IBM ILOG CPLEX Optimization Studio CPLEX Parameters Reference

Version 12 Release 8

IBM ILOG CPLEX Optimization Studio CPLEX Parameters Reference

Version 12 Release 8

Contents

Chapter 1. Parameters of CPLEX

Parameters, accessible and manageable by users, control the behavior of CPLEX.

Accessing parameters

Users access and modify parameters by means of methods in the various APIs.

Documentation about CPLEX parameters specific to the Python API is available as online help inside a Python session. A brief introduction to CPLEX parameters is available in the topic Using CPLEX parameters in the CPLEX Python API in the tutorial about Python in *Getting Stated with CPLEX*.

Documentation about CPLEX parameters specific to the CPLEX for MATLAB connector is available in the topic Using parameters in the *Getting Started with CPLEX for MATLAB* manual. This information is also available as online help inside a MATLAB session.

The following methods set and access parameters for objects of the class `IloCplex` in C++ and Java:

```
setParam
getParam
getMin
getMax
getDefault
setDefaults
```

The names of the corresponding accessors in the class `Cplex` in the .NET API follow the usual conventions of names and capitalization of languages in that framework. For example, the class `Cplex` and its method `Solve` are denoted `Cplex.Solve`. Likewise, the methods `Cplex.GetParam` and `SetParam` access and set parameters in the .NET API.

C applications and applications written in other languages callable from C access and set parameters with the following routines:

Callable Library routine to access parameters	**Purpose**
`CPXgetdblparam`	Accesses a parameter of type double
`CPXsetdblparam`	Changes a parameter of type double
`CPXinfodblparam`	Gets the default value and range of a parameter of type double
`CPXgetintparam`	Accesses a parameter of type integer
`CPXsetintparam`	Changes a parameter of type integer
`CPXinfointparam`	Gets the default value and range of a parameter of type integer
`CPXgetlongparam`	Accesses a parameter of type long
`CPXsetlongparam`	Changes a parameter of type long

Callable Library routine to access parameters	Purpose
CPXinfolongparam	Gets the default value and range of a parameter of type long
CPXgetstrparam	Accesses a parameter of type string
CPXsetstrparam	Changes a parameter of type string
CPXinfostrparam	Gets the default value of a parameter of type string
CPXsetdefaults	Resets all parameters to their standard default values
CPXgetparamname	Accesses the name of a parameter
CPXgetparamnum	Access the identifying number assigned to a parameter
CPXgetchgparams	Accesses all parameters not currently at their default value

Managing sets of parameters

Users may groups parameters into sets in the object-oriented APIs.

The object-oriented APIs of CPLEX also allow you to group parameters into a **set** and then manage that set of parameters together.

- In the C++ API, use the member functions of an instance of the class `IloCplex::ParameterSet`.
- In the Java API, use the methods of an object of the class `IloCplex.ParameterSet`.
- In the .NET API, use the methods of the class `Cplex.ParameterSet`.
- In the Python API, use the methods of the class `Cplex.ParameterSet`.
- In the CPLEX for MATLAB Toolbox API, use the function `cplexoptimset`.
- In the MATLAB Cplex class API, use the structure `Cplex.Param`.

Parameter names

Names of CPLEX parameters follow the coding conventions of each API.

In the parameter table, each parameter has a name (that is, a symbolic constant) to refer to it within an application.

- For the Callable Library (C API), these constants start with `CPXPARAM_`; for example, `CPXPARAM_Simplex_Limits_Iterations`. They are used as the second argument in all parameter routines (except `CPXXsetdefaults`, which does not require them).

  Legacy names

  For the C API, these constants are capitalized and start with `CPX_PARAM_`; for example, `CPX_PARAM_ITLIM`. They are used as the second argument in all parameter routines (except `CPXsetdefaults`, which does not require them).
- For C++ applications, the parameters are defined in classes that specify a hierarchy of applicability of the parameter; for example, `IloCplex::Param::Simplex::Limits::Iterations`.

  Legacy names

For C++ applications, the parameters are defined in nested enumeration types for Boolean, integer, floating-point, and string parameters. The `enum` names use mixed (lower and upper) case letters and must be prefixed with the class name `IloCplex::` for scope. For example, `IloCplex::ItLim` is the `IloCplex` equivalent of `CPX_PARAM_ITLIM`.

- For Java applications, the parameters are defined as final static objects in nested classes that specify a hierarchy of applicability of the parameter; for example, `IloCplex.Param.Simplex.Limits.Iterations`.

  Legacy names

  For Java applications, the parameters are defined as final static objects in nested classes called `IloCplex.BooleanParam`, `IloCplex.IntParam`, `IloCplex.LongParam`, `IloCplex.DoubleParam`, and `IloCplex.StringParam` for Boolean, integer, long, floating-point, and string parameters, respectively. The parameter object names use mixed (lower and upper) case letters and must be prefixed with the appropriate class for scope. For example, `IloCplex.IntParam.ItLim` is the object representing the parameter `CPX_PARAM_ITLIM`.
- For .NET applications, the parameters follow the usual conventions for capitalizing attributes and defining scope within a namespace.
- For Python applications, the names of parameters resemble the names in the CPLEX Interactive Optimizer, modified for the syntax of a Python application. For example, the command in the Interactive Optimizer **`set mip cuts mcfcut`** looks like this in a Python application:`cplex.parameters.mip.cuts.set(mcfcut)`.
- For MATLAB Cplex Class API applications, the names of parameters resemble the names in the CPLEX Interactive Optimizer. For example, setting the MIP variable selection strategy looks like this in a MATLAB Cplex Class API application: `cplex.Param.mip.strategy.variableselect = 3;` where 3 indicates strong branching. The parameters in CPLEX for MATLAB Toolbox applications are named similarly. For example, setting the MIP variable selection strategy looks like this in a toolbox application: `options.mip.strategy.variableselect.Cur = 3;` where `options` is a structure created with the function `cplexoptimset('cplex')`. In addition, in order maintain compatibility with the MATLAB Optimization Toolbox, a number of parameters may be set using the MATLAB Optimization Toolbox parameter names; for example, the maximum nodes can be set with: `options = cplexoptimset('MaxNodes', 400);`.

The reference page of each parameter documents an integer that serves as a reference number or unique identifier for each parameter. That integer reference number corresponds to the value that each symbolic constant represents, as found in the `cplex.h` , `cplexx.h` , and `cpxconst.h` header files of the Callable Library (C API).

Correspondence of parameters between APIs

Certain specialized parameters of one API may not have an equivalent in another API.

Some parameters available for the C API are not supported as parameters for the object-oriented APIs or have a slightly different name there. In particular:

- "epsilon (degree of tolerance) used in linearization" on page 63 (`EpLin`), the parameter specifying the tolerance to use in linearization in the object oriented APIs (C++, Java, .NET), is not applicable in the C API, nor in the Python API.
- "MIP callback switch between original model and reduced, presolved model" on page 84(`CPX_PARAM_MIPCBREDLP`), the parameter specifying whether to use the

reduced or original model in MIP callbacks, has no equivalent in the object-oriented APIs (C++, Java, .NET) nor in the Python API, nor in the MATLAB connector.

- Logging output is controlled by a parameter in the C API (`CPX_PARAM_SCRIND`), but when using the object-oriented APIs, you control logging by configuring the output channel:
  - `IloCplex::out` in C++

    For example, to turn off output to the screen, use `cplex.setOut(env.getNullStream())` .
  - `IloCplex.output` in Java

    For example, to turn off output to the screen, use `cplex.setOut(null)`.
  - `Cplex.Out` in .NET

    For example, to turn off output to the screen, use `Cplex.SetOut(Null)`.
  - `cplex.set_results_stream` in Python

    For example, to turn off output to the screen, use `cplex.set_results_stream(None)`.
  - `DisplayFunc` in the MATLAB Cplex Class API

    For example, to turn off output to the screen, use`cplex.DisplayFunc();`.
  - `display` and `Display` in the CPLEX for MATLAB Toolbox API

    For example, to turn off output to the screen, use `optimset('Display','off');` or `options.display = 'off';`.
- The parameter `IloCplex::RootAlg` in the C++ API corresponds to these parameters in the C API:
  - "algorithm for initial MIP relaxation" on page 139: `CPX_PARAM_STARTALG`
  - "algorithm for continuous linear problems" on page 136: `CPX_PARAM_LPMETHOD`
  - "algorithm for continuous quadratic optimization" on page 138: `CPX_PARAM_QPMETHOD`
- The parameter `IloCplex::NodeAlg` in the C++ API corresponds to the parameter "MIP subproblem algorithm" on page 99 `CPX_PARAM_SUBALG` in the C API.

Saving parameter settings to a file in the C API

Users of the Callable Library (C API) can save current parameter settings in a PRM file.

You can read and write a file of parameter settings with the C API. The file extension is `.prm` . The C routine `CPXreadcopyparam` reads parameter values from a file with the `.prm` extension. The routine `CPXwriteparam` writes a file of the current non-default parameter settings to a file with the `.prm` extension. Here is the format of such a file:

```
CPLEX Parameter File Version number
  parameter_name   parameter_value
```

Tip:

The heading with a version number in the first line of a PRM file is significant to CPLEX. An easy way to produce a correctly formatted PRM file with a proper heading is to have CPLEX write the file for you.

CPLEX reads the entire file before changing any of the parameter settings. After successfully reading a parameter file, the C API first sets all parameters to their default value. Then it applies the settings it read in the parameter file. No changes are made if the parameter file contains errors, such as missing or illegal values. There is no checking for duplicate entries in the file. In the case of duplicate entries, the last setting in the file is applied.

When you create a parameter file from the C API, only the non-default values are written to the file. You can double-quote string values or not when you create a PRM file, but CPLEX always writes string-valued parameters with double quotation marks.

The comment character in a parameter file is #. After that character, CPLEX ignores the rest of the line.

The C API issues a warning if the version recorded in the parameter file does not match the version of the product. A warning is also issued if a nonintegral value is given for an integer-valued parameter.

Here is an example of a correct CPLEX parameter file:

```
CPLEX Parameter File Version 11.0
CPX_PARAM_EPPER                 3.45000000000000e-06
CPX_PARAM_OBJULIM               1.23456789012345e+05
CPX_PARAM_PERIND                1
CPX_PARAM_SCRIND                1
CPX_PARAM_WORKDIR               "tmp"
```

The environment variable **ILOG_CPLEX_PARAMETER_FILE** lets you specify the existence and location of a PRM file to CPLEX.

Chapter 2. Topical list of parameters

Users can browse CPLEX parameters, organized by topics.

Simplex

Here are links to parameters of interest to users of the simplex optimizers.

Selects the “algorithm for continuous linear problems” on page 136

“advanced start switch” on page 17

“Benders strategy” on page 30

“lower objective value limit” on page 105

“upper objective value limit” on page 105

“dual simplex pricing algorithm” on page 59

dynamic row-management

“primal simplex pricing algorithm” on page 118

“simplex crash ordering” on page 50

“Markowitz tolerance” on page 64

“optimality tolerance” on page 64

“perturbation constant” on page 65

“simplex perturbation switch” on page 110

“simplex perturbation limit” on page 111

“relaxation for FeasOpt” on page 66

“feasibility tolerance” on page 67

“simplex maximum iteration limit” on page 80

“memory reduction switch” on page 83

“numerical precision emphasis” on page 102

“simplex pricing candidate list size” on page 122

“sifting subproblem algorithm” on page 143

“simplex iteration information display” on page 145

“simplex singularity repair limit” on page 146

Barrier

Here are links to parameters of interest to users of the barrier optimizer.

“advanced start switch” on page 17

“barrier algorithm” on page 22

“barrier starting point algorithm” on page 30

“barrier crossover algorithm” on page 24

“sifting subproblem algorithm” on page 143

“barrier ordering algorithm” on page 28

“barrier display information” on page 25

“barrier growth limit” on page 26

“barrier column nonzeros” on page 23

“barrier iteration limit” on page 26

“barrier maximum correction limit” on page 27

“barrier objective range” on page 28

“convergence tolerance for LP and QP problems” on page 25

“convergence tolerance for QC problems” on page 29

“memory reduction switch” on page 83

“numerical precision emphasis” on page 102

MIP

Here are topics of interest to users of the MIP optimizer.

The parameters controlling MIP behavior are accessible through the following topics:

- “MIP general” on page 9
- “MIP strategies” on page 9
- “MIP cuts” on page 10
- “MIP tolerances” on page 11
- “MIP limits” on page 11
- Distributed MIP

MIP general

Here are links to parameters of general interest to users of the MIP optimizer.

"advanced start switch" on page 17

"MIP emphasis switch" on page 86

"MIP repeat presolve switch" on page 134

"relaxed LP presolve switch" on page 132

"indefinite MIQP switch" on page 125

"optimality target" on page 106

"solution type for LP and QP" on page 152

"bound strengthening switch" on page 37

"memory reduction switch" on page 83

"numerical precision emphasis" on page 102

"MIP callback switch between original model and reduced, presolved model" on page 84

"MIP node log display information" on page 85

"MIP node log interval" on page 88

"node storage file switch" on page 100

MIP strategies

Here are links to parameters controlling MIP strategies.

"algorithm for initial MIP relaxation" on page 139

"Benders strategy" on page 30

"MIP subproblem algorithm" on page 99

"MIP variable selection strategy" on page 166

"MIP strategy best bound interval" on page 36

"MIP branching direction" on page 39

"backtracking tolerance" on page 39

"MIP dive strategy" on page 58

"MIP heuristic frequency" on page 75

"local branching heuristic" on page 81

“MIP priority order switch” on page 90

“MIP priority order generation” on page 91

“MIP node selection strategy” on page 101

“node presolve switch” on page 122

“MIP probing level” on page 123

“RINS heuristic frequency” on page 135

“feasibility pump switch” on page 72

scale parameter for subMIPs

algorithm for initial MIP relaxation of a subMIP of a MIP

algorithm for subproblems of a subMIP of a MIP

MIP cuts

Here are links to parameters controlling cuts.

These parameters set limits on cut generation.

“constraint aggregation limit for cut generation” on page 18

“cut factor row-multiplier limit” on page 52

“type of cut limit” on page 60

“number of cutting plane passes” on page 52

These parameters control specific types of cuts.

“Boolean Quadric Polytope cuts” on page 38

“MIP cliques switch” on page 42

“MIP covers switch” on page 47

“MIP disjunctive cuts switch” on page 57

“MIP flow cover cuts switch” on page 70

“MIP flow path cut switch” on page 71

“MIP Gomory fractional cuts switch” on page 73

- “pass limit for generating Gomory fractional cuts” on page 74
- “candidate limit for generating Gomory fractional cuts” on page 73

“MIP GUB cuts switch” on page 75

“MIP globally valid implied bound cuts switch” on page 76

“MIP locally valid implied bound cuts switch” on page 77

“Lift-and-project cuts switch for MIP and MIQCP” on page 81

“MCF cut switch” on page 82

“MIP MIR (mixed integer rounding) cut switch” on page 94

“Reformulation Linearization Technique (RLT) cuts” on page 135

“MIP zero-half cuts switch” on page 170

MIP tolerances

Here are links to parameters setting MIP tolerances.

“backtracking tolerance” on page 39

“lower cutoff” on page 51

“upper cutoff” on page 53

“absolute objective difference cutoff” on page 104

“relative objective difference cutoff” on page 132

“absolute MIP gap tolerance” on page 61

“relative MIP gap tolerance” on page 61

“integrality tolerance” on page 62

“relaxation for FeasOpt” on page 66

MIP limits

Here are links to parameters setting MIP limits.

“MIP integer solution limit” on page 79

“MIP integer solution-file switch and prefix” on page 78

“pass limit for generating Gomory fractional cuts” on page 74

“candidate limit for generating Gomory fractional cuts” on page 73

“constraint aggregation limit for cut generation” on page 18

“type of cut limit” on page 60

“cut factor row-multiplier limit” on page 52

“number of cutting plane passes” on page 52

“MIP node limit” on page 101

“time spent probing” on page 124

“number of attempts to repair infeasible MIP start” on page 133

“MIP strong branching candidate list limit” on page 154

“MIP strong branching iterations limit” on page 154

“limit on nodes explored when CPLEX solves a subMIP” on page 155

“tree memory limit” on page 160

“time spent in ramp up during distributed parallel optimization” on page 127

“deterministic time spent in ramp up during distributed parallel optimization” on page 127

“ramp up duration” on page 128

Solution polishing

Here are links to parameters controlling starting conditions for solution polishing

“absolute MIP gap before starting to polish a feasible solution” on page 112

“relative MIP gap before starting to polish a feasible solution” on page 113

“MIP integer solutions to find before starting to polish a feasible solution” on page 114

“nodes to process before starting to polish a feasible solution” on page 115

“time before starting to polish a feasible solution” on page 115

Solution pool

Here are links to parameters controlling the solution pool.

“solution pool intensity” on page 149

“solution pool replacement strategy” on page 151

“maximum number of solutions generated for solution pool by populate” on page 117

“maximum number of solutions kept in solution pool” on page 147

“absolute gap for solution pool” on page 146

“relative gap for solution pool” on page 148

Network

Here are links to parameters of interest to users of the network flow optimizer.

"network optimality tolerance" on page 96

"network primal feasibility tolerance" on page 96

"simplex network extraction level" on page 97

"network simplex iteration limit" on page 98

"network simplex pricing algorithm" on page 98

"network logging display switch" on page 95

Parallel optimization

Here are links to parameters controlling parallel optimization.

"parallel mode switch" on page 108

"CPU mask to bind threads to cores" on page 48

"global thread count" on page 157

"Benders strategy" on page 30

Sifting

Here are links to parameters of interest to users of the sifting optimizer.

"sifting subproblem algorithm" on page 143

"sifting information display" on page 144

"upper limit on sifting iterations" on page 145

sifting from simplex switch

Preprocessing: aggregator, presolver

Here are links to parameters related to preprocessing.

"symmetry breaking" on page 156

"preprocessing aggregator fill" on page 18

"preprocessing aggregator application limit" on page 19

"bound strengthening switch" on page 37

"coefficient reduction setting" on page 44

"dependency switch" on page 55

“presolve dual setting” on page 119

“presolve switch” on page 120

“linear reduction switch” on page 120

“limit on the number of presolve passes made” on page 121

“node presolve switch” on page 122

“relaxed LP presolve switch” on page 132

“MIP repeat presolve switch” on page 134

“primal and dual reduction type” on page 130

Tolerances

Here are links to parameters setting tolerances.

“convergence tolerance for LP and QP problems” on page 25

“convergence tolerance for QC problems” on page 29

“backtracking tolerance” on page 39

“Benders feasibility cut tolerance” on page 35

“Benders optimality cut tolerance” on page 36

“lower cutoff” on page 51

“upper cutoff” on page 53

“absolute MIP gap tolerance” on page 61

“absolute MIP gap before starting to polish a feasible solution” on page 112

“relative MIP gap tolerance” on page 61

“relative MIP gap before starting to polish a feasible solution” on page 113

“integrality tolerance” on page 62

“epsilon (degree of tolerance) used in linearization” on page 63

“Markowitz tolerance” on page 64

“optimality tolerance” on page 64

“network optimality tolerance” on page 96

“feasibility tolerance” on page 67

“relaxation for FeasOpt” on page 66

“absolute objective difference cutoff” on page 104

“relative objective difference cutoff” on page 132

“perturbation constant” on page 65

“absolute gap for solution pool” on page 146

“relative gap for solution pool” on page 148

Limits

Here are links to parameters setting general limits.

“memory available for working storage” on page 168

“global thread count” on page 157

“optimizer time limit in seconds” on page 158

“variable (column) read limit” on page 45

“constraint (row) read limit” on page 141

“nonzero element read limit” on page 103

“QP Q-matrix nonzero read limit” on page 126

Display and output

Here are links to parameters controlling screen displays, logs, and files.

“messages to screen switch” on page 142

“tuning information display” on page 162

“barrier display information” on page 25

“simplex iteration information display” on page 145

“sifting information display” on page 144

“MIP node log display information” on page 85

“MIP node log interval” on page 88

“network logging display switch” on page 95

“clock type for computation time” on page 43

“conflict information display” on page 46

“data consistency checking and modeling assistance” on page 54 including modeling assistance

“precision of numerical output in MPS and REW file formats” on page 94 “directory for working files” on page 167

“write level for MST, SOL files” on page 169

changed parameter display switch

Chapter 3. List of CPLEX parameters

CPLEX parameters, documented here alphabetically by name in the Callable Library (C API), are available in the C++, Java, .NET, and Python APIs, as well as in the Interactive Optimizer and the MathWorks MATLAB connector.

advanced start switch

If set to 1 or 2, this parameter specifies that CPLEX should use advanced starting information when it initiates optimization.

Purpose

Advanced start switch

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_Advance	CPX_PARAM_ADVIND (int)
C++	IloCplex::Param::Advance	AdvInd (int)
Java	IloCplex.Param.Advance	AdvInd (int)
.NET	Cplex.Param.Advance	AdvInd (int)
OPL		advind
Python	parameters.advance	advance
MATLAB	Cplex.Param.advance	advance
Interactive	advance	advance
Identifier	1001	1001

Description

If set to 1 or 2, this parameter specifies that CPLEX should use advanced starting information when optimization is initiated.

For MIP models, setting 1 (one) will cause CPLEX to continue with a partially explored MIP tree if one is available. If tree exploration has not yet begun, setting 1 (one) specifies that CPLEX should use a loaded MIP start, if available. Setting 2 retains the current incumbent (if there is one), re-applies presolve, and starts a new search from a new root.

Setting 2 is useful for continuous models. Consequently, it can be particularly useful for solving fixed MIP models, where a start vector but no corresponding basis is available.

For continuous models solved with simplex, setting 1 (one) will use the currently loaded basis. If a basis is available only for the original, unpresolved model, or if CPLEX has a start vector rather than a simplex basis, then the simplex algorithm will proceed on the unpresolved model. With setting 2, CPLEX will first perform presolve on the model and on the basis or start vector, and then proceed with optimization on the presolved problem.

For continuous models solved with the barrier algorithm, settings 1 or 2 will continue simplex optimization from the last available barrier iterate.

Tip:

If you optimize your MIP model, then change a tolerance (such as “upper cutoff” on page 53, “lower cutoff” on page 51, “integrality tolerance” on page 62), and then re-optimize, the change in tolerance may not be taken into account in certain circumstances, depending on characteristics of your model and parameter settings. In order for CPLEX to take into account your change in tolerance, you must restart the second optimization from the beginning. That is, you must set CPX_PARAM_ADVIND, AdvInd to 0 (zero).

Table 1. Values.

Value	Meaning
0	Do not use advanced start information
1	Use an advanced basis supplied by the user; **default**
2	Crush an advanced basis or starting vector supplied by the user

constraint aggregation limit for cut generation

Limits the number of constraints that can be aggregated for generating flow cover and mixed integer rounding (MIR) cuts.

Purpose

Constraint aggregation limit for cut generation

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_MIP_Limits_AggForCut	CPX_PARAM_AGGCUTLIM
C++	IloCplex::Param::MIP::Limits::AggForCut	AggCutLim (int)
Java	IloCplex.Param.MIP.Limits.AggForCut	AggCutLim (int)
.NET	Cplex.Param.MIP.Limits.AggForCut	AggCutLim (int)
OPL		aggcutlim
Python	parameters.mip.limits.aggforcut	mip.limits.aggforcut
MATLAB	Cplex.Param.mip.limits.aggforcut	mip.limits.aggforcut
Interactive	mip limits aggforcut	mip limits aggforcut
Identifier	2054	2054

Description

Limits the number of constraints that can be aggregated for generating flow cover and mixed integer rounding (MIR) cuts.

Values

Any nonnegative integer; **default**: 3

preprocessing aggregator fill

Limits variable substitutions by the aggregator.

Purpose

Preprocessing aggregator fill

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_Preprocessing_Fill	CPX_PARAM_AGGFILL
C++	IloCplex::Param::Preprocessing::Fill	AggFill (CPXLONG)
Java	IloCplex.Param.Preprocessing.Fill	AggFill (CPXLONG)
.NET	Cplex.Param.Preprocessing.Fill	AggFill (CPXLONG)
OPL		aggfill
Python	parameters.preprocessing.fill	preprocessing.fill
MATLAB	Cplex.Param.preprocessing.fill	preprocessing.fill
Interactive	preprocessing fill	preprocessing fill
Identifier	1002	1002

Description

Limits number of variable substitutions by the aggregator. If the net result of a single substitution is more nonzeros than this value, the substitution is not made.

Tip:

The symbols `CPXINT` and `CPXLONG` declare a type of integer appropriate to your specification of a relatively small or large model by means of the symbol `CPX_APIMODEL`.

Values

Any nonnegative integer; **default**: 10

preprocessing aggregator application limit

Invokes the aggregator to use substitution where possible to reduce the number of rows and columns before the problem is solved.

Purpose

Preprocessing aggregator application limit

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_Preprocessing_Aggregator	CPX_PARAM_AGGIND (int)
C++	IloCplex::Param::Preprocessing::Aggregator	AggInd (int)
Java	IloCplex.Param.Preprocessing.Aggregator	AggInd (int)
.NET	Cplex.Param.Preprocessing.Aggregator	AggInd (int)
OPL		aggind
Python	parameters.preprocessing.aggregator	preprocessing.aggregator
MATLAB	Cplex.Param.preprocessing.aggregator	preprocessing.aggregator
Interactive	preprocessing aggregator	preprocessing aggregator
Identifier	1003	1003

Description

Invokes the aggregator to use substitution where possible to reduce the number of rows and columns before the problem is solved. If set to a positive value, the aggregator is applied the specified number of times or until no more reductions are possible.

Table 2. Values

Value	Meaning
-1	Automatic (1 for LP, infinite for MIP) **default**
0	Do not use any aggregator
Any positive integer	Number of times to apply aggregator

API string encoding switch

API string encoding switch

Purpose

API string encoding switch

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_Read_APIEncoding	CPX_PARAM_APIENCODING
C++	IloCplex::Param::Read::APIEncoding	APIEncoding
Java	not in this API	not in this API
.NET	not in this API	not in this API
OPL	not in this API	not in this API
Python	parameters.read.apiencoding	read.apiencoding
MATLAB	Cplex.Param.read.apiencoding	read.apiencoding
Interactive	not in this component	not in this component
Identifier	1130	1130

Description

Specifies which encoding (also known as the code page) that CPLEX uses to represent strings. That is, this parameter tells CPLEX which characters to expect as input and how to represent as output such strings as the name of a model, of a variable, of a constraint. If, for example, your application uses an accent in the name of a model, an umlaut in the name of a variable, or a Chinese character for the name of a constraint, then this parameter is of interest to you.

Note:

This parameter has no effect on IBM CPLEX Optimizer for z/OS, where only EBCDIC IBM-1047 encoding is available.

Which features does this parameter govern?

Generally, this parameter impacts names, such as the name of a model, the names of variables, the names of constraints, and other modeling elements. Also CPLEX respects this parameter (in the application programming interfaces (APIs) where the parameter is available) when CPLEX provides information (such as a log, a warning, an error message) on an output stream.

In the Callable Library (**C API**), this parameter specifies the encoding in which CPLEX passes a string to a function destination added to a **channel** by means of the routine `CPXaddfuncdest`.

In the **C++ API**, this parameter specifies the encoding of **streams** accessed by the methods `setWarning` and `setOut`. CPLEX also encodes **exceptions** according to the value of this parameter.

In the **Python API**, this parameter specifies the encoding of **streams** accessed by methods such as Cplex.set_log_stream, Cplex.set_warning_stream, or Cplex.set_error_stream.

This parameter is not available in the **C#.NET API** where CPLEX relies on the native encoding.

This parameter is not available in the **Java API** where CPLEX relies on the native encoding.

For a brief description of the advantages of `UTF-8`, see the topic Selecting an encoding in the *CPLEX User's Manual*.

Which values does this parameter accept?

This parameter accepts a **string** specifying the user's choice of encoding, such as `UTF-8`, `ISO-8859-1`, `US-ASCII`, and so forth.

An asterisk (*) is an acceptable value in some situations. This wildcard instructs CPLEX to use the default converter according to the protocols and standards of International Components for Unicode (ICU). In other words, CPLEX calls ICU with a NULL encoding. When ICU is called with a NULL encoding (as in this case), ICU takes the name of the encoding from the terminal where the application started.

Other acceptable values of this parameter depend on the API.

- In the Callable Library (**C API**), this parameter accepts any string that is the name of a valid code page supported by CPLEX. For example, `UTF-8` is a multi-byte encoding that is an acceptable value for this parameter; it encompasses the ASCII character set; it does not allow valid characters to include a NULL byte.

  For a complete list of valid strings that are the name of an encoding (that is, the name of a code page), consult the web site of a standards organization such as:
  - A brief introduction to code pages
  - ICU: International Components for Unicode
  - International Components for Unicode at IBM
- In the **C++ API**, the value of this parameter must be the name of an encoding that is a superset of ASCII. For example, ASCII is a subset of the encoding or code page `UTF-8`, so `UTF-8` is an acceptable value for this parameter. Likewise, ASCII is a subset of `ISO-8859-1`, also known as `Latin-1`, so `ISO-8859-1` is an acceptable value for this parameter. However, the code page `UTF-16` is **not** acceptable, nor is `UTF-32` because both allow valid characters that contain a NULL byte.

What is the default value of this parameter?

The **default** value of this parameter depends on the API.

- In the **C API**, the default value is the string `ISO-8859-1` (also known as Latin-1).
- In the **C++ API**, the default value is the string `ISO-8859-1` (also known as Latin-1).
- In the **Python API** of CPLEX, the default value is UTF-8.
- In the **CPLEX for MATLAB APIs**, the default value is the asterisk (`*`).

The encoding `ISO-8859-1` is a superset of the familiar ASCII encoding, so it supports many widely used character sets. However, this default encoding cannot represent multi-byte character sets, such as Chinese, Japanese, Korean, Vietnamese, or Indian characters, for example. If you want to represent a character set that requires multiple bytes per character, then a better choice for the value of this parameter is `UTF-8`.

If you change the value of this parameter, you must verify that your choice is compatible with the "file encoding switch" on page 69 (`CPXPARAM_Read_FileEncoding`, `FileEncoding`). In fact, the encoding or code page specified by the API encoding parameter must be a subset of the encoding or code page specified by the file encoding parameter. For example, if the value of the file encoding parameter is `UTF-8`, then `US-ASCII` is an acceptable value of the API encoding parameter because `US-ASCII` is a subset of `UTF-8`. For examples of the hazards of incompatible choices of the encoding parameters, see the topic Selecting an encoding in the *CPLEX User's Manual*.

What about errors?

In situations where CPLEX encounters a string, such as content in a file, that is not compatible with the specified encoding, the behavior is not defined. Because of the incompatibility, CPLEX silently converts the string to an inappropriate character of the specified encoding, or CPLEX raises the error 1235 CPXERR_ENCODING_CONVERSION. For an example of why such unpredictable behavior occurs, see the topic Selecting an encoding in the *CPLEX User's Manual*.

Values

valid string for the name of an encoding (code page) that is a superset of ASCII; **default**: `ISO-8859-1` or empty string.

See also

"file encoding switch" on page 69

barrier algorithm

The default setting 0 uses the "infeasibility - estimate start" algorithm (setting 1) when solving subproblems in a MIP problem, and the standard barrier algorithm (setting 3) in other cases.

Purpose

Barrier algorithm

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM Barrier Algorithm	CPX_PARAM_BARALG
C++	IloCplex::Param::Barrier::Algorithm	BarAlg (int)
Java	IloCplex.Param.Barrier.Algorithm	BarAlg (int)
.NET	Cplex.Param.Barrier.Algorithm	BarAlg (int)
OPL		baralg
Python	parameters.barrier.algorithm	barrier.algorithm
MATLAB	Cplex.Param.barrier.algorithm	barrier.algorithm
Interactive	barrier algorithm	barrier algorithm
Identifier	3007	3007

Description

The default setting 0 uses the "infeasibility - estimate start" algorithm (setting 1) when solving subproblems in a MIP problem, and the standard barrier algorithm (setting 3) in other cases. The standard barrier algorithm is almost always fastest. However, on problems that are primal or dual infeasible (common for MIP subproblems), the standard algorithm may not work as well as the alternatives. The two alternative algorithms (settings 1 and 2) may eliminate numerical difficulties related to infeasibility, but are generally slower.

Value	Meaning
0	**Default** setting
1	Infeasibility-estimate start
2	Infeasibility-constant start
3	Standard barrier

barrier column nonzeros

Used in the recognition of dense columns.

Purpose

Barrier column nonzeros

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_Barrier_ColNonzeros	CPX_PARAM_BARCOLNZ
C++	IloCplex::Param::Barrier::ColNonzeros	BarColNz (int)
Java	IloCplex.Param.Barrier.ColNonzeros	BarColNz (int)
.NET	Cplex.Param.Barrier.ColNonzeros	BarColNz (int)
OPL		barcolnz
Python	parameters.barrier.colnonzeros	barrier.colnonzeros
MATLAB	Cplex.Param.barrier.colnonzeros	barrier.colnonzeros
Interactive	barrier colnonzeros	barrier colnonzeros
Identifier	3009	3009

Description

Used in the recognition of dense columns. If columns in the presolved and aggregated problem exist with more entries than this value, such columns are considered dense and are treated specially by the CPLEX barrier optimizer to reduce their effect.

Value	Meaning
0	Dynamically calculated; **default**
Any positive integer	Number of nonzero entries that make a column dense

barrier crossover algorithm

Decides which crossover is performed at the end of a barrier optimization.

Purpose

Barrier crossover algorithm

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_Barrier_Crossover	CPX_PARAM_BARCROSSALG
C++	IloCplex::Param::Barrier::Crossover	BarCrossAlg (int)
Java	IloCplex.Param.Barrier.Crossover	BarCrossAlg (int)
.NET	Cplex.Param.Barrier.Crossover	BarCrossAlg (int)
OPL		barcrossalg
Python	parameters.barrier.crossover	barrier.crossover
MATLAB	Cplex.Param.barrier.crossover	barrier.crossover
Interactive	barrier crossover	barrier crossover
Identifier	3018	3018

Description

Decides which crossover is performed at the end of a barrier optimization. This parameter applies when CPLEX uses the Barrier Optimizer to solve an LP or QP problem, or when it is used to solve the continuous relaxation of an MILP or MIQP at a node in a MIP.

By default, CPLEX does not invoke crossover on a QP problem. On an LP problem, it invokes primal and dual crossover in parallel when multiple threads are available.

Tip: Do not use the **deprecated** value -1 (minus one) of this parameter to turn off crossover. Instead, to turn off crossover, use the parameter “solution type for LP and QP” on page 152 set to the value 2, to search for a nonbasic solution.

Value	Meaning
-1	No crossover **deprecated**
0	Automatic: let CPLEX choose; **default**
1	Primal crossover
2	Dual crossover

barrier display information

Sets the level of barrier progress information to be displayed.

Purpose

Barrier display information

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_Barrier_Display	CPX_PARAM_BARDISPLAY
C++	IloCplex::Param::Barrier::Display	BarDisplay (int)
Java	IloCplex.Param.Barrier.Display	BarDisplay (int)
.NET	Cplex.Param.Barrier.Display	BarDisplay (int)
OPL		bardisplay
Python	parameters.barrier.display	barrier.display
MATLAB	Cplex.Param.barrier.display	barrier.display
Interactive	barrier display	barrier display
Identifier	3010	3010

Description

Sets the level of barrier progress information to be displayed.

Value	Meaning
0	No progress information
1	Normal setup and iteration information; **default**
2	Diagnostic information

convergence tolerance for LP and QP problems

Sets the tolerance on complementarity for convergence.

Purpose

Convergence tolerance for LP and QP problems

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_Barrier_ConvergeTol	CPX_PARAM_BAREPCOMP
C++	IloCplex::Param::Barrier::ConvergeTol	BarEpComp (double)
Java	IloCplex.Param.Barrier.ConvergeTol	BarEpComp (double)
.NET	Cplex.Param.Barrier.ConvergeTol	BarEpComp (double)
OPL		barepcomp
Python	parameters.barrier.convergetol	barrier.convergetol
MATLAB	Cplex.Param.barrier.convergetol	barrier.convergetol
Interactive	barrier convergetol	barrier convergetol
Identifier	3002	3002

Description

Sets the tolerance on complementarity for convergence. The barrier algorithm terminates with an optimal solution if the relative complementarity is smaller than this value.

Changing this tolerance to a smaller value may result in greater numerical precision of the solution, but also increases the chance of failure to converge in the algorithm and consequently may result in no solution at all. Therefore, caution is advised in deviating from the default setting.

Values

Any positive number greater than or equal to 1e-12; **default**: 1e-8.

See also

For problems with quadratic constraints (QCP), see “convergence tolerance for QC problems” on page 29

barrier growth limit

Used to detect unbounded optimal faces.

Purpose

Barrier growth limit

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_Barrier_Limits_Growth	CPX_PARAM_BARGROWTH
C++	IloCplex::Param::Barrier::Limits::Growth	BarGrowth (double)
Java	IloCplex.Param.Barrier.Limits.Growth	BarGrowth (double)
.NET	Cplex.Param.Barrier.Limits.Growth	BarGrowth (double)
OPL		bargrowth
Python	parameters.barrier.limits.growth	barrier.limit.growth
MATLAB	Cplex.Param.barrier.limits.growth	barrier.limits.growth
Interactive	barrier limits growth	barrier limits growth
Identifier	3003	3003

Description

Used to detect unbounded optimal faces. At higher values, the barrier algorithm is less likely to conclude that the problem has an unbounded optimal face, but more likely to have numerical difficulties if the problem has an unbounded face.

Values

1.0 or greater; **default**: 1e12.

barrier iteration limit

Sets the number of barrier iterations before termination.

