



Changing the rules of business™

ILOG CPLEX 10.1 Parameters

July 2006

COPYRIGHT NOTICE

ILOG CPLEX 10.1 Copyright © 1997-2006, by ILOG S.A., 9 Rue de Verdun, 94253 Gentilly Cedex, France, and ILOG, Inc., 1080 Linda Vista Ave., Mountain View, California 94043, USA. All rights reserved.

General Use Restrictions

This document and the software described in this document are the property of ILOG and are protected as ILOG trade secrets. They are furnished under a license or nondisclosure agreement, and may be used or copied only within the terms of such license or nondisclosure agreement.

No part of this work may be reproduced or disseminated in any form or by any means, without the prior written permission of ILOG S.A, or ILOG, Inc.

Trademarks

ILOG, the ILOG design, CPLEX, and all other logos and product and service names of ILOG are registered trademarks or trademarks of ILOG in France, the U.S. and/or other countries.

All other company and product names are trademarks or registered trademarks of their respective holders.

Java and all Java-based marks are either trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries.

Microsoft and Windows are either trademarks or registered trademarks of Microsoft Corporation in the United States and other countries.

document version 10.1

Table of Contents

Parameters of ILOG CPLEX	9
Accessing Parameters.....	9
Parameter Names.....	10
Correspondence of Parameters	11
Saving Parameter Settings to a File.....	12
Parameter Table	12
CPX_PARAM_ADVIND	13
CPX_PARAM_AGGCUTLIM	13
CPX_PARAM_AGGFILL	13
CPX_PARAM_AGGIND.....	14
CPX_PARAM_BARALG	14
CPX_PARAM_BARCOLNZ	14
CPX_PARAM_BARCROSSALG	14
CPX_PARAM_BARDISPLAY	15
CPX_PARAM_BAREPCOMP	15
CPX_PARAM_BARGROWTH.....	15
CPX_PARAM_BARITLIM	15
CPX_PARAM_BARMAXCOR	16
CPX_PARAM_BAROBRNG.....	16

CPX_PARAM_BARORDER.....	16
CPX_PARAM_BARQCPEPCOMP	16
CPX_PARAM_BARSTARTALG.....	17
CPX_PARAM_BARTHREADS	17
CPX_PARAM_BBINTERVAL.....	17
CPX_PARAM_BNDSTRENIND.....	17
CPX_PARAM_BRDIR	18
CPX_PARAM_BTTOL.....	18
CPX_PARAM_CLIQUES	18
CPX_PARAM_CLOCKTYPE.....	19
CPX_PARAM_COEREDIND.....	19
CPX_PARAM_COLREADLIM.....	19
CPX_PARAM_CONFLICTDISPLAY	19
CPX_PARAM_COVERS	20
CPX_PARAM_CRAIND.....	20
CPX_PARAM_CUTLO.....	20
CPX_PARAM_CUTPASS	21
CPX_PARAM_CUTSFACOR	21
CPX_PARAM_CUTUP.....	21
CPX_PARAM_DATACHECK.....	21
CPX_PARAM_DEPIND.....	22
CPX_PARAM_DISJCUTS	22
CPX_PARAM_DIVETYPE.....	22
CPX_PARAM_DPRIIND	23
CPX_PARAM_EPAGAP	23
CPX_PARAM_EPGAP.....	23
CPX_PARAM_EPINT	23
CPX_PARAM_EPLIN (n/a)	24
CPX_PARAM_EPMRK	24
CPX_PARAM_EPOPT.....	24
CPX_PARAM_EPPER.....	25

CPX_PARAM_EPRELAX	25
CPX_PARAM_EPRHS.....	25
CPX_PARAM_FEASOPTMODE	26
CPX_PARAM_FLOWCOVERS	26
CPX_PARAM_FLOWPATHS.....	27
CPX_PARAM_FRACCAND.....	27
CPX_PARAM_FRACCUTS	27
CPX_PARAM_FRACPASS	27
CPX_PARAM_GUBCOVERS	28
CPX_PARAM_HEURFREQ.....	28
CPX_PARAM_IMPLBD	28
CPX_PARAM_INTSOLLIM	28
CPX_PARAM_ITLIM	29
CPX_PARAM_LBHEUR	29
CPX_PARAM_LPMETHOD.....	29
CPX_PARAM_MEMORYEMPHASIS	30
CPX_PARAM_MIPCBREDLP	30
CPX_PARAM_MIPDISPLAY	30
CPX_PARAM_MIPEMPHASIS	31
CPX_PARAM_MIPINTERVAL.....	31
CPX_PARAM_MIPORDIND	31
CPX_PARAM_MIPORDTYPE.....	32
CPX_PARAM_MIPTHREADS.....	32
CPX_PARAM_MIRCUTS.....	32
CPX_PARAM_MPSLONGNUM.....	32
CPX_PARAM_NETDISPLAY.....	33
CPX_PARAM_NETEPOPT	33
CPX_PARAM_NETEPRHS	33
CPX_PARAM_NETFIND	33
CPX_PARAM_NETITLIM.....	33
CPX_PARAM_NETPPRIIND.....	34

