Detailed Description

`template<class T>class SmiTreeNode< T >`

Scenario Tree.

This class is used for storing and accessing scenario trees. [SmiTreeNode](#) template class.

Manages pointers to parent, child and sibling for tree navigation. Template class instance is a pointer to an object that must be created with "new" operator.

Definition at line 27 of file SmiScenarioTree.hpp.

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

- SmiScenarioTree.hpp

File Documentation

Index

- addPathtoLeaf
 - SmiScenarioTree, [16](#)
- find
 - SmiScenarioTree, [16](#)
- generateScenario
 - SmiScnModel, [19](#), [20](#)
- getLeaf
 - SmiScenarioTree, [16](#)
- getRoot
 - SmiScenarioTree, [15](#)
- scenBegin
 - SmiScenarioTree, [15](#)
- SmiCoreCombineAdd, [6](#)
- SmiCoreCombineReplace, [7](#)
- SmiCoreCombineRule, [8](#)
- SmiCoreData, [9](#)
- SmiDiscreteDistribution, [9](#)
- SmiDiscreteEvent, [10](#)
- SmiDiscreteRV, [10](#)
- SmiLinearData, [11](#)
- SmiMessage, [11](#)
- SmiNodeData, [12](#)
- SmiQuadraticData, [13](#)
- SmiQuadraticDataDC, [13](#)
- SmiScenarioTree
 - addPathtoLeaf, [16](#)
 - find, [16](#)
 - getLeaf, [16](#)
 - getRoot, [15](#)
 - scenBegin, [15](#)
- SmiScenarioTree< T >, [14](#)
- SmiScnModel, [16](#)
 - generateScenario, [19](#), [20](#)
- SmiScnModelAddNode, [20](#)
- SmiScnModelDeleteNode, [20](#)
- SmiScnNode, [20](#)
- SmiSmplsCardReader, [21](#)
- SmiSmplsIO, [22](#)
- SmiTreeNode< T >, [22](#)