Purpose

Barrier iteration limit

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_Barrier_Limits_Iteration	CPX_PARAM_BARITLIM
C++	IloCplex::Param::Barrier::Limits::Iteration	BarItLim (long)
Java	IloCplex.Param.Barrier.Limits.Iteration	BarItLim (long)
.NET	Cplex.Param.Barrier.Limits.Iteration	BarItLim (long)
OPL		baritlim
Python	parameters.barrier.limits.iteration	barrier.limit.iteration
MATLAB	Cplex.Param.barrier.limits.iteration	barrier.limits.iteration
Interactive	barrier limits iteration	barrier limits iteration
Identifier	3012	3012

Description

Sets the number of barrier iterations before termination. When this parameter is set to 0 (zero), no barrier iterations occur, but problem setup occurs and information about the setup is displayed (such as Cholesky factor statistics).

Table 3. Values

Value	Meaning
0	No barrier iterations
9223372036800000000	**default**
Any positive integer	Number of barrier iterations before termination

barrier maximum correction limit

Sets the maximum number of centering corrections done on each iteration.

Purpose

Barrier maximum correction limit

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_Barrier_Limits_Corrections	CPX_PARAM_BARMAXCOR
C++	IloCplex::Param::Barrier::Limits::Corrections	BarMaxCor (long)
Java	IloCplex.Param.Barrier.Limits.Corrections	BarMaxCor (long)
.NET	Cplex.Param.Barrier.Limits.Corrections	BarMaxCor (long)
OPL		barmaxcor
Python	parameters.barrier.limits.corrections	barrier.limit.corrections
MATLAB	Cplex.Param.barrier.limits.corrections	barrier.limits.corrections
Interactive	barrier limits corrections	barrier limits corrections
Identifier	3013	3013

Description

Sets the maximum number of centering corrections done on each iteration. An explicit value greater than 0 (zero) may improve the numerical performance of the algorithm at the expense of computation time.

Table 4. Values

Value	Meaning
-1	Automatic; let CPLEX choose; **default**
0	None
Any positive integer	Maximum number of centering corrections per iteration

barrier objective range

Sets the maximum absolute value of the objective function.

Purpose

Barrier objective range

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_Barrier_Limits_ObjRange	CPX_PARAM_BAROBJRNG
C++	IloCplex::Param::Barrier::Limits::ObjRange	BarObjRng (double)
Java	IloCplex.Param.Barrier.Limits.ObjRange	BarObjRng (double)
.NET	Cplex.Param.Barrier.Limits.ObjRange	BarObjRng (double)
OPL		barobjrng
Python	parameters.barrier.limits.objrange	barrier.limit.objrange
MATLAB	Cplex.Param.barrier.limits.objrange	barrier.limits.objrange
Interactive	barrier limits objrange	barrier limits objrange
Identifier	3004	3004

Description

Sets the maximum absolute value of the objective function. The barrier algorithm looks at this limit to detect unbounded problems.

Values

Any nonnegative number; **default**: 1e20

barrier ordering algorithm

Sets the algorithm to be used to permute the rows of the constraint matrix in order to reduce fill in the Cholesky factor.

Purpose

Barrier ordering algorithm

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_Barrier_Ordering	CPX_PARAM_BARORDER
C++	IloCplex::Param::Barrier::Ordering	BarOrder (int)
Java	IloCplex.Param.Barrier.Ordering	BarOrder (int)
.NET	Cplex.Param.Barrier.Ordering	BarOrder (int)
OPL		barorder
Python	parameters.barrier.ordering	barrier.ordering
MATLAB	Cplex.Param.barrier.ordering	barrier.ordering

API	Parameter Name	Name prior to V12.6.0
Interactive	barrier ordering	barrier ordering
Identifier	3014	3014

Description

Sets the algorithm to be used to permute the rows of the constraint matrix in order to reduce fill in the Cholesky factor.

Table 5. Values

Value	Meaning
0	Automatic: let CPLEX choose; **default**
1	Approximate minimum degree (AMD)
2	Approximate minimum fill (AMF)
3	Nested dissection (ND)

convergence tolerance for QC problems

Sets the tolerance on complementarity for convergence in quadratically constrained problems (QCPs).

Purpose

Convergence tolerance for quadratically constrained problems

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_Barrier_QCPConvergeTol	CPX_PARAM_BARQCPEPCOMP
C++	IloCplex::Param::Barrier::QCPConvergeTol	BarQCPEpComp (double)
Java	IloCplex.Param.Barrier.QCPConvergeTol	BarQCPEpComp (double)
.NET	Cplex.Param.Barrier.QCPConvergeTol	BarQCPEpComp (double)
OPL		barqcpepcomp
Python	parameters.barrier.qcpconvergetol	barrier.qcpconvergetol
MATLAB	Cplex.Param.barrier.qcpconvergetol	barrier.qcpconvergetol
Interactive	barrier qcpconvergetol	barrier qcpconvergetol
Identifier	3020	3020

Description

Sets the tolerance on complementarity for convergence in quadratically constrained problems (QCPs). The barrier algorithm terminates with an optimal solution if the relative complementarity is smaller than this value.

Changing this tolerance to a smaller value may result in greater numerical precision of the solution, but also increases the chance of a convergence failure in the algorithm and consequently may result in no solution at all. Therefore, caution is advised in deviating from the default setting.

Values

Any positive number greater than or equal to 1e-12; **default**: 1e-7.

For LPs and for QPs (that is, when all the constraints are linear) see "convergence tolerance for LP and QP problems" on page 25.

barrier starting point algorithm

Sets the algorithm to be used to compute the initial starting point for the barrier optimizer.

Purpose

Barrier starting point algorithm

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_Barrier_StartAlg	CPX_PARAM_BARSTARTALG
C++	IloCplex::Param::Barrier::StartAlg	BarStartAlg (int)
Java	IloCplex.Param.Barrier.StartAlg	BarStartAlg (int)
.NET	Cplex.Param.Barrier.StartAlg	BarStartAlg (int)
OPL		barstartalg
Python	parameters.barrier.startalg	barrier.startalg
MATLAB	Cplex.Param.barrier.startalg	barrier.startalg
Interactive	barrier startalg	barrier startalg
Identifier	3017	3017

Description

Sets the algorithm to be used to compute the initial starting point for the barrier optimizer.

Value	Meaning
1	Dual is 0 (zero); **default**
2	Estimate dual
3	Average of primal estimate, dual 0 (zero)
4	Average of primal estimate, estimate dual

Benders strategy

This parameter specifies whether CPLEX should apply Benders algorithm as a strategy to solve a model.

Purpose

Benders decomposition algorithm as a strategy

API	Parameter Name
C	CPXPARAM_Benders_Strategy
C++	IloCplex::Param::Benders::Strategy
Java	IloCplex.Param.Benders.Strategy
.NET	Cplex.Param.Benders.Strategy
OPL	bendersstrategy
Python	parameters.benders.strategy
MATLAB	Cplex.Param.benders.strategy
Interactive	benders strategy *value*

API	Parameter Name
Identifier	1501

Description

Given a formulation of a problem, CPLEX can decompose the model into a single master and (possibly multiple) subproblems. To do so, CPLEX can make use of annotations that you supply for your model. The strategy can be applied to mixed-integer linear programs (MILP). For certain types of problems, this approach offers significant performance improvements as subproblems can be solved in parallel.

For mixed integer programs (MIP), under certain conditions, CPLEX can apply Benders algorithm to improve the search to find more feasible solutions more quickly.

Tolerances for Benders

These other CPLEX parameters allow you to specify cut tolerances particular to Benders:

- "Benders feasibility cut tolerance" on page 35
- "Benders optimality cut tolerance" on page 36

Strategies

By **default**, if you did not annotate your model to specify a decomposition, CPLEX executes conventional branch and bound. If you annotated your model, CPLEX attempts to apply your annotations and to refine your decomposition before it solves the model. With this parameter, you can direct CPLEX to decompose your model and to apply its implementation of Benders algorithm in one of these alternative ways:

- 1 USER: CPLEX attempts to decompose your model strictly according to your annotations.
- 2 WORKERS: CPLEX decomposes your model by using your annotations as hints and refining the decomposition where it can.
  - CPLEX initially decomposes your model according to your annotations.
  - Cplex then attempts to refine that decomposition by further decomposing the specified subproblems.

  This approach can be useful if you annotate certain variables to go into master, and all others to go into a single subproblem, which CPLEX can then decompose further for you.
- 3 FULL: CPLEX automatically decomposes your model, ignoring any annotations you may have supplied.
  - CPLEX puts all integer variables into the master.
  - CPLEX puts all continuous variables into a subproblem.
  - CPLEX further decomposes that subproblem, if possible.

Annotations for decomposition

If you want to specify a decomposition to CPLEX, you use **annotations**. First, create a new annotation `cpxBendersPartition` of type `long`. Next, use `cpxBendersPartition` to annotate some or all of the variables in your model. These annotations specify to CPLEX whether certain variables belong to the master or to one of the subproblems assigned to workers (where the subproblems are numbered from 1 (one) to N, the number of subproblems).

- If you annotate a given variable with the value 0 (zero), CPLEX assigns that variable to the master.
- If you do not annotate a given variable, CPLEX assumes the **default** value annotation.
- If you annotate a given variable with the value i, where i is greater than or equal to 1 (one), CPLEX assigns that variable to subproblem i.
- If you annotate a given variable with a value strictly less than zero, CPLEX raises an error CPXERR_NO_DECOMPOSITION.

CPLEX produces an error CPXERR_BAD_DECOMPOSITION if the annotated decomposition does not yield disjoint subproblems. For example, if your annotations specify that two (or more) variables belong in different subproblems, yet your model specifies that these variables participate in the same constraint, then these variables are linked. Consequently, the subproblems where these variables appear according to your annotations are not disjoint with respect to the partition.

Tip: It is a good idea to verify that the N subproblems plus master actually define a complete decomposition of the original model into disjoint subproblems. That is, make sure you have a complete partition for your decomposition into disjoint subproblems.

Benders without annotations

If you want CPLEX to apply a Benders strategy as it solves your problem, but you do not specify `cpxBendersPartition` annotations yourself, CPLEX puts all **integer** variables in master and **continuous linear** variables into subproblems.

Errors in automatic decomposition

When you specify a Benders strategy, but you do not annotate your model, CPLEX attempts to partition the variables into a Benders decomposition. In such a case, if there are no integer variables in your model, or if there are no continuous variables in your model, CPLEX raises an error (CPXERR_BAD_DECOMPOSITION or CPXERR_NO_DECOMPOSITION) stating that it cannot automatically decompose the model to apply a Benders strategy.

Table 6. Values.

Value	Name	Meaning
-1	OFF	Execute conventional branch and bound; ignore any Benders annotations. That is, do not use Benders algorithm even if a Benders partition of the current model is present

Table 6. Values (continued).

Value	Name	Meaning
0	AUTO	**default** Let CPLEX decide. • case 1: If the user supplies no annotations with the model, CPLEX executes conventional branch and bound. • case 2: If annotations specifying a Benders partition of the current model are available, CPLEX attempts to decompose the model. CPLEX uses the master as given by the annotations, and attempts to partition the subproblems further, if possible, before applying Benders algorithm to solve the model. If the user supplied annotations, but the annotations supplied do not lead to a complete decomposition into master and disjoint subproblems (that is, if the annotations are wrong in that sense), CPLEX produces the error CPXERR_BAD_DECOMPOSITION.
1	USER	CPLEX applies Benders algorithm to a decomposition based on annotations supplied by the user. If no annotations to decompose the model are available, this setting produces the error CPXERR_NO_DECOMPOSITION. If the user supplies annotations, but the supplied annotations do not lead to a complete partition of the original model into disjoint master and subproblems, then this setting produces the error CPXERR_BAD_DECOMPOSITION.

Table 6. Values (continued).

Value	Name	Meaning
2	WORKERS	CPLEX accepts the master as given and attempts to decompose the remaining elements into disjoint subproblems to assign to workers. It then solves the Benders decomposition of the model. If no annotations to decompose the model are available, this setting produces the error CPXERR_NO_DECOMPOSITION. If the user supplies annotations, but the supplied annotations do not lead to a complete partition of the original model into disjoint master and subproblems, then this setting produces the error CPXERR_BAD_DECOMPOSITION.

Table 6. Values (continued).

Value	Name	Meaning
3	FULL	CPLEX ignores any annotation file supplied with the model; CPLEX applies presolve; CPLEX then automatically generates a Benders partition, putting **integer** variables in master and **continuous linear** variables into disjoint subproblems. CPLEX then solves the Benders decomposition of the model. If the problem is a strictly linear program (LP), that is, there are no integer-constrained variables to put into master, then CPLEX reports the error CPXERR_PARAM_INCOMPATIBLE. If the problem is a mixed integer linear program (MILP) where all variables are integer-constrained, (that is, there are no continuous linear variables to decompose into disjoint subproblems) then CPLEX reports the error CPXERR_NO_DECOMPOSITION. If the problem is a mixed integer linear program (MILP) where all variables are continuous, (that is, there are no integer-constrained variables to decompose into master) then CPLEX reports the error CPXERR_NO_DECOMPOSITION.

Benders feasibility cut tolerance

This parameter specifies a tolerance for the feasibility of cuts when CPLEX solves a model decomposed according to Benders algorithm.

Purpose

Tolerance for whether a feasibility cut has been violated in Benders decomposition

API	Parameter Name
C	CPXPARAM_Benders_Tolerances_feasibilitycut
C++	IloCplex::Param::Benders::Tolerances::feasibilitycut
Java	IloCplex.Param.Benders.Tolerances.feasibilitycut
.NET	Cplex.Param.Benders.Tolerances.feasibilitycut

API	Parameter Name
Python	parameters.benders.tolerances.feasibilitycut
MATLAB	Cplex.Param.benders.tolerances.feasibilitycut
Interactive	benders tolerances feasibilitycut
Identifier	1509

Description

This parameter sets a tolerance for the violation of a feasibility cut in Benders algorithm.

Values

Any number from 1e-9 to 1e-1; **default**: 1e-06.

Benders optimality cut tolerance

This parameter specifies a tolerance for the optimality of cuts when CPLEX solves a model decomposed according to Benders algorithm.

Purpose

Tolerance for optimality cuts in Benders decomposition

API	Parameter Name
C	CPXPARAM_Benders_Tolerances_optimalitycut
C++	IloCplex::Param::Benders::Tolerances:optimalitycut
Java	IloCplex.Param.Benders.Tolerances.optimalitycut
.NET	Cplex.Param.Benders.Tolerances.optimalitycut
Python	parameters.benders.tolerances.optimalitycut
MATLAB	Cplex.Param.benders.tolerances.optimalitycut
Interactive	benders tolerances optimalitycut
Identifier	1510

Description

This parameter sets a tolerance for the violation of an optimality cut in Benders algorithm.

Values

Any number from 1e-9 to 1e-1; **default**: 1e-06.

MIP strategy best bound interval

Sets the best bound interval for MIP strategy.

Purpose

MIP strategy best bound interval

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_MIP_Strategy_BBInterval	CPX_PARAM_BBINTERVAL

API	Parameter Name	Name prior to V12.6.0
C++	IloCplex::Param::MIP::Strategy::BBInterval	BBInterval (long)
Java	IloCplex.Param.MIP.Strategy.BBInterval	BBInterval (long)
.NET	Cplex.Param.MIP.Strategy.BBInterval	BBInterval (long)
OPL		bbinterval
Python	parameters.mip.strategy.bbinterval	mip.strategy.bbinterval
MATLAB	Cplex.Param.mip.strategy.bbinterval	mip.strategy.bbinterval
Interactive	mip strategy bbinterval	mip strategy bbinterval
Identifier	2039	2039

Description

Sets the best bound interval for MIP strategy.

When you set this parameter to best estimate node selection, the best bound interval is the interval at which the best bound node, instead of the best estimate node, is selected from the tree. A best bound interval of 0 (zero) means "never select the best bound node." A best bound interval of 1 (one) means "always select the best bound node," and is thus equivalent to node select 1 (one).

Higher values of this parameter mean that the best bound node will be selected less frequently; experience has shown it to be beneficial to select the best bound node occasionally, and therefore the default value of this parameter is 7.

Table 7. Values

Value	Meaning
0	Never select best bound node; always select best estimate
1	Always select best bound node
7	Select best bound node occasionally; **default**
Any positive integer	Select best bound node less frequently than best estimate node

See also

"MIP node selection strategy" on page 101

bound strengthening switch

Decides whether to apply bound strengthening in mixed integer programs (MIPs).

Purpose

Bound strengthening switch

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_Preprocessing_BoundStrength	CPX_PARAM_BNDSTRENIND
C++	IloCplex::Param::Preprocessing::BoundStrength	BndStrenInd (int)
Java	IloCplex.Param.Preprocessing.BoundStrength	BndStrenInd (int)
.NET	Cplex.Param.Preprocessing.BoundStrength	BndStrenInd (int)
OPL		bndstrenind
Python	parameters.preprocessing.boundstrength	preprocessing.boundstrength

API	Parameter Name	Name prior to V12.6.0
MATLAB	Cplex.Param.preprocessing.boundstrength	preprocessing.boundstrength
Interactive	preprocessing boundstrength	preprocessing boundstrength
Identifier	2029	2029

Description

Decides whether to apply bound strengthening in mixed integer programs (MIPs). Bound strengthening tightens the bounds on variables, perhaps to the point where the variable can be fixed and thus removed from consideration during branch and cut.

Tip:

Strengthening means to replace one row of a model with another such that an integer vector is feasible in the new row if and only if the integer vector was feasible in the original row. Strengthening improves the LP relaxation of the row by finding a dominating row. In other words, the LP region defined by the strengthened row plus the bounds on the variables will be strictly contained in the LP region defined by the original row plus bounds on the variables.

Value	Meaning
-1	Automatic: let CPLEX choose; **default**
0	Do not apply bound strengthening
1	Apply bound strengthening

Boolean Quadric Polytope cuts

Controls the addition of BQP cuts for nonconvex QP or MIQP solved to global optimality.

Purpose

Boolean Quadric Polytope (BQP) cuts switch

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_MIP_Cuts_BQP	CPX_PARAM_BQPCUTS
C++	IloCplex::Param::MIP::Cuts::BQP	
Java	IloCplex.Param.MIP.Cuts.BQP	
.NET	Cplex.Param.MIP.Cuts.BQP	
OPL	bqpcuts	
Python	parameters.mip.cuts.bqp	
MATLAB	Cplex.Param.mip.cuts.bqp	
Interactive	mip cuts bqp	
Identifier	2195	

Description

This parameter controls the addition of cuts based on the Boolean Quadric Polytope (BQP) for **nonconvex** quadratic programs (QP) or **nonconvex** mixed integer quadratic programs (MIQP) solved to **global optimality**.

For a definition of the cuts based on a Boolean Quadric Polytope (BQP), see the topic Boolean Quadric Polytope (BQP) cuts in the general topic Cuts in the *CPLEX User's Manual*. The table Parameters for controlling cuts, also in the user's manual, includes links to the documentation of other parameters affecting other types of cuts.

Value	Meaning
-1	Do not generate BQP cuts
0	Automatic: let CPLEX decide; **default**
1	Generate BQP cuts moderately
2	Generate BQP cuts aggressively
3	Generate BQP cuts very aggressively

MIP branching direction

Decides which branch, the up or the down branch, should be taken first at each node.

Purpose

MIP branching direction

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_MIP_Strategy_Branch	CPX_PARAM_BRDIR
C++	IloCplex::Param::MIP::Strategy::Branch	BrDir (int)
Java	IloCplex.Param.MIP.Strategy.Branch	BrDir (int)
.NET	Cplex.Param.MIP.Strategy.Branch	BrDir (int)
OPL		brdir
Python	parameters.mip.strategy.branch	mip.strategy.branch
MATLAB	Cplex.Param.mip.strategy.branch	mip.strategy.branch
Interactive	mip strategy branch	mip strategy branch
Identifier	2001	2001

Description

Decides which branch, the up or the down branch, should be taken first at each node.

Value	Symbol	Meaning
-1	`CPX_BRDIR_DOWN`	Down branch selected first
0	`CPX_BRDIR_AUTO`	Automatic: let CPLEX choose; **default**
1	`CPX_BRDIR_UP`	Up branch selected first

backtracking tolerance

Controls how often backtracking is done during the branching process.

Purpose

Backtracking tolerance

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_MIP_Strategy_Backtrack	CPX_PARAM_BTTOL
C++	IloCplex::Param::MIP::Strategy::Backtrack	BtTol (double)
Java	IloCplex.Param.MIP.Strategy.Backtrack	BtTol (double)
.NET	Cplex.Param.MIP.Strategy.Backtrack	BtTol (double)
OPL		bttol
Python	parameters.mip.strategy.backtrack	mip.strategy.backtrack
MATLAB	Cplex.Param.mip.strategy.backtrack	mip.strategy.backtrack
Interactive	mip strategy backtrack	mip strategy backtrack
Identifier	2002	2002

Description

Controls how often backtracking is done during the branching process. The decision when to backtrack depends on three values that change during the course of the optimization:

- the objective function value of the best integer feasible solution (*incumbent*);
- the best remaining objective function value of any unexplored node (*best node*);
- the objective function value of the most recently solved node (*current objective*).

If a cutoff tolerance ("upper cutoff" on page 53 or "lower cutoff" on page 51) has been set by the user, then that value is used as the incumbent until an integer feasible solution is found.

The *target gap* is defined to be the absolute value of the difference between the incumbent and the best node, multiplied by this backtracking parameter. CPLEX does not backtrack until the absolute value of the difference between the objective of the current node and the best node is at least as large as the target gap.

Low values of this backtracking parameter thus tend to increase the amount of backtracking, which makes the search process more of a pure best-bound search. Higher parameter values tend to decrease backtracking, making the search more of a pure depth-first search.

The backtracking value has effect only after an integer feasible solution is found or when a cutoff has been specified. Note that this backtracking value merely permits backtracking but does not force it; CPLEX may choose to continue searching a limb of the tree if that limb seems a promising candidate for finding an integer feasible solution.

Values

Any number from 0.0 to 1.0; **default**: 0.9999

See also

"upper cutoff" on page 53, "lower cutoff" on page 51

calculate QCP dual values

Instructs CPLEX to calculate the dual values of a quadratically constrained problem

Purpose

calculating QCP dual values

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_Preprocessing_QCPDuals	CPX_PARAM_CALCQCPDUALS
C++	IloCplex::Param::Preprocessing::QCPDuals	CalcQCPDuals (int)
Java	IloCplex.Param.Preprocessing.QCPDuals	CalcQCPDuals (int)
.NET	Cplex.Param.Preprocessing.QCPDuals	CalcQCPDuals (int)
OPL		not available
Python	parameters.preprocessing.qcpduals	preprocessing.qcpduals
MATLAB	Cplex.Param.preprocessing.qcpduals	preprocessing.qcpduals
Interactive	preprocessing qcpduals	preprocessing qcpduals
Identifier	4003	4003

Description

This parameter determines whether CPLEX preprocesses a quadratically constrained program (QCP) so that the user can access dual values for the QCP.

If this parameter is set to 0 (zero), then CPLEX does not calculate dual values for the QCP.

If this parameter is set to 1 (one), its default value, then CPLEX calculates dual values for the QCP as long as the calculations do not interfere with presolve reductions.

If this parameter is set to 2, then CPLEX calculates dual values and moreover, CPLEX disables any presolve reductions that interfere with these dual-value calculations.

For more information about accessing dual values of a QCP, see the topic Accessing dual values and reduced costs of QCP solutions in the *CPLEX User's Manual*.

For more information about presolve reductions, see the topic Advanced presolve routines in the *CPLEX User's Manual*.

Values

Table 8. Values.

Value	Meaning	Symbol in Callable Library (C API)	Symbol in C++, Java, .NET APIs	Symbol in Python API
0	Do not calculate dual values for the QCP	CPX_QCPDUALS_NO	QCPDualsNo	no

Table 8. Values (continued).

Value	Meaning	Symbol in Callable Library (C API)	Symbol in C++, Java, .NET APIs	Symbol in Python API
1	Calculate dual values for the QCP as long as the calculations do not interfere with presolve reductions. **default**	CPX_QCPDUALS_IFPOSSIBLE	QCPDualsIfPossible	if_possible
2	Calculate dual values and disable any presolve reductions that interfere with these calculations.	CPX_QCPDUALS_FORCE	QCPDualsForce	force

MIP cliques switch

Decides whether or not clique cuts should be generated for the problem.

Purpose

MIP cliques switch

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_MIP_Cuts_Cliques	CPX_PARAM_CLIQUES
C++	IloCplex::Param::MIP::Cuts::Cliques	Cliques (int)
Java	IloCplex.Param.MIP.Cuts.Cliques	Cliques (int)
.NET	Cplex.Param.MIP.Cuts.Cliques	Cliques (int)
OPL		cliques
Python	parameters.mip.cuts.cliques	mip.cuts.cliques
MATLAB	Cplex.Param.mip.cuts.cliques	mip.cuts.cliques
Interactive	mip cuts cliques	mip cuts cliques
Identifier	2003	2003

Description

Decides whether or not clique cuts should be generated for the problem. Setting the value to 0 (zero), the default, specifies that the attempt to generate cliques should continue only if it seems to be helping.

For a definition of a clique cut, see the topic Clique cuts in the general topic Cuts in the *CPLEX User's Manual*. The table Parameters for controlling cuts, also in the user's manual, includes links to the documentation of other parameters affecting other types of cuts.

Value	Meaning
-1	Do not generate clique cuts
0	Automatic: let CPLEX choose; **default**
1	Generate clique cuts moderately
2	Generate clique cuts aggressively
3	Generate clique cuts very aggressively

clock type for computation time

Decides how computation times are measured for both reporting performance and terminating optimization when a time limit has been set.

Purpose

Clock type for computation time

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_ClockType	CPX_PARAM_CLOCKTYPE
C++	IloCplex::Param::ClockType	ClockType (int)
Java	IloCplex.Param.ClockType	ClockType (int)
.NET	Cplex.Param.ClockType	ClockType (int)
OPL		clocktype
Python	parameters.clocktype	clocktype
MATLAB	Cplex.Param.clocktype	clocktype
Interactive	clocktype	clocktype
Identifier	1006	1006

Description

Decides how computation times are measured for both reporting performance and terminating optimization when a time limit has been set. Small variations in measured time on identical runs may be expected on any computer system with any setting of this parameter.

The default setting 2 supports wall clock time.

Value	Meaning
0	Automatic: let CPLEX choose
1	CPU time
2	Wall clock time (total physical time elapsed); **default**

clone log in parallel optimization

Specifies whether to create clone log files during parallel optimization.

Purpose

Creates clone log files in parallel optimization

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_Output_CloneLog	CPX_PARAM_CLONELOG
C++	IloCplex::Param::Output::CloneLog	CloneLog
Java	IloCplex.Param.Output.CloneLog	CloneLog
.NET	Cplex.Param.Output.CloneLog	CloneLog
OPL		not available
Python	parameters.output.clonelog	output.clonelog
MATLAB	Cplex.Param.output.clonelog	output.clonelog
Interactive	output clonelog	output clonelog
Identifier		

Description

Specifies whether CPLEX clones the log files of nodes during parallel or concurrent optimization. When you use parallel or concurrent CPLEX, this feature makes it more convenient to check node logs when you use more than one thread to solve models. For parallel optimization on N threads, for example, turning on this parameter creates N logs,`clone[0].log` through `clone[N-1].log`. This feature is available only during concurrent optimization and mixed integer programming (MIP) optimization.

Value	Meaning
-1	CPLEX does not clone log files. (off)
0	Automatic: CPLEX clones log files if log file is specified. **default**
1	CPLEX clones log files. (on)

coefficient reduction setting

Decides how coefficient reduction is used.

Purpose

Coefficient reduction setting

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_Preprocessing_CoeffReduce	CPX_PARAM_COEREDIND
C++	IloCplex::Param::Preprocessing::CoeffReduce	CoeRedInd (int)
Java	IloCplex.Param.Preprocessing.CoeffReduce	CoeRedInd (int)
.NET	Cplex.Param.Preprocessing.CoeffReduce	CoeRedInd (int)
OPL		coeredind
Python	parameters.preprocessing.coeffreduce	preprocessing.coeffreduce
MATLAB	Cplex.Param.preprocessing.coeffreduce	preprocessing.coeffreduce
Interactive	preprocessing coeffreduce	preprocessing coeffreduce
Identifier	2004	2004

Description

Decides how coefficient reduction is used. Coefficient reduction improves the objective value of the initial (and subsequent) LP relaxations solved during branch and cut by reducing the number of non-integral vertices. By **default**, CPLEX applies coefficient reductions during preprocessing of a model.

The value 0 (zero) turns off coefficient reduction during preprocessing.

The value 1 (one) applies limited coefficient reduction to achieve only integral coefficients.

The value 2, applies coefficient reduction somewhat more aggressively, reducing all coefficients that can be reduced.

The value 3, the most aggressive setting of this parameter, applies a technique known as tilting. Tilting can cut off additional fractional solutions in some models. Cutting off these fractional solutions potentially yields more progress in both the best node and best integer solution in those particular models.

Tip:

Tilting means to replace one row of a model with another such that an integer vector is feasible in the new row if and only if the integer vector was feasible in the original row. In contrast to a dominating row, the LP region for a tilted row can contain fractional points that are excluded by the LP region of the original row and the bounds of the variables. But, the aim is to tilt the row in such a way that it gets tighter in those areas of the LP polyhedron that are not already covered by other constraints.

Value	Meaning
-1	Automatic: let CPLEX decide; **default**
0	Do not use coefficient reduction
1	Reduce only to integral coefficients
2	Reduce all potential coefficients
3	Reduce aggressively with tilting

variable (column) read limit

Specifies a limit for the number of columns (variables) to read for an allocation of memory.

Purpose

Limit the number of variables (columns) read for memory allocation

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_Read_Variables	CPX_PARAM_COLREADLIM
C++	IloCplex::Param::Read::Variables	ColReadLim (int)
Java	IloCplex.Param.Read.Variables	ColReadLim (int)
.NET	Cplex.Param.Read.Variables	ColReadLim (int)
OPL		not available
Python	parameters.read.variables	read.variables
MATLAB	Cplex.Param.read.variables	read.variables
Interactive	read variables	read variables
Identifier	1023	1023

Description

Specifies a limit for the number of columns (variables) to read for an allocation of memory.

This parameter does not restrict the size of a problem. Rather, it indirectly specifies the default amount of memory that will be pre-allocated before a problem is read from a file. If the limit is exceeded, more memory is automatically allocated.

Values

Any integer from 0 (zero) to CPX_BIGINT; **default**: 60 000.

conflict information display

Decides how much information CPLEX reports when the conflict refiner is working.

Purpose

Conflict information display

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_Conflict_Display	CPX_PARAM_CONFLICTDISPLAY
C++	IloCplex::Param::Conflict::Display	ConflictDisplay (int)
Java	IloCplex.Param.Conflict.Display	ConflictDisplay (int)
.NET	Cplex.Param.Conflict.Display	ConflictDisplay (int)
OPL		conflictdisplay
Python	parameters.conflict.display	conflict.display
MATLAB	Cplex.Param.conflict.display	conflict.display
Interactive	conflict display i	conflict display i
Identifier	1074	1074

Description

Decides how much information CPLEX reports when the conflict refiner is working.

Table 9. Values

Value	Meaning
0	No display
1	Summary display; **default**
2	Detailed display

conflict refiner algorithm

Determines which algorithm CPLEX uses in the conflict refiner to find a minimal conflict in an infeasible model.

Purpose

Conflict refiner algorithm

API	Parameter Name
C	CPXPARAM_Conflict_Algorithm
C++	IloCplex::Param::Conflict::Algorithm
Java	IloCplex.Param.Conflict.Algorithm
.NET	Cplex.Param.Conflict.Algorithm
OPL	
Python	parameters.conflict_algorithm
MATLAB	Cplex.Param.
Interactive	conflict i
Identifier	1073

Description

This parameter determines which algorithm CPLEX uses in the conflict refiner to discover a minimal set of conflicting constraints in an infeasible model.

By default, CPLEX automatically chooses the algorithm in the conflict refiner according to the way that CPLEX determined the infeasibility of the original problem.

To see which algorithm that CPLEX chose by default, use the parameter to manage "conflict information display" on page 46.

The algorithm selected by default by CPLEX ordinarily yields the fastest possible performance, while it makes sure that CPLEX finds a minimal conflict if you allow it to run to completion. However, in certain models, a nondefault selection can work better. In particular, settings 1 (one), 2, or 3 entail preprocessing or propagation algorithms that are very fast, but those same algorithms can be inefficient at discarding large numbers of constraints quickly. In such a case, explicitly setting this parameter to 5 for a limited solve by the conflict refiner or to 6 for a full solve can improve performance.

Also, only settings 0, 4, and 6 guarantee a provably minimal conflict. The faster but more limited settings of 1, 2, 3, and 5 are less effective at proving whether individual constraints are included in or excluded from a minimal conflict.

Setting 4 of this parameter is primarily for continuous models.

Table 10. Values

Value	Meaning
0	Automatic: let CPLEX choose; **default**
1	Simple, fast algorithm
2	Bound propagation
3	Presolve
4	Irreducibly inconsistent set (IIS) for continuous models
5	Limited solve
6	Full solve

See also

"conflict information display" on page 46

MIP covers switch

Decides whether or not cover cuts should be generated for the problem.

Purpose

MIP covers switch

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_MIP_Cuts_Covers	CPX_PARAM_COVERS
C++	IloCplex::Param::MIP::Cuts::Covers	Covers (int)

API	Parameter Name	Name prior to V12.6.0
Java	IloCplex.Param.MIP.Cuts.Covers	Covers (int)
.NET	Cplex.Param.MIP.Cuts.Covers	Covers (int)
OPL		covers
Python	parameters.mip.cuts.covers	mip.cuts.covers
MATLAB	Cplex.Param.mip.cuts.covers	mip.cuts.covers
Interactive	mip cuts covers	mip cuts covers
Identifier	2005	2005

Description

Decides whether or not cover cuts should be generated for the problem. Setting the value to 0 (zero), the default, indicates that the attempt to generate covers should continue only if it seems to be helping.

For a definition of a cover cut, see the topic Cover cuts in the general topic Cuts, in the *CPLEX User's Manual*. The table Parameters for controlling cuts, also in the user's manual, includes links to the documentation of other parameters affecting other types of cuts

Table 11. Values

Value	Meaning
-1	Do not generate cover cuts
0	Automatic: let CPLEX choose; **default**
1	Generate cover cuts moderately
2	Generate cover cuts aggressively
3	Generate cover cuts very aggressively

CPU mask to bind threads to cores

Specifies how CPLEX binds threads to cores

Purpose

Specifies whether and how CPLEX binds threads to cores

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_CPUmask	CPX_PARAM_CPUMASK
C++	IloCplex::Param::CPUmask	
Java	IloCplex.Param.CPUmask	
.NET	Cplex.Param.CPUmask	
OPL		
Python	parameters.cpumask	
MATLAB	Cplex.Param.cpumask	
Interactive	cpumask	
Identifier	1144	

Description

This parameter specifies how CPLEX should bind threads to cores on platforms where CPLEX supports binding threads to cores. Currently, CPLEX supports CPU binding of threads to cores on AIX and Linux platforms.

This feature is known as CPU binding. It is also sometimes known as *processor affinity*.

- If the value is "off" in lower case, then CPLEX does not bind threads to cores.
- If the value is "auto" in lower case, then by **default** CPLEX decides whether or not to bind threads to cores.
- Otherwise, CPLEX treats the value of this parameter as a hexadecimal number without the usual `0x` prefix. Thus a valid string consists of these elements:
  - any digit from 0 (zero) through 9 (inclusive)
  - any lower case character in the range a through f (inclusive)
  - any upper case character in the range A through F (inclusive)

  CPLEX rejects a string containing any other digits or characters than those.

When the value of this parameter is a valid string (that is, a hexadecimal number, not "off" and not "auto"), it serves as a mask to specify which cores CPLEX is allowed to use. Each bit of this hexadecimal number corresponds to a core. CPLEX uses the i^{th} core if and only if the i^{th} bit of this string is set to 1 (one). For example,

- `00000001` designates core #0.
- `00000003` designates cores #0 and #1.
- `FFFFFFFF` designates all cores #0 through #31.

Note that the CPU mask can designate non-existent logical cores; for example, `FFFFFFFF` is a valid value on a 4-core machine. In such cases, CPLEX disregards the bits corresponding to non-existent logical cores.

Tip: For GNU/Linux users, this parameter behaves like the `taskset` command (except that values of this parameter lack the prefix `0x`).

Examples

- For example, if a user sets this CPU mask parameter to the hexadecimal value "f" on a 16-core machine, and the user sets the "global thread count" on page 157 parameter 8, then CPLEX binds two threads to each of the first four cores, resulting in a total of 8 threads. CPLEX will not use any of the remaining 12 cores.
- For example, on a 16 core machine, consider the difference between the value "off" and the value "ffff". If the value of this parameter is "off" CPLEX does no binding. If the value of this parameter is "ffff", CPLEX binds threads to cores.

Terminology

The description of this parameter observes the following conventions in terminology.

The term *CPU* (that is, central processing unit) corresponds to a single chip; that chip may include multiple physical cores.

The term *logical core* is synonymous with the term *virtual core*; the operating system reports the number of these logical cores as the number of cores present on the machine. This number can be different from the number of physical cores in the machine when features such as hyperthreading are enabled in the CPU.

Round-robin refers to the algorithm that repeats the cycle of assigning one thread to each core until all threads are assigned to a core.

Table 12. Values

String	Meaning
"off"	CPLEX performs no binding.
"auto"	**Default** CPLEX decides whether to bind threads to cores.
A string consisting of digits or characters from the set {0-9, a-f, A-F}	CPLEX binds the threads in round-robin fashion to the cores specified by the mask.

See also

"global thread count" on page 157

simplex crash ordering

Decides how CPLEX orders variables relative to the objective function when selecting an initial basis.

Purpose

Simplex crash ordering

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_Simplex_Crash	CPX_PARAM_CRAIND
C++	IloCplex::Param::Simplex::Crash	CraInd (int)
Java	IloCplex.Param.Simplex.Crash	CraInd (int)
.NET	Cplex.Param.Simplex.Crash	CraInd (int)
OPL		craind
Python	parameters.simplex.crash	simplex.crash
MATLAB	Cplex.Param.simplex.crash	simplex.crash
Interactive	simplex crash	simplex crash
Identifier	1007	1007

Description

Decides how CPLEX orders variables relative to the objective function when selecting an initial basis.

Table 13. Values

Value	Meaning
LP Primal	
-1	Alternate ways of using objective coefficients
0	Ignore objective coefficients during crash
1	Alternate ways of using objective coefficients; **default**
LP Dual	

Table 13. Values (continued)

Value	Meaning
-1	Aggressive starting basis
0	Aggressive starting basis
1	Default starting basis; **default**
QP Primal	
-1	Slack basis
0	Ignore Q terms and use LP solver for crash
1	Ignore objective and use LP solver for crash; **default**
QP Dual	
-1	Slack basis
0	Use Q terms for crash
1	Use Q terms for crash; **default**

lower cutoff

Sets lower cutoff tolerance.

Purpose

Lower cutoff

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_MIP_Tolerances_LowerCutoff	CPX_PARAM_CUTLO
C++	IloCplex::Param::MIP::Tolerances::LowerCutoff	CutLo (double)
Java	IloCplex.Param.MIP.Tolerances.LowerCutoff	CutLo (double)
.NET	Cplex.Param.MIP.Tolerances.LowerCutoff	CutLo (double)
OPL		cutlo
Python	parameters.mip.tolerances.lowercutoff	mip.tolerances.lowercutoff
MATLAB	Cplex.Param.mip.tolerances.lowercutoff	mip.tolerances.lowercutoff
Interactive	mip tolerances lowercutoff	mip tolerances lowercutoff
Identifier	2006	2006

Description

Sets the lower cutoff tolerance in a MIP. When the problem is a maximization problem, CPLEX cuts off or discards solutions that are less than the specified cutoff value. If the model has no solution with an objective value greater than or equal to the cutoff value, then CPLEX declares the model infeasible. In other words, setting the lower cutoff value c for a maximization problem is similar to adding this constraint to the objective function of the model: `obj >= c`.

Tip:

This parameter is effective only when the branch and bound algorithm is invoked, for example, in a mixed integer program (MIP). It does not have the expected effect when branch and bound is not invoked.

This parameter is not effective with the conflict refiner nor with FeasOpt. That is, neither of those tools can analyze an infeasibility introduced by this parameter. If you want to analyze such a condition, add an explicit objective constraint to your model instead before you invoke either of those tools.

Values

Any number; **default**: -1e+75.

number of cutting plane passes

Sets the upper limit on the number of cutting plane passes CPLEX performs when solving the root node of a MIP model.

Purpose

Number of cutting plane passes

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_MIP_Limits_CutPasses	CPX_PARAM_CUTPASS
C++	IloCplex::Param::MIP::Limits::CutPasses	CutPass (long)
Java	IloCplex.Param.MIP.Limits.CutPasses	CutPass (long)
.NET	Cplex.Param.MIP.Limits.CutPasses	CutPass (long)
OPL		cutpass
Python	parameters.mip.limits.cutpasses	mip.limits.cutpasses
MATLAB	Cplex.Param.mip.limits.cutpasses	mip.limits.cutpasses
Interactive	mip limits cutpasses	mip limits cutpasses
Identifier	2056	2056

Description

Sets the upper limit on the number of cutting plane passes CPLEX performs when solving the root node of a MIP model.

Table 14. Values

Value	Meaning
-1	None
0	Automatic: let CPLEX choose; **default**
Any positive integer	Number of passes to perform

cut factor row-multiplier limit

Limits the number of cuts that can be added.

Purpose

Cut factor to limit the number of cuts added in terms of original number of rows

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_MIP_Limits_CutsFactor	CPX_PARAM_CUTSFACTOR
C++	IloCplex::Param::MIP::Limits::CutsFactor	CutsFactor (double)
Java	IloCplex.Param.MIP.Limits.CutsFactor	CutsFactor (double)

API	Parameter Name	Name prior to V12.6.0
.NET	Cplex.Param.MIP.Limits.CutsFactor	CutsFactor (double)
OPL		cutsfactor
Python	parameters.mip.limits.cutsfactor	mip.limits.cutsfactor
MATLAB	Cplex.Param.mip.limits.cutsfactor	mip.limits.cutsfactor
Interactive	mip limits cutsfactor	mip limits cutsfactor
Identifier	2033	2033

Description

Limits the number of cuts that can be added. CPLEX limits the total number of rows in the model including **cuts added** to `CutsFactor` times the original number of rows.

The range of acceptable values of this parameter is `[-1.0, BIGREAL]` (where the symbol `BIGREAL` depends on your platform).

For **values between zero and one** (that is, in the range `[0.0, 1.0]`, CPLEX generates no cuts.

For **values strictly greater than 1.0 (one)**, CPLEX limits the number of rows in the model with cuts added. The limit on this total is the product of CutsFactor times the original number of rows.

If the model is presolved, the "original number of rows" is that of the presolved model.

The **default value** of this parameter is -1.0 (that is, negative one). The default value -1.0 means that CPLEX dynamically adjusts the limit on the number of cuts added to the model.

CPLEX regards **negative values** of this parameter as equivalent to the default value -1.0. That is, a negative value specifies no particular limit on the number of cuts. CPLEX computes and dynamically adjusts such a limit automatically.

Because cuts can be added and removed during the course of optimization, `CutsFactor` may not correspond directly to the number of cuts seen in the node log or in the summary table at the end of optimization.

Values

Any value in the range [-1.0, BIGREAL]; **default**: -1.0

upper cutoff

Sets the upper cutoff tolerance.

Purpose

Upper cutoff

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_MIP_Tolerances_UpperCutoff	CPX_PARAM_CUTUP
C++	IloCplex::Param::MIP::Tolerances::UpperCutoff	CutUp (double)

API	Parameter Name	Name prior to V12.6.0
Java	IloCplex.Param.MIP.Tolerances.UpperCutoff	CutUp (double)
.NET	Cplex.Param.MIP.Tolerances.UpperCutoff	CutUp (double)
OPL		cutup
Python	parameters.mip.tolerances.uppercutoff	mip.tolerances.uppercutoff
MATLAB	Cplex.Param.mip.tolerances.uppercutoff	mip.tolerances.uppercutoff
Interactive	mip tolerances uppercutoff	mip tolerances uppercutoff
Identifier	2007	2007

Description

Sets the upper cutoff tolerance. When the problem is a minimization problem, CPLEX cuts off or discards any solutions that are greater than the specified upper cutoff value. If the model has no solution with an objective value less than or equal to the cutoff value, CPLEX declares the model infeasible. In other words, setting an upper cutoff value c for a minimization problem is similar to adding this constraint to the objective function of the model: `obj <= c`.

Tip:

This parameter is effective only in the branch and bound algorithm, for example, in a mixed integer program (MIP). It does not have the expected effect when branch and bound is not invoked.

This parameter is not effective with the conflict refiner nor with FeasOpt. That is, neither of those tools can analyze an infeasibility introduced by this parameter. If you want to analyze such a condition, add an explicit objective constraint to your model instead before you invoke either of those tools.

Values

Any number; **default**: 1e+75.

data consistency checking and modeling assistance

Turns on or off data consistency checking and modeling assistance.

Purpose

Data consistency checking and modeling assistance switch

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_Read_DataCheck	CPX_PARAM_DATACHECK
C++	IloCplex::Param::Read::DataCheck	DataCheck
Java	IloCplex.Param.Read.DataCheck	DataCheck
.NET	Cplex.Param.Read.DataCheck	DataCheck
OPL		datacheck
Python	parameters.read.datacheck	read.datacheck
MATLAB	Cplex.Param.read.datacheck	read.datacheck
Interactive	read datacheck	read datacheck
Identifier	1056	1056

Description

Decides whether data should be checked for consistency and whether CPLEX should offer modeling assistance. When this parameter is set to 1 (one), the routines CPXcopy____, CPXread____, and CPXchg____ of the C API perform extensive checking of data in their array arguments, such as checking that indices are within range, that there are no duplicate entries, and that values are valid for the type of data or are valid numbers. This checking is useful for debugging applications. When this checking identifies trouble, you can gather more specific detail by calling one of the routines in `check.c`, as described in the *CPLEX User's Manual* in the topic Checking and debugging problem data.

Note: For most application programming interfaces (APIs) of CPLEX, the default value of this parameter is 0 (zero); that is, data checking is off by default. However, data checking in the CPLEX Python API is 1 (one), that is, on by default.

When the value of this parameter is set to level 2, CPX_DATACHECK_ASSIST, CPLEX turns on both data consistency checking and modeling assistance. At this level, CPLEX issues warnings at the start of the optimization about disproportionate values (too large, too small) in coefficients, bounds, and righthand sides (RHS). For all APIs, see the documentation of these warnings in the reference manual of the Callable Library (C API) in the topic Modeling information in the CPLEX Callable Library (C API) .

Table 15. Values

int	Symbol	Interactive	Meaning	
0	`CPX_DATACHECK_OFF`	no	Data checking off; do not check; **default**	
1	`CPX_DATACHECK_WARN`	yes	Data checking on; **default** in Python API	
2	CPX_DATACHECK_ASSIST	assist	Data checking on; modeling assistance on; warnings issued for both	

dependency switch

Decides whether to activate the dependency checker.

Purpose

Dependency switch

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_Preprocessing_Dependency	CPX_PARAM_DEPIND
C++	IloCplex::Param::Preprocessing::Dependency	DepInd (int)
Java	IloCplex.Param.Preprocessing.Dependency	DepInd (int)
.NET	Cplex.Param.Preprocessing.Dependency	DepInd (int)
OPL		depind
Python	parameters.preprocessing.dependency	preprocessing.dependency
MATLAB	Cplex.Param.preprocessing.dependency	preprocessing.dependency

API	Parameter Name	Name prior to V12.6.0
Interactive	preprocessing dependency	preprocessing dependency
Identifier	1008	1008

Description

Decides whether to activate the dependency checker. If on, the dependency checker searches for dependent rows during preprocessing. If off, dependent rows are not identified.

Table 16. Values

Value	Meaning
-1	Automatic: let CPLEX choose; **default**
0	Off: do not use dependency checker
1	Turn on only at the beginning of preprocessing
2	Turn on only at the end of preprocessing
3	Turn on at the beginning and at the end of preprocessing

deterministic time limit

Deterministic time limit

Purpose

Deterministic time limit

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_DetTimeLimit	CPX_PARAM_DETTILIM
C++	IloCplex::Param::DetTimeLimit	DetTiLim (double)
Java	IloCplex.Param.DetTimeLimit	DetTiLim (double)
.NET	Cplex.Param.DetTimeLimit	DetTiLim (double)
OPL		not available
Python	parameters.dettimelimit	dettimelimit
MATLAB	Cplex.Param.dettimelimit	dettimelimit
Interactive	dettimelimit	dettimelimit
Identifier	1127	1127

Description

Sets a time limit expressed in **ticks**, a unit to measure work done deterministically.

The length of a deterministic tick may vary by platform. Nevertheless, ticks are normally consistent measures for a given platform (combination of hardware and software) carrying the same load. In other words, the correspondence of ticks to clock time depends on the hardware, software, and the current load of the machine. For the same platform and same load, the ratio of ticks per second stays roughly constant, independent of the model solved. However, for very short optimization runs, the variation of this ratio is typically high.

CPLEX measures deterministic time only for work inside CPLEX. In other words, deterministic time does not include user algorithms implemented by means of callbacks. However, each application programming interface (API) of CPLEX offers routines or methods that access the deterministic clock and provide deterministic time stamps. In the APIs that support callbacks, you can use such deterministic time stamps in your application to mark time even from callbacks. For more detail about these deterministic time stamps, see the reference manual of the API that you use.

- In the Callable Library (C API), see the documentation of CPXgetdettime and CPXgetcallbackinfo.
- In the C++ API, see the documentation of IloCplex::CallbackI::getStartDetTime and IloCplex::CallbackI::getEndDetTime.
- In the Java API, see the documentation of IloCplex.Callback.getStartDetTime and IloCplex.Callback.getEndDetTime.
- In the .NET API, see the documentation of Cplex.ICallback.GetStartDetTime and GetEndDetTime.
- In the Python API, see the documentation of Callback.get_start_dettime and Callback.get_end_dettime.
- In the MATLAB connector, see the documentation of cplex.Solution.dettime.

At the end of optimization, the Interactive Optimizer displays the deterministic time spent to optimize the model as well as the ratio of ticks per second. For example, consider these lines, typical of output from the Interactive Optimizer:

```
MIP - Integer optimal solution:  Objective =  1.1580000000e+03
Solution time =    2.81 sec.  Iterations = 72793  Nodes = 2666
Deterministic time = 1996.47 ticks  (709.54 ticks/sec)
```

See also

For a nondeterministic time limit measured in seconds, see “optimizer time limit in seconds” on page 158 (CPX_PARAM_TILIM, TiLim).

For more detail about use of time limits, see the topic Timing interface in the *CPLEX User's Manual*.

Value

Any nonnegative double value in deterministic ticks; **default**:1.0E+75

MIP disjunctive cuts switch

Decides whether or not disjunctive cuts should be generated for the problem.

Purpose

MIP disjunctive cuts switch

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_MIP_Cuts_Disjunctive	CPX_PARAM_DISJCUTS
C++	IloCplex::Param::MIP::Cuts::Disjunctive	DisjCuts (int)
Java	IloCplex.Param.MIP.Cuts.Disjunctive	DisjCuts (int)
.NET	Cplex.Param.MIP.Cuts.Disjunctive	DisjCuts (int)
OPL		disjcuts
Python	parameters.mip.cuts.disjunctive	mip.cuts.disjunctive

API	Parameter Name	Name prior to V12.6.0
MATLAB	Cplex.Param.mip.cuts.disjunctive	mip.cuts.disjunctive
Interactive	mip cuts disjunctive	mip cuts disjunctive
Identifier	2053	2053

Description

Decides whether or not disjunctive cuts should be generated for the problem. Setting the value to 0 (zero), the default, specifies that the attempt to generate disjunctive cuts should continue only if it seems to be helping.

For a definition of a disjunctive cut, see the topic Disjunctive cuts in the general topic Cuts in the CPLEX User's Manual. The table Parameters for controlling cuts, also in the user's manual, includes links to the documentation of other parameters affecting other types of cuts.

Table 17. Values

Value	Meaning
-1	Do not generate disjunctive cuts
0	Automatic: let CPLEX choose; **default**
1	Generate disjunctive cuts moderately
2	Generate disjunctive cuts aggressively
3	Generate disjunctive cuts very aggressively

MIP dive strategy

Controls the MIP dive strategy.

Purpose

MIP dive strategy

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_MIP_Strategy_Dive	CPX_PARAM_DIVETYPE
C++	IloCplex::Param::MIP::Strategy::Dive	DiveType (int)
Java	IloCplex.Param.MIP.Strategy.Dive	DiveType (int)
.NET	Cplex.Param.MIP.Strategy.Dive	DiveType (int)
OPL		divetype
Python	parameters.mip.strategy.dive	mip.strategy.dive
MATLAB	Cplex.Param.mip.strategy.dive	mip.strategy.dive
Interactive	mip strategy dive	mip strategy dive
Identifier	2060	2060

Description

Controls the MIP dive strategy. The MIP traversal strategy occasionally performs probing dives, where it looks ahead at both children nodes before deciding which node to choose. The default (automatic) setting lets CPLEX choose when to perform a probing dive, 1 (one) directs CPLEX never to perform probing dives, 2 always to probe, 3 to spend more time exploring potential solutions that are similar to the current incumbent. Setting 2, always to probe, is helpful for finding integer solutions.

Table 18. Values

Value	Meaning
0	Automatic: let CPLEX choose; **default**
1	Traditional dive
2	Probing dive
3	Guided dive

dual simplex pricing algorithm

Decides the type of pricing applied in the dual simplex algorithm.

Purpose

Dual simplex pricing algorithm

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_Simplex_DGradient	CPX_PARAM_DPRIIND
C++	IloCplex::Param::Simplex::DGradient	DPriInd (int)
Java	IloCplex.Param.Simplex.DGradient	DPriInd (int)
.NET	Cplex.Param.Simplex.DGradient	DPriInd (int)
OPL		dpriind
Python	parameters.simplex.dgradient	simplex.dgradient
MATLAB	Cplex.Param.simplex.dgradient	simplex.dgradient
Interactive	simplex dgradient	simplex dgradient
Identifier	1009	1009

Description

Decides the type of pricing applied in the dual simplex algorithm. The default pricing (0) usually provides the fastest solution time, but many problems benefit from alternate settings.

Table 19. Values

Value	Symbol	Meaning
0	`CPX_DPRIIND_AUTO`	Automatic: let CPLEX choose; **default**
1	`CPX_DPRIIND_FULL`	Standard dual pricing
2	`CPX_DPRIIND_STEEP`	Steepest-edge pricing
3	`CPX_DPRIIND_FULL_STEEP`	Steepest-edge pricing in slack space
4	`CPX_DPRIIND_STEEPQSTART`	Steepest-edge pricing, unit initial norms
5	`CPX_DPRIIND_DEVEX`	devex pricing

See also

“candidate limit for generating Gomory fractional cuts” on page 73, “MIP Gomory fractional cuts switch” on page 73, “pass limit for generating Gomory fractional cuts” on page 74

type of cut limit

Sets a limit for each type of cut.

Purpose

Type of cut limit

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_MIP_Limits_EachCutLimit	CPX_PARAM_EACHCUTLIM
C++	IloCplex::Param::MIP::Limits::EachCutLimit	EachCutLim (int)
Java	IloCplex.Param.MIP.Limits.EachCutLimit	EachCutLim (int)
.NET	Cplex.Param.MIP.Limits.EachCutLimit	EachCutLim (int)
OPL		eachcutlim
Python	parameters.mip.limits.eachcutlimit	mip.limits.eachcutlimit
MATLAB	Cplex.Param.mip.limits.eachcutlimit	mip.limits.eachcutlimit
Interactive	mip limits eachcutlimit	mip limits eachcutlimit
Identifier	2102	2102

Description

This parameter allows you to set a uniform limit on the number of cuts of each type that CPLEX generates at every cut pass. That is, every time CPLEX attempts to generate new cuts, the number of generated cuts of each type will not exceed such a limit.

By default, the limit is the largest integer supported by a given platform; that is, there is no effective limit by default.

Tighter limits on the number of cuts of each type may benefit certain models. For example, a limit on each type of cut will prevent any one type of cut from being created in such large number that the limit on the total number of all types of cuts is reached before other types of cuts have an opportunity to be created.

A setting of 0 (zero) means no cuts.

This parameter does **not** influence the number of Gomory cuts. For means to control the number of Gomory cuts, see also the fractional cut parameters:
- "candidate limit for generating Gomory fractional cuts" on page 73: `CPX_PARAM_FRACCAND`, `FracCand`;
- "MIP Gomory fractional cuts switch" on page 73: `CPX_PARAM_FRACCUTS`, `FracCuts`;
- "pass limit for generating Gomory fractional cuts" on page 74: `CPX_PARAM_FRACPASS`, `FracPass`.

Table 20. Values

Value	Meaning
0	No cuts
Any positive number	Limit each type of cut
2100000000	**default**

absolute MIP gap tolerance

Sets an absolute tolerance on the gap between the best integer objective and the objective of the best node remaining.

Purpose

Absolute MIP gap tolerance

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_MIP_Tolerances_AbsMIPGap	CPX_PARAM_EPAGAP
C++	IloCplex::Param::MIP::Tolerances::AbsMIPGap	EpAGap (double)
Java	IloCplex.Param.MIP.Tolerances.AbsMIPGap	EpAGap (double)
.NET	Cplex.Param.MIP.Tolerances.AbsMIPGap	EpAGap (double)
OPL		epagap
Python	parameters.mip.tolerances.absmipgap	mip.tolerances.absmipgap
MATLAB	Cplex.Param.mip.tolerances.absmipgap	mip.tolerances.absmipgap
Interactive	mip tolerances absmipgap	mip tolerances absmipgap
Identifier	2008	2008

Description

Sets an absolute tolerance on the gap between the best integer objective and the objective of the best node remaining. When this difference falls below the value of this parameter, the mixed integer optimization is stopped.

Values

Any nonnegative number; **default**: 1e-06.

relative MIP gap tolerance

Sets a relative tolerance on the gap between the best integer objective and the objective of the best node remaining.

Purpose

Relative MIP gap tolerance

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_MIP_Tolerances_MIPGap	CPX_PARAM_EPGAP
C++	IloCplex::Param::MIP::Tolerances::MIPGap	EpGap (double)
Java	IloCplex.Param.MIP.Tolerances.MIPGap	EpGap (double)
.NET	Cplex.Param.MIP.Tolerances.MIPGap	EpGap (double)
OPL		epgap
Python	parameters.mip.tolerances.mipgap	mip.tolerances.mipgap
MATLAB	Cplex.Param.mip.tolerances.mipgap	mip.tolerances.mipgap
Interactive	mip tolerances mipgap	mip tolerances mipgap
Identifier	2009	2009

Description

When the value

`|bestbound-bestinteger|/(1e-10+|bestinteger|)`

falls below the value of this parameter, the mixed integer optimization is stopped.

For example, to instruct CPLEX to stop as soon as it has found a feasible integer solution proved to be within five percent of optimal, set the relative MIP gap tolerance to 0.05.

Values

Any number from 0.0 to 1.0; **default**: 1e-04.

integrality tolerance

Specifies the amount by which an integer variable can be different from an integer and still be considered feasible.

Purpose

Integrality tolerance

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_MIP_Tolerances_Integrality	CPX_PARAM_EPINT
C++	IloCplex::Param::MIP::Tolerances::Integrality	EpInt (double)
Java	IloCplex.Param.MIP.Tolerances.Integrality	EpInt (double)
.NET	Cplex.Param.MIP.Tolerances.Integrality	EpInt (double)
OPL		epint
Python	parameters.mip.tolerances.integrality	mip.tolerances.integrality
MATLAB Cplex class API	Cplex.Param.mip.tolerances.integrality	mip.tolerances.integrality
MATLAB	CPLEX Toolbox compatible	mip.tolerances.integrality
MATLAB	Optimization Toolbox compatible	TolXInteger
Interactive	mip tolerances integrality	mip tolerances integrality
Identifier	2010	2010

Description

Specifies the amount by which an integer variable can be different from an integer and still be considered feasible.

A value of zero is permitted, and the optimizer will attempt to meet this tolerance.

However, in some models, computer round-off may still result in small, nonzero deviations from integrality. If any of these deviations exceed the value of this parameter, or exceed 1e-10 in the case where this parameter has been set to a value less than that, a solution status of `CPX_STAT_OPTIMAL_INFEAS` will be returned instead of the usual `CPX_STAT_OPTIMAL`.

Tip: This parameter sets the amount by which a computed solution value for an integer variable can violate integrality; it does **not** specify an amount by which CPLEX relaxes integrality.

Values

Any number from 0.0 to 0.5; **default**: 1e-05.

epsilon (degree of tolerance) used in linearization

Sets the epsilon (degree of tolerance) used in linearization in the object-oriented APIs.

Purpose

Epsilon used in linearization

API	Parameter Name	Name prior to V12.6.0
C		
C++	IloCplex::Param::MIP::Tolerances::Linearization	EpLin (double)
Java	IloCplex::Param::MIP::Tolerances::Linearization	EpLin (double)
.NET	IloCplex::Param::MIP::Tolerances::Linearization	EpLin (double)
OPL		
Python		
MATLAB		
Interactive		
Identifier		2068

Description

Sets the epsilon (degree of tolerance) used in linearization in the object-oriented APIs.

Not applicable in the C API.

Not applicable in the Python API.

Not applicable in the CPLEX connector for MATLAB.

Not available in the Interactive Optimizer.

This parameter controls how strict inequalities are managed during linearization. In other words, it provides an epsilon for deciding when two values are not equal during linearization. For example, when x is a numeric variable (that is, an instance of `IloNumVar`),

`x < a`

becomes

`x <= a-eplin .`

Similarly, `x!=a`

becomes

```
{(x < a) || (x > a)}
```

which is linearized automatically for you in the object-oriented APIs as

```
{( x <= a-eplin) || (x >= a+eplin)} .
```

Exercise caution in changing this parameter from its default value: the smaller the epsilon, the more numerically unstable the model will tend to become. If you are not getting an expected solution for an object-oriented model that uses linearization, it might be that this solution is cut off because of the relatively high `EpLin` value. In such a case, carefully try reducing it.

Values

Any positive value greater than zero; **default**: 1e-3.

Markowitz tolerance

Influences pivot selection during basis factoring.

Purpose

Markowitz tolerance

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_Simplex_Tolerances_Markowitz	CPX_PARAM_EPMRK
C++	IloCplex::Param::Simplex::Tolerances::Markowitz	EpMrk (double)
Java	IloCplex.Param.Simplex.Tolerances.Markowitz	EpMrk (double)
.NET	Cplex.Param.Simplex.Tolerances.Markowitz	EpMrk (double)
OPL		epmrk
Python	parameters.simplex.tolerances.markowitz	simplex.tolerances.markowitz
MATLAB	Cplex.Param.simplex.tolerances.markowitz	simplex.tolerances.markowitz
Interactive	simplex tolerances markowitz	simplex tolerances markowitz
Identifier	1013	1013

Description

Influences pivot selection during basis factoring. Increasing the Markowitz threshold may improve the numerical properties of the solution.

Values

Any number from 0.0001 to 0.99999; **default**: 0.01.

optimality tolerance

Influences the reduced-cost tolerance for optimality.

Purpose

Optimality tolerance

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_Simplex_Tolerances_Optimality	CPX_PARAM_EPOPT
C++	IloCplex::Param::Simplex::Tolerances::Optimality	EpOpt (double)
Java	IloCplex.Param.Simplex.Tolerances.Optimality	EpOpt (double)
.NET	Cplex.Param.Simplex.Tolerances.Optimality	EpOpt (double)
OPL		epopt
Python	parameters.simplex.tolerances.optimality	simplex.tolerances.optimality
MATLAB Cplex class API	Cplex.Param.simplex.tolerances.optimality	simplex.tolerances.optimality
MATLAB CPLEX Toolbox compatible	simplex.tolerances.optimality	simplex.tolerances.optimality
MATLAB Optimization Toolbox compatible	TolFun and TolRLPFun	TolFun and TolRLPFun
Interactive	simplex tolerances optimality	simplex tolerances optimality
Identifier	1014	1014

Description

Influences the reduced-cost tolerance for optimality. This parameter governs how closely CPLEX must approach the theoretically optimal solution.

The simplex algorithm halts when it has found a basic feasible solution with all reduced costs nonnegative. CPLEX uses this optimality tolerance to make the decision of whether or not a given reduced cost should be considered nonnegative. CPLEX considers "nonnegative" a negative reduced cost having absolute value less than the optimality tolerance. For example, if your optimality tolerance is set to 1e-6, then CPLEX considers a reduced cost of -1e-9 as nonnegative for the purpose of deciding whether the solution is optimal.

Values

Any number from 1e-9 to 1e-1; **default**: 1e-06.

perturbation constant

Sets the amount by which CPLEX perturbs the upper and lower bounds or objective coefficients on the variables when a problem is perturbed in the simplex algorithm.

Purpose

Perturbation constant

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_Simplex_Perturbation_Constant	CPX_PARAM_EPPER
C++	IloCplex::Param::Simplex::Perturbation::Constant	EpPer (double)

API	Parameter Name	Name prior to V12.6.0
Java	IloCplex.Param.Simplex.Perturbation.Constant	EpPer (double)
.NET	Cplex.Param.Simplex.Perturbation.Constant	EpPer (double)
OPL		epper
Python	parameters.simplex.perturbation.constant	simplex.perturbation.constant
MATLAB	Cplex.Param.simplex.perturbation.constant	simplex.perturbation.constant
Interactive	simplex perturbationlimit no/yes C	simplex perturbationlimit no/yes C
Identifier	1015	1015

Description

Sets the amount by which CPLEX perturbs the upper and lower bounds or objective coefficients on the variables when a problem is perturbed in the simplex algorithm. This parameter can be set to a smaller value if the default value creates too large a change in the problem.

In the **Interactive Optimizer**, the command

```
set simplex perturbationlimit
```

accepts two arguments and actually sets two parameters simultaneously. The first argument is a switch or indicator; its value is yes to turn on perturbation or no to turn off perturbation. See the parameter "simplex perturbation switch" on page 110 for more detail about this effect. The second argument is a constant value to set an amount of perturbation.

Values

Any positive number greater than or equal to 1e-8; **default**: 1e-6.

relaxation for FeasOpt

Controls the amount of relaxation for the routine `CPXfeasopt` in the C API or for the method `feasOpt` in the object-oriented APIs.

Purpose

Relaxation for `feasOpt`

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_Feasopt_Tolerance	CPX_PARAM_EPRELAX
C++	IloCplex::Param::Feasopt::Tolerance	EpRelax (double)
Java	IloCplex.Param.Feasopt.Tolerance	EpRelax (double)
.NET	Cplex.Param.Feasopt.Tolerance	EpRelax (double)
OPL		eprelax
Python	parameters.feasopt.tolerance	feasopt.tolerance
MATLAB	Cplex.Param.feasopt.tolerance	feasopt.tolerance
Interactive	feasopt tolerance	feasopt tolerance
Identifier	2073	2073

Description

Controls the amount of relaxation for the routine `CPXfeasopt` in the C API or for the method `feasOpt` in the object-oriented APIs.

In the case of a MIP, it serves the purpose of the absolute gap for the `feasOpt` model in Phase I (the phase to minimize relaxation).

Using this parameter, you can implement other stopping criteria as well. To do so, first call `feasOpt` with the stopping criteria that you prefer; then set this parameter to the resulting objective of the Phase I model; unset the other stopping criteria, and call `feasOpt` again. Since the solution from the first call already matches this parameter, Phase I will terminate immediately in this second call to `feasOpt`, and Phase II will start.

In the case of an LP, this parameter controls the lower objective limit for Phase I of `feasOpt` and is thus relevant only when the primal optimizer is in use.

Values

Any nonnegative value; **default**: 1e-6.

See also

"lower objective value limit" on page 105

feasibility tolerance

Specifies the feasibility tolerance, that is, the degree to which the basic variables of a model may violate their bounds.

Purpose

Feasibility tolerance

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_Simplex_Tolerances_Feasibility	CPX_PARAM_EPRHS
C++	IloCplex::Param::Simplex::Tolerances::Feasibility	EpRHS (double)
Java	IloCplex.Param.Simplex.Tolerances.Feasibility	EpRHS (double)
.NET	Cplex.Param.Simplex.Tolerances.Feasibility	EpRHS (double)
OPL		eprhs
Python	parameters.simplex.tolerances.feasibility	simplex.tolerances.feasibility
MATLAB	Cplex.Param.simplex.tolerances.feasibility	simplex.tolerances.feasibility
Interactive	simplex tolerances feasibility	simplex tolerances feasibility
Identifier	1016	1016

Description

Specifies the feasibility tolerance, that is, the degree to which values of the basic variables calculated by the simplex method may violate their bounds. CPLEX® does **not** use this tolerance to relax the variable bounds nor to relax right hand side values. This parameter specifies an allowable violation. Feasibility influences the selection of an optimal basis and can be reset to a higher value when a problem is having difficulty maintaining feasibility during optimization. You can also lower this tolerance after finding an optimal solution if there is any doubt that the solution is truly optimal. If the feasibility tolerance is set too low, CPLEX may falsely conclude that a problem is infeasible. If you encounter reports of infeasibility during Phase II of the optimization, a small adjustment in the feasibility tolerance may improve performance.

Values

Any number from 1e-9 to 1e-1; **default**: 1e-06.

mode of FeasOpt

Decides how FeasOpt measures the relaxation when finding a minimal relaxation in an infeasible model.

Purpose

Mode of FeasOpt

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_Feasopt_Mode	CPX_PARAM_FEASOPTMODE
C++	IloCplex::Param::Feasopt::Mode	FeasOptMode (int)
Java	IloCplex.Param.Feasopt.Mode	FeasOptMode (int)
.NET	Cplex.Param.Feasopt.Mode	FeasOptMode (int)
OPL		feasoptmode
Python	parameters.feasopt.mode	feasopt.mode
MATLAB	Cplex.Param.feasopt.mode	feasopt.mode
Interactive	feasopt mode	feasopt mode
Identifier	1084	1084

Description

Decides how FeasOpt measures the relaxation when finding a minimal relaxation in an infeasible model. FeasOpt works in two phases. In its first phase, it attempts to minimize its relaxation of the infeasible model. That is, it attempts to find a feasible solution that requires minimal change. In its second phase, it finds an optimal solution among those that require only as much relaxation as it found necessary in the first phase. Values of this parameter indicate two aspects to CPLEX:

- whether to stop in phase one or continue to phase two and
- how to measure the relaxation, according to one of the following criteria:
  - as a sum of required relaxations;
  - as the number of constraints and bounds required to be relaxed;
  - as a sum of the squares of required relaxations.

Table 21. Values

Value	Symbol	Symbol (C API)	Meaning
0	`MinSum`	`CPX_FEASOPT_MIN_SUM`	Minimize the sum of all required relaxations in first phase only; **default**
1	`OptSum`	`CPX_FEASOPT_OPT_SUM`	Minimize the sum of all required relaxations in first phase and execute second phase to find optimum among minimal relaxations
2	`MinInf`	`CPX_FEASOPT_MIN_INF`	Minimize the number of constraints and bounds requiring relaxation in first phase only

Table 21. Values (continued)

Value	Symbol	Symbol (C API)	Meaning
3	`OptInf`	`CPX_FEASOPT_OPT_INF`	Minimize the number of constraints and bounds requiring relaxation in first phase and execute second phase to find optimum among minimal relaxations
4	`MinQuad`	`CPX_FEASOPT_MIN_QUAD`	Minimize the sum of squares of required relaxations in first phase only
5	`OptQuad`	`CPX_FEASOPT_OPT_QUAD`	Minimize the sum of squares of required relaxations in first phase and execute second phase to find optimum among minimal relaxations

file encoding switch

file encoding switch

Purpose

File encoding switch

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_Read_FileEncoding	CPX_PARAM_FILEENCODING
C++	IloCplex::Param::Read::FileEncoding	FileEncoding (string)
Java	IloCplex.Param.Read.FileEncoding	FileEncoding (string)
.NET	Cplex.Param.Read.FileEncoding	FileEncoding (string)
OPL		not available
Python	parameters.read.fileencoding	read.fileencoding
MATLAB	Cplex.Param.read.fileencoding	read.fileencoding
Interactive	read fileencoding	read fileencoding
Identifier	1129	1129

Description

Specifies which encoding (also known as the code page) that CPLEX uses for reading and writing files. This parameter accepts a **string**, such as `UTF-8`, `ISO-8859-1`, `US-ASCII`, and so forth, specifying the user's choice of encoding for reading and writing files.

Note:

This parameter has no effect on IBM CPLEX Optimizer for z/OS, where only EBCDIC IBM-1047 encoding is available.

The **default** value of this parameter depends on the CPLEX component.

- In the CPLEX connector for **MATLAB**, the default value of the file encoding parameter is the empty string (" ") for consistency with MATLAB conventions.
- In the **Python API**, the default value of the file encoding parameter is the empty string (" ") for consistency with Python conventions.
- In other APIs of CPLEX, such as the **C, C++, Java, .NET API**, the default value of the file encoding parameter is the string `ISO-8859-1` (also known as Latin-1).

The encoding ISO-8859-1 is a superset of the familiar ASCII encoding, so it supports many widely used character sets. However, this default encoding cannot represent multi-byte character sets, such as Chinese, Japanese, or Korean characters, for example. If you want CPLEX to represent a character set that requires multiple bytes per character, then a better choice for the value of this parameter is UTF-8. The encoding UTF-8 is compatible with ASCII encoding; it represents every character in Unicode; it does not include a NULL byte in a valid character; it does not require specification of big-end or little-end byte order; it does not require a byte-order mark.

When you change the value of this parameter, you also need to verify that the "API string encoding switch" on page 20 (CPX_PARAM_APIENCODING, APIEncoding) is compatible. The encoding specified by the API encoding parameter must be a subset of the encoding specified by the file encoding parameter. For example, if the API encoding parameter specifies US-ASCII, then UTF-8 is a reasonable choice for the file encoding parameter because the code page US-ASCII is a subset of UTF-8.

For a complete list of valid strings that are the name of an encoding (that is, the name of a code page), consult the web site of a standards organization such as:
- A brief introduction to code pages
- ICU: International Components for Unicode
- International Components for Unicode at IBM

In situations where CPLEX encounters a string, such as content in a file, that is not compatible with the specified encoding, the behavior is not defined. Because of the incompatibility, CPLEX silently converts the string to an inappropriate character of the specified encoding, or CPLEX raises the error 1235 CPXERR_ENCODING_CONVERSION.

Values

valid string for the name of an encoding (code page); **default**: ISO-8859-1 or the empty string (" ")

See also

"API string encoding switch" on page 20

MIP flow cover cuts switch

Decides whether or not to generate flow cover cuts for the problem.

Purpose

MIP flow cover cuts switch

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_MIP_Cuts_FlowCovers	CPX_PARAM_FLOWCOVERS
C++	IloCplex::Param::MIP::Cuts::FlowCovers	FlowCovers (int)
Java	IloCplex.Param.MIP.Cuts.FlowCovers	FlowCovers (int)
.NET	Cplex.Param.MIP.Cuts.FlowCovers	FlowCovers (int)
OPL		flowcovers
Python	parameters.mip.cuts.flowcovers	mip.cuts.flowcovers
MATLAB	Cplex.Param.mip.cuts.flowcovers	mip.cuts.flowcovers
Interactive	mip cuts flowcovers	mip cuts flowcovers

API	Parameter Name	Name prior to V12.6.0
Identifier	2040	2040

Description

Decides whether or not to generate flow cover cuts for the problem. Setting the value to 0 (zero), the default, indicates that the attempt to generate flow cover cuts should continue only if it seems to be helping.

For a definition of a flow cover cut, see the topic Flow cover cuts in the general topic Cuts in the *CPLEX User's Manual*. The table Parameters for controlling cuts, also in the user's manual, includes links to the documentation of other parameters affecting other types of cuts.

Table 22. Values

Value	Meaning
-1	Do not generate flow cover cuts
0	Automatic: let CPLEX choose; **default**
1	Generate flow cover cuts moderately
2	Generate flow cover cuts aggressively

MIP flow path cut switch

Decides whether or not flow path cuts should be generated for the problem.

Purpose

MIP flow path cut switch

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_MIP_Cuts_PathCut	CPX_PARAM_FLOWPATHS
C++	IloCplex::Param::MIP::Cuts::PathCut	FlowPaths (int)
Java	IloCplex.Param.MIP.Cuts.PathCut	FlowPaths (int)
.NET	Cplex.Param.MIP.Cuts.PathCut	FlowPaths (int)
OPL		flowpaths
Python	parameters.mip.cuts.pathcut	mip.cuts.pathcut
MATLAB	Cplex.Param.mip.cuts.pathcut	mip.cuts.pathcut
Interactive	mip cuts pathcut	mip cuts pathcut
Identifier	2051	2051

Description

Decides whether or not flow path cuts should be generated for the problem. Setting the value to 0 (zero), the default, indicates that the attempt to generate flow path cuts should continue only if it seems to be helping.

For a definition of a flow path cut, see the topic Flow path cuts in the general topic Cuts in the *CPLEX User's Manual*. The table Parameters for controlling cuts, also in the user's manual, includes links to the documentation of other parameters affecting other types of cuts.

Table 23. Values

Value	Meaning
-1	Do not generate flow path cuts
0	Automatic: let CPLEX choose; **default**
1	Generate flow path cuts moderately
2	Generate flow path cuts aggressively

feasibility pump switch

Turns on or off the feasibility pump heuristic for mixed integer programming (MIP) models.

Purpose

Feasibility pump switch

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_MIP_Strategy_FPHeur	CPX_PARAM_FPHEUR
C++	IloCplex::Param::MIP::Strategy::FPHeur	FPHeur (int)
Java	IloCplex.Param.MIP.Strategy.FPHeur	FPHeur (int)
.NET	Cplex.Param.MIP.Strategy.FPHeur	FPHeur (int)
OPL		fpheur
Python	parameters.mip.strategy.fpheur	mip.strategy.fpheur
MATLAB	Cplex.Param.mip.strategy.fpheur	mip.strategy.fpheur
Interactive	mip strategy fpheur	mip strategy fpheur
Identifier	2098	2098

Description

Turns on or off the feasibility pump heuristic for mixed integer programming (MIP) models.

At the default setting 0 (zero), CPLEX automatically chooses whether or not to apply the feasibility pump heuristic on the basis of characteristics of the model. The feasibility pump does **not** apply to models of the type mixed integer quadratically constrained programs (MIQCP).

To turn off the feasibility pump heuristic, set the parameter to -1 (minus one).

To turn on the feasibility pump heuristic, set the parameter to 1 (one) or 2.

If the parameter is set to 1 (one), the feasibility pump tries to find a feasible solution without taking the objective function into account.

If the parameter is set to 2, the heuristic usually finds solutions of better objective value, but is more likely to fail to find a feasible solution.

For more detail about the feasibility pump heuristic, see research by Fischetti, Glover, and Lodi (2003, 2005), by Bertacco, Fischetti, and Lodi (2005), and by Achterberg and Berthold (2005, 2007).

Table 24. Values

Value	Meaning
-1	Do not apply the feasibility pump heuristic
0	Automatic: let CPLEX choose; **default**
1	Apply the feasibility pump heuristic with an emphasis on finding a feasible solution
2	Apply the feasibility pump heuristic with an emphasis on finding a feasible solution with a good objective value

candidate limit for generating Gomory fractional cuts

Limits the number of candidate variables for generating Gomory fractional cuts.

Purpose

Candidate limit for generating Gomory fractional cuts

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_MIP_Limits_GomoryCand	CPX_PARAM_FRACCAND
C++	IloCplex::Param::MIP::Limits::GomoryCand	FracCand (int)
Java	IloCplex.Param.MIP.Limits.GomoryCand	FracCand (int)
.NET	Cplex.Param.MIP.Limits.GomoryCand	FracCand (int)
OPL		fraccand
Python	parameters.mip.limits.gomorycand	mip.limits.gomorycand
MATLAB	Cplex.Param.mip.limits.gomorycand	mip.limits.gomorycand
Interactive	mip limits gomorycand	mip limits gomorycand
Identifier	2048	2048

Description

Limits the number of candidate variables for generating Gomory fractional cuts.

Values

Any positive integer; **default**: 200.

MIP Gomory fractional cuts switch

Decides whether or not Gomory fractional cuts should be generated for the problem.

Purpose

MIP Gomory fractional cuts switch

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_MIP_Cuts_Gomory	CPX_PARAM_FRACCUTS
C++	IloCplex::Param::MIP::Cuts::Gomory	FracCuts (int)
Java	IloCplex.Param.MIP.Cuts.Gomory	FracCuts (int)
.NET	Cplex.Param.MIP.Cuts.Gomory	FracCuts (int)
OPL		fraccuts

API	Parameter Name	Name prior to V12.6.0
Python	parameters.mip.cuts.gomory	mip.cuts.gomory
MATLAB	Cplex.Param.mip.cuts.gomory	mip.cuts.gomory
Interactive	mip cuts gomory	mip cuts gomory
Identifier	2049	2049

Description

Decides whether or not Gomory fractional cuts should be generated for the problem. Setting the value to 0 (zero), the default, indicates that the attempt to generate Gomory fractional cuts should continue only if it seems to be helping.

For a definition of a Gomory fractional cut, see the topic Gomory fractional cuts in the general topic Cuts in the *CPLEX User's Manual*. The table Parameters for controlling cuts, also in the user's manual, includes links to the documentation of other parameters affecting other types of cuts.

Table 25. Values

Value	Meaning
-1	Do not generate Gomory fractional cuts
0	Automatic: let CPLEX choose; **default**
1	Generate Gomory fractional cuts moderately
2	Generate Gomory fractional cuts aggressively

pass limit for generating Gomory fractional cuts

Limits the number of passes for generating Gomory fractional cuts.

Purpose

Pass limit for generating Gomory fractional cuts

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_MIP_Limits_GomoryPass	CPX_PARAM_FRACPASS
C++	IloCplex::Param::MIP::Limits::GomoryPass	FracPass (long)
Java	IloCplex.Param.MIP.Limits.GomoryPass	FracPass (long)
.NET	Cplex.Param.MIP.Limits.GomoryPass	FracPass (long)
OPL		fracpass
Python	parameters.mip.limits.gomorypass	mip.limits.gomorypass
MATLAB	Cplex.Param.mip.limits.gomorypass	mip.limits.gomorypass
Interactive	mip limits gomorypass	mip limits gomorypass
Identifier	2050	2050

Description

Limits the number of passes for generating Gomory fractional cuts. At the default setting of 0 (zero), CPLEX decides the number of passes to make. The parameter is ignored if the Gomory fractional cut parameter ("MIP Gomory fractional cuts switch" on page 73: `CPX_PARAM_FRACCUTS`, `FracCuts`) is set to a nonzero value.

Table 26. Values

Value	Meaning
0	Automatic: let CPLEX choose; **default**
Any positive integer	Number of passes to generate Gomory fractional cuts

MIP GUB cuts switch

Decides whether or not to generate GUB cuts for the problem.

Purpose

MIP GUB cuts switch

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_MIP_Cuts_GUBCovers	CPX_PARAM_GUBCOVERS
C++	IloCplex::Param::MIP::Cuts::GUBCovers	GUBCovers (int)
Java	IloCplex.Param.MIP.Cuts.GUBCovers	GUBCovers (int)
.NET	Cplex.Param.MIP.Cuts.GUBCovers	GUBCovers (int)
OPL		gubcovers
Python	parameters.mip.cuts.gubcovers	mip.cuts.gubcovers
MATLAB	Cplex.Param.mip.cuts.gubcovers	mip.cuts.gubcovers
Interactive	mip cuts gubcovers	mip cuts gubcovers
Identifier	2044	2044

Description

Decides whether or not to generate generalized upper bound (GUB) cover cuts for the problem. Setting the value to 0 (zero), the default, indicates that the attempt to generate GUB cuts should continue only if it seems to be helping.

For a definition of a GUB cover cut, see the topic Generalized upper bound (GUB) cover cutsin the general topic Cuts in the *CPLEX User's Manual*. The table Parameters for controlling cuts, also in the user's manual, includes links to the documentation of other parameters affecting other types of cuts.

Table 27. Values

Value	Meaning
-1	Do not generate GUB cuts
0	Automatic: let CPLEX choose; **default**
1	Generate GUB cuts moderately
2	Generate GUB cuts aggressively

MIP heuristic frequency

Decides how often to apply the periodic heuristic.

Purpose

MIP heuristic frequency

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_MIP_Strategy_HeuristicFreq	CPX_PARAM_HEURFREQ
C++	IloCplex::Param::MIP::Strategy::HeuristicFreq	HeurFreq (long)
Java	IloCplex.Param.MIP.Strategy.HeuristicFreq	HeurFreq (long)
.NET	Cplex.Param.MIP.Strategy.HeuristicFreq	HeurFreq (long)
OPL		heurfreq
Python	parameters.mip.strategy.heuristicfreq	mip.strategy.heuristicfreq
MATLAB	Cplex.Param.mip.strategy.heuristicfreq	mip.strategy.heuristicfreq
Interactive	mip strategy heuristicfreq	mip strategy heuristicfreq
Identifier	2031	2031

Description

Decides how often to apply the periodic heuristic. Setting the value to -1 turns off the periodic heuristic. Setting the value to 0 (zero), the default, applies the periodic heuristic at an interval chosen automatically. Setting the value to a positive number applies the heuristic at the requested node interval. For example, setting this parameter to 20 dictates that the heuristic be called at node 0, 20, 40, 60, etc.

For an introduction to heuristics in CPLEX, see the topic Applying heuristicsamong the topics in Tuning performance features of the mixed integer optimizerin the *CPLEX User's Manual*. For more about other heuristics, see the topics in Heuristics (also in the *CPLEX User's Manual*). There, the topic Node heuristic refers specifically to this parameter.

Table 28. Values

Value	Meaning
-1	None
0	Automatic: let CPLEX choose; **default**
Any positive integer	Apply the periodic heuristic at this frequency

MIP globally valid implied bound cuts switch

Decides whether or not to generate globally valid implied bound cuts for the model.

Purpose

MIP globally valid implied bound cuts switch

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_MIP_Cuts_Implied	CPX_PARAM_IMPLBD
C++	IloCplex::Param::MIP::Cuts::Implied	ImplBd (int)
Java	IloCplex.Param.MIP.Cuts.Implied	ImplBd (int)
.NET	Cplex.Param.MIP.Cuts.Implied	ImplBd (int)
OPL		implbd
Python	parameters.mip.cuts.implied	mip.cuts.implied
MATLAB	Cplex.Param.mip.cuts.implied	mip.cuts.implied

API	Parameter Name	Name prior to V12.6.0
Interactive	mip cuts implied	mip cuts implied
Identifier	2041	2041

Description

Decides whether or not to generate globally valid implied bound cuts for the model. Setting the value to 0 (zero), the default, specifies that the attempt to generate globally valid implied bound cuts should continue only if it seems to be helping.

For a definition of an implied bound cut, see the topic Implied bound cuts: global and local in the general topic Cuts in the *CPLEX User's Manual*. The table Parameters for controlling cuts, also in the user's manual, includes links to the documentation of other parameters affecting other types of cuts.

Table 29. Values

Value	Meaning
-1	Do not generate implied bound cuts
0	Automatic: let CPLEX choose; **default**
1	Generate implied bound cuts moderately
2	Generate implied bound cuts aggressively

See also

"MIP locally valid implied bound cuts switch"

MIP locally valid implied bound cuts switch

Decides whether or not to generate locally valid implied bound cuts for the problem.

Purpose

MIP locally valid implied bound cuts switch

API	Parameter Name
C	CPXPARAM_MIP_Cuts_LocalImplied
C++	IloCplex::Param::MIP::Cuts::LocalImplied
Java	IloCplex.Param.MIP.Cuts.LocalImplied
.NET	Cplex.Param.MIP.Cuts.LocalImplied
OPL	localimplied
Python	parameters.mip.cuts.localimplied
MATLAB	Cplex.Param.mip.cuts.localimplied
Interactive	mip cuts localimplied
Identifier	2181

Description

Instructs CPLEX whether or not to generate locally valid implied bound cuts for the model. The default value 0 (zero) specifies that the attempt to generate locally valid implied bound cuts should continue only if it seems to be helping.

For a definition of locally valid implied bound cuts, see the topic Implied bound cuts: global and local in the *CPLEX User's Manual*.

Table 30. Values

Value	Meaning
-1	Do not generate locally valid implied bound cuts
0	Automatic: let CPLEX choose; **default**
1	Generate locally valid implied bound cuts moderately
2	Generate locally valid implied bound cuts aggressively
3	Generate locally valid implied bound cuts very aggressively

See also

"MIP globally valid implied bound cuts switch" on page 76

MIP integer solution-file switch and prefix

MIP integer solution file switch and filename prefix.

Purpose

MIP integer solution-file switch and filename prefix.

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_Output_IntSolFilePrefix	CPX_PARAM_INTSOLFILEPREFIX
C++	IloCplex::Param::Output::IntSolFilePrefix	IntSolFilePrefix (string)
Java	IloCplex.Param.Output.IntSolFilePrefix	IntSolFilePrefix (string)
.NET	Cplex.Param.Output.IntSolFilePrefix	IntSolFilePrefix (string)
OPL		not available
Python	parameters.output.intsolfileprefix	output.intsolfileprefix
MATLAB	Cplex.Param.output.intsolfileprefix	output.intsolfileprefix
Interactive	output intsolfileprefix	output intsolfileprefix
Identifier	2143	2143

Description

Decides whether CPLEX writes the current MIP incumbent integer solution to a file and (if so) sets a prefix for the name of that file.

By default, the value of this parameter is the empty string, and file-writing is turned off. When this parameter is set to a non empty string, CPLEX writes each new incumbent to a file at the time the MIP integer solution is found.

In addition to switching on the writing of a file of solutions, this parameter also specifies the prefix of the name of the file to use. The prefix can contain a relative or absolute path. If the prefix does not contain a relative or absolute path, CPLEX writes to a file in the "directory for working files" on page 167.

The complete file name of the file that CPLEX writes is `PREFIX-NNNNN.sol`, where:

- `PREFIX` is the prefix specified by this parameter;
- `NNNNN` is the sequence number of the solution; the sequence starts at 00001;
- `sol` represents the solution file format, documented in the topic SOL file format: solution files in the reference manual, *File formats supported by CPLEX*.

Note:

Existing files of the same name will be overwritten.

If the specified file cannot be written (for example, in case of lack of disk space, or no write access to the specified location), optimization stops with an error status code.

This parameter accepts a `string` as its value. If you change either the "API string encoding switch" on page 20 or the "file encoding switch" on page 69 from their default value to a multi-byte encoding where a `NULL` byte can occur within the encoding of a character, you must take into account the issues documented in the topic Selecting an encoding in the *CPLEX User's Manual*. Especially consider the possibility that a `NULL` byte occurring in the encoding of a character can inadvertently signal the termination of a string, such as a filename or directory path, and thus provoke surprising or incorrect results.

Values

valid string for the prefix of a file name; **default**: " " (the empty string; that is, the switch is off)

See also

"directory for working files" on page 167

MIP integer solution limit

Sets the number of MIP solutions to be found before stopping.

Purpose

MIP integer solution limit

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_MIP_Limits_Solutions	CPX_PARAM_INTSOLLIM
C++	IloCplex::Param::MIP::Limits::Solutions	IntSolLim (long)
Java	IloCplex.Param.MIP.Limits.Solutions	IntSolLim (long)
.NET	Cplex.Param.MIP.Limits.Solutions	IntSolLim (long)
OPL		intsollim
Python	parameters.mip.limits.solutions	mip.limits.solutions
MATLAB	Cplex.Param.mip.limits.solutions	mip.limits.solutions
Interactive	mip limits solutions	mip limits solutions
Identifier	2015	2015

Description

Sets the number of MIP solutions to be found before stopping.

This integer solution limit does **not** apply to the populate procedure, which generates solutions to store in the solution pool. For a limit on the number of solutions generated by populate, see the populate limit parameter: "maximum number of solutions generated for solution pool by populate" on page 117.

Values

Any positive integer strictly greater than zero; zero is **not** allowed; **default**: 9223372036800000000.

See also

"maximum number of solutions generated for solution pool by populate" on page 117

simplex maximum iteration limit

Sets the maximum number of simplex iterations to be performed before the algorithm terminates without reaching optimality.

Purpose

Simplex maximum iteration limit

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_Simplex_Limits_Iterations	CPX_PARAM_ITLIM
C++	IloCplex::Param::Simplex::Limits::Iterations	ItLim (long)
Java	IloCplex.Param.Simplex.Limits.Iterations	ItLim (long)
.NET	Cplex.Param.Simplex.Limits.Iterations	ItLim (long)
OPL		itlim
Python	parameters.simplex.limits.iterations	simplex.limits.iterations
MATLAB Cplex class API	Cplex.Param.simplex.limits.iterations	simplex.limits.iterations
MATLAB CPLEX Toolbox compatibility	simplex.limits.iterations	simplex.limits.iterations
MATLAB Optimization Toolbox compatibility	MaxIter	MaxIter
Interactive	simplex limits iterations	simplex limits iterations
Identifier	1020	1020

Description

Sets the maximum number of simplex iterations to be performed before the algorithm terminates without reaching optimality. When set to 0 (zero), no simplex method iteration occurs. However, CPLEX factors the initial basis from which solution routines provide information about the associated initial solution.

Values

Any nonnegative integer; **default**: 9223372036800000000.

local branching heuristic

Controls whether CPLEX applies a local branching heuristic to try to improve new incumbents found during a MIP search.

Purpose

Local branching heuristic

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_MIP_Strategy_LBHeur	CPX_PARAM_LBHEUR
C++	IloCplex::Param::MIP::Strategy::LBHeur	LBHeur (bool)
Java	IloCplex.Param.MIP.Strategy.LBHeur	LBHeur (bool)
.NET	Cplex.Param.MIP.Strategy.LBHeur	LBHeur (bool)
OPL		lbheur
Python	parameters.mip.strategy.lbheur	mip.strategy.lbheur
MATLAB	Cplex.Param.mip.strategy.lbheur	mip.strategy.lbheur
Interactive	mip strategy lbheur	mip strategy lbheur
Identifier	2063	2063

Description

Controls whether CPLEX applies a local branching heuristic to try to improve new incumbents found during a MIP search. By default, this parameter is off. If you turn it on, CPLEX will invoke a local branching heuristic only when it finds a new incumbent. If CPLEX finds multiple incumbents at a single node, the local branching heuristic will be applied only to the last one found.

Table 31. Values

Value	bool	Symbol	Interactive	Meaning
0	false	CPX_OFF	no	Local branching heuristic is off; **default**
1	true	CPX_ON	yes	Apply local branching heuristic to new incumbent

Lift-and-project cuts switch for MIP and MIQCP

Decides whether or not lift-and-project cuts are generated for the problem.

Purpose

Lift-and-project cuts switch

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_MIP_Cuts_LiftProj	CPX_PARAM_LANDPCUTS
C++	IloCplex::Param::MIP::Cuts::LiftProj	LiftProjCuts (int)
Java	IloCplex.Param.MIP.Cuts.LiftProj	LiftProjCuts (int)
.NET	Cplex.Param.MIP.Cuts.LiftProj	LiftProjCuts (int)
OPL		not available
Python	parameters.mip.cuts.liftproj	mip.cuts.liftproj
MATLAB	Cplex.Param.mip.cuts.liftproj	mip.cuts.liftproj
Interactive	mip cuts liftproj	mip cuts liftproj
Identifier	2152	2152

Description

Decides whether or not lift-and-project cuts are generated for the problem. Setting the value of this parameter to 0 (zero), the default, specifies that the attempt to generate lift-and-project cuts should continue only if it seems to be helping.

For a brief definition of lift-and-project cuts, see the topic Lift-and-project cuts in the general topic Cuts in the CPLEX User's Manual. That same topic also includes a bibliography for further reading about lift-and-project cuts.

The table Parameters for controlling cuts, also in the user's manual, includes links to the documentation of other parameters affecting other types of cuts.

Table 32. Values

Value	Meaning
-1	Do not generate lift-and-project cuts
0	Automatic: let CPLEX choose; **default**
1	Generate lift-and-project cuts moderately
2	Generate lift-and-project cuts aggressively
3	Generate lift-and-project cuts very aggressively

MCF cut switch

Switches on or off generation of multi-commodity flow cuts in a MIP.

Purpose

Switches on or off generation of multi-commodity flow cuts in a MIP.

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_MIP_Cuts_MCFCut	CPX_PARAM_MCFCUTS
C++	IloCplex::Param::MIP::Cuts::MCFCut	MCFCuts (int)
Java	IloCplex.Param.MIP.Cuts.MCFCut	MCFCuts (int)
.NET	Cplex.Param.MIP.Cuts.MCFCut	MCFCuts (int)
OPL		mcfcuts
Python	parameters.mip.cuts.mcfcut	mip.cuts.mcfcut

API	Parameter Name	Name prior to V12.6.0
MATLAB	Cplex.Param.mip.cuts.mcfcut	mip.cuts.mcfcut
Interactive	mip cuts mcfcut	mip cuts mcfcut
Identifier	2134	2134

Description

Specifies whether CPLEX should generate **multi-commodity flow cuts** in a problem where CPLEX detects the characteristics of a multi-commodity flow network with **arc capacities**. By default, CPLEX decides whether or not to generate such cuts.

To turn off generation of such cuts, set this parameter to -1 (minus one).

CPLEX is able to recognize the structure of a network as represented in many real-world models. When it recognizes such a network structure, CPLEX is able to generate cutting planes that usually help solve such problems. In this case, the cuts that CPLEX generates state that the capacities installed on arcs pointing into a component of the network must be at least as large as the total flow demand of the component that cannot be satisfied by flow sources within the component.

For a definition of a multi-commodity flow cut, see the topic Multi-commodity flow (MCF) cuts in the general topic Cuts in the *CPLEX User's Manual*. The table Parameters for controlling cuts, also in the user's manual, includes links to the documentation of other parameters affecting other types of cuts.

Table 33. Values

Value	Meaning
-1	Turn off MCF cuts
0	Automatic: let CPLEX decide whether to generate MCF cuts; **default**
1	Generate a moderate number of MCF cuts
2	Generate MCF cuts aggressively

memory reduction switch

Directs CPLEX that it should conserve memory where possible.

Purpose

Reduces use of memory

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_Emphasis_Memory	CPX_PARAM_MEMORYEMPHASIS
C++	IloCplex::Param::Emphasis::Memory	MemoryEmphasis (bool)
Java	IloCplex.Param.Emphasis.Memory	MemoryEmphasis (bool)
.NET	Cplex.Param.Emphasis.Memory	MemoryEmphasis (bool)
OPL		memoryemphasis
Python	parameters.emphasis.memory	emphasis.memory
MATLAB	Cplex.Param.emphasis.memory	emphasis.memory
Interactive	emphasis memory	emphasis memory
Identifier	1082	1082

Description

Directs CPLEX that it should conserve memory where possible. When you set this parameter to its nondefault value, CPLEX will choose tactics, such as data compression or disk storage, for some of the data computed by the simplex, barrier, and MIP optimizers. Of course, conserving memory may impact performance in some models. Also, while solution information will be available after optimization, certain computations that require a basis that has been factored (for example, for the computation of the condition number Kappa) may be unavailable.

Table 34. Values

Value	bool	Symbol	Interactive	Meaning
0	false	`CPX_OFF`	no	Off; do not conserve memory; **default**
1	true	`CPX_ON`	yes	On; conserve memory where possible

MIP callback switch between original model and reduced, presolved model

Controls whether your callback accesses node information of the original model (off) or node information of the reduced, presolved model (on, default).

Purpose

MIP callback switch between original model and reduced, presolved model

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_MIP_Strategy_CallbackReducedLP	CPX_PARAM_MIPCBREDLP (int)
C++		
Java		
.NET		
OPL		
Python		
MATLAB		
Interactive		
Identifier	2055	2055

Description

Controls whether your callback accesses node information of the original model (off) or node information of the reduced, presolved model (on, default); also known as the MIP callback reduced LP parameter.

Advanced routines to control MIP callbacks (such as `CPXgetcallbacklp` , `CPXsetheuristiccallbackfunc` , `CPXsetbranchcallbackfunc` , `CPXgetbranchcallbackfunc` , `CPXsetcutcallbackfunc` , `CPXsetincumbentcallbackfunc` , `CPXgetcallbacksosinfo` , `CPXcutcallbackadd` , CPXcutcallbackaddlocal , and others) consider the setting of this parameter and access the original model or the reduced, presolved model accordingly.

The routine CPXgetcallbacknodelp is an exception: it always accesses the current node LP associated with the presolved model, regardless of the setting of this parameter.

For certain routines, such as CPXcutcallbackadd , when you set the parameter CPX_PARAM_MIPCBREDLP to zero, you should also set CPX_PARAM_PRELINEAR to zero as well.

In the C++, Java, .NET, Python, and MATLAB APIs of CPLEX, only the original model is available to callbacks. In other words, this parameter is effective only for certain advanced routines of the C API.

Table 35. Values.

Value	Symbol	Meaning
0	CPX_OFF	Off: use original model
1	CPX_ON	On: use reduced, presolved model; **default**

MIP node log display information

Decides what CPLEX reports to the screen during mixed integer optimization (MIP).

Purpose

MIP node log display information

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_MIP_Display	CPX_PARAM_MIPDISPLAY
C++	IloCplex::Param::MIP::Display	MIPDisplay (int)
Java	IloCplex.Param.MIP.Display	MIPDisplay (int)
.NET	Cplex.Param.MIP.Display	MIPDisplay (int)
OPL		mipdisplay
Python	parameters.mip.display	mip.display
MATLAB	Cplex.Param.mip.display	mip.display
Interactive	mip display	mip display
Identifier	2012	2012

Description

Decides what CPLEX reports to the screen and records in a log during mixed integer optimization (MIP).

The amount of information displayed increases with increasing values of this parameter.

- A setting of 0 (zero) causes no node log to be displayed until the **optimal solution** is found.
- A setting of 1 (one) displays an entry for each **integer feasible solution** found.
  Each entry contains:

– the value of the **objective function**;
– the **node count**;
– the **number of unexplored nodes** in the tree;
– the current **optimality gap**.

- A setting of 2 also generates an entry at a frequency determined by the "MIP node log interval" on page 88 parameter. At a lower frequency, the log additionally displays elapsed time in seconds and deterministic time in ticks.
- A setting of 3 gives all the information of option 2 plus additional information:
  – At the same frequency as option 2, the node log adds a line specifying the number of **cutting planes** added to the problem since the last node log line was displayed; this additional line is omitted if the number of cuts added since the last log line is 0 (zero).
  – Whenever a MIP start was successfully used to find a new incumbent solution, that success is recorded in the node log. (This information about MIP starts is independent of the MIP interval frequency in option 2.)
  – For each new incumbent that is found, the node log displays how much time in seconds and how many deterministic ticks elapsed since the beginning of optimization. (This information about elapsed time between new incumbents is independent of the MIP interval frequency in option 2.)
- A setting of 4 additionally generates entries for the **LP root relaxation** according to the setting of the parameter to control the "simplex iteration information display" on page 145 (`SimDisplay`, `CPX_PARAM_SIMDISPLAY`).
- A setting of 5 additionally generates entries for the **LP subproblems**, also according to the setting of the parameter to control the "simplex iteration information display" on page 145 (`SimDisplay`, `CPX_PARAM_SIMDISPLAY`).

Table 36. Values

Value	Meaning
0	No display until optimal solution has been found
1	Display integer feasible solutions
2	Display integer feasible solutions plus an entry at a frequency set by "MIP node log interval" on page 88; **default**
3	Display the number of cuts added since previous display; information about the processing of each successful MIP start; elapsed time in seconds and elapsed time in deterministic ticks for integer feasible solutions
4	Display information available from previous options and information about the LP subproblem at root
5	Display information available from previous options and information about the LP subproblems at root **and** at nodes

See also

"MIP node log interval" on page 88, "simplex iteration information display" on page 145, "network logging display switch" on page 95, and "messages to screen switch" on page 142

MIP emphasis switch

Controls trade-offs between speed, feasibility, optimality, and moving bounds in MIP.

Purpose

MIP emphasis switch

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_Emphasis_MIP	CPX_PARAM_MIPEMPHASIS
C++	IloCplex::Param::Emphasis::MIP	MIPEmphasis (int)
Java	IloCplex.Param.Emphasis.MIP	MIPEmphasis (int)
.NET	Cplex.Param.Emphasis.MIP	MIPEmphasis (int)
OPL		mipemphasis
Python	parameters.emphasis.mip	emphasis.mip
MATLAB	Cplex.Param.emphasis.mip	emphasis.mip
Interactive	emphasis mip	emphasis mip
Identifier	2058	2058

Description

Controls trade-offs between speed, feasibility, optimality, and moving bounds in MIP.

With the default setting of BALANCED, CPLEX works toward a rapid proof of an optimal solution, but balances that with effort toward finding high quality feasible solutions early in the optimization.

When this parameter is set to FEASIBILITY, CPLEX frequently will generate more feasible solutions as it optimizes the problem, at some sacrifice in the speed to the proof of optimality.

When set to OPTIMALITY, less effort may be applied to finding feasible solutions early.

With the setting BESTBOUND, even greater emphasis is placed on proving optimality through moving the best bound value, so that the detection of feasible solutions along the way becomes almost incidental.

When the parameter is set to HIDDENFEAS, the MIP optimizer works hard to find high quality feasible solutions that are otherwise very difficult to find, so consider this setting when the FEASIBILITY setting has difficulty finding solutions of acceptable quality.

Table 37. Values

Value	Symbol	Meaning
0	`CPX_MIPEMPHASIS_BALANCED`	Balance optimality and feasibility; **default**
1	`CPX_MIPEMPHASIS_FEASIBILITY`	Emphasize feasibility over optimality
2	`CPX_MIPEMPHASIS_OPTIMALITY`	Emphasize optimality over feasibility
3	`CPX_MIPEMPHASIS_BESTBOUND`	Emphasize moving best bound
4	`CPX_MIPEMPHASIS_HIDDENFEAS`	Emphasize finding hidden feasible solutions

MIP node log interval

Controls the frequency of node logging when the MIP display parameter is set higher than 1 (one).

Purpose

MIP node log interval

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_MIP_Interval	CPX_PARAM_MIPINTERVAL
C++	IloCplex::Param::MIP::Interval	MIPInterval (long)
Java	IloCplex.Param.MIP.Interval	MIPInterval (long)
.NET	Cplex.Param.MIP.Interval	MIPInterval (long)
OPL		mipinterval
Python	parameters.mip.interval	mip.interval
MATLAB CPLEX class API	Cplex.Param.mip.interval	mip.interval
MATLAB CPLEX Toolbox compatibility	mip.interval	mip.interval
MATLAB Optimization Toolbox compatibility	NodeDisplayInterval	NodeDisplayInterval
Interactive	mip interval	mip interval
Identifier	2013	2013

Description

Controls the frequency of node logging when the MIP display parameter ("MIP node log display information" on page 85) is set higher than 1 (one). Frequency must be an integer; it may be 0 (zero), positive, or negative.

By **default**, CPLEX displays new information in the node log during a MIP solve at relatively high frequency during the early stages of solving a MIP model, and adds lines to the log at progressively longer intervals as solving continues. In other words, CPLEX logs information frequently in the beginning and progressively less often as it works.

When the value is a **positive integer** n, CPLEX displays new incumbents, plus it displays a new line in the log every n nodes.

When the value is a **negative integer** n, CPLEX displays new incumbents, and the negative value determines how much processing CPLEX does before it displays a new line in the node log. A negative value close to zero means that CPLEX displays new lines in the log frequently. A negative value far from zero means that CPLEX displays new lines in the log less frequently. In other words, a negative value of this parameter contracts or dilates the interval at which CPLEX displays information in the node log.

Table 38. Values

Value	Meaning
n < 0	Display new incumbents, and display a log line frequently at the beginning of solving and less frequently as solving progresses
0 (zero)	automatic: let CPLEX decide the frequency to log nodes (**default**)
n > 0	Display new incumbents, and display a log line every n nodes

See also

"MIP node log display information" on page 85

MIP kappa computation

Sets the strategy for computing statistics about MIP kappa

Purpose

MIP kappa computation

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_MIP_Strategy_KappaStats	CPX_PARAM_MIPKAPPASTATS
C++	IloCplex::Param::MIP::Strategy::KappaStats	MIPKappaStats (int)
Java	IloCplex.Param.MIP.Strategy.KappaStats	MIPKappaStats (int)
.NET	Cplex.Param.MIP.Strategy.KappaStats	MIPKappaStats (int)
OPL		mipkappastats
Python	parameters.mip.strategy.kappastats	mip.strategy.kappastats
MATLAB	Cplex.Param.mip.strategy.kappastats	mip.strategy.kappastats
Interactive	mip strategy kappastats	mip strategy kappastats
Identifier	2137	2137

Description

Sets the strategy for CPLEX to gather statistics about the MIP kappa of subproblems of a MIP.

What is MIP kappa?

MIP kappa summarizes the **distribution** of the **condition number** of the **optimal bases** CPLEX encountered during the solution of a MIP model. That summary may let you know more about the numerical difficulties of your MIP model.

When can you compute MIP kappa?

Because MIP kappa (as a statistical distribution) requires CPLEX to compute the condition number of the optimal bases of the subproblems during branch-and-cut search, you can compute the MIP kappa only when CPLEX solves the subproblem with its simplex optimizer. In other words, in order to obtain results with this parameter, you can **not** use the sifting optimizer nor the barrier without crossover to solve the subproblems. See the parameters "MIP subproblem algorithm" on page 99 (`CPX_PARAM_SUBALG`, `NodeAlg`) and "algorithm for initial MIP relaxation" on page 139 (`CPX_PARAM_STARTALG`, `RootAlg`) for more details about those choices.

What are the performance trade-offs for computing MIP kappa?

Computing the kappa of a subproblem has a cost. In fact, computing MIP kappa for the basis matrices can be computationally expensive and thus generally slows down the solution of a problem. Therefore, the automatic setting `CPX_MIPKAPPA_AUTO` tells CPLEX generally not to compute MIP kappa, but in cases where the parameter "numerical precision emphasis" on page 102 (`CPX_PARAM_NUMERICALEMPHASIS`, `NumericalEmphasis`) is turned on, that is, set to 1 (one), CPLEX computes MIP kappa for a sample of subproblems.

The value `CPX_MIPKAPPA_SAMPLE` leads to a negligible performance degradation on average, but can slow down the branch-and-cut exploration by as much as 10% on certain models.

The value `CPX_MIPKAPPA_FULL` leads to a 2% performance degradation on average, but can significantly slow the branch-and-cut exploration on certain models.

In practice, the value `CPX_MIPKAPPA_SAMPLE` is a good trade-off between performance and accuracy of statistics.

If you need very accurate statistics, then use value `CPX_MIPKAPPA_FULL`.

Table 39. Values

Value	Symbol	Meaning
–1	`CPX_MIPKAPPA_OFF`	No MIP kappa statistics
0	`CPX_MIPKAPPA_AUTO`	Automatic: let CPLEX decide; **default**
1	`CPX_MIPKAPPA_SAMPLE`	Compute MIP kappa for a sample of subproblems
2	`CPX_MIPKAPPA_FULL`	Compute MIP kappa for all subproblems

MIP priority order switch

Decides whether to use the priority order, if one exists, for the next mixed integer optimization.

Purpose

MIP priority order switch

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_MIP_Strategy_Order	CPX_PARAM_MIPORDIND
C++	IloCplex::Param::MIP::Strategy::Order	MIPOrdInd (bool)
Java	IloCplex.Param.MIP.Strategy.Order	MIPOrdInd (bool)
.NET	Cplex.Param.MIP.Strategy.Order	MIPOrdInd (bool)
OPL		mipordind
Python	parameters.mip.strategy.order	mip.strategy.order
MATLAB	Cplex.Param.mip.strategy.order	mip.strategy.order
Interactive	mip strategy order	mip strategy order
Identifier	2020	2020

Description

Decides whether to use the priority order, if one exists, for the next mixed integer optimization.

Table 40. Values

Value	bool	Symbol	Interactive	Meaning
0	false	`CPX_OFF`	no	Off: do not use priority order
1	true	`CPX_ON`	yes	On: use priority order, if it exists; **default**

MIP priority order generation

Selects the type of generic priority order to generate when no priority order is present.

Purpose

MIP priority order generation

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_MIP_OrderType	CPX_PARAM_MIPORDTYPE
C++	IloCplex::Param::MIP::OrderType	MIPOrdType (int)
Java	IloCplex.Param.MIP.OrderType	MIPOrdType (int)
.NET	Cplex.Param.MIP.OrderType	MIPOrdType (int)
OPL		mipordtype
Python	parameters.mip.ordertype	mip.ordertype
MATLAB	Cplex.Param.mip.ordertype	mip.ordertype
Interactive	mip ordertype	mip ordertype
Identifier	2032	2032

Description

Selects the type of generic priority order to generate when no priority order is present.

Table 41. Values

Value	Symbol	Meaning
0	**default**	Do not generate a priority order
1	`CPX_MIPORDER_COST`	Use decreasing cost
2	`CPX_MIPORDER_BOUNDS`	Use increasing bound range
3	`CPX_MIPORDER_SCALEDCOST`	Use increasing cost per coefficient count

MIP dynamic search switch

Sets the search strategy for a mixed integer program (MIP).

Purpose

MIP dynamic search switch

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_MIP_Strategy_Search	CPX_PARAM_MIPSEARCH
C++	IloCplex::Param::MIP::Strategy::Search	MIPSearch (int)
Java	IloCplex.Param.MIP.Strategy.Search	MIPSearch (int)
.NET	Cplex.Param.MIP.Strategy.Search	MIPSearch (int)
OPL		mipsearch
Python	parameters.mip.strategy.search	mip.strategy.search
MATLAB	Cplex.Param.mip.strategy.search	mip.strategy.search
Interactive	mip strategy search	mip strategy search
Identifier	2109	2109

Description

Sets the search strategy for a mixed integer program (MIP). By default, CPLEX chooses whether to apply dynamic search or conventional branch and cut based on characteristics of the model and the presence (or absence) of callbacks.

Only informational callbacks are compatible with dynamic search. For more detail about informational callbacks and how to create and install them in your application, see Informational callbacks in the *CPLEX User's Manual*.

To benefit from dynamic search, a MIP must **not** include query callbacks. In other words, query callbacks are not compatible with dynamic search. For a more detailed definition of query or diagnostic callbacks, see Query or diagnostic callbacks in the *CPLEX User's Manual*.

To benefit from dynamic search, a MIP must **not** include control callbacks (that is, callbacks that alter the search path through the solution space). In other words, control callbacks are not compatible with dynamic search. These control callbacks are identified as **advanced** in the reference manuals of the APIs. If control callbacks are present in your application, CPLEX will disable dynamic search, issue a warning, and apply only static branch and cut. If you want to control the search yourself, for example, through advanced control callbacks, then you should set this parameter to 1 (one) to disable dynamic search and to apply conventional branch and cut.

Table 42. Values

Value	Symbolic Name	Meaning
0	CPX_MIPSEARCH_AUTO	Automatic: let CPLEX choose; **default**
1	CPX_MIPSEARCH_TRADITIONAL	Apply traditional branch and cut strategy; disable dynamic search
2	CPX_MIPSEARCH_DYNAMIC	Apply dynamic search

MIQCP strategy switch

Sets the strategy that CPLEX uses to solve a quadratically constrained mixed integer program (MIQCP).

Purpose

MIQCP strategy switch

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_MIP_Strategy_MIQCPStrat	CPX_PARAM_MIQCPSTRAT
C++	IloCplex::Param::MIP::Strategy::MIQCPStrat	MIQCPStrat (int)
Java	IloCplex.Param.MIP.Strategy.MIQCPStrat	MIQCPStrat (int)
.NET	Cplex.Param.MIP.Strategy.MIQCPStrat	MIQCPStrat (int)
OPL		miqcpstrat
Python	parameters.mip.strategy.miqcpstrat	mip.strategy.miqcpstrat
MATLAB	Cplex.Param.mip.strategy.miqcpstrat	mip.strategy.miqcpstrat
Interactive	mip strategy miqcpstrat	mip strategy miqcpstrat
Identifier	2110	2110

Description

Sets the strategy that CPLEX uses to solve a quadratically constrained mixed integer program (MIQCP).

This parameter controls how MIQCPs (that is, mixed integer programs with one or more constraints including quadratic terms) are solved.

For more detail about the types of quadratically constrained models that CPLEX solves, see Identifying a quadratically constrained program (QCP) in the *CPLEX User's Manual*.

For more information about the strategy that CPLEX applies to solve MIQCPs, see Features of the MIP optimizer for MIQCP also in the *CPLEX User's Manual*.

At the default setting of 0 (zero), CPLEX automatically chooses a strategy.

When you set this parameter to the value 1 (one), you tell CPLEX to solve a **QCP relaxation** of the model at each node.

When you set this parameter to the value 2, you tell CPLEX to attempt to solve an **LP relaxation** of the model at each node.

For some models, the setting 2 may be more effective than 1 (one). For best performance, you may need to experiment with this parameter to determine the appropriate setting for your model.

Specifically, if the node log indicates long solve times for a QCP relaxation, consider setting this parameter to the value 2. Conversely, if you see that the best node value appears to move very slowly, the linear approximation may not be particularly accurate; in such cases, setting the parameter to value 1 (one) may improve performance.

Table 43. Values

Value	Meaning
0	Automatic: let CPLEX choose; **default**
1	Solve a QCP node relaxation at each node
2	Solve an LP node relaxation at each node

MIP MIR (mixed integer rounding) cut switch

Decides whether or not to generate MIR cuts (mixed integer rounding cuts) for the problem.

Purpose

MIP MIR (mixed integer rounding) cut switch

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_MIP_Cuts_MIRCut	CPX_PARAM_MIRCUTS
C++	IloCplex::Param::MIP::Cuts::MIRCut	MIRCuts (int)
Java	IloCplex.Param.MIP.Cuts.MIRCut	MIRCuts (int)
.NET	Cplex.Param.MIP.Cuts.MIRCut	MIRCuts (int)
OPL		mircuts
Python	parameters.mip.cuts.mircut	mip.cuts.mircut
MATLAB	Cplex.Param.mip.cuts.mircut	mip.cuts.mircut
Interactive	mip cuts mircut	mip cuts mircut
Identifier	2052	2052

Description

Decides whether or not to generate MIR cuts (mixed integer rounding cuts) for the problem. The value 0 (zero), the default, specifies that the attempt to generate MIR cuts should continue only if it seems to be helping.

For a definition of a MIR cut, see the topic Mixed integer rounding (MIR) cuts in the general topic Cuts in the *CPLEX User's Manual*. The table Parameters for controlling cuts, also in the user's manual, includes links to the documentation of other parameters affecting other types of cuts.

Value	Meaning
-1	Do not generate MIR cuts
0	Automatic: let CPLEX choose; **default**
1	Generate MIR cuts moderately
2	Generate MIR cuts aggressively

precision of numerical output in MPS and REW file formats

Decides the precision of numerical output in the MPS and REW file formats.

Purpose

Precision of numerical output in MPS and REW file formats

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_Output_MPSLong	CPX_PARAM_MPSLONGNUM
C++	IloCplex::Param::Output::MPSLong	MPSLongNum (bool)
Java	IloCplex.Param.Output.MPSLong	MPSLongNum (bool)
.NET	Cplex.Param.Output.MPSLong	MPSLongNum (bool)
OPL		not available
Python	parameters.output.mpslong	output.mpslong

API	Parameter Name	Name prior to V12.6.0
MATLAB	Cplex.Param.output.mpslong	output.mpslong
Interactive	output mpslong	output mpslong
Identifier	1081	1081

Description

Decides the precision of numerical output in the MPS and REW file formats. When this parameter is set to its default value 1 (one), numbers are written to MPS files in full-precision; that is, up to 15 significant digits may be written. The setting 0 (zero) writes files that correspond to the standard MPS format, where at most 12 characters can be used to represent a value. This limit may result in loss of precision.

Value	bool	Symbol	Interactive	Meaning
0	false	`CPX_OFF`	no	Off: use limited MPS precision
1	true	`CPX_ON`	yes	On: use full-precision; **default**

See also

MPS file format: industry standard

network logging display switch

Decides what CPLEX reports to the screen during network optimization.

Purpose

Network logging display switch

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_Network_Display	CPX_PARAM_NETDISPLAY
C++	IloCplex::Param::Network::Display	NetDisplay (int)
Java	IloCplex.Param.Network.Display	NetDisplay (int)
.NET	Cplex.Param.Network.Display	NetDisplay (int)
OPL		netdisplay
Python	parameters.network.display	not available
MATLAB	Cplex.Param.network.display	not available
Interactive	network display	network display
Identifier	5005	5005

Description

Decides what CPLEX reports to the screen during network optimization. Settings 1 and 2 differ only during Phase I. Setting 2 shows monotonic values, whereas 1 usually does not.

Value	Symbol	Meaning
0	`CPXNET_NO_DISPLAY_OBJECTIVE`	No display

Value	Symbol	Meaning
1	CPXNET_TRUE_OBJECTIVE	Display true objective values
2	CPXNET_PENALIZE_OBJECTIVE	Display penalized objective values; **default**

network optimality tolerance

Specifies the optimality tolerance for network optimization.

Purpose

Optimality tolerance for network optimization

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_Network_Tolerances_Optimality	CPX_PARAM_NETEPOPT
C++	IloCplex::Param::Network::Tolerances::Optimality	NetEpOpt (double)
Java	IloCplex.Param.Network.Tolerances.Optimality	NetEpOpt (double)
.NET	Cplex.Param.Network.Tolerances.Optimality	NetEpOpt (double)
OPL		netepopt
Python	parameters.network.tolerances.optimality	not available
MATLAB	Cplex.Param.network.tolerances.optimality	not available
Interactive	network tolerances optimality	network tolerances optimality
Identifier	5002	5002

Description

Specifies the optimality tolerance for network optimization; that is, the amount a reduced cost may violate the criterion for an optimal solution.

Values

Any number from 1e-11 to 1e-1; **default**: 1e-6.

network primal feasibility tolerance

Specifies feasibility tolerance for network primal optimization. The feasibility tolerance specifies the degree to which the flow value of a model may violate its bounds.

Purpose

Feasibility tolerance for network primal optimization

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_Network_Tolerances_Feasibility	CPX_PARAM_NETEPRHS
C++	IloCplex::Param::Network::Tolerances::Feasibility	NetEpRHS (double)
Java	IloCplex.Param.Network.Tolerances.Feasibility	NetEpRHS (double)
.NET	Cplex.Param.Network.Tolerances.Feasibility	NetEpRHS (double)
OPL		neteprhs
Python	parameters.network.tolerances.feasibility	not available
MATLAB	Cplex.Param.network.tolerances.feasibility	not available
Interactive	network tolerances feasibility	network tolerances feasibility
Identifier	5003	5003

Description

Specifies feasibility tolerance for network primal optimization. The feasibility tolerance specifies the degree to which the flow value of a model may violate its bounds. This tolerance influences the selection of an optimal basis and can be reset to a higher value when a problem is having difficulty maintaining feasibility during optimization. You may also wish to lower this tolerance after finding an optimal solution if there is any doubt that the solution is truly optimal. If the feasibility tolerance is set too low, CPLEX may falsely conclude that a problem is infeasible. If you encounter reports of infeasibility during Phase II of the optimization, a small adjustment in the feasibility tolerance may improve performance.

Values

Any number from 1e-11 to 1e-1; **default**: 1e-6.

simplex network extraction level

Establishes the level of network extraction for network simplex optimization.

Purpose

Simplex network extraction level

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_Network_NetFind	CPX_PARAM_NETFIND
C++	IloCplex::Param::Network::NetFind	NetFind (int)
Java	IloCplex.Param.Network.NetFind	NetFind (int)
.NET	Cplex.Param.Network.NetFind	NetFind (int)
OPL		netfind
Python	parameters.network.netfind	not available
MATLAB	Cplex.Param.network.netfind	not available
Interactive	network netfind	network netfind
Identifier	1022	1022

Description

Establishes the level of network extraction for network simplex optimization. The default value is suitable for recognizing commonly used modeling approaches when representing a network problem within an LP formulation.

Table 44. Values

Value	Symbol	Meaning
1	`CPX_NETFIND_PURE`	Extract pure network only
2	`CPX_NETFIND_REFLECT`	Try reflection scaling; **default**
3	`CPX_NETFIND_SCALE`	Try general scaling

network simplex iteration limit

Sets the maximum number of iterations to be performed before the algorithm terminates without reaching optimality.

Purpose

Network simplex iteration limit

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_Network_Iterations	CPX_PARAM_NETITLIM
C++	IloCplex::Param::Network::Iterations	NetItLim (long)
Java	IloCplex::Param::Network::Iterations	NetItLim (long)
.NET	Cplex::Param::Network::Iterations	NetItLim (long)
OPL		netitlim
Python	parameters.network.iterations	not available
MATLAB	Cplex.Param.network.iterations	not available
Interactive	network iterations	network iterations
Identifier	5001	5001

Description

Sets the maximum number of iterations to be performed before the algorithm terminates without reaching optimality.

Values

Any nonnegative integer; **default**: 9223372036800000000.

network simplex pricing algorithm

Specifies the pricing algorithm for network simplex optimization.

Purpose

Network simplex pricing algorithm

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_Network_Pricing	CPX_PARAM_NETPPRIIND
C++	IloCplex::Param::Network::Pricing	NetPPriInd (int)
Java	IloCplex.Param.Network.Pricing	NetPPriInd (int)
.NET	Cplex.Param.Network.Pricing	NetPPriInd (int)
OPL		netppriind
Python	parameters.network.pricing	not available
MATLAB	Cplex.Param.network.pricing	not available
Interactive	network pricing	network pricing
Identifier	5004	5004

Description

Specifies the pricing algorithm for network simplex optimization. The default (0) shows best performance for most problems, and currently is equivalent to 3.

Table 45. Values

Value	Symbol	Meaning
0	CPXNET_PRICE_AUTO	Automatic: let CPLEX choose; **default**
1	CPXNET_PRICE_PARTIAL	Partial pricing
2	CPXNET_PRICE_MULT_PART	Multiple partial pricing
3	CPXNET_PRICE_SORT_MULT_PART	Multiple partial pricing with sorting

MIP subproblem algorithm

Decides which continuous optimizer will be used to solve the subproblems in a MIP, after the initial relaxation.

Purpose

MIP subproblem algorithm

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_MIP_Strategy_SubAlgorithm	CPX_PARAM_SUBALG
C++	IloCplex::Param::NodeAlgorithm	NodeAlg (int)
Java	IloCplex.Param.NodeAlgorithm	NodeAlg (int)
.NET	Cplex.Param.NodeAlgorithm	NodeAlg (int)
OPL		nodealg
Python	parameters.mip.strategy.subalgorithm	mip.strategy.subalgorithm
MATLAB	Cplex.Param.mip.strategy.subalgorithm	mip.strategy.subalgorithm
Interactive	mip strategy subalgorithm	mip strategy subalgorithm
Identifier	2026	2026

Description

Decides which continuous optimizer will be used to solve the subproblems in a MIP, after the initial relaxation.

The default Automatic setting (0 zero) of this parameter currently selects the dual simplex optimizer for subproblem solution for MILP and MIQP. The Automatic setting may be expanded in the future so that CPLEX chooses the algorithm based on additional characteristics of the model.

For MILP (integer constraints and otherwise continuous variable), all settings are permitted.

For MIQP (integer constraints and positive semi-definite quadratic terms in objective), setting 3 (Network) is not permitted, and setting 5 (Sifting) reverts to 0 (Automatic).

For MIQCP (integer constraints and positive semi-definite quadratic terms among the constraints), only the setting 0 (Automatic) is permitted. In other words, given a problem of type MIQCP, and any setting of this parameter other than 0 (Automatic), CPLEX will exit because of the quadratic constraint or quadratic constraints, and CPLEX displays an error message specifying that a mixed integer problem cannot be solved by this setting.

Table 46. Values

Value	Symbol	Meaning
0	`CPX_ALG_AUTOMATIC`	Automatic: let CPLEX choose; **default**
1	`CPX_ALG_PRIMAL`	Primal simplex
2	`CPX_ALG_DUAL`	Dual simplex
3	`CPX_ALG_NET`	Network simplex
4	`CPX_ALG_BARRIER`	Barrier
5	`CPX_ALG_SIFTING`	Sifting

node storage file switch

Used when working memory (`CPX_PARAM_WORKMEM`, `WorkMem`) has been exceeded by the size of the search tree.

Purpose

Node storage file switch

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_MIP_Strategy_File	CPX_PARAM_NODEFILEIND
C++	IloCplex::Param::MIP::Strategy::File	NodeFileInd (int)
Java	IloCplex.Param.MIP.Strategy.File	NodeFileInd (int)
.NET	Cplex.Param.MIP.Strategy.File	NodeFileInd (int)
OPL		nodefileind
Python	parameters.mip.strategy.file	mip.strategy.file
MATLAB	Cplex.Param.mip.strategy.file	mip.strategy.file
Interactive	mip strategy file	mip strategy file
Identifier	2016	2016

Description

Used when working memory specified by the “memory available for working storage” on page 168 parameter, (`CPX_PARAM_WORKMEM`, `WorkMem`) has been exceeded by the size of the tree. If the node file parameter is set to zero when the tree memory limit is reached, optimization is terminated. Otherwise, a group of nodes is removed from the in-memory set as needed. By default, CPLEX transfers nodes to node files when the in-memory set is larger than the current value of “memory available for working storage” on page 168, and it keeps the resulting node files in compressed form in memory. At settings 2 and 3, the node files are transferred to disk, in uncompressed and compressed form respectively, into the “directory for working files” on page 167 (`CPX_PARAM_WORKDIR`, `WorkDir`), and CPLEX actively manages which nodes remain in memory for processing.

Value	Meaning
0	No node file
1	Node file in memory and compressed; **default**
2	Node file on disk
3	Node file on disk and compressed

Related reference:

"directory for working files" on page 167
Specifies the name of an existing directory into which CPLEX may store temporary working files.

"memory available for working storage" on page 168
Specifies an upper limit on the amount of central memory, in megabytes, that CPLEX is permitted to use for working memory.

MIP node limit

Sets the maximum number of nodes solved before the algorithm terminates without reaching optimality.

Purpose

MIP node limit

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_MIP_Limits_Nodes	CPX_PARAM_NODELIM
C++	IloCplex::Param::MIP::Limits::Nodes	NodeLim (long)
Java	IloCplex.Param.MIP.Limits.Nodes	NodeLim (long)
.NET	Cplex.Param.MIP.Limits.Nodes	NodeLim (long)
OPL		nodelim
Python	parameters.mip.limits.nodes	mip.limits.nodes
MATLAB Cplex class compatible	Cplex.Param.mip.limits.nodes	mip.limits.nodes
MATLAB CPLEX Toolbox compatible	mip.limits.nodes	mip.limits.nodes
MATLAB Optimization Toolbox compatible	MaxNodes	MaxNodes
Interactive	mip limits nodes	mip limits nodes
Identifier	2017	2017

Description

Sets the maximum number of nodes solved before the algorithm terminates without reaching optimality. When this parameter is set to 0 (zero), CPLEX completes processing at the root; that is, it creates cuts and applies heuristics at the root. When this parameter is set to 1 (one), it allows branching from the root; that is, nodes are created but not solved.

Values

Any nonnegative integer; **default**: 9223372036800000000.

MIP node selection strategy

Used to set the rule for selecting the next node to process when backtracking.

Purpose

MIP node selection strategy

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_MIP_Strategy_NodeSelect	CPX_PARAM_NODESEL
C++	IloCplex::Param::MIP::Strategy::NodeSelect	NodeSel (int)
Java	IloCplex.Param.MIP.Strategy.NodeSelect	NodeSel (int)
.NET	Cplex.Param.MIP.Strategy.NodeSelect	NodeSel (int)
OPL		nodesel
Python	parameters.mip.strategy.nodeselect	mip.strategy.nodeselect
MATLAB Cplex class compatibity	Cplex.Param.mip.strategy.nodeselect	mip.strategy.nodeselect
MATLAB CPLEX Toolbox compatibility	mip.strategy.nodeselect	mip.strategy.nodeselect
MATLAB Optimization Toolbox compatibility	NodeSearchStrategy	NodeSearchStrategy
Interactive	mip strategy nodeselect	mip strategy nodeselect
Identifier	2018	2018

Description

Sets the rule for selecting the next node to process when the search is backtracking. The depth-first search strategy chooses the most recently created node. The best-bound strategy chooses the node with the best objective function for the associated LP relaxation. The best-estimate strategy selects the node with the best estimate of the integer objective value that would be obtained from a node once all integer infeasibilities are removed. An alternative best-estimate search is also available.

Table 47. Values

Value	Symbol	Meaning
0	`CPX_NODESEL_DFS`	Depth-first search
1	`CPX_NODESEL_BESTBOUND`	Best-bound search; **default**
2	`CPX_NODESEL_BESTEST`	Best-estimate search
3	`CPX_NODESEL_BESTEST_ALT`	Alternative best-estimate search

numerical precision emphasis

Emphasizes precision in numerically unstable or difficult problems.

Purpose

Numerical precision emphasis

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_Emphasis_Numerical	CPX_PARAM_NUMERICALEMPHASIS

API	Parameter Name	Name prior to V12.6.0
C++	IloCplex::Param::Emphasis::Numerical	NumericalEmphasis (bool)
Java	IloCplex.Param.Emphasis.Numerical	NumericalEmphasis (bool)
.NET	Cplex.Param.Emphasis.Numerical	NumericalEmphasis (bool)
OPL		numericalemphasis
Python	parameters.emphasis.numerical	emphasis.numerical
MATLAB	Cplex.Param.emphasis.numerical	emphasis.numerical
Interactive	emphasis numerical	emphasis numerical
Identifier	1083	1083

Description

Emphasizes precision in numerically unstable or difficult problems. This parameter lets you specify to CPLEX that it should emphasize precision in numerically difficult or unstable problems, with consequent performance trade-offs in time and memory.

Table 48. Values

Value	bool	Symbol	Interactive	Meaning
0	false	`CPX_OFF`	no	Do not emphasize numerical precision; **default**
1	true	`CPX_ON`	yes	Exercise extreme caution in computation

nonzero element read limit

Specifies a limit for the number of nonzero elements to read for an allocation of memory.

Purpose

Nonzero element read limit

API	Parameter Name	Name prior to V12.6.0
C 64-bit	CPXPARAM_Read_Nonzeros	CPX_PARAM_NZREADLIM (CPXLONG)
C 32-bit	CPXPARAM_Read_Nonzeros	CPX_PARAM_NZREADLIM (CPXINT)
C++	IloCplex::Param::Read::Nonzeros	NzReadLim (CPXLONG)
Java	IloCplex.Param.Read.Nonzeros	NzReadLim (CPXLONG)
.NET	Cplex.Param.Read.Nonzeros	NzReadLim (CPXLONG)
OPL		not available
Python	parameters.read.nonzeros	read.nonzeros
MATLAB	Cplex.Param.read.nonzeros	read.nonzeros
Interactive	read nonzeros	read nonzeros
Identifier	1024	1024

Description

Specifies a limit for the number of nonzero elements to read for an allocation of memory. This parameter does not restrict the size of a problem. Rather, it indirectly specifies the default amount of memory that will be pre-allocated before a problem is read from a file. If the limit is exceeded, more memory is automatically allocated.

Values

Any integer from 0 to CPX_BIGINT or CPX_BIGLONG, depending on integer type; **default**: 250 000.

absolute objective difference cutoff

Used to update the cutoff each time a mixed integer solution is found.

Purpose

Absolute objective difference cutoff

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_MIP_Tolerances_ObjDifference	CPX_PARAM_OBJDIF
C++	IloCplex::Param::MIP::Tolerances::ObjDifference	ObjDif (double)
Java	IloCplex.Param.MIP.Tolerances.ObjDifference	ObjDif (double)
.NET	Cplex.Param.MIP.Tolerances.ObjDifference	ObjDif (double)
OPL		objdif
Python	parameters.mip.tolerances.objdifference	mip.tolerances.objdifference
MATLAB	Cplex.Param.mip.tolerances.objdifference	mip.tolerances.objdifference
Interactive	mip tolerances objdifference	mip tolerances objdifference
Identifier	2019	2019

Description

Used to update the cutoff each time a mixed integer solution is found. This absolute value is subtracted from (added to) the newly found integer objective value when minimizing (maximizing). This forces the mixed integer optimization to ignore integer solutions that are not at least this amount better than the best one found so far.

The objective difference parameter can be adjusted to improve problem solving efficiency by limiting the number of nodes; however, setting this parameter at a value other than zero (the default) can cause some integer solutions, including the true integer optimum, to be missed.

Negative values for this parameter can result in some integer solutions that are worse than or the same as those previously generated, but does not necessarily result in the generation of all possible integer solutions.

Values

Any number; **default**: 0.0.

Related reference:

"relative objective difference cutoff" on page 132
Used to update the cutoff each time a mixed integer solution is found.

lower objective value limit

Sets a lower limit on the value of the objective function in the simplex algorithms.

Purpose

Lower objective value limit

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_Simplex_Limits_LowerObj	CPX_PARAM_OBJLLIM
C++	IloCplex::Param::Simplex::Limits::LowerObj	ObjLLim (double)
Java	IloCplex.Param.Simplex.Limits.LowerObj	ObjLLim (double)
.NET	Cplex.Param.Simplex.Limits.LowerObj	ObjLLim (double)
OPL		objllim
Python	parameters.simplex.limits.lowerobj	simplex.limits.lowerobj
MATLAB	Cplex.Param.simplex.limits.lowerobj	simplex.limits.lowerobj
Interactive	simplex limits lowerobj	simplex limits lowerobj
Identifier	1025	1025

Description

Sets a lower limit on the value of the objective function in the simplex algorithms. Setting a lower objective function limit causes CPLEX to halt the optimization process when the minimum objective function value limit has been reached. This limit applies only during Phase II of the simplex algorithm in minimization problems.

Tip:

This parameter is not effective with the conflict refiner nor with FeasOpt. That is, neither of those tools can analyze an infeasibility introduced by this parameter. If you want to analyze such a condition, add an explicit objective constraint, such as `obj >= c`, to your model instead before you invoke either of those tools.

Values

Any number; **default**: -1e+75.

upper objective value limit

Sets an upper limit on the value of the objective function in the simplex algorithms.

Purpose

Upper objective value limit

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_Simplex_Limits_UpperObj	CPX_PARAM_OBJULIM
C++	IloCplex::Param::Simplex::Limits::UpperObj	ObjULim (double)
Java	IloCplex.Param.Simplex.Limits.UpperObj	ObjULim (double)

API	Parameter Name	Name prior to V12.6.0
.NET	Cplex.Param.Simplex.Limits.UpperObj	ObjULim (double)
OPL		objulim
Python	parameters.simplex.limits.upperobj	simplex.limits.upperobj
MATLAB	Cplex.Param.simplex.limits.upperobj	simplex.limits.upperobj
Interactive	simplex limits upperobj	simplex limits upperobj
Identifier	1026	1026

Description

Sets an upper limit on the value of the objective function in the simplex algorithms. Setting an upper objective function limit causes CPLEX to halt the optimization process when the maximum objective function value limit has been reached. This limit applies only during Phase II of the simplex algorithm in maximization problems.

Tip:

This parameter is not effective with the conflict refiner nor with FeasOpt. That is, neither of those tools can analyze an infeasibility introduced by this parameter. If you want to analyze such a condition, add an explicit objective constraint, such as `obj <= c`. to your model instead before you invoke either of those tools.

Values

Any number; **default**: 1e+75.

optimality target

Specifies type of optimality that CPLEX targets (optimal convex or first-order satisfaction) as it searches for a solution

Purpose

Optimality target

API	Parameter Name
C	CPXPARAM_OptimalityTarget
C++	IloCplex::Param::OptimalityTarget
Java	IloCplex.Param.OptimalityTarget
.NET	Cplex.Param.OptimalityTarget
OPL	optimalitytarget
Python	parameters.optimalitytarget
MATLAB	Cplex.Param.optimalitytarget
Interactive	optimalitytarget
Identifier	1131

Description

Specifies the type of solution that CPLEX attempts to compute with respect to the optimality of that solution when CPLEX solves a continuous (QP) or mixed integer (MIQP) **quadratic model**. In other words, the variables of the model can be continuous or mixed integer and continuous; the objective function includes a quadratic term, and perhaps the objective function is not positive semi-definite (non PSD).

Tip: This parameter does **not** apply to quadratically constrained problems whether they contain integer variables (MIQCP) or not (QCP). For those problems, use only the default value of this parameter.

By **default**, CPLEX first attempts to compute a provably optimal solution to a QP or MIQP problem. If CPLEX cannot compute a provably optimal solution because the objective function is not convex, CPLEX terminates and returns Error 5002, CPXERR_Q_NOT_POS_DEF.

When this parameter is set to 1 (one), CPLEX assumes that the model is convex and searches for a globally optimal solution. In problems of type QP or MIQP (that is, the objective function includes one or more quadratic terms), this setting interacts with linearization switch for QP, MIQP, the linearization switch for QP and MIQP models.

When this parameter is set to 2, CPLEX searches for a solution that satisfies first-order optimality conditions. If the model is convex, such a solution is also globally optimal. If the model is not convex, the solution is not necessarily globally optimal. You can query the solution status to determine the kind of solution that CPLEX found.

This value 2 of the parameter is valid only for continuous QP models. If the model includes integer variables, CPLEX returns Error 1017, CPXERR_NOT_FOR_MIP. If the model includes quadratic constraints, CPLEX returns Error 1031, CPXERR_NOT_FOR_QCP.

When this parameter is set to 3, if the problem type is QP, CPLEX first changes the problem type to MIQP. CPLEX then solves the problem (whether originally QP or MIQP) to global optimality. In problems of type QP or MIQP, this setting interacts with linearization switch for QP, MIQP.

For MIQCP (that is, mixed integer models with positive semi-definite quadratic terms among the constraints), only the setting 0 (zero) Automatic is valid. Other values raise Error 1031, CPXERR_NOT_FOR_QCP.

Tip: When the value of this parameter is 3 (that is, you have instructed CPLEX to search for a globally optimal solution to a nonconvex QP or MIQP), then information about dual values is not available for the solution.

Table 49. Values

Value	Symbol	Meaning
0	`CPX_OPTIMALITYTARGET_AUTO`	Automatic: let CPLEX decide; **default**
1	`CPX_OPTIMALITYTARGET_OPTIMALCONVEX`	Searches for a globally optimal solution to a convex model.
2	`CPX_OPTIMALITYTARGET_FIRSTORDER`	Searches for a solution that satisfies first-order optimality conditions, but is not necessarily globally optimal.

Table 49. Values (continued)

Value	Symbol	Meaning
3	CPX_OPTIMALITYTARGET_OPTIMALGLOBAL	Searches for a globally optimal solution to a nonconvex model; changes problem type to MIQP if necessary.

parallel mode switch

Sets the parallel optimization mode. Possible modes are automatic, deterministic, and opportunistic.

Purpose

Parallel mode switch

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_Parallel	CPX_PARAM_PARALLELMODE
C++	IloCplex::Param::Parallel	ParallelMode (int)
Java	IloCplex.Param.Parallel	ParallelMode (int)
.NET	Cplex.Param.Parallel	ParallelMode (int)
OPL		parallelmode
Python	parameters.parallel	parallel
MATLAB	Cplex.Param.parallel	parallel
Interactive	parallel	parallel
Identifier	1109	1109

Description

Sets the parallel optimization mode. Possible modes are automatic, deterministic, and opportunistic.

In this context, *deterministic* means that multiple runs with the same model at the same parameter settings on the same platform will reproduce the same solution path and results. In contrast, *opportunistic* implies that even slight differences in timing among threads or in the order in which tasks are executed in different threads may produce a different solution path and consequently different timings or different solution vectors during optimization executed in parallel threads. In multithreaded applications, the opportunistic setting entails less synchronization between threads and consequently may provide better performance.

By default, CPLEX applies as much parallelism as possible while still achieving deterministic results. That is, when you run the same model twice on the same platform with the same parameter settings, you will see the same solution and optimization run. This condition is referred to as the *deterministic* mode.

More opportunities to exploit parallelism are available if you do not require determinism. In other words, CPLEX can find more possibilities for parallelism if you do not require an invariant, repeatable solution path and precisely the same solution vector. To use all available parallelism, you need to select the *opportunistic* parallel mode. In this mode, CPLEX will use all opportunities for parallelism in order to achieve best performance.

However, in opportunistic mode, the actual optimization may differ from run to run, including the solution time itself and the path traveled in the search.

Deterministic and sequential optimization

Parallel MIP optimization can be opportunistic or deterministic.

Parallel barrier optimization is only deterministic.

Concurrent optimization can be opportunistic or deterministic. In either mode, when six or more threads are available, concurrent optimization launches primal simplex, dual simplex, and barrier optimizers by default. If ten threads are available to CPLEX, concurrent optimization launches primal simplex, dual simplex, barrier, and sifting (if sifting is applicable to the current type of problem).

Tip: This parallel mode parameter also applies to **distributed** parallel MIP optimization. That is, you can instruct CPLEX to execute distributed parallel MIP optimization either opportunistically or deterministically.

Callbacks and MIP optimization

If callbacks other than informational callbacks are used for solving a MIP, the order in which the callbacks are called cannot be guaranteed to remain deterministic, not even in deterministic mode. Thus, to make sure of deterministic runs when the parallel mode parameter and the global default thread count parameter are at their default settings, CPLEX reverts to sequential solving of the MIP in the presence of query callbacks, diagnostic callbacks, or control callbacks.

Consequently, if your application invokes query, diagnostic, or control callbacks, and you still prefer deterministic parallel search, you can choose value 1 (one), overriding the automatic setting and turning on deterministic parallel search. It is then your responsibility to make sure that your callbacks do not perform operations that could lead to opportunistic behavior and are implemented in a thread-safe way. To meet these conditions, your application must **not** store and must **not** update any information in the callbacks.

Determinism versus opportunism

This parameter also allows you to turn off this default setting by choosing value -1 (minus one). Cases where you might wish to turn off deterministic search include situations where you want to take advantage of possibly faster performance of opportunistic parallel MIP optimization in multiple threads after you have confirmed that deterministic parallel MIP optimization produced the results you expected.

Table 50. Values

Value	Symbolic Constant Callable Library	Symbolic Constant Concert Technology	Meaning
-1	`CPX_PARALLEL_OPPORTUNISTIC`	`Opportunistic`	Enable opportunistic parallel search mode
0	`CPX_PARALLEL_AUTO`	`AutoParallel`	Automatic: let CPLEX decide whether to invoke deterministic or opportunistic search; **default**

Table 50. Values (continued)

Value	Symbolic Constant Callable Library	Symbolic Constant Concert Technology	Meaning
1	`CPX_PARALLEL_DETERMINISTIC`	`Deterministic`	Enable deterministic parallel search mode

See also: "global thread count" on page 157: `CPX_PARAM_THREADS`, `Threads`

simplex perturbation switch

Decides whether to perturb problems.

Purpose

Simplex perturbation switch

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_Simplex_Perturbation_Indicator	CPX_PARAM_PERIND
C++	IloCplex::Param::Simplex::Perturbation::Indicator	PerInd (bool)
Java	IloCplex.Param.Simplex.Perturbation.Indicator	PerInd (bool)
.NET	Cplex.Param.Simplex.Perturbation.Indicator	PerInd (bool)
OPL		perind
Python	parameters.simplex.perturbation.indicator	simplex.perturbation
MATLAB	Cplex.Param.simplex.perturbation.indicator	simplex.perturbation
Interactive	simplex perturbationlimit	simplex perturbationlimit
Identifier	1027	1027

Description

Decides whether to perturb problems.

Setting this parameter to 1 (one) causes all problems to be automatically perturbed as optimization begins. A setting of 0 (zero) allows CPLEX to decide dynamically, during solution, whether progress is slow enough to merit a perturbation. The situations in which a setting of 1 (one) helps are rare and restricted to problems that exhibit extreme degeneracy.

In the **Interactive Optimizer**, the command

```
set simplex perturbationlimit
```

accepts two arguments and actually sets two parameters simultaneously. The first argument is a switch or indicator; its value is yes to turn on perturbation or no to turn off perturbation. The second argument is a constant value to set an amount of perturbation. See the parameter "perturbation constant" on page 65 for more information about the value of this second argument.

Table 51. Values.

Value	bool	Symbol	Interactive	Meaning
0	false	`CPX_OFF`	no	Automatic: let CPLEX choose; **default**

Table 51. Values (continued).

Value	bool	Symbol	Interactive	Meaning
1	true	CPX_ON	yes	Turn on perturbation from beginning

simplex perturbation limit

Sets the number of degenerate iterations before perturbation is performed.

Purpose

Simplex perturbation limit

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_Simplex_Limits_Perturbation	CPX_PARAM_PERLIM
C++	IloCplex::Param::Simplex::Limits::Perturbation	PerLim (int)
Java	IloCplex.Param.Simplex.Limits.Perturbation	PerLim (int)
.NET	Cplex.Param.Simplex.Limits.Perturbation	PerLim (int)
OPL		perlim
Python	parameters.simplex.limits.perturbation	simplex.limits.perturbation
MATLAB	Cplex.Param.simplex.limits.perturbation	simplex.limits.perturbation
Interactive	simplex limits perturbation	simplex limits perturbation
Identifier	1028	1028

Description

Sets the number of degenerate iterations before perturbation is performed.

Table 52. Values

Value	Meaning
0	Automatic: let CPLEX choose; **default**
Any positive integer	Number of degenerate iterations before perturbation

deterministic time before starting to polish a feasible solution

Sets the amount of time expressed in deterministic ticks to spend during a normal mixed integer optimization after which CPLEX starts to polish a feasible solution

Purpose

Deterministic time before starting to polish a feasible solution

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_MIP_PolishAfter_DetTime	CPX_PARAM_POLISHAFTERDETTIME
C++	IloCplex::Param::MIP::PolishAfter::DetTime	PolishAfterDetTime (double)
Java	IloCplex.Param.MIP.PolishAfter.DetTime	PolishAfterDetTime (double)
.NET	Cplex.Param.MIP.PolishAfter.DetTime	PolishAfterDetTime (double)
OPL		polishafterdettime
Python	parameters.mip.polishafter.dettime	mip.polishafter.dettime

API	Parameter Name	Name prior to V12.6.0
MATLAB	Cplex.Param.mip.polishafter.dettime	mip.polishafter.dettime
Interactive	mip polishafter dettime	mip polishafter dettime
Identifier	2151	2151

Description

Tells CPLEX how much time (expressed in deterministic ticks) to spend during mixed integer optimization before CPLEX starts polishing a feasible solution. The default value (1.0E+75 seconds) is such that CPLEX does **not** start solution polishing by default.

Starting conditions

CPLEX must have a feasible solution in order to start polishing. It must also have certain internal structures in place to support solution polishing. Consequently, when the criterion specified by this parameter is met, CPLEX begins solution polishing only after these starting conditions are also met. That is, there may be a delay between the moment when the criterion specified by this parameter is met and when solution polishing starts.

Values

Any nonnegative value in deterministic ticks; **default**:1.0E+75 ticks.

See also

"time before starting to polish a feasible solution" on page 115

absolute MIP gap before starting to polish a feasible solution

Sets an absolute MIP gap after which CPLEX starts to polish a feasible solution

Purpose

Absolute MIP gap before starting to polish a feasible solution

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_MIP_PolishAfter_AbsMIPGap	CPX_PARAM_POLISHAFTEREPAGAP
C++	IloCplex::Param::MIP::PolishAfter::AbsMIPGap	PolishAfterEpAGap (double)
Java	IloCplex.Param.MIP.PolishAfter.AbsMIPGap	PolishAfterEpAGap (double)
.NET	Cplex.Param.MIP.PolishAfter.AbsMIPGap	PolishAfterEpAGap (double)
OPL		polishafterepagap
Python	parameters.mip.polishafter.absmipgap	mip.polishafter.absmipgap
MATLAB	Cplex.Param.mip.polishafter.absmipgap	mip.polishafter.absmipgap
Interactive	mip polishafter absmipgap	mip polishafter absmipgap
Identifier	2126	2126

Description

Sets an absolute MIP gap (that is, the difference between the best integer objective and the objective of the best node remaining) after which CPLEX stops branch-and-cut and begins polishing a feasible solution. The default value (0.0) is such that CPLEX does not invoke solution polishing by default.

Starting conditions

CPLEX must have a feasible solution in order to start polishing. It must also have certain internal structures in place to support solution polishing. Consequently, when the criterion specified by this parameter is met, CPLEX begins solution polishing only after these starting conditions are also met. That is, there may be a delay between the moment when the criterion specified by this parameter is met and when solution polishing starts.

Values

Any nonnegative value; **default**: 0.0.

See also

"absolute MIP gap tolerance" on page 61

relative MIP gap before starting to polish a feasible solution

Sets a relative MIP gap after which CPLEX starts to polish a feasible solution

Purpose

Relative MIP gap before starting to polish a solution

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_MIP_PolishAfter_MIPGap	CPX_PARAM_POLISHAFTEREPGAP
C++	IloCplex::Param::MIP::PolishAfter::MIPGap	PolishAfterEpGap (double)
Java	IloCplex.Param.MIP.PolishAfter.MIPGap	PolishAfterEpGap (double)
.NET	Cplex.Param.MIP.PolishAfter.MIPGap	PolishAfterEpGap (double)
OPL		polishafterepgap
Python	parameters.mip.polishafter.mipgap	mip.polishafter.mipgap
MATLAB	Cplex.Param.mip.polishafter.mipgap	mip.polishafter.mipgap
Interactive	mip polishafter mipgap	mip polishafter mipgap
Identifier	2127	2127

Description

Sets a relative MIP gap after which CPLEX will stop branch-and-cut and begin polishing a feasible solution. The default value (0.0) is such that CPLEX does not invoke solution polishing by default. The relative MIP gap is calculated like this:

```
|bestbound-bestinteger|/(1e-10+|bestinteger|)
```

Starting conditions

CPLEX must have a feasible solution in order to start polishing. It must also have certain internal structures in place to support solution polishing. Consequently, when the criterion specified by this parameter is met, CPLEX begins solution polishing only after these starting conditions are also met. That is, there may be a delay between the moment when the criterion specified by this parameter is met and when solution polishing starts.

Values

Any number from 0.0 to 1.0, inclusive; **default**: 0.0.

See also

"relative MIP gap tolerance" on page 61

MIP integer solutions to find before starting to polish a feasible solution

Sets the number of integer solutions to find after which CPLEX starts to polish a feasible solution

Purpose

MIP integer solutions to find before starting to polish a feasible solution

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_MIP_PolishAfter_Solutions	CPX_PARAM_POLISHAFTERINTSOL
C++	IloCplex::Param::MIP::PolishAfter::Solutions	PolishAfterIntSol (long)
Java	IloCplex.Param.MIP.PolishAfter.Solutions	PolishAfterIntSol (long)
.NET	Cplex.Param.MIP.PolishAfter.Solutions	PolishAfterIntSol (long)
OPL		polishafterintsol
Python	parameters.mip.polishafter.solutions	mip.polishafter.solutions
MATLAB	Cplex.Param.mip.polishafter.solutions	mip.polishafter.solutions
Interactive	mip polishafter solutions	mip polishafter solutions
Identifier	2129	2129

Description

Sets the number of integer solutions to find before CPLEX stops branch-and-cut and begins to polish a feasible solution. The default value is such that CPLEX does not invoke solution polishing by default.

Starting conditions

CPLEX must have a feasible solution in order to start polishing. It must also have certain internal structures in place to support solution polishing. Consequently, when the criterion specified by this parameter is met, CPLEX begins solution polishing only after these starting conditions are also met. That is, there may be a delay between the moment when the criterion specified by this parameter is met and when solution polishing starts.

Values

Any positive integer strictly greater than zero; zero is **not** allowed; **default**: 9223372036800000000

See also

"MIP integer solution-file switch and prefix" on page 78

nodes to process before starting to polish a feasible solution

Sets the number of nodes to process after which CPLEX starts to polish a feasible solution

Purpose

Nodes to process before starting to polish a feasible solution

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_MIP_PolishAfter_Nodes	CPX_PARAM_POLISHAFTERNODE
C++	IloCplex::Param::MIP::PolishAfter::Nodes	PolishAfterNode (long)
Java	IloCplex.Param.MIP.PolishAfter.Nodes	PolishAfterNode (long)
.NET	Cplex.Param.MIP.PolishAfter.Nodes	PolishAfterNode (long)
OPL		polishafternode
Python	parameters.mip.polishafter.nodes	mip.polishafter.nodes
MATLAB	Cplex.Param.mip.polishafter.nodes	mip.polishafter.nodes
Interactive	mip polishafter nodes	mip polishafter nodes
Identifier	2128	2128

Description

Sets the number of nodes processed in branch-and-cut before CPLEX starts solution polishing, if a feasible solution is available.

When this parameter is set to 0 (zero), CPLEX completes processing at the root; that is, it creates cuts and applies heuristics at the root.

When this parameter is set to 1 (one), it allows branching from the root; that is, nodes are created but not solved.

When no feasible solution is available yet, CPLEX explores more nodes than the number specified by this parameter.

Starting conditions

CPLEX must have a feasible solution in order to start polishing. It must also have certain internal structures in place to support solution polishing. Consequently, when the criterion specified by this parameter is met, CPLEX begins solution polishing only after these starting conditions are also met. That is, there may be a delay between the moment when the criterion specified by this parameter is met and when solution polishing starts.

Values

Any nonnegative integer; **default**: 9223372036800000000

See also

"MIP node limit" on page 101

time before starting to polish a feasible solution

Sets the amount of time in seconds to spend during a normal mixed integer optimization after which CPLEX starts to polish a feasible solution

Purpose

Time before starting to polish a feasible solution

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_MIP_PolishAfter_Time	CPX_PARAM_POLISHAFTERTIME
C++	IloCplex::Param::MIP::PolishAfter::Time	PolishAfterTime (double)
Java	IloCplex.Param.MIP.PolishAfter.Time	PolishAfterTime (double)
.NET	Cplex.Param.MIP.PolishAfter.Time	PolishAfterTime (double)
OPL		polishaftertime
Python	parameters.mip.polishafter.time	mip.polishafter.time
MATLAB	Cplex.Param.mip.polishafter.time	mip.polishafter.time
Interactive	mip polishafter time	mip polishafter time
Identifier	2130	2130

Description

Tells CPLEX how much time in seconds to spend during mixed integer optimization before CPLEX starts polishing a feasible solution. The default value (1.0E+75 seconds) is such that CPLEX does **not** start solution polishing by default.

Whether CPLEX measures CPU time or wall clock time (also known as real time) depends on the parameter "clock type for computation time" on page 43.

Starting conditions

CPLEX must have a feasible solution in order to start polishing. It must also have certain internal structures in place to support solution polishing. Consequently, when the criterion specified by this parameter is met, CPLEX begins solution polishing only after these starting conditions are also met. That is, there may be a delay between the moment when the criterion specified by this parameter is met and when solution polishing starts.

Values

Any nonnegative value in seconds; **default**:1.0E+75 seconds.

See also

"clock type for computation time" on page 43

"deterministic time before starting to polish a feasible solution" on page 111

time spent polishing a solution (deprecated)

Deprecated parameter

Purpose

Time spent polishing a solution (deprecated)

C Name	CPX_PARAM_POLISHTIME (double)
C++ Name	PolishTime (double)
Java Name	PolishTime (double)
.NET Name	PolishTime (double)

OPL Name	polishtime
Python Name	deprecated and not available
MATLAB Name	deprecated and not available
Interactive Optimizer	mip limit polishtime
Identifier	2066

Description

This **deprecated** parameter told CPLEX how much time in seconds to spend after a normal mixed integer optimization in polishing a solution. The default was zero, no polishing time.

Instead of this **deprecated** parameter, use one of the following parameters to control the effort that CPLEX spends in branch-and-cut before it begins polishing a feasible solution:

- "absolute MIP gap before starting to polish a feasible solution" on page 112
- "relative MIP gap before starting to polish a feasible solution" on page 113
- "MIP integer solutions to find before starting to polish a feasible solution" on page 114
- "nodes to process before starting to polish a feasible solution" on page 115
- "time before starting to polish a feasible solution" on page 115
- "optimizer time limit in seconds" on page 158

Values

Any nonnegative value in seconds; **default**: 0.0 (zero) seconds.

maximum number of solutions generated for solution pool by populate

Sets the maximum number of mixed integer programming (MIP) solutions generated for the solution pool during each call to the populate procedure.

Purpose

Maximum number of solutions generated for the solution pool by populate

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_MIP_Limits_Populate	CPX_PARAM_POPULATELIM
C++	IloCplex::Param::MIP::Limits::Populate	PopulateLim (int)
Java	IloCplex.Param.MIP.Limits.Populate	PopulateLim (int)
.NET	Cplex.Param.MIP.Limits.Populate	PopulateLim (int)
OPL		populatelim
Python	parameters.mip.limits.populate	mip.limits.populate
MATLAB	Cplex.Param.mip.limits.populate	mip.limits.populate
Interactive	mip limits populate	mip limits populate
Identifier	2108	2108

Description

Sets the maximum number of mixed integer programming (MIP) solutions generated for the solution pool during each call to the populate procedure. Populate stops when it has generated `PopulateLim` solutions. A solution is counted if it is valid for all filters, consistent with the relative and absolute pool gap parameters, and has not been rejected by the incumbent callback (if any exists), whether or not it improves the objective of the model.

In parallel, populate may not respect this parameter exactly due to disparities between threads. That is, it may happen that populate stops when it has generated a number of solutions slightly more than or slightly less than this limit because of differences in synchronization between threads.

This parameter does **not** apply to MIP optimization generally; it applies only to the populate procedure.

If you are looking for a parameter to control the number of solutions stored in the solution pool, consider instead the solution pool capacity parameter ("maximum number of solutions kept in solution pool" on page 147: `SolnPoolCapacity`, `CPX_PARAM_SOLNPOOLCAPACITY`).

Populate will stop before it reaches the limit set by this parameter if it reaches another limit, such as a time limit set by the user. Additional stopping criteria can be specified by these parameters:

- "relative gap for solution pool" on page 148: `SolnPoolGap`, `CPX_PARAM_SOLNPOOLGAP`
- "absolute gap for solution pool" on page 146: `SolnPoolAGap`, `CPX_PARAM_SOLNPOOLAGAP`
- "MIP node limit" on page 101: `NodeLim`, `CPX_PARAM_NODELIM`
- "optimizer time limit in seconds" on page 158: `TiLim`, `CPX_PARAM_TILIM`

Values

Any nonnegative integer; **default:** 20.

primal simplex pricing algorithm

Sets the primal simplex pricing algorithm.

Purpose

Primal simplex pricing algorithm

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_Simplex_PGradient	CPX_PARAM_PPRIIND
C++	IloCplex::Param::Simplex::PGradient	PPriInd (int)
Java	IloCplex.Param.Simplex.PGradient	PPriInd (int)
.NET	Cplex.Param.Simplex.PGradient	PPriInd (int)
OPL		ppriind
Python	parameters.simplex.pgradient	simplex.pgradient
MATLAB	Cplex.Param.simplex.pgradient	simplex.pgradient
Interactive	simplex pgradient	simplex pgradient
Identifier	1029	1029

Description

Sets the primal simplex pricing algorithm. The default pricing (0) usually provides the fastest solution time, but many problems benefit from alternative settings.

Table 53. Values

Value	Symbol	Meaning
-1	CPX_PPRIIND_PARTIAL	Reduced-cost pricing
0	CPX_PPRIIND_AUTO	Hybrid reduced-cost & devex pricing; **default**
1	CPX_PPRIIND_DEVEX	Devex pricing
2	CPX_PPRIIND_STEEP	Steepest-edge pricing
3	CPX_PPRIIND_STEEPQSTART	Steepest-edge pricing with slack initial norms
4	CPX_PPRIIND_FULL	Full pricing

presolve dual setting

Decides whether CPLEX presolve should pass the primal or dual linear programming problem to the linear programming optimization algorithm.

Purpose

Presolve dual setting

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_Preprocessing_Dual	CPX_PARAM_PREDUAL
C++	IloCplex::Param::Preprocessing::Dual	PreDual (int)
Java	IloCplex.Param.Preprocessing.Dual	PreDual (int)
.NET	Cplex.Param.Preprocessing.Dual	PreDual (int)
OPL		predual
Python	parameters.preprocessing.dual	preprocessing.dual
MATLAB	Cplex.Param.preprocessing.dual	preprocessing.dual
Interactive	preprocessing dual	preprocessing dual
Identifier	1044	1044

Description

Decides whether CPLEX presolve should pass the primal or dual linear programming problem to the linear programming optimization algorithm. By default, CPLEX chooses automatically.

If this parameter is set to 1 (one), the CPLEX presolve algorithm is applied to the primal problem, but the resulting dual linear program is passed to the optimizer. This is a useful technique for problems with more constraints than variables.

When this parameter is set to 0 (zero, its default value) or 1 (one, turned on), CPLEX disables crushing and uncrushing of the model by such routines as CPXuncrushx. To use CPXuncrushx effectively, you must set the value of this parameter to -1, turning off this feature.

Table 54. Values

Value	Meaning
-1	Turn off this feature
0	Automatic: let CPLEX choose; **default**
1	Turn on this feature

presolve switch

Decides whether CPLEX applies presolve during preprocessing.

Purpose

Presolve switch

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_Preprocessing_Presolve	CPX_PARAM_PREIND
C++	IloCplex::Param::Preprocessing::Presolve	PreInd (bool)
Java	IloCplex.Param.Preprocessing.Presolve	PreInd (bool)
.NET	Cplex.Param.Preprocessing.Presolve	PreInd (bool)
OPL		preind
Python	parameters.preprocessing.presolve	preprocessing.presolve
MATLAB	Cplex.Param.preprocessing.presolve	preprocessing.presolve
Interactive	preprocessing presolve	preprocessing presolve
Identifier	1030	1030

Description

Decides whether CPLEX applies presolve during preprocessing. When set to 1 (one), the default, this parameter invokes CPLEX presolve to simplify and reduce problems. In other words, this parameter turns on or off presolve during preprocessing.

To limit the number of passes that CPLEX carries out in presolve, use another parameter: "limit on the number of presolve passes made" on page 121.

Table 55. Values

Value	bool	Symbol	Interactive	Meaning
0	false	`CPX_OFF`	no	Do not apply presolve
1	true	`CPX_ON`	yes	Apply presolve; **default**

linear reduction switch

Decides whether linear or full reductions occur during preprocessing.

Purpose

Linear reduction switch

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_Preprocessing_Linear	CPX_PARAM_PRELINEAR
C++	IloCplex::Param::Preprocessing::Linear	PreLinear (int)
Java	IloCplex.Param.Preprocessing.Linear	PreLinear (int)
.NET	Cplex.Param.Preprocessing.Linear	PreLinear (int)
OPL		prelinear
Python	parameters.preprocessing.linear	preprocessing.linear
MATLAB	Cplex.Param.preprocessing.linear	preprocessing.linear
Interactive	preprocessing linear	preprocessing linear

API	Parameter Name	Name prior to V12.6.0
Identifier	1058	1058

Description

Decides whether linear or full reductions occur during preprocessing. If only linear reductions are performed, each variable in the original model can be expressed as a linear form of variables in the presolved model. This condition guarantees, for example, that users can add their own custom cuts to the presolved model.

If your application uses **lazy constraints** (for example, you have explicitly added lazy constraints to the model before optimization, or you have registered lazy constraints from a callback by means of a method or routine, such as `CPXsetlazyconstraintcallbackfunc`) then CPLEX turns **off** nonlinear reductions.

Table 56. Values.

Value	Meaning
0	Perform only linear reductions
1	Perform full reductions; **default**

limit on the number of presolve passes made

Limits the number of presolve passes that CPLEX makes during preprocessing.

Purpose

Limit on the number of presolve passes during preprocessing

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_Preprocessing_NumPass	CPX_PARAM_PREPASS
C++	IloCplex::Param::Preprocessing::NumPass	PrePass (int)
Java	IloCplex.Param.Preprocessing.NumPass	PrePass (int)
.NET	Cplex.Param.Preprocessing.NumPass	PrePass (int)
OPL		prepass
Python	parameters.preprocessing.numpass	preprocessing.numpass
MATLAB	Cplex.Param.preprocessing.numpass	preprocessing.numpass
Interactive	preprocessing numpass	preprocessing numpass
Identifier	1052	1052

Description

Limits the number of presolve passes that CPLEX makes during preprocessing.

When this parameter is set to a positive value, presolve is applied the specified number of times, or until no more reductions are possible.

At the default value of -1, presolve continues only if it seems to be helping.

When this parameter is set to zero, CPLEX does not enter its main presolve loop, but other reductions may occur, depending on settings of other parameters and characteristics of your model. In other words, setting this parameter to 0 (zero) is **not** equivalent to turning off the “presolve switch” on page 120

(CPX_PARAM_PREIND, PreInd). To turn off presolve, use the "presolve switch" on page 120 (CPX_PARAM_PREIND, PreInd) instead.

Table 57. Values

Value	Meaning
-1	Automatic: let CPLEX choose; presolve continues as long as helpful; **default**
0	Do not use presolve; other reductions may still occur
Any positive integer	Apply presolve specified number of times

node presolve switch

Decides whether node presolve should be performed at the nodes of a mixed integer programming (MIP) solution.

Purpose

Node presolve switch

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_MIP_Strategy_PresolveNode	CPX_PARAM_PRESLVND
C++	IloCplex::Param::MIP::Strategy::PresolveNode	PreslvNd (int)
Java	IloCplex.Param.MIP.Strategy.PresolveNode	PreslvNd (int)
.NET	Cplex.Param.MIP.Strategy.PresolveNode	PreslvNd (int)
OPL		preslvnd
Python	parameters.mip.strategy.presolvenode	mip.strategy.presolvenode
MATLAB	Cplex.Param.mip.strategy.presolvenode	mip.strategy.presolvenode
Interactive	mip strategy presolvenode	mip strategy presolvenode
Identifier	2037	2037

Description

Decides whether node presolve should be performed at the nodes of a mixed integer programming (MIP) solution. Node presolve can significantly reduce solution time for some models. The default setting is generally effective at deciding whether to apply node presolve, although runtimes can be reduced for some models by the user turning node presolve off.

Value	Meaning
-1	No node presolve
0	Automatic: let CPLEX choose; **default**
1	Force presolve at nodes
2	Perform probing on integer-infeasible variables
3	Perform aggressive node probing

simplex pricing candidate list size

Sets the maximum number of variables kept in the list of pricing candidates for the simplex algorithms.

Purpose

Simplex pricing candidate list size

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_Simplex_Pricing	CPX_PARAM_PRICELIM
C++	IloCplex::Param::Simplex::Pricing	PriceLim (int)
Java	IloCplex.Param.Simplex.Pricing	PriceLim (int)
.NET	Cplex.Param.Simplex.Pricing	PriceLim (int)
OPL		pricelim
Python	parameters.simplex.pricing	simplex.pricing
MATLAB	Cplex.Param.simplex.pricing	simplex.pricing
Interactive	simplex pricing	simplex pricing
Identifier	1010	1010

Description

Sets the maximum number of variables kept in the list of pricing candidates for the simplex algorithms.

Table 58. Values

Value	Meaning
0	Automatic: let CPLEX choose; **default**
Any positive integer	Number of pricing candidates

MIP probing level

Sets the amount of probing on variables to be performed before MIP branching.

Purpose

MIP probing level

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_MIP_Strategy_Probe	CPX_PARAM_PROBE
C++	IloCplex::Param::MIP::Strategy::Probe	Probe (int)
Java	IloCplex.Param.MIP.Strategy.Probe	Probe (int)
.NET	Cplex.Param.MIP.Strategy.Probe	Probe (int)
OPL		probe
Python	parameters.mip.strategy.probe	mip.strategy.probe
MATLAB	Cplex.Param.mip.strategy.probe	mip.strategy.probe
Interactive	mip strategy probe	mip strategy probe
Identifier	2042	2042

Description

Sets the amount of probing on variables to be performed before MIP branching. Higher settings perform more probing. Probing can be very powerful but very time-consuming at the start. Setting the parameter to values above the default of 0 (automatic) can result in dramatic reductions or dramatic increases in solution time, depending on the model.

Table 59. Values

Value	Meaning
-1	No probing
0	Automatic: let CPLEX choose; **default**
1	Moderate probing level
2	Aggressive probing level
3	Very aggressive probing level

deterministic time spent probing

Limits the amount of time (expressed in deterministic ticks) spent probing.

Purpose

Time spent probing, measured deterministically

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_MIP_Limits_ProbeDetTime	CPX_PARAM_PROBEDETTIME
C++	IloCplex::Param::MIP::Limits::ProbeDetTime	ProbeDetTime (double)
Java	IloCplex.Param.MIP.Limits.ProbeDetTime	ProbeDetTime (double)
.NET	Cplex.Param.MIP.Limits.ProbeDetTime	ProbeDetTime (double)
OPL		probedettime
Python	parameters.mip.limits.probedettime	mip.limits.probedettime
MATLAB	Cplex.Param.mip.limits.probedettime	mip.limits.probedettime
Interactive	mip limits probedettime	mip limits probedettime
Identifier	2150	2150

Description

Limits the amount of time (expressed in deterministic ticks) spent probing.

For a parameter limiting the amount of time spent probing in seconds, (rather than deterministic ticks) see "time spent probing" (CPX_PARAM_PROBETIME, ProbeTime).

Values

Any nonnegative number; **default**: 1e+75.

time spent probing

Limits the amount of time in seconds spent probing.

Purpose

Time spent probing

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_MIP_Limits_ProbeTime	CPX_PARAM_PROBETIME
C++	IloCplex::Param::MIP::Limits::ProbeTime	ProbeTime (double)
Java	IloCplex.Param.MIP.Limits.ProbeTime	ProbeTime (double)

API	Parameter Name	Name prior to V12.6.0
.NET	Cplex.Param.MIP.Limits.ProbeTime	ProbeTime (double)
OPL		probetime
Python	parameters.mip.limits.probetime	mip.limits.probetime
MATLAB	Cplex.Param.mip.limits.probetime	mip.limits.probetime
Interactive	mip limits probetime	mip limits probetime
Identifier	2065	2065

Description

Limits the amount of time in seconds spent probing.

For a parameter limiting the amount of time spent probing in deterministic ticks (rather than seconds) see "deterministic time spent probing" on page 124 (CPX_PARAM_PROBEDETTIME, ProbeDetTime).

Values

Any nonnegative number; **default**: 1e+75.

indefinite MIQP switch

Decides whether CPLEX will attempt to reformulate a MIQP or MIQCP model that contains only binary variables.

Purpose

Indefinite MIQP switch

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_Preprocessing_QPMakePSD	CPX_PARAM_QPMAKEPSDIND (int)
C++	IloCplex::Param::Preprocessing::QPMakePSD	QPmakePSDInd (bool)
Java	IloCplex.Param.Preprocessing.QPMakePSD	QPmakePSDInd (bool)
.NET	Cplex.Param.Preprocessing.QPMakePSD	QPmakePSDInd (bool)
OPL		qpmakepsdind
Python	parameters.preprocessing.qpmakepsd	preprocessing.qpmakepsd
MATLAB	Cplex.Param.preprocessing.qpmakepsd	preprocessing.qpmakepsd
Interactive	preprocessing qpmakepsd	preprocessing qpmakepsd
Identifier	4010	4010

Description

Decides whether CPLEX will attempt to reformulate a MIQP or MIQCP model that contains only binary variables. When this feature is active, adjustments will be made to the elements of a quadratic matrix that is not nominally positive semi-definite (PSD, as required by CPLEX for all QP and most QCP formulations), to make it PSD, and CPLEX will also attempt to tighten an already PSD matrix for better numerical behavior. The default setting of 1 (one) means yes, CPLEX should attempt to reformulate, but you can turn it off if necessary; most models benefit from the default setting.

Table 60. Values

Value	bool	Symbol	Interactive	Meaning
0	false	`CPX_OFF`	no	Turn off attempts to make binary model PSD
1	true	`CPX_ON`	yes	On: CPLEX attempts to make binary model PSD; **default**

QP Q-matrix nonzero read limit

Specifies a limit for the number of nonzero elements to read for an allocation of memory in a model with a quadratic matrix.

Purpose

QP Q-matrix nonzero read limit

API	Parameter Name	Name prior to V12.6.0
C 64-bit	CPXPARAM_Read_QPNonzeros	CPX_PARAM_QPNZREADLIM (CPXLONG)
C 32-bit	CPXPARAM_Read_QPNonzeros	CPX_PARAM_QPNZREADLIM (CPXINT)
C++	IloCplex::Param::Read::QPNonzeros	QPNzReadLim (CPXLONG)
Java	IloCplex.Param.Read.QPNonzeros	QPNzReadLim (CPXLONG)
.NET	Cplex.Param.Read.QPNonzeros	QPNzReadLim (CPXLONG)
OPL		not available
Python	parameters.read.qpnonzeros	read.qpnonzeros
MATLAB	Cplex.Param.read.qpnonzeros	read.qpnonzeros
Interactive	read qpnonzeros	read qpnonzeros
Identifier	4001	

Description

Specifies a limit for the number of nonzero elements to read for an allocation of memory in a model with a quadratic matrix.

This parameter does not restrict the size of a problem. Rather, it indirectly specifies the default amount of memory that will be pre-allocated before a problem is read from a file. If the limit is exceeded, more memory is automatically allocated.

Values

Any integer from 0 (zero) to `CPX_BIGINT` or `CPX_BIGLONG`, depending on the type of integer; **default**: 5 000.

deterministic time spent in ramp up during distributed parallel optimization

Limits the amount of time in deterministic ticks spent during ramp up of distributed parallel optimization.

Purpose

Time spent (measured in deterministic ticks) during ramp up of distributed parallel optimization

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_MIP_Limits_RampupDetTimeLimit	
C++	IloCplex::Param::MIP::Limits::RampupDetTimeLimit	
Java	IloCplex.Param.MIP.Limits.RampupDetTimeLimit	
.NET	Cplex.Param.MIP.Limits.RampupDetTimeLimit	
OPL		
Python	parameters.mip.limits.rampupdettimelimit	
MATLAB	Cplex.Param.mip.limits.rampupdettimelimit	
Interactive	mip limits rampupdettimelimit	
Identifier	2164	

Description

This parameters specifies a limit on the amount of time measured in deterministic ticks to spend in the ramp up phase of distributed parallel optimization. This parameter is effective **only** when the "ramp up duration" on page 128 parameter has a value of 0 (zero) or 1 (one), where 0 (zero) designates the default automatic value that CPLEX decides the ramp up duration, and 1 (one) designates dynamic ramp up. See "ramp up duration" on page 128 for more detail about the conditions for time limits in ramp up.

For a parameter limiting the amount of time spent in ramp up in seconds (rather than deterministic ticks) see "time spent in ramp up during distributed parallel optimization."

Values

The value 0 (zero) specifies that no time should be spent in ramp up.

Any positive number strictly greater than zero specifies a time limit in deterministic ticks.

The **default** value is BIGREAL deterministic ticks; that is, (1e+75) deterministic ticks on most platforms.

time spent in ramp up during distributed parallel optimization

Limits the amount of time in seconds spent during ramp up of distributed parallel optimization.

Purpose

Time spent in seconds during ramp up of distributed parallel optimization

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_MIP_Limits_RampupTimeLimit	
C++	IloCplex::Param::MIP::Limits::RampupTimeLimit	
Java	IloCplex.Param.MIP.Limits.RampupTimeLimit	
.NET	Cplex.Param.MIP.Limits.RampupTimeLimit	
OPL		
Python	parameters.mip.limits.rampuptimelimit	
MATLAB	Cplex.Param.mip.limits.rampuptimelimit	
Interactive	mip limits rampuptimelimit	
Identifier	2165	

Description

This parameters specifies a limit on the amount of time in seconds to spend in the ramp up phase of distributed parallel optimization. This parameter is effective **only** when the "ramp up duration" parameter has a value of 0 (zero) or 1 (one), where 0 (zero) designates the default automatic value that CPLEX decides the ramp up duration, and 1 (one) designates dynamic ramp up. See "ramp up duration" for more detail about the conditions for time limits in ramp up.

For a parameter limiting the amount of time spent in ramp up in deterministic ticks (rather than seconds) see "deterministic time spent in ramp up during distributed parallel optimization" on page 127.

Values

The value 0 (zero) specifies that no time should be spent in ramp up.

Any positive number strictly greater than zero specifies a time limit in seconds.

The **default** value is BIGREAL seconds; that is, (1e+75) seconds on most platforms.

ramp up duration

Customizes ramp up for distributed parallel optimization.

Purpose

Ramp up duration

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_DistMIP_Rampup_Duration	
C++	IloCplex::Param::DistMIP::Rampup::Duration	
Java	IloCplex.Param.DistMIP.Rampup.Duration	
.NET	Cplex.Param.DistMIP.Rampup.Duration	
OPL		
Python	parameters.distmip.rampup.duration	
MATLAB	Cplex.Param.distmip.rampup.duration	
Cplex class compatible		
Interactive	distmip rampup duration	

API	Parameter Name	Name prior to V12.6.0
Identifier	2163	

Description

During the ramp up phase of distributed parallel optimization, each worker applies different parameter settings to the same problem as the other workers. In other words, there is a competition among the workers to process the greatest number of nodes in parallel in the search tree of the distributed problem. At any given time, each worker is a candidate to be the winner of this competition.

This parameter enables you to customize the ramp up phase for your model. Its value has an impact on both timing parameters: "time spent in ramp up during distributed parallel optimization" on page 127 and "deterministic time spent in ramp up during distributed parallel optimization" on page 127.

When the value of this parameter is -1, CPLEX turns off ramp up and ignores both of the parameters "time spent in ramp up during distributed parallel optimization" on page 127 and "deterministic time spent in ramp up during distributed parallel optimization" on page 127. CPLEX directly begins distributed parallel tree search.

When the value of this parameter is 2, CPLEX observes ramp up with an infinite horizon. CPLEX ignores both of the parameters "time spent in ramp up during distributed parallel optimization" on page 127 and "deterministic time spent in ramp up during distributed parallel optimization" on page 127. CPLEX never switches to distributed parallel tree search. This situation is also known as concurrent mixed integer programming (concurrent MIP).

When the value of this parameter is 1 (one), CPLEX considers the values of both "time spent in ramp up during distributed parallel optimization" on page 127 and "deterministic time spent in ramp up during distributed parallel optimization" on page 127.

- If both ramp up timing parameters are at their default value (effectively, an infinite amount of time), then CPLEX terminates ramp up according to internal criteria before switching to distributed parallel tree search.
- If one or both of the ramp up timing parameters is set to a non default finite value, the CPLEX observes that time limit by executing ramp up for that given amount of time. If the two time limits differ, CPLEX observes the smaller time limit before terminating ramp up and switching to distributed parallel tree search.

When the value of this parameter remains at its default, 0 (zero), CPLEX considers the values of both timeing parameters "time spent in ramp up during distributed parallel optimization" on page 127 and "deterministic time spent in ramp up during distributed parallel optimization" on page 127.

- If at least one of the ramp up timing parameters is set to a finite value, then CPLEX behaves as it does when the value of this parameter is 1 (one): first ramping up, then switching to distributed parallel tree search.
- If both of the ramp up timing parameters are at their default value (effectively an infinite amount of time), then CPLEX behaves as it does when the value of this parameter is 2: concurrent MIP.

Tip: CPLEX behavior at default values is subject to change in future releases.

Values

Table 61. Values

Value	Symbol	Meaning
-1	CPX_RAMPUP_DISABLED	CPLEX turns off ramp up.
0	CPX_RAMPUP_AUTO	Automatic: let CPLEX decide.
1	CPX_RAMPUP_DYNAMIC	CPLEX dynamically switches to distributed tree search.
2	CPX_RAMPUP_INFINITE	CPLEX observes an infinite horizon for ramp up; also known as concurrent MIP optimization.

random seed

This parameter sets the random seed differently for diversity of solutions.

Purpose

Set random seed differently for diversity of solutions.

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_RandomSeed	CPX_PARAM_RANDOMSEED
C++	IloCplex::Param::RandomSeed	RandomSeed (int)
Java	IloCplex.Param.RandomSeed	RandomSeed (int)
.NET	Cplex.Param.RandomSeed	RandomSeed (int)
OPL		randomseed
Python	parameters.randomseed	randomseed
MATLAB	Cplex.Param.randomseed	randomseed
Interactive	randomseed	randomseed
Identifier	1124	1124

Description

This parameter makes it possible for your application to manage the random seed that CPLEX uses in some of its internal operations. Variation in the random seed can increase diversity of results.

Values

Any nonnegative integer; that is, an integer in the interval [0, BIGINT].

The **default** value of this parameter changes with each release.

primal and dual reduction type

Specifies whether primal reductions, dual reductions, both, or neither are performed during preprocessing.

Purpose

Primal and dual reduction type

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_Preprocessing_Reduce	CPX_PARAM_REDUCE
C++	IloCplex::Param::Preprocessing::Reduce	Reduce (int)
Java	IloCplex.Param.Preprocessing.Reduce	Reduce (int)
.NET	Cplex.Param.Preprocessing.Reduce	Reduce (int)
OPL		reduce
Python	parameters.preprocessing.reduce	preprocessing.reduce
MATLAB	Cplex.Param.preprocessing.reduce	preprocessing.reduce
Interactive	preprocessing reduce	preprocessing reduce
Identifier	1057	1057

Description

Specifies whether primal reductions, dual reductions, both, or neither are performed during preprocessing. These preprocessing reductions are also known as **presolve reductions**.

If your application uses **lazy constraints** (for example, you have explicitly added lazy constraints to the model before optimization, or you have registered lazy constraints from a callback by means of a method or routine, such as `CPXsetlazyconstraintcallbackfunc`) then CPLEX turns **off** dual reductions.

Table 62. Values.

Value	Symbol	Meaning
0	`CPX_PREREDUCE_NOPRIMALORDUAL`	No primal or dual reductions
1	`CPX_PREREDUCE_PRIMALONLY`	Only primal reductions
2	`CPX_PREREDUCE_DUALONLY`	Only dual reductions
3	`CPX_PREREDUCE_PRIMALANDDUAL`	Both primal and dual reductions; **default**

simplex refactoring frequency

Sets the number of iterations between refactoring of the basis matrix.

Purpose

Simplex refactoring frequency

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_Simplex_Refactor	CPX_PARAM_REINV
C++	IloCplex::Param::Simplex::Refactor	ReInv (int)
Java	IloCplex.Param.Simplex.Refactor	ReInv (int)
.NET	Cplex.Param.Simplex.Refactor	ReInv (int)
OPL		reinv
Python	parameters.simplex.refactor	simplex.refactor
MATLAB	Cplex.Param.simplex.refactor	simplex.refactor
Interactive	simplex refactor	simplex refactor
Identifier	1031	1031

Description

Sets the number of iterations between refactoring of the basis matrix.

Table 63. Values

Value	Meaning
0	Automatic: let CPLEX choose; **default**
Integer from 1 to 10 000	Number of iterations between refactoring of the basis matrix

relaxed LP presolve switch

Decides whether LP presolve is applied to the root relaxation in a mixed integer program (MIP).

Purpose

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_Preprocessing_Relax	CPX_PARAM_RELAXPREIND
C++	IloCplex::Param::Preprocessing::Relax	RelaxPreInd (int)
Java	IloCplex.Param.Preprocessing.Relax	RelaxPreInd (int)
.NET	Cplex.Param.Preprocessing.Relax	RelaxPreInd (int)
OPL		relaxpreind
Python	parameters.preprocessing.relax	preprocessing.relax
MATLAB	Cplex.Param.preprocessing.relax	preprocessing.relax
Interactive	preprocessing relax	preprocessing relax
Identifier	2034	2034

Relaxed LP presolve switch

Description

Decides whether LP presolve is applied to the root relaxation in a mixed integer program (MIP). Sometimes additional reductions can be made beyond any MIP presolve reductions that were already done. By default, CPLEX applies presolve to the initial relaxation in order to hasten time to the initial solution.

Value	Symbol	Meaning
-1		Automatic: let CPLEX choose; **default**
0	`CPX_OFF`	Off: do not use presolve on initial relaxation
1	`CPX_ON`	On: use presolve on initial relaxation

relative objective difference cutoff

Used to update the cutoff each time a mixed integer solution is found.

Purpose

Relative objective difference cutoff

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_MIP_Tolerances_RelObjDifference	CPX_PARAM_RELOBJDIF
C++	IloCplex::Param::MIP::Tolerances::RelObjDifference	RelObjDif (double)
Java	IloCplex.Param.MIP.Tolerances.RelObjDifference	RelObjDif (double)
.NET	Cplex.Param.MIP.Tolerances.RelObjDifference	RelObjDif (double)
OPL		relobjdif
Python	parameters.mip.tolerances.relobjdifference	mip.tolerances.relobjdifference
MATLAB	Cplex.Param.mip.tolerances.relobjdifference	mip.tolerances.relobjdifference
Interactive	mip tolerances relobjdifference	mip tolerances relobjdifference
Identifier	2022	2022

Description

Used to update the cutoff each time a mixed integer solution is found. The value is multiplied by the absolute value of the integer objective and subtracted from (added to) the newly found integer objective when minimizing (maximizing). This computation forces the mixed integer optimization to ignore integer solutions that are not at least this amount better than the one found so far.

The relative objective difference parameter can be adjusted to improve problem solving efficiency by limiting the number of nodes; however, setting this parameter at a value other than zero (the default) can cause some integer solutions, including the true integer optimum, to be missed.

If both the relative objective difference and the "absolute objective difference cutoff" on page 104 (`CPX_PARAM_OBJDIF`, `ObjDif`) are nonzero, the value of the absolute objective difference is used.

Values

Any number from 0.0 to 1.0; **default**: 0.0.

See also

"absolute objective difference cutoff" on page 104

number of attempts to repair infeasible MIP start

Limits the attempts to repair an infeasible MIP start.

Purpose

Number of attempts to repair infeasible MIP start

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_MIP_Limits_RepairTries	CPX_PARAM_REPAIRTRIES
C++	IloCplex::Param::MIP::Limits::RepairTries	RepairTries (long)
Java	IloCplex.Param.MIP.Limits.RepairTries	RepairTries (long)
.NET	Cplex.Param.MIP.Limits.RepairTries	RepairTries (long)
OPL		repairtries

API	Parameter Name	Name prior to V12.6.0
Python	parameters.mip.limits.repairtries	mip.limits.repairtries
MATLAB	Cplex.Param.mip.limits.repairtries	mip.limits.repairtries
Interactive	mip limits repairtries	mip limits repairtries
Identifier	2067	2067

Description

Limits the attempts to repair an infeasible MIP start. This parameter lets you tell CPLEX whether and how many times it should try to repair an infeasible MIP start that you supplied. The parameter has no effect if the MIP start you supplied is feasible. It has no effect if no MIP start was supplied.

Table 64. Values

Value	Meaning
-1	None: do not try to repair
0	Automatic: let CPLEX choose; **default**
Any positive integer	Number of attempts

MIP repeat presolve switch

Specifies whether to re-apply presolve, with or without cuts, to a MIP model after processing at the root is otherwise complete.

Purpose

Reapply presolve after processing the root node

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_Preprocessing_RepeatPresolve	CPX_PARAM_REPEATPRESOLVE
C++	IloCplex::Param::Preprocessing::RepeatPresolve	RepeatPresolve (int)
Java	IloCplex.Param.Preprocessing.RepeatPresolve	RepeatPresolve (int)
.NET	Cplex.Param.Preprocessing.RepeatPresolve	RepeatPresolve (int)
OPL		repeatpresolve
Python	parameters.preprocessing.repeatpresolve	preprocessing.repeatpresolve
MATLAB	Cplex.Param.preprocessing.repeatpresolve	preprocessing.repeatpresolve
Interactive	preprocessing repeatpresolve	preprocessing repeatpresolve
Identifier	2064	2064

Description

Specifies whether to re-apply presolve, with or without cuts, to a MIP model after processing at the root is otherwise complete.

Table 65. Values.

Value	Symbol
-1	Automatic: let CPLEX choose; **default**
0	Turn off represolve
1	Represolve without cuts
2	Represolve with cuts

Table 65. Values (continued).

Value	Symbol
3	Represolve with cuts and allow new root cuts

RINS heuristic frequency

Decides how often to apply the relaxation induced neighborhood search (RINS) heuristic.

Purpose

RINS heuristic frequency

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_MIP_Strategy_RINSHeur	CPX_PARAM_RINSHEUR
C++	IloCplex::Param::MIP::Strategy::RINSHeur	RINSHeur (long)
Java	IloCplex.Param.MIP.Strategy.RINSHeur	RINSHeur (long)
.NET	Cplex.Param.MIP.Strategy.RINSHeur	RINSHeur (long)
OPL		rinsheur
Python	parameters.mip.strategy.rinsheur	mip.strategy.rinsheur
MATLAB	Cplex.Param.mip.strategy.rinsheur	mip.strategy.rinsheur
Interactive	mip strategy rinsheur	mip strategy rinsheur
Identifier	2061	2061

Description

Decides how often to apply the relaxation induced neighborhood search (RINS) heuristic. This heuristic attempts to improve upon the best solution found so far. It will not be applied until CPLEX has found at least one incumbent solution.

Setting the value to -1 turns off the RINS heuristic. Setting the value to 0 (zero), the default, applies the RINS heuristic at an interval chosen automatically by CPLEX. Setting the value to a positive number applies the RINS heuristic at the requested node interval. For example, setting `RINSHeur` to 20 dictates that the RINS heuristic be called at node 0, 20, 40, 60, etc.

RINS is a powerful heuristic for finding high quality feasible solutions, but it may be expensive.

Table 66. Values

Value	Meaning
-1	None: do not apply RINS heuristic
0	Automatic: let CPLEX choose; **default**
Any positive integer	Frequency to apply RINS heuristic

Reformulation Linearization Technique (RLT) cuts

Controls the addition of RLT cuts for nonconvex QP solved to global optimality.

Purpose

Reformulation Linearization Technique (RLT) cuts switch

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_MIP_Cuts_RLT	CPX_PARAM_RLTCUTS
C++	IloCplex::Param::MIP::Cuts::RLT	
Java	IloCplex.Param.MIP.Cuts.RLT	
.NET	Cplex.Param.MIP.Cuts.RLT	
OPL		
Python	parameters.mip.cuts.rlt	
MATLAB	Cplex.Param.mip.cuts.rlt	
Interactive	mip cuts rlt	
Identifier	2196	

Description

This parameter controls the addition of cuts based on the Reformulation Linearization Technique (RLT) for **nonconvex** quadratic programs (QP) or mixed integer quadratic programs (MIQP) solved to **global optimality**. That is, the "optimality target" on page 106 parameter must be set to 3 (CPX_OPTIMALITYTARGET_OPTIMALGLOBAL).

For a definition of the cuts based on a Reformulation Linearization Technique (RLT), see the topic Reformulation Linearization Technique (RLT) cuts in the general topic Cuts in the *CPLEX User's Manual*. The table Parameters for controlling cuts, also in the user's manual, includes links to the documentation of other parameters affecting other types of cuts.

Value	Meaning
-1	Do not generate RLT cuts
0	Automatic: let CPLEX decide; **default**
1	Generate RLT cuts moderately
2	Generate RLT cuts aggressively
3	Generate RLT cuts very aggressively

algorithm for continuous linear problems

Controls which algorithm is used to solve continuous linear models or to solve the root relaxation of a MIP.

Purpose

Solution algorithm for continuous problems

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_LPMethod	CPX_PARAM_LPMETHOD
C++	IloCplex::Param::RootAlgorithm	RootAlg (int)
Java	IloCplex.Param.RootAlgorithm	RootAlg (int)
.NET	Cplex.Param.RootAlgorithm	RootAlg (int)
OPL		rootalg
Python	parameters.lpmethod	lpmethod

API	Parameter Name	Name prior to V12.6.0
MATLAB Cplex class API	Cplex.Param.lpmethod	lpmethod
MATLAB CPLEX Toolbox compatible	lpmethod	lpmethod
MATLAB Optimization Toolbox compatible	Simplex	Simplex
Interactive	lpmethod	lpmethod
Identifier	1062	1062

Description

Controls which algorithm CPLEX uses to solve continuous linear models (LPs).

In the object-oriented APIs (such as C++, Java, or .NET APIs), this parameter, as `RootAlg`, also controls which algorithm CPLEX uses to solve the root relaxation of a MIP.

In the C API and the Interactive Optimizer, there are separate parameters to control LP, QP, and MIP optimizers, depending on the problem type. See, for example, “algorithm for continuous quadratic optimization” on page 138 or “algorithm for initial MIP relaxation” on page 139.

In all cases, the **default** setting is 0 (zero). The default setting means that CPLEX will select the algorithm in a way that should give best overall performance.

For specific problem classes, the following details document the automatic settings. Note that future versions of CPLEX could adopt different strategies. Therefore, if you select any nondefault settings, you should review them periodically.

Currently, the behavior of the automatic setting is that CPLEX almost always invokes the dual simplex algorithm when it is solving an LP model from scratch. When it is continuing from an advanced basis, it will check whether the basis is primal or dual feasible, and choose the primal or dual simplex algorithm accordingly.

If multiple threads have been requested, in either deterministic or opportunistic mode, the concurrent optimization algorithm is selected by the automatic setting when CPLEX is solving a continuous linear programming model (LP) from scratch.

When three or more threads are available, and you select concurrent optimization for the value of this parameter, its behavior depends on whether parallel mode is opportunistic or deterministic (default parallel mode). Concurrent optimization in **opportunistic** parallel mode runs the dual simplex optimizer on one thread, the primal simplex optimizer on a second thread, the parallel barrier optimizer on a third thread and on any additional available threads. In contrast, concurrent optimization in **deterministic** parallel mode runs the dual optimizer on one thread and the parallel barrier optimizer on any additional available threads.

The automatic setting may be expanded in the future so that CPLEX chooses the algorithm based on additional problem characteristics.

Tip: In the presence of a linear programming (LP) **callback**, CPLEX does **not** invoke concurrent optimization, regardless of the value of this parameter.

Table 67. Values

Value	Symbol	Meaning
0	`CPX_ALG_AUTOMATIC`	Automatic: let CPLEX choose; **default**
1	`CPX_ALG_PRIMAL`	Primal simplex
2	`CPX_ALG_DUAL`	Dual simplex
3	`CPX_ALG_NET`	Network simplex
4	`CPX_ALG_BARRIER`	Barrier
5	`CPX_ALG_SIFTING`	Sifting
6	`CPX_ALG_CONCURRENT`	Concurrent (Dual, Barrier, and Primal in opportunistic parallel mode; Dual and Barrier in deterministic parallel mode)

algorithm for continuous quadratic optimization

Sets which algorithm to use when the C routine `CPXqpopt` (or the command `optimize` in the Interactive Optimizer) is invoked.

Purpose

Algorithm for continuous quadratic optimization

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_QPMethod	CPX_PARAM_QPMETHOD
C++	IloCplex::Param::RootAlgorithm	RootAlg (int)
Java	IloCplex.Param.RootAlgorithm	RootAlg (int)
.NET	Cplex.Param.RootAlgorithm	RootAlg (int)
OPL		rootalg
Python	parameters.qpmethod	qpmethod
MATLAB	Cplex.Param.qpmethod	qpmethod
Interactive	qpmethod	qpmethod
Identifier	1063	1063

Description

Sets which algorithm to use when the C routine CPXXqpopt and CPXqpopt (or the command `optimize` in the Interactive Optimizer) is invoked.

Currently, the behavior of the Automatic setting is that CPLEX invokes the Barrier Optimizer for continuous QP models. The Automatic setting may be expanded in the future so that CPLEX chooses the algorithm based on additional problem characteristics.

Tip: In the presence of a linear programming (LP) **callback**, CPLEX does **not** invoke concurrent optimization, regardless of the value of this parameter.

Table 68. Values

Value	Symbol	Meaning
0	`CPX_ALG_AUTOMATIC`	Automatic: let CPLEX choose; **default**
1	`CPX_ALG_PRIMAL`	Use the primal simplex optimizer.
2	`CPX_ALG_DUAL`	Use the dual simplex optimizer.
3	`CPX_ALG_NET`	Use the network optimizer.
4	`CPX_ALG_BARRIER`	Use the barrier optimizer.
6	`CPX_ALG_CONCURRENT`	Use the concurrent optimizer.

algorithm for initial MIP relaxation

Sets which continuous optimizer will be used to solve the initial relaxation of a MIP.

Purpose

MIP starting algorithm

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_MIP_Strategy_StartAlgorithm	CPX_PARAM_STARTALG
C++	IloCplex::Param::RootAlgorithm	RootAlg
Java	IloCplex.Param.RootAlgorithm	RootAlg
.NET	Cplex.Param.RootAlgorithm	RootAlg
OPL		rootalg
Python	parameters.mip.strategy.startalgorithm	mip.strategy.startalgorithm
MATLAB	Cplex.Param.mip.strategy.startalgorithm	mip.strategy.startalgorithm
Interactive	mip strategy startalgorithm	mip strategy startalgorithm
Identifier	2025	2025

Description

Sets which continuous optimizer will be used to solve the initial relaxation of a MIP.

The default Automatic setting (0 zero) of this parameter currently selects the concurrent optimizer for root relaxations of mixed integer linear programming models (MILP) and selects the dual simplex optimizer for root relaxations of mixed integer quadratic programming models (MIQP). The Automatic setting may be expanded in the future so that CPLEX chooses the algorithm based on additional characteristics of the model.

For MILP (integer constraints and otherwise continuous variables), all settings are permitted.

For MIQP (integer constraints and positive semi-definite quadratic terms in the objective), settings 5 (Sifting) and 6 (Concurrent) are **not** implemented; if you happen to choose them, setting 5 (Sifting) reverts to 0 (zero) and setting 6 (Concurrent) reverts to 4.

For MIQCP (integer constraints and positive semi-definite quadratic terms among the constraints), only the setting 0 (zero) Automatic is permitted.

Table 69. Values

Value	Symbol	Meaning
0	`CPX_ALG_AUTOMATIC`	Automatic: let CPLEX choose; **default**
1	`CPX_ALG_PRIMAL`	Primal Simplex
2	`CPX_ALG_DUAL`	Dual Simplex
3	`CPX_ALG_NET`	Network Simplex
4	`CPX_ALG_BARRIER`	Barrier
5	`CPX_ALG_SIFTING`	Sifting
6	`CPX_ALG_CONCURRENT`	Concurrent (Dual, Barrier, and Primal in opportunistic mode; Dual and Barrier in deterministic mode)

auxiliary root threads

Partitions the number of threads to manage tasks at the root node.

Purpose

Auxiliary root threads

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_MIP_Limits_AuxRootThreads	CPX_PARAM_AUXROOTTHREADS
C++	IloCplex::Param::MIP::Limits::AuxRootThreads	AuxRootThreads
Java	IloCplex.Param.MIP.Limits.AuxRootThreads	AuxRootThreads
.NET	Cplex.Param.MIP.Limits.AuxRootThreads	AuxRootThreads
OPL		auxrootthreads
Python	parameters.mip.limits.auxrootthreads	mip.limits.auxrootthreads
MATLAB	Cplex.Param.mip.limits.auxrootthreads	mip.limits.auxrootthreads
Interactive	mip limits auxrootthreads n	mip limits auxrootthreads n
Identifier	2139	2139

Description

Partitions the number of threads for CPLEX to use for auxiliary tasks while it solves the root node of a problem.

On a system that offers N global threads, if you set this parameter to n, where

```
N > n > 0
```

then CPLEX uses at most n threads for auxiliary tasks and at most N-n threads to solve the root node.

See also the parameter "global thread count" on page 157, for more general information about **parallel solving and threads**.

Tip:

You cannot set n, the value of this parameter, to a value greater than or equal to N, the number of global threads offered on your system.

Independent of the auxiliary root threads parameter, CPLEX will never use more threads than those defined by the "global thread count" on page 157 parameter, whether that parameter is 0 (zero), its default value, or N, a value that you set. CPLEX also makes sure that there is at least one thread available for the main root tasks. For example, if you set the global threads parameter to 3 and the auxiliary root threads parameter to 4, CPLEX still uses only two threads for auxiliary root tasks in order to keep one thread available for the main root tasks.

At its **default** value, 0 (zero), CPLEX automatically chooses the number of threads to use for the primary root tasks and for auxiliary tasks. The number of threads that CPLEX uses to solve the root node depends on these factors:

- the number of threads available to your application on your system (for example, as a result of limited resources or competition with other applications);
- the value of the "global thread count" on page 157 parameter (`CPX_PARAM_THREADS`, `Threads`).

Table 70. Values

Value	Meaning
-1	Off: do not use additional threads for auxiliary tasks.
0	Automatic: let CPLEX choose the number of threads to use; **default**
N > n > 0	Use n threads for auxiliary root tasks

constraint (row) read limit

Specifies a limit for the number of rows (constraints) to read for an allocation of memory.

Purpose

Constraint (row) read limit

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_Read_Constraints	CPX_PARAM_ROWREADLIM
C++	IloCplex::Param::Read::Constraints	RowReadLim (int)
Java	IloCplex.Param.Read.Constraints	RowReadLim (int)
.NET	Cplex.Param.Read.Constraints	RowReadLim (int)
OPL		not available
Python	parameters.read.constraints	read.constraints
MATLAB	Cplex.Param.read.constraints	read.constraints
Interactive	read constraints	read constraints
Identifier	1021	1021

Description

Specifies a limit for the number of rows (constraints) to read for an allocation of memory.

This parameter does not restrict the size of a problem. Rather, it indirectly specifies the default amount of memory that will be pre-allocated before a problem is read from a file. If the limit is exceeded, more memory is automatically allocated.

Values

Any integer from 0 (zero) to `CPX_BIGINT` ; **default**: 30 000.

scale parameter

Decides how to scale the problem matrix.

Purpose

Scale parameter

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_Read_Scale	CPX_PARAM_SCAIND
C++	IloCplex::Param::Read::Scale	ScaInd (int)
Java	IloCplex.Param.Read.Scale	ScaInd (int)
.NET	Cplex.Param.Read.Scale	ScaInd (int)
OPL		scaind
Python	parameters.read.scale	read.scale
MATLAB	Cplex.Param.read.scale	read.scale
Interactive	read scale	read scale
Identifier	1034	1034

Description

Decides how to scale the problem matrix.

Table 71. Values

Value	Meaning
-1	No scaling
0	Equilibration scaling; **default**
1	More aggressive scaling

messages to screen switch

Decides whether or not results are displayed on screen in an application of the C API, MATLAB API, or MATLAB Toolbox.

Purpose

Messages to screen switch

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_ScreenOutput	CPX_PARAM_SCRIND
MATLAB Cplex class API	DisplayFunc	DisplayFunc
MATLAB Optimization Toolbox	diagnostics or Diagnostics	diagnostics or Diagnostics
Identifier	1035	1035

Description

Decides whether or not results are displayed on screen in a application of the Callable Library (C API), MATLAB API, or MATLAB Toolbox.

To turn off output to the screen, in a **C++** application, where `cplex` is an instance of the class `IloCplex` and `env` is an instance of the class `IloEnv` , the environment, use `cplex.setOut(env.getNullStream())` .

In a **Java** application, use `cplex.setOut(null)`.

In a **.NET** application, use `Cplex.SetOut(Null)`.

In a **Python** application, where `c` is an instance of the class `cplex.Cplex`, use `c.set_results_stream(None)`.

Table 72. Values

Value	Symbol	Meaning
0	`CPX_OFF`	Turn off display of messages to screen; **default**
1	`CPX_ON`	Display messages on screen

sifting subproblem algorithm

Sets the algorithm to be used for solving sifting subproblems.

Purpose

Sifting subproblem algorithm

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_Sifting_Algorithm	CPX_PARAM_SIFTALG
C++	IloCplex::Param::Sifting::Algorithm	SiftAlg (int)
Java	IloCplex.Param.Sifting.Algorithm	SiftAlg (int)
.NET	Cplex.Param.Sifting.Algorithm	SiftAlg (int)
OPL		siftalg
Python	parameters.sifting.algorithm	sifting.algorithm
MATLAB	Cplex.Param.sifting.algorithm	sifting.algorithm
Interactive	sifting algorithm	sifting algorithm

API	Parameter Name	Name prior to V12.6.0
Identifier	1077	1077

Description

Sets the algorithm to be used for solving sifting subproblems. The default automatic setting will typically use a mix of barrier and primal simplex.

For more information about CPLEX sifting, see the topic Sifting optimizer in the *CPLEX User's Manual*.

Table 73. Values

Value	Symbol	Meaning
0	`CPX_ALG_AUTOMATIC`	Automatic: let CPLEX choose; **default**
1	`CPX_ALG_PRIMAL`	Primal Simplex
2	`CPX_ALG_DUAL`	Dual Simplex
3	`CPX_ALG_NET`	Network Simplex
4	`CPX_ALG_BARRIER`	Barrier

sifting information display

Sets the amount of information to display about the progress of sifting.

Purpose

Sifting information display

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_Sifting_Display	CPX_PARAM_SIFTDISPLAY
C++	IloCplex::Param::Sifting::Display	SiftDisplay (int)
Java	IloCplex.Param.Sifting.Display	SiftDisplay (int)
.NET	Cplex.Param.Sifting.Display	SiftDisplay (int)
OPL		siftdisplay
Python	parameters.sifting.display	sifting.display
MATLAB	Cplex.Param.sifting.display	sifting.display
Interactive	sifting display	sifting display
Identifier	1076	1076

Description

Sets the amount of information to display about the progress of sifting.

For more information about CPLEX sifting, see the topic Sifting optimizer in the *CPLEX User's Manual*.

Table 74. Values

Value	Meaning
0	No display of sifting information
1	Display major iterations; **default**

Table 74. Values (continued)

Value	Meaning
2	Display LP subproblem information within each sifting iteration

upper limit on sifting iterations

Sets the maximum number of sifting iterations that may be performed if convergence to optimality has not been reached.

Purpose

Upper limit on sifting iterations

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_Sifting_Iterations	CPX_PARAM_SIFTITLIM
C++	IloCplex::Param::Sifting::Iterations	SiftItLim (long)
Java	IloCplex.Param.Sifting.Iterations	SiftItLim (long)
.NET	Cplex.Param.Sifting.Iterations	SiftItLim (long)
OPL		siftitlim
Python	parameters.sifting.iterations	sifting.iterations
MATLAB	Cplex.Param.sifting.iterations	sifting.iterations
Interactive	sifting iterations	sifting iterations
Identifier	1078	1078

Description

Sets the maximum number of sifting iterations that may be performed if convergence to optimality has not been reached.

Values

Any nonnegative integer; **default**: 9223372036800000000.

simplex iteration information display

Sets how often CPLEX reports about iterations during simplex optimization.

Purpose

Simplex iteration information display

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_Simplex_Display	CPX_PARAM_SIMDISPLAY
C++	IloCplex::Param::Simplex::Display	SimDisplay (int)
Java	IloCplex.Param.Simplex.Display	SimDisplay (int)
.NET	Cplex.Param.Simplex.Display	SimDisplay (int)
OPL		simdisplay
Python	parameters.simplex.display	simplex.display
MATLAB	Cplex.Param.simplex.display	simplex.display
Interactive	simplex display	simplex display
Identifier	1019	1019

Description

Sets how often CPLEX reports about iterations during simplex optimization.

Table 75. Values

Value	Meaning
0	No iteration messages until solution
1	Iteration information after each refactoring; **default**
2	Iteration information for each iteration

simplex singularity repair limit

Restricts the number of times CPLEX attempts to repair the basis when singularities are encountered during the simplex algorithm.

Purpose

Simplex singularity repair limit

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_Simplex_Limits_Singularity	CPX_PARAM_SINGLIM
C++	IloCplex::Param::Simplex::Limits::Singularity	SingLim (int)
Java	IloCplex.Param.Simplex.Limits.Singularity	SingLim (int)
.NET	Cplex.Param.Simplex.Limits.Singularity	SingLim (int)
OPL		singlim
Python	parameters.simplex.limits.singularity	simplex.limits.singularity
MATLAB	Cplex.Param.simplex.limits.singularity	simplex.limits.singularity
Interactive	simplex limits singularity	simplex limits singularity
Identifier	1037	1037

Description

Restricts the number of times CPLEX attempts to repair the basis when singularities are encountered during the simplex algorithm. When this limit is exceeded, CPLEX replaces the current basis with the best factorable basis that has been found.

Values

Any nonnegative integer; **default**: 10.

absolute gap for solution pool

Sets an absolute tolerance on the objective value for the solutions in the solution pool.

Purpose

Absolute gap for solution pool

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_MIP_Pool_AbsGap	CPX_PARAM_SOLNPOOLAGAP

API	Parameter Name	Name prior to V12.6.0
C++	IloCplex::Param::MIP::Pool::AbsGap	SolnPoolAGap (double)
Java	IloCplex.Param.MIP.Pool.AbsGap	SolnPoolAGap (double)
.NET	Cplex.Param.MIP.Pool.AbsGap	SolnPoolAGap (double)
OPL		solnpoolagap
Python	parameters.mip.pool.absgap	mip.pool.absgap
MATLAB	Cplex.Param.mip.pool.absgap	mip.pool.absgap
Interactive	mip pool absgap	mip pool absgap
Identifier	2106	2106

Description

Sets an absolute tolerance on the objective value for the solutions in the solution pool. Solutions that are worse (either greater in the case of a minimization, or less in the case of a maximization) than the objective of the incumbent solution according to this measure are not kept in the solution pool.

Values of the solution pool *absolute* gap (`SolnPoolAGap` or `CPX_PARAM_SOLNPOOLAGAP`) and the solution pool *relative* gap ("relative gap for solution pool" on page 148: `SolnPoolGap` or `CPX_PARAM_SOLNPOOLGAP`) may differ: For example, you may specify that solutions must be within 15 units by means of the solution pool absolute gap and also within 1% of the incumbent by means of the solution pool relative gap. A solution is accepted in the pool only if it is valid for both the relative and the absolute gaps.

The solution pool absolute gap parameter can also be used as a stopping criterion for the populate procedure: if populate cannot enumerate any more solutions that satisfy this objective quality, then it will stop. In the presence of both an absolute and a relative solution pool gap parameter, populate will stop when the smaller of the two is reached.

Values

Any nonnegative real number; **default**: 1.0e+75.

maximum number of solutions kept in solution pool

Limits the number of solutions kept in the solution pool

Purpose

Maximum number of solutions kept in the solution pool

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_MIP_Pool_Capacity	CPX_PARAM_SOLNPOOLCAPACITY
C++	IloCplex::Param::MIP::Pool::Capacity	SolnPoolCapacity (int)
Java	IloCplex.Param.MIP.Pool.Capacity	SolnPoolCapacity (int)
.NET	Cplex.Param.MIP.Pool.Capacity	SolnPoolCapacity (int)
OPL		solnpoolcapacity
Python	parameters.mip.pool.capacity	mip.pool.capacity
MATLAB	Cplex.Param.mip.pool.capacity	mip.pool.capacity
Interactive	mip pool capacity	mip pool capacity
Identifier	2103	2103

Description

Sets the maximum number of solutions kept in the solution pool. At most, SolnPoolCapacity solutions will be stored in the pool. Superfluous solutions are managed according to the strategy set by the "solution pool replacement strategy" on page 151 parameter (SolnPoolReplace, CPX_PARAM_SOLNPOOLREPLACE).

The optimization (whether by MIP optimization or the populate procedure) will not stop if more than SolnPoolCapacity solutions are generated. Instead, stopping criteria can be specified by these parameters:

- "maximum number of solutions generated for solution pool by populate" on page 117 (PopulateLim, CPX_PARAM_POPULATELIM)
- "relative gap for solution pool" (SolnPoolGap, CPX_PARAM_SOLNPOOLGAP)
- "absolute gap for solution pool" on page 146 (SolnPoolAGap, CPX_PARAM_SOLNPOOLAGAP)
- "MIP node limit" on page 101 (NodeLim, CPX_PARAM_NODELIM)
- "optimizer time limit in seconds" on page 158 (TiLim, CPX_PARAM_TILIM)

The default value for `SolnPoolCapacity` is 2100000000, but it may be set to any nonnegative integer value. If set to zero, it will turn off all features related to the solution pool.

If you are looking for a parameter to control the number of solutions generated by the populate procedure, consider the parameter "maximum number of solutions generated for solution pool by populate" on page 117.

Values

Any nonnegative integer; 0 (zero) turns off all features of the solution pool; **default**: 2100000000.

relative gap for solution pool

Sets a relative tolerance on the objective value for the solutions in the solution pool.

Purpose

Relative gap for the solution pool

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_MIP_Pool_RelGap	CPX_PARAM_SOLNPOOLGAP
C++	IloCplex::Param::MIP::Pool::RelGap	SolnPoolGap (double)
Java	IloCplex.Param.MIP.Pool.RelGap	SolnPoolGap (double)
.NET	Cplex.Param.MIP.Pool.RelGap	SolnPoolGap (double)
OPL		solnpoolgap
Python	parameters.mip.pool.relgap	mip.pool.relgap
MATLAB	Cplex.Param.mip.pool.relgap	mip.pool.relgap
Interactive	mip pool relgap	mip pool relgap
Identifier	2105	2105

Description

Sets a relative tolerance on the objective value for the solutions in the solution pool. Solutions that are worse (either greater in the case of a minimization, or less in the case of a maximization) than the incumbent solution by this measure are not kept in the solution pool. For example, if you set this parameter to 0.01, then solutions worse than the incumbent by 1% or more will be discarded.

Values of the "absolute gap for solution pool" on page 146 (`SolnPoolAGap` or `CPX_PARAM_SOLNPOOLAGAP`) and the "relative gap for solution pool" on page 148 (`SolnPoolGap` or `CPX_PARAM_SOLNPOOLGAP`) may differ: For example, you may specify that solutions must be within 15 units by means of the solution pool absolute gap and within 1% of the incumbent by means of the solution pool relative gap. A solution is accepted in the pool only if it is valid for both the relative and the absolute gaps.

The solution pool relative gap parameter can also be used as a stopping criterion for the populate procedure: if populate cannot enumerate any more solutions that satisfy this objective quality, then it will stop. In the presence of both an absolute and a relative solution pool gap parameter, populate will stop when the smaller of the two is reached.

Values

Any nonnegative real number; **default**: 1.0e+75.

solution pool intensity

Controls the trade-off between the number of solutions generated for the solution pool and the amount of time or memory consumed.

Purpose

Solution pool intensity

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_MIP_Pool_Intensity	CPX_PARAM_SOLNPOOLINTENSITY
C++	IloCplex::Param::MIP::Pool::Intensity	SolnPoolIntensity (int)
Java	IloCplex.Param.MIP.Pool.Intensity	SolnPoolIntensity (int)
.NET	Cplex.Param.MIP.Pool.Intensity	SolnPoolIntensity (int)
OPL		solnpoolintensity
Python	parameters.mip.pool.intensity	mip.pool.intensity
MATLAB	Cplex.Param.mip.pool.intensity	mip.pool.intensity
Interactive	mip pool intensity	mip pool intensity
Identifier	2107	2107

Description

Controls the trade-off between the number of solutions generated for the solution pool and the amount of time or memory consumed. This parameter applies both to MIP optimization and to the populate procedure.

Values from 1 (one) to 4 invoke increasing effort to find larger numbers of solutions. Higher values are more expensive in terms of time and memory but are likely to yield more solutions.

Effect

For MIP optimization, increasing the value of the parameter corresponds to increasing the amount of effort spent setting up the branch and cut tree to prepare for a subsequent call to the populate procedure.

For populate, increasing the value of this parameter corresponds, in addition, to increasing the amount of effort spent exploring the tree to generate more solutions. If MIP optimization is called before populate, populate will reuse the information computed and stored during MIP optimization only if this parameter has not been increased between calls. Similarly, if populate is called several times successively, populate will re-use the information computed and stored during previous calls to populate only if the solution pool intensity has not increased between calls. Therefore, it is most efficient **not** to change the value of this parameter between calls to MIP optimization and populate, nor between successive calls of populate. Increase the value of this parameter only if too few solutions are generated.

Settings

Its default value, 0 (zero), lets CPLEX choose which intensity to apply. If MIP optimization is called first after the model is read, CPLEX sets the intensity to 1 (one) for this call to MIP optimization and to subsequent calls of populate. In contrast, if populate is called directly after the model is read, CPLEX sets the intensity to 2 for this call and subsequent calls of populate.

For value 1 (one), the performance of MIP optimization is not affected. There is no slowdown and no additional consumption of memory due to this setting. However, populate will quickly generate only a small number of solutions. Generating more than a few solutions with this setting will be slow. When you are looking for a larger number of solutions, use a higher value of this parameter.

For value 2, some information is stored in the branch and cut tree so that it is easier to generate a larger number of solutions. This storage has an impact on memory used but does not lead to a slowdown in the performance of MIP optimization. With this value, calling populate is likely to yield a number of solutions large enough for most purposes. This value is a good choice for most models.

For value 3, the algorithm is more aggressive in computing and storing information in order to generate a large number of solutions. Compared to values 1 (one) and 2, this value will generate a larger number of solutions, but it will slow MIP optimization and increase memory consumption. Use this value only if setting this parameter to 2 does not generate enough solutions.

For value 4, the algorithm generates all solutions to your model. Even for small models, the number of possible solutions is likely to be huge; thus enumerating all of them will take time and consume a large quantity of memory. In this case, remember to set the "maximum number of solutions generated for solution pool by populate" on page 117 (PopulateLim, CPX_PARAM_POPULATELIM) to a value appropriate for your model; otherwise, the populate procedure will stop prematurely because of this stopping criterion instead of enumerating all solutions. In addition, a few limitations apply to this exhaustive enumeration, as explained in Enumerating all solutions in the *CPLEX User's Manual*.

Table 76. Values

Value	Meaning
0	Automatic: let CPLEX choose; **default**
1	Mild: generate few solutions quickly
2	Moderate: generate a larger number of solutions
3	Aggressive: generate many solutions and expect performance penalty
4	Very aggressive: enumerate all practical solutions

solution pool replacement strategy

Designates the strategy for replacing a solution in the solution pool when the solution pool has reached its capacity.

Purpose

Solution pool replacement strategy

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_MIP_Pool_Replace	CPX_PARAM_SOLNPOOLREPLACE
C++	IloCplex::Param::MIP::Pool::Replace	SolnPoolReplace (int)
Java	IloCplex.Param.MIP.Pool.Replace	SolnPoolReplace (int)
.NET	Cplex.Param.MIP.Pool.Replace	SolnPoolReplace (int)
OPL		solnpoolreplace
Python	parameters.mip.pool.replace	mip.pool.replace
MATLAB	Cplex.Param.mip.pool.replace	mip.pool.replace
Interactive	mip pool replace	mip pool replace
Identifier	2104	2104

Description

Designates the strategy for replacing a solution in the solution pool when the solution pool has reached its capacity.

The value 0 (`CPX_SOLNPOOL_FIFO`) replaces solutions according to a first-in, first-out policy. The value 1 (`CPX_SOLNPOOL_OBJ`) keeps the solutions with the best objective values. The value 2 (`CPX_SOLNPOOL_DIV`) replaces solutions in order to build a set of diverse solutions. When the value is 2, CPLEX considers only variables of the type binary or integer (not continuous variables) to calculate diversity in the replacement strategy.

If the solutions you obtain are too similar to each other, try setting `SolnPoolReplace` to 2.

The replacement strategy applies only to the subset of solutions created in the current call of MIP optimization or populate. Solutions already in the pool are not affected by the replacement strategy. They will not be replaced, even if they satisfy the criterion of the replacement strategy.

Table 77. Values

Value	Symbol	Meaning
0	CPX_SOLNPOOL_FIFO	Replace the first solution (oldest) by the most recent solution; first in, first out; **default**
1	CPX_SOLNPOOL_OBJ	Replace the solution which has the worst objective
2	CPX_SOLNPOOL_DIV	Replace solutions in order to build a set of diverse solutions

solution type for LP and QP

Specifies type of solution (basic or non basic) that CPLEX produces for a linear program (LP) or quadratic program (QP).

Purpose

Solution type for LP or QP

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_SolutionType	CPX_PARAM_SOLUTIONTYPE
C++	IloCplex::Param::SolutionType	
Java	IloCplex.Param.SolutionType	
.NET	Cplex.Param.SolutionType	
OPL	solutiontype	
Python	parameters.solutiontype	
MATLAB	Cplex.Param.solutiontype	
Interactive	solutiontype	
Identifier	1147	

Description

Specifies the type of solution (basic or non basic) that CPLEX attempts to compute for a linear program (LP) or for a quadratic program (QP). In this context, *basic* means having to do with the basis. When CPLEX offers a *non basic* solution, the solution consists of a pair of primal-dual solution vectors.

By **default** (that is, when the value of this parameter is 0 (zero) automatic), CPLEX seeks a basic solution (that is, a solution with a basis) for all linear programs (LP) and for all quadratic programs (QP).

When the value of this parameter is 1 (one), CPLEX seeks a basic solution, that is, a solution that includes a basis with a basic status for variables and constraints. In other words, CPLEX behaves the same way for the values 0 (zero) and 1 (one) of this parameter.

When the value of this parameter is 2, CPLEX seeks a pair of primal-dual solution vectors. This setting does not prevent CPLEX from producing status information, but in seeking a pair of primal-dual solution vectors, CPLEX possibly may not produce *basic* status information; that is, it is possible that CPLEX does not produce status information about which variables and constraints participate in the *basis* at this setting.

Solution type interacts with symmetry detection in LP simplex

If your setting of this parameter requires CPLEX to find a **basis**, and your model is a linear program (LP) with a simplex algorithm, then CPLEX does not apply "symmetry breaking" on page 156.

If your setting of this parameter does **not** require CPLEX to find a **basis**, then CPLEX applies the **default** "symmetry breaking" on page 156 to a linear program (LP) using a simplex algorithm.

Solution type interacts with barrier crossover algorithm

Your setting of "barrier crossover algorithm" on page 24 interacts with your setting of this parameter. For example, the Callable Library (C API) routine CPXXhybbaropt and CPXhybbaropt produces a basic solution, solution vectors, or an error, depending on these settings.

Specifically, to avoid errors, set the parameter controlling the barrier crossover algorithm to a non default value **only** when the solution type parameter is at its default. Likewise, set the solution type parameter to a non default value **only** when the parameter controlling the barrier crossover algorithm is at its default setting.

Table 78. Solution type interacts with barrier crossover algorithm.

solution type	barrier crossover algorithm: no crossover	barrier crossover algorithm: auto	barrier crossover algorithm: primal	barrier crossover algorithm: dual
auto 0	solution vectors	basic solution	basic solution	basic solution
basic 1	error	basic solution	error	error
non basic 2	error	solution vectors	error	error

Tip: The value 2 of this solution type parameter tells CPLEX to seek a nonbasic solution. In the case of barrier optimization, this setting turns off crossover.

Tip: At the value 2 of this solution type parameter, it is possible under certain conditions that CPLEX can produce a basis anyway.

Table 79. Values

Value	Symbol	Meaning
0	`CPX_AUTO_SOLN`	Automatic: let CPLEX decide; **default**
1	`CPX_BASIC_SOLN`	CPLEX computes a basic solution.
2	`CPX_NONBASIC_SOLN`	CPLEX computes a primal-dual pair of solution-vectors

See also

"symmetry breaking" on page 156

MIP strong branching candidate list limit

Controls the length of the candidate list when CPLEX uses variable selection as the setting for strong branching.

Purpose

MIP strong branching candidate list limit

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_MIP_Limits_StrongCand	CPX_PARAM_STRONGCANDLIM
C++	IloCplex::Param::MIP::Limits::StrongCand	StrongCandLim (int)
Java	IloCplex.Param.MIP.Limits.StrongCand	StrongCandLim (int)
.NET	Cplex.Param.MIP.Limits.StrongCand	StrongCandLim (int)
OPL		strongcandlim
Python	parameters.mip.limits.strongcand	mip.limits.strongcand
MATLAB	Cplex.Param.mip.limits.strongcand	mip.limits.strongcand
Interactive	mip limits strongcand	mip limits strongcand
Identifier	2045	2045

Description

Controls the length of the candidate list when CPLEX uses strong branching as the way to select variables. For more detail about that parameter, see "MIP variable selection strategy" on page 166:

- `VarSel` in the C++, Java, or .NET API;
- `CPX_PARAM_VARSEL` in the C API;
- `set mip strategy variableselect 3` in the Interactive Optimizer.

Values

Any positive number; **default**: 10.

MIP strong branching iterations limit

Controls the number of simplex iterations performed on each variable in the candidate list when CPLEX uses variable selection as the setting for strong branching.

Purpose

MIP strong branching iterations limit

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_MIP_Limits_StrongIt	CPX_PARAM_STRONGITLIM
C++	IloCplex::Param::MIP::Limits::StrongIt	StrongItLim (long)
Java	IloCplex.Param.MIP.Limits.StrongIt	StrongItLim (long)
.NET	Cplex.Param.MIP.Limits.StrongIt	StrongItLim (long)
OPL		strongitlim
Python	parameters.mip.limits.strongit	mip.limits.strongit
MATLAB	Cplex.Param.mip.limits.strongit	mip.limits.strongit
Interactive	mip limits strongit	mip limits strongit
Identifier	2046	2046

Description

Controls the number of simplex iterations performed on each variable in the candidate list when CPLEX uses strong branching as the way to select variables. For more detail about that parameter, see "MIP variable selection strategy" on page 166:

- `VarSel` in the C++, Java, or .NET API;
- `CPX_PARAM_VARSEL` in the C API;
- `set mip strategy variableselect 3` in the Interactive Optimizer.

The default setting 0 (zero) chooses the iteration limit automatically.

Table 80. Values

Value	Meaning
0	Automatic: let CPLEX choose; **default**
Any positive integer	Limit of the simplex iterations performed on each candidate variable

limit on nodes explored when CPLEX solves a subMIP

Limits the number of nodes explored when CPLEX solves a subMIP.

Purpose

Limits the number of nodes explored when CPLEX solves a subMIP

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_MIP_SubMIP_NodeLimit	CPX_PARAM_SUBMIPNODELIMIT
C++	IloCplex::Param::MIP::SubMIP::NodeLimit	SubMIPNodeLim (long)
Java	IloCplex.Param.MIP.SubMIP.NodeLimit	SubMIPNodeLim (long)
.NET	Cplex.Param.MIP.SubMIP.NodeLimit	SubMIPNodeLim (long)
OPL		submipnodelim
Python	parameters.mip.submip.nodelimit	mip.limits.submipnodelim
MATLAB	Cplex.Param.mip.submip.nodelimit	mip.limits.submipnodelim
Interactive	mip submip nodelimit	mip limits submipnodelim
Identifier	2212	

Description

This parameter sets the number of nodes explored when CPLEX is solving a subMIP.

With this parameter, you can change the limit on the number of nodes explored when CPLEX solves a subMIP of a MIP. See the documentation of these other subMIP parameters for further instructions that you can impose when CPLEX solves a subMIP:

- scale parameter for subMIPs
- algorithm for initial MIP relaxation of a subMIP of a MIP
- algorithm for subproblems of a subMIP of a MIP

Tip: This rarely used parameter is helpful **only** in rare "corner" cases where there is clear evidence that the default limit is really inappropriate for this particular problem.

For a definition of **subMIP** and for a list of conditions in which CPLEX solves a subMIP, see the topic What is a subMIP? in the *CPLEX User's Manual*.

Values

Any positive integer; **default**: 500.

symmetry breaking

Decides whether symmetry breaking reductions will be automatically executed, during the preprocessing phase, in a MIP or LP model.

Purpose

Symmetry breaking

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_Preprocessing_Symmetry	CPX_PARAM_SYMMETRY
C++	IloCplex::Param::Preprocessing::Symmetry	Symmetry (int)
Java	IloCplex.Param.Preprocessing.Symmetry	Symmetry (int)
.NET	Cplex.Param.Preprocessing.Symmetry	Symmetry (int)
OPL		symmetry
Python	parameters.preprocessing.symmetry	preprocessing.symmetry
MATLAB	Cplex.Param.preprocessing.symmetry	preprocessing.symmetry
Interactive	preprocessing symmetry	preprocessing symmetry
Identifier	2059	2059

Description

Decides whether symmetry breaking reductions will be automatically executed, during the preprocessing phase, in either a MIP or LP model. The default level, -1, allows CPLEX to choose the degree of symmetry breaking to apply. The value 0 (zero) turns off symmetry breaking. Levels 1 through 5 apply increasingly aggressive symmetry breaking.

Symmetry detection interacts with solution type

At the **default** setting of this parameter, CPLEX considers the current value of “solution type for LP and QP” on page 152 as well.

When CPLEX solves a linear program (LP), and this parameter is at its **default** value, then whether CPLEX attempts to detect symmetry in the model depends on your choice of algorithm and on your choice of “solution type for LP and QP” on page 152.

- If your choice of algorithm is barrier optimization **without** crossover, then CPLEX applies symmetry detection according to the settings of this parameter.
- If your choice of algorithm is barrier optimization with crossover, then CPLEX applies symmetry detection during barrier optimization, according to the settings of this parameter.

- If your choice of algorithm is simplex optimization, then CPLEX considers your choice of "solution type for LP and QP" on page 152 as well.
  - If your choice of solution type requires a basis, then CPLEX does **not** apply symmetry detection.
  - If your choice of solution type does **not** require a basis, then CPLEX applies symmetry detection when this parameter is set at its **default**.

Table 81. Values

Value	Meaning
-1	Automatic: let CPLEX choose; **default**
0	Turn off symmetry breaking
1	Exert a moderate level of symmetry breaking
2	Exert an aggressive level of symmetry breaking
3	Exert a very aggressive level of symmetry breaking
4	Exert a highly aggressive level of symmetry breaking
5	Exert an extremely aggressive level of symmetry breaking

See also

"solution type for LP and QP" on page 152

global thread count

Sets the default number of parallel threads that will be invoked by any CPLEX parallel optimizer.

Purpose

Global default thread count

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_Threads	CPX_PARAM_THREADS
C++	IloCplex::Param::Threads	Threads (int)
Java	IloCplex.Param.Threads	Threads (int)
.NET	Cplex.Param.Threads	Threads (int)
OPL		threads
Python	parameters.threads	threads
MATLAB	Cplex.Param.threads	threads
Interactive	threads	threads
Identifier	1067	1067

Description

Sets the default maximal number of parallel threads that will be invoked by any CPLEX parallel optimizer.

For a single thread, the parallel algorithms behave deterministically, regardless of thread parameter settings; that is, the algorithm proceeds sequentially in a single thread.

In this context, **sequential** means that the algorithm proceeds step by step, consecutively, in a predictable and repeatable order within a single thread. **Deterministic** means that repeated solving of the same model with the same parameter settings on the same computing platform will follow exactly the same solution path, yielding the same level of performance and the same values in the solution. Sequential execution is deterministic. In multithreaded computing, a deterministic setting requires synchronization between threads. **Opportunistic** entails less synchronization between threads and thus may offer better performance at the sacrifice of repeatable, invariant solution paths and values in repeated runs on multiple threads or multiple processors.

When this parameter is at its default setting 0 (zero), and your application includes **no callbacks** or only an informational callback, CPLEX can use all available threads; that is, at most 32 threads or the number of cores of the machine, whichever is smaller. If your machine offers more than 32 threads, you can take advantage of them by increasing the value of this parameter.

When this parameter is at its default setting 0 (zero), and your application includes **callbacks** other than informational callbacks (that is, the application includes a query, diagnostic, or control callback), then CPLEX uses one thread. In other words, the presence of a callback turns off parallel processing when the value of this parameter is at its default.

In order to use **parallel optimization** in conjunction with **callbacks**, you need to set this parameter to a positive value. However, when you do so, you need to be aware of the fact that the callbacks may be invoked concurrently.

For a description of informational, query, diagnostic, and control callbacks, see the topic Using legacy optimization callbacks in the *CPLEX User's Manual*.

Tip: This parameter interacts with "CPU mask to bind threads to cores" on page 48. In particular, when this threads parameter is left at its default value 0 (that is, zero), then the value of "CPU mask to bind threads to cores" on page 48 determines how many threads are used.

Table 82. Values

Value	Meaning
0	Automatic: let CPLEX decide; **default**
1	Sequential; single threaded
N	Uses up to N threads; N is limited by available processors and Processor Value Units (PVU).

See also

"parallel mode switch" on page 108

optimizer time limit in seconds

Sets the maximum time, in seconds, for a call to an optimizer.

Purpose

Optimizer time limit in seconds

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_TimeLimit	CPX_PARAM_TILIM
C++	IloCplex::Param::TimeLimit	TiLim (double)
Java	IloCplex.Param.TimeLimit	TiLim (double)
.NET	Cplex.Param.TimeLimit	TiLim (double)
OPL		tilim
Python	parameters.timelimit	timelimit
MATLAB Cplex class API	Cplex.Param.timelimit	timelimit
MATLAB Optimization Toolbox compatible	MaxTime	MaxTime
Interactive	timelimit	timelimit
Identifier	1039	1039

Description

Sets the maximum time, in seconds, for a call to an optimizer. This time limit applies also to the conflict refiner.

The time is measured in terms of either CPU time or elapsed time, according to the setting of the "clock type for computation time" on page 43 parameter (`CPX_PARAM_CLOCKTYPE`, `ClockType`).

The time limit for an optimizer applies to the sum of all its steps, such as preprocessing, crossover, and internal calls to other optimizers.

In a sequence of calls to optimizers, the limit is not cumulative but applies to each call individually. For example, if you set a time limit of 10 seconds, and you call MIP optimization twice then there could be a total of (at most) 20 seconds of running time if each call consumes its maximum allotment.

See also

For a **deterministic** time limit on optimization, see "deterministic time limit" on page 56 (CPX_PARAM_DETTILIM, DetTiLim).

For an introduction to time stamps measured in seconds, see the topic Timing interface in the *CPLEX User's Manual*. For more detail about time stamps measured in seconds, see the reference manual of the API that you use.

- In the Callable Library (C API), see the documentation of CPXXgettime and CPXXgetcallbackinfo.
- In the C++ API, see the documentation of IloCplex::CallbackI::getStartTime and IloCplex::CallbackI::getEndTime.
- In the Java API, see the documentation of IloCplex.Callback.getStartTime and IloCplex.Callback.getEndTime.
- In the .NET API, see the documentation of Cplex.ICallback.GetStartTime and GetEndTime.

- In the Python API, see the documentation of Callback.get_start_time and Callback.get_end_time.
- In the MATLAB connector, see the documentation of cplex.Solution.time.

Values

Any nonnegative value in seconds; **default**: 1e+75.

See also

"clock type for computation time" on page 43

tree memory limit

Sets an absolute upper limit on the size (in megabytes, uncompressed) of the branch-and-cut tree.

Purpose

Tree memory limit

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_MIP_Limits_TreeMemory	CPX_PARAM_TRELIM
C++	IloCplex::Param::MIP::Limits::TreeMemory	TreLim (double)
Java	IloCplex.Param.MIP.Limits.TreeMemory	TreLim (double)
.NET	Cplex.Param.MIP.Limits.TreeMemory	TreLim (double)
OPL		trelim
Python	parameters.mip.limits.treememory	mip.limits.treememory
MATLAB	Cplex.Param.mip.limits.treememory	mip.limits.treememory
Interactive	mip limits treememory	mip limits treememory
Identifier	2027	2027

Description

Sets an absolute upper limit on the size (in megabytes, uncompressed) of the branch-and-cut tree. If this limit is exceeded, CPLEX terminates optimization.

Values

Any nonnegative number; **default**: 1e+75.

deterministic tuning time limit

Sets a time limit in deterministic ticks per model and per test set (that is, suite of models) applicable in tuning.

Purpose

Deterministic tuning time limit

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_Tune_DetTimeLimit	CPX_PARAM_TUNINGDETTILIM
C++	IloCplex::Param::Tune::DetTimeLimit	TuningDetTiLim (double)
Java	IloCplex.Param.Tune.DetTimeLimit	TuningDetTiLim (double)

API	Parameter Name	Name prior to V12.6.0
.NET	Cplex.Param.Tune.DetTimeLimit	TuningDetTiLim (double)
OPL		tuningdettilim
Python	parameters.tune.dettimelimit	tuning.dettimelimit
MATLAB	Cplex.Param.tune.dettimelimit	tune.timelimit
Interactive	tune dettimelimit	tune dettimelimit
Identifier	1139	1139

Description

Sets a deterministic time limit per model and per test set (that is, suite of models) applicable in tuning and measured in **ticks**.

When this deterministic tuning time limit is set to a finite value, then tuning finds appropriate settings of other CPLEX parameters to minimize the deterministic time of optimization. Furthermore, the tuning process itself is deterministic. In this context, "a finite value" means any value strictly less than 1e+75 (such as the finite value 1e+74).

Interaction with other parameters: nondeterministic tuning time limit

This deterministic time limit on tuning is **not** compatible with the wall-clock "tuning time limit in seconds" on page 164 (CPX_PARAM_TUNINGTILIM, TuningTiLim). Only one of these two parameters can be set to a finite value at a time. Any attempt to set either of these parameters to a finite value while the other is already set to a finite value results in the error CPXERR_PARAM_INCOMPATIBLE from the routine CPXsetdblparam or the method setDblParam (depending on the API you are using).

Finite values of tuning time limits

If this deterministic time limit on tuning is set to a finite value, then the tuning process itself is deterministic, and CPLEX recommends appropriate parameter settings to minimize the deterministic optimization time.

If the wall-clock "tuning time limit in seconds" on page 164 (CPX_PARAM_TUNINGTILIM, TuningTiLim) is set to a finite value, then the tuning process itself is **nondeterministic**, and it recommends appropriate parameter settings to minimize the wall-clock optimization time.

The default value of this parameter is 1e+75 (effectively, infinite).

Likewise, the default value of the wall-clock "tuning time limit in seconds" on page 164 (CPX_PARAM_TUNINGTILIM, TuningTiLim) is also 1e+75 (effectively, infinite).

If this parameter is set at its default value 1e+75, and if the "tuning time limit in seconds" on page 164 (CPX_PARAM_TUNINGTILIM, TuningTiLim) is also set at its default value 1e+75, then the combination is equivalent to setting the deterministic tuning time limit to 10 000 000 ticks. Consequently, these combined default settings make the tuning process deterministic, and CPLEX recommends settings to minimize the deterministic optimization time.

Unlimited time per model

If you want to run a tuning session with unlimited time per model, then set one of the tuning time limit parameters (either wall-clock "tuning time limit in seconds" on page 164 (CPX_PARAM_TUNINGTILIM, TuningTiLim) or "deterministic tuning time limit" on page 160 (CPX_PARAM_TUNINGDETTILIM, TuningDetTiLim) to a very large value that is strictly less than 1e+75 (for example, 1e+74). If you set CPX_PARAM_TUNINGDETTILIM, TuningDetTiLim to a finite value, then the tuning process will be deterministic. If you set CPX_PARAM_TUNINGTILIM, TuningTiLim to a finite value, then the tuning process will be nondeterministic.

Ticks

A tick is a unit to measure work done deterministically. The length of a deterministic tick may vary by platform. Nevertheless, ticks are normally consistent measures for a given platform (combination of hardware and software) carrying the same load. In other words, the correspondence of ticks to clock time depends on the hardware, software, and the current load of the machine. For the same platform and same load, the ratio of ticks per second stays roughly constant, independent of the model solved. However, for very short optimization runs, the variation of this ratio is typically high.

Values

Any nonnegative number; **default**: 1e+75 ticks.

See also

"optimizer time limit in seconds" on page 158

"deterministic time limit" on page 56

"tuning time limit in seconds" on page 164

tuning information display

Specifies the level of information reported by the tuning tool as it works.

Purpose

Tuning information display

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_Tune_Display	CPX_PARAM_TUNINGDISPLAY
C++	IloCplex::Param::Tune::Display	TuningDisplay (int)
Java	IloCplex.Param.Tune.Display	TuningDisplay (int)
.NET	Cplex.Param.Tune.Display	TuningDisplay (int)
OPL		tuningdisplay
Python	parameters.tune.display	tuning.display
MATLAB	Cplex.Param.tune.display	tune.display
Interactive	tune display	tune display
Identifier	1113	1113

Description

Specifies the level of information reported by the tuning tool as it works.

Use level 0 (zero) to turn off reporting from the tuning tool.

Use level 1 (one), the **default**, to display a minimal amount of information.

Use level 2 to display the minimal amount plus the parameter settings that the tuning tool is trying.

Use level 3 to display an exhaustive report of minimal information, plus settings that are being tried, plus logs.

Table 83. Values

Value	Meaning
0	Turn off display
1	Display standard, minimal reporting; **default**
2	Display standard report plus parameter settings being tried
3	Display exhaustive report and log

tuning measure

Controls the measure for evaluating progress when a suite of models is being tuned.

Purpose

Tuning measure

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_Tune_Measure	CPX_PARAM_TUNINGMEASURE
C++	IloCplex::Param::Tune::Measure	TuningMeasure
Java	IloCplex.Param.Tune.Measure	TuningMeasure
.NET	Cplex.Param.Tune.Measure	TuningMeasure
OPL		tuningmeasure
Python	parameters.tune.measure	tuning.measure
MATLAB	Cplex.Param.tune.measure	tuning.measure
Interactive	tune measure	tune measure
Identifier	1110	1110

Description

Controls the measure for evaluating progress when a suite of models is being tuned.

Possible values are:

- `CPX_TUNE_AVERAGE` uses the mean average of time to compare different parameter sets over a suite of models.
- `CPX_TUNE_MINMAX` uses a minmax approach to compare the time of different parameter sets over a suite of models.

Table 84. Values

Value	**Meaning**
`CPX_TUNE_AVERAGE`	mean time; **default**
`CPX_TUNE_MINMAX`	minmax time

tuning repeater

Specifies the number of times tuning is to be repeated on reordered versions of a given problem.

Purpose

Tuning repeater

API	**Parameter Name**	**Name prior to V12.6.0**
C	CPXPARAM_Tune_Repeat	CPX_PARAM_TUNINGREPEAT
C++	IloCplex::Param::Tune::Repeat	TuningRepeat (int)
Java	IloCplex.Param.Tune.Repeat	TuningRepeat (int)
.NET	Cplex.Param.Tune.Repeat	TuningRepeat (int)
OPL		tuningrepeat
Python	parameters.tune.repeat	tuning.repeat
MATLAB	Cplex.Param.tune.repeat	tuning.repeat
Interactive	tune repeat	tune repeat
Identifier	1111	1111

Description

Specifies the number of times tuning is to be repeated on reordered versions of a given problem. The problem is reordered automatically by CPLEX permuting its rows and columns. This repetition is helpful when only one problem is being tuned, as repeated reordering and re-tuning may lead to more robust tuning results.

This parameter applies to only one problem in a tuning session. That is, in the Interactive Optimizer, this parameter is effective only when you are tuning a single problem; in the Callable Library (C API), this parameter is effective only when you are tuning a single problem with the routine `CPXtuneparam` .

Values

Any nonnegative integer; **default**: 1 (one)

tuning time limit in seconds

Sets a nondeterministic time limit in seconds per model and per test set (that is, suite of models) applicable in tuning.

Purpose

Nondeterministic tuning time limit (wall-clock time)

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_Tune_TimeLimit	CPX_PARAM_TUNINGTILIM
C++	IloCplex::Param::Tune::TimeLimit	TuningTiLim (double)
Java	IloCplex.Param.Tune.TimeLimit	TuningTiLim (double)
.NET	Cplex.Param.Tune.TimeLimit	TuningTiLim (double)
OPL		tuningtilim
Python	parameters.tune.timelimit	tuning.timelimit
MATLAB	Cplex.Param.tune.timelimit	tuning.timelimit
Interactive	tune timelimit	tune timelimit
Identifier	1112	1112

Description

Sets a nondeterministic time limit in seconds per model and per test set (that is, suite of models) applicable in tuning. This parameter is also known as the wall-clock time limit on tuning.

Interaction with other parameters: deterministic tuning time limit

The "deterministic tuning time limit" on page 160 (CPX_PARAM_TUNINGDETTILIM, TuningDetTiLim) is **not** compatible with this wall-clock, nondeterministic tuning time limit (CPX_PARAM_TUNINGTILIM, TuningTiLim). Only one of these two parameters can be set to a finite value at a time. Any attempt to set either of these parameters to a finite value while the other is already set to a finite value results in the error CPXERR_PARAM_INCOMPATIBLE from the routine CPXsetdblparam or the method setDblParam (depending on your choice of API).

Finite values of tuning time limits

If this wall-clock, nondeterministic tuning time parameter (CPX_PARAM_TUNINGTILIM, TuningTiLim) is set to a finite value, then the tuning process itself is **nondeterministic**, and CPLEX recommends appropriate parameter settings to minimize the wall-clock optimization time.

If the "deterministic tuning time limit" on page 160 (CPX_PARAM_TUNINGDETTILIM, TuningDetTiLim) is set to a finite value, then the tuning process itself is deterministic, and CPLEX recommends appropriate parameter settings to minimize the deterministic optimization time.

The default value of this parameter is 1e+75 (effectively, infinite).

Likewise, the default value of the "deterministic tuning time limit" on page 160 (CPX_PARAM_TUNINGDETTILIM, TuningDetTiLim) is also 1e+75 (effectively, infinite).

If this parameter is set at its default value 1e+75, and if the "deterministic tuning time limit" on page 160 (CPX_PARAM_TUNINGDETTILIM, TuningDetTiLim) is also set at its default value 1e+75, then the combination is equivalent to setting the deterministic tuning time limit to 10 000 000 ticks. Consequently, these combined default settings make the tuning process deterministic, and CPLEX recommends settings to minimize the deterministic optimization time.

Unlimited time per model

If you want to run a tuning session with unlimited time per model, then set one of the tuning time limit parameters (either wall-clock "tuning time limit in seconds" on page 164 (CPX_PARAM_TUNINGTILIM, TuningTiLim) or "deterministic tuning time limit" on page 160 (CPX_PARAM_TUNINGDETTILIM, TuningDetTiLim) to a very large value that is strictly less than 1e+75 (for example, 1e+74). If you set CPX_PARAM_TUNINGDETTILIM, TuningDetTiLim to a finite value, then the tuning process will be deterministic. If you set CPX_PARAM_TUNINGTILIM, TuningTiLim to a finite value, then the tuning process will be nondeterministic.

Examples

For an example of how to use general and tuning-specific time limit parameters together, see Examples: time limits on tuning in the Interactive Optimizer in the *CPLEX User's Manual*.

Values

Any nonnegative number; **default**: 1e+75 seconds.

See also

"optimizer time limit in seconds" on page 158

"deterministic time limit" on page 56

"deterministic tuning time limit" on page 160

MIP variable selection strategy

Sets the rule for selecting the branching variable at the node which has been selected for branching.

Purpose

MIP variable selection strategy

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_MIP_Strategy_VariableSelect	CPX_PARAM_VARSEL
C++	IloCplex::Param::MIP::Strategy::VariableSelect	VarSel (int)
Java	IloCplex.Param.MIP.Strategy.VariableSelect	VarSel (int)
.NET	Cplex.Param.MIP.Strategy.VariableSelect	VarSel (int)
OPL		varsel
Python	parameters.mip.strategy.variableselect	mip.strategy.variableselect
MATLAB Cplex class API	Cplex.Param.mip.strategy.variableselect	mip.strategy.variableselect
MATLAB Optimization Toolbox compatible	BranchStrategy	BranchStrategy
Interactive	mip strategy variableselect	mip strategy variableselect
Identifier	2028	2028

Description

Sets the rule for selecting the branching variable at the node which has been selected for branching.

The minimum infeasibility rule chooses the variable with the value closest to an integer but still fractional. The minimum infeasibility rule (-1) may lead more quickly to a first integer feasible solution, but is usually slower overall to reach the optimal integer solution.

The maximum infeasibility rule chooses the variable with the value furtherest from an integer. The maximum infeasibility rule (1 one) forces larger changes earlier in the tree.

Pseudo cost (2) variable selection is derived from pseudo-shadow prices.

Strong branching (3) causes variable selection based on partially solving a number of subproblems with tentative branches to see which branch is the most promising. This strategy can be effective on large, difficult MIP problems.

Pseudo reduced costs (4) are a computationally less-intensive form of pseudo costs.

The default value (0 zero) allows CPLEX to select the best rule based on the problem and its progress.

Table 85. Values

Value	Symbol	Meaning
-1	CPX_VARSEL_MININFEAS	Branch on variable with minimum infeasibility
0	CPX_VARSEL_DEFAULT	Automatic: let CPLEX choose variable to branch on; **default**
1	CPX_VARSEL_MAXINFEAS	Branch on variable with maximum infeasibility
2	CPX_VARSEL_PSEUDO	Branch based on pseudo costs
3	CPX_VARSEL_STRONG	Strong branching
4	CPX_VARSEL_PSEUDOREDUCED	Branch based on pseudo reduced costs

directory for working files

Specifies the name of an existing directory into which CPLEX may store temporary working files.

Purpose

Directory for working files

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_WorkDir	CPX_PARAM_WORKDIR
C++	IloCplex::Param::WorkDir	WorkDir (string)
Java	IloCplex.Param.WorkDir	WorkDir (string)
.NET	Cplex.Param.WorkDir	WorkDir (string)

API	Parameter Name	Name prior to V12.6.0
OPL		workdir
Python	parameters.workdir	workdir
MATLAB	Cplex.Param.workdir	workdir
Interactive	workdir	workdir
Identifier	1064	1064

Description

Specifies the name of an existing directory into which CPLEX may store temporary working files, such as for MIP node files or for out-of-core barrier files. The default is the current working directory.

This parameter accepts a `string` as its value. If you change either the "API string encoding switch" on page 20 or the "file encoding switch" on page 69 from their default value to a multi-byte encoding where a `NULL` byte can occur within the encoding of a character, you must take into account the issues documented in the topic Selecting an encoding in the *CPLEX User's Manual*. Especially consider the possibility that a `NULL` byte occurring in the encoding of a character can inadvertently signal the termination of a string, such as a filename or directory path, and thus provoke surprising or incorrect results.

Tip: If the string designating the path to the target directory includes one or more spaces, be sure to include the entire string in double quotation marks.

Values

Any existing directory; **default**: '.'

memory available for working storage

Specifies an upper limit on the amount of central memory, in megabytes, that CPLEX is permitted to use for working memory.

Purpose

Memory available for working storage

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_WorkMem	CPX_PARAM_WORKMEM
C++	IloCplex::Param::WorkMem	WorkMem (double)
Java	IloCplex.Param.WorkMem	WorkMem (double)
.NET	Cplex.Param.WorkMem	WorkMem (double)
OPL		workmem
Python	parameters.workmem	workmem
MATLAB	Cplex.Param.workmem	workmem
Interactive	workmem	workmem
Identifier	1065	1065

Description

Specifies an upper limit on the amount of central memory, in megabytes, that CPLEX is permitted to use for working memory before swapping to disk files, compressing memory, or taking other actions.

Values

Any nonnegative number, in megabytes; **default**: 2048

See also

"directory for working files" on page 167

write level for MST, SOL files

Sets a level of detail for CPLEX to write a file in MST or SOL format.

Purpose

Write level for MST, SOL files

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_Output_WriteLevel	CPX_PARAM_WRITELEVEL
C++	IloCplex::Param::Output::WriteLevel	WriteLevel (int)
Java	IloCplex.Param.Output.WriteLevel	WriteLevel (int)
.NET	Cplex.Param.Output.WriteLevel	WriteLevel (int)
OPL		not available
Python	parameters.output.writelevel	output.writelevel
MATLAB	Cplex.Param.output.writelevel	output.writelevel
Interactive	output writelevel	output writelevel
Identifier	1114	1114

Description

Sets the level of detail for CPLEX to write a solution to a file in SOL format or a MIP start to a file in MST format. CPLEX writes information about a MIP start to a formatted file of type MST with the file extension `.mst`. CPLEX writes information about a solution to a formatted file of type SOL with the file extension `.sol`. CPLEX records the write level at which it created a file in that file, so that the file can be read back accurately later.

The default setting of this parameter is 0 (zero) AUTO; that is, let CPLEX decide the level of detail. CPLEX behaves differently, depending on whether the format is SOL or MST and on whether it is writing a solution or MIP start. For SOL files, AUTO resembles level 1 (one): CPLEX writes all variables and their respective values to the file. For MST files, AUTO resembles level 2: CPLEX writes discrete variables and their respective values to the file.

When the value of this parameter is 1 (one), CPLEX writes **all** variables, both discrete and continuous, with their values.

When the value of this parameter is 2, CPLEX writes values for **discrete** variables only.

When the value of this parameter is 3, CPLEX writes values of **nonzero** variables only.

When the value of this parameter is 4, CPLEX writes values of **nonzero discrete** variables only.

Treatment of nonzeros

With respect to levels 3 and 4, where **nonzero** values are significant, CPLEX considers a value nonzero if the absolute value is strictly less than 1e-16. In the case of **SOL** files, CPLEX applies this test to **primal** and **dual variable** values, that is, both x and pi variable values. In the case of **MST** files, CPLEX applies this test only to x values.

Restrictions due to reduced file size

Levels 3 and 4 reduce the size of files, of course. However, this reduced file entails restrictions and may create surprising results when the file is re-used. Levels 3 and 4 are not equivalent to levels 1 and 2. Indeed, if a MIP start does not contain a value for a variable expected at level 3 or 4, then this variable will be fixed to 0 (zero) when that MIP start file is processed. Specifically, at level 3, if the MIP start does not specify a value for a variable of any type, or at level 4, if the MIP start does not specify a value for a discrete variable, such a variable will be fixed to 0 (zero). Consequently, the same MIP start written at level 1 or 2 may produce satisfactory solutions, but the reduced MIP start file, written at level 3 or 4, perhaps does not lead to solutions. This surprising situation arises typically in the case of model changes with the addition of new variables.

Table 86. Values

Value	Symbol	Meaning
0	`AUTO`	Automatic: let CPLEX decide
1	`CPX_WRITELEVEL_ALLVARS`	CPLEX writes all variables and their values
2	`CPX_WRITELEVEL_DISCRETEVARS`	CPLEX writes only discrete variables and their values
3	`CPX_WRITELEVEL_NONZEROVARS`	CPLEX writes only nonzero variables and their values
4	`CPX_WRITELEVEL_NONZERODISCRETEVARS`	CPLEX writes only nonzero discrete variables and their values

MIP zero-half cuts switch

Decides whether or not to generate zero-half cuts for the problem.

Purpose

MIP zero-half cuts switch

API	Parameter Name	Name prior to V12.6.0
C	CPXPARAM_MIP_Cuts_ZeroHalfCut	CPX_PARAM_ZEROHALFCUTS
C++	IloCplex::Param::MIP::Cuts::ZeroHalfCut	ZeroHalfCuts (int)
Java	IloCplex.Param.MIP.Cuts.ZeroHalfCut	ZeroHalfCuts (int)
.NET	Cplex.Param.MIP.Cuts.ZeroHalfCut	ZeroHalfCuts (int)
OPL		zerohalfcuts
Python	parameters.mip.cuts.zerohalfcut	mip.cuts.zerohalfcut

API	Parameter Name	Name prior to V12.6.0
MATLAB	Cplex.Param.mip.cuts.zerohalfcut	mip.cuts.zerohalfcut
Interactive	mip cuts zerohalfcut	mip cuts zerohalfcut
Identifier	2111	2111

Description

Decides whether or not to generate zero-half cuts for the problem. The value 0 (zero), the default, specifies that the attempt to generate zero-half cuts should continue only if it seems to be helping.

If you find that too much time is spent generating zero-half cuts for your model, consider setting this parameter to -1 (minus one) to turn off zero-half cuts.

If the dual bound of your model does not make sufficient progress, consider setting this parameter to 2 to generate zero-half cuts more aggressively.

For a definition of a zero-half cut, see the topic Zero-half cuts in the general topic Cuts in the *CPLEX User's Manual*. The table Parameters for controlling cuts, also in the user's manual, includes links to the documentation of other parameters affecting other types of cuts.

Table 87. Values

Value	Meaning
-1	Do not generate zero-half cuts
0	Automatic: let CPLEX choose; **default**
1	Generate zero-half cuts moderately
2	Generate zero-half cuts aggressively

Acknowledgment of use: dtoa routine of the gdtoa package

IBM ILOG CPLEX acknowledges use of the `dtoa` routine of the `gdtoa` package.

IBM ILOG CPLEX acknowledges use of the `dtoa` routine of the `gdtoa` package, available at

`http://www.netlib.org/fp/`.

The author of this software is David M. Gay.

All Rights Reserved.

Copyright (C) 1998, 1999 by Lucent Technologies

Permission to use, copy, modify, and distribute this software and its documentation for any purpose and without fee is hereby granted, provided that the above copyright notice appears in all copies and that both that the copyright notice and this permission notice and warranty disclaimer appear in supporting documentation, and that the name of Lucent or any of its entities not be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission.

LUCENT DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL LUCENT OR ANY OF ITS ENTITIES BE LIABLE FOR ANY SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

(end of acknowledgment of use of `dtoa` routine of the `gdtoa` package)

Further acknowledgments: AMPL

IBM ILOG CPLEX acknowledges use of AMPL Drivers and Solver Library.

Acknowledgement: AMPL Driver for CPLEX

CPLEX acknowledges copyright, authorship, and the disclaimer of warranty for the AMPL Driver for CPLEX.

Copyright (C) 1997-2001 by Lucent Technologies

The author of this software is David M. Gay.

All Rights Reserved

Permission to use, copy, modify, and distribute this software and its documentation for any purpose and without fee is hereby granted, provided that the above copyright notice appear in all copies and that both that the copyright notice and this permission notice and warranty disclaimer appear in supporting documentation, and that the name of Lucent or any of its entities not be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission.

LUCENT DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL LUCENT OR ANY OF ITS ENTITIES BE LIABLE FOR ANY SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

(end of acknowledgment for AMPL Driver for CPLEX)

Acknowledgement: AMPL Solver Library

CPLEX acknowledges copyright, authorship, and the disclaimer of warranty for the AMPL Solver Library.

Copyright (C) 1990, 2001 Lucent Technologies

All Rights Reserved

Permission to use, copy, modify, and distribute this software and its documentation for any purpose and without fee is hereby granted, provided that the above copyright notice appear in all copies and that both that the copyright notice and this permission notice and warranty disclaimer appear in supporting documentation, and that the name of Lucent or any of its entities not be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission.

LUCENT DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND

FITNESS. IN NO EVENT SHALL LUCENT OR ANY OF ITS ENTITIES BE LIABLE FOR ANY SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

Copyright (C) 2007 David M. Gay

Permission to use, copy, modify, and distribute this software and its documentation for any purpose and without fee is hereby granted, provided that the above copyright notice appear in all copies and that both that the copyright notice and this permission notice and warranty disclaimer appear in supporting documentation.

The author disclaims all warranties with regard to this software, including all implied warranties of merchantability and fitness. In no event shall the author be liable for any special, indirect or consequential damages or any damages whatsoever resulting from loss of use, data or profits, whether in an action of contract, negligence or other tortious action, arising out of or in connection with the use or performance of this software.

Copyright (C) 2011 AMPL Optimization LLC, written by David M. Gay.

Permission to use, copy, modify, and distribute this software and its documentation for any purpose and without fee is hereby granted, provided that the above copyright notice appear in all copies and that both that the copyright notice and this permission notice and warranty disclaimer appear in supporting documentation.

The author and AMPL Optimization LLC disclaim all warranties with regard to this software, including all implied warranties of merchantability and fitness. In no event shall the author be liable for any special, indirect or consequential damages or any damages whatsoever resulting from loss of use, data or profits, whether in an action of contract, negligence or other tortious action, arising out of or in connection with the use or performance of this software.

Copyright (C) 2002 AMPL Optimization LLC

All Rights Reserved

Based largely on suf_sos.c, which bears the following Copyright notice and disclaimer. AMPL Optimization LLC similarly disclaims all warranties with regard to this software and grants permission to use, copy, modify, and distribute it and its documentation.

(end of acknowledgment for AMPL Solver Library)

Index

D

E

F

G

H

I

K

L

M

N

T

U

V

W

Z

Printed in USA