CPX_PARAM_NODEFILEIND.....	34
CPX_PARAM_NODELIM.....	34
CPX_PARAM_NODESEL.....	35
CPX_PARAM_NUMERICAL EMPHASIS.....	35
CPX_PARAM_NZREADLIM.....	35
CPX_PARAM_OBJDIF.....	35
CPX_PARAM_OBJLLIM.....	36
CPX_PARAM_OBJULIM.....	36
CPX_PARAM_PERIND.....	36
CPX_PARAM_PERLIM.....	36
CPX_PARAM_POLISHTIME.....	36
CPX_PARAM_PPRIIND.....	37
CPX_PARAM_PREDUAL.....	37
CPX_PARAM_PREIND.....	37
CPX_PARAM_PRELINEAR.....	37
CPX_PARAM_PREPASS.....	38
CPX_PARAM_PRESLVND.....	38
CPX_PARAM_PRICELIM.....	38
CPX_PARAM_PROBE.....	38
CPX_PARAM_PROBETIME.....	39
CPX_PARAM_QPMAKEPSDIND.....	39
CPX_PARAM_QPMETHOD.....	39
CPX_PARAM_QPNZREADLIM.....	39
CPX_PARAM_REDUCE.....	40
CPX_PARAM_REINV.....	40
CPX_PARAM_RELAXPREIND.....	40
CPX_PARAM_RELOBJDIF.....	40
CPX_PARAM_REPAIRTRIES.....	41
CPX_PARAM_REPEATPRESOLVE.....	41
CPX_PARAM_RINSHEUR.....	41
CPX_PARAM_ROWREADLIM.....	41

CPX_PARAM_SCAIND	42
CPX_PARAM_SCRIND	42
CPX_PARAM_SIFTALG	42
CPX_PARAM_SIFTDISPLAY	42
CPX_PARAM_SIFTITLIM	43
CPX_PARAM_SIMDISPLAY	43
CPX_PARAM_SINGLIM	43
CPX_PARAM_STARTALG	44
CPX_PARAM_STRONGCANDLIM	44
CPX_PARAM_STRONGITLIM	44
CPX_PARAM_STRONGTHREADLIM	45
CPX_PARAM_SUBALG	45
CPX_PARAM_SUBMIPNODELIM	45
CPX_PARAM_SYMMETRY	46
CPX_PARAM_THREADS	46
CPX_PARAM_TILIM	46
CPX_PARAM_TRELIM	46
CPX_PARAM_VARSEL	47
CPX_PARAM_WORKDIR	47
CPX_PARAM_WORKMEM	47
Index	49

Parameters of ILOG CPLEX

The behavior of ILOG CPLEX is controlled by a variety of parameters that are each accessible and settable by the user. This manual lists these parameters and explains their settings in the ILOG CPLEX Component Libraries and the Interactive Optimizer. It also explains how to read and write parameter settings of the Callable Library to a file.

- ◆ *Accessing Parameters* on page 9
- ◆ *Parameter Names* on page 10
- ◆ *Correspondence of Parameters* on page 11
- ◆ *Saving Parameter Settings to a File* on page 12
- ◆ *Parameter Table* on page 12

Accessing Parameters

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

```
setParam  
getParam  
getMin  
getMax  
getDefault
```

`setDefault`s

The names of the corresponding accessors in the class `Cplex` in .NET 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`.

Callable Library programs (C and other languages) access and set parameters with the following routines:

<code>CPXgetdblparam</code>	Accesses a parameter of type double
<code>CPXsetdblparam</code>	Changes a parameter of type double
<code>CPXinfodblparam</code>	Gets the default value and range of a parameter of type double
<code>CPXgetintparam</code>	Accesses a parameter of type integer
<code>CPXsetintparam</code>	Changes a parameter of type integer
<code>CPXinfointparam</code>	Gets the default value and range of a parameter of type integer
<code>CPXgetstrparam</code>	Accesses a parameter of type string
<code>CPXsetstrparam</code>	Changes a parameter of type string
<code>CPXinfostrparam</code>	Gets the default value of a parameter of type string
<code>CPXsetdefaults</code>	Resets all parameters to their standard default values
<code>CPXgetparamname</code>	Accesses the name of a parameter
<code>CPXgetparamnum</code>	Access the identifying number assigned to a parameter
<code>CPXgetchgparams</code>	Accesses all parameters not currently at their default value

Parameter Names

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

- ◆ For the Callable Library 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 using Concert Technology, 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 using Concert Technology, the parameters are defined as final static objects in nested classes called `IloCplex.BooleanParam`, `IloCplex.IntParam`, `IloCplex.DoubleParam`, and `IloCplex.StringParam` for Boolean, integer, 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 using Concert Technology, the parameters follow the usual conventions for capitalizing attributes and defining scope within a namespace.

An integer that serves as a reference number for each parameter is shown in the table. That integer reference number corresponds to the value that each symbolic constant represents, as found in the `cplex.h` header file, but it is strongly recommended that the symbolic constants be used instead of their integer equivalents whenever possible, for the sake of portability to future versions of ILOG CPLEX.

Correspondence of Parameters

Some parameters available for the Callable Library are not supported as parameters for Concert Technology or have a slightly different name there. In particular:

- ◆ Logging output is controlled by a parameter in the Callable Library (`CPX_PARAM_SCRIND`), but when using Concert Technology, 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)`.
- ◆ The parameter `IloCplex::RootAlg` in Concert Technology corresponds to these parameters in the Callable Library:
 - `CPX_PARAM_STARTALG`
 - `CPX_PARAM_LPMETHOD`
 - `CPX_PARAM_QPMETHOD`
- ◆ The parameter `IloCplex::NodeAlg` in Concert Technology corresponds to the parameter `CPX_PARAM_SUBALG` in the Callable Library.

Saving Parameter Settings to a File

It is possible to read and write a file of parameter settings with the Callable Library. The file extension is `.prm`. The Callable Library routine `CPXreadcopyparam` reads parameter values from a file with the `.prm` extension. The routine `CPXwriteparam` writes a file of the current nondefault 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
```

ILOG CPLEX reads the entire file before changing any of the parameter settings. After successfully reading a parameter file, the Callable Library 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 write a parameter file from the Callable Library, only the non-default values are written to the file. String values may be double-quoted or not, but are always written with double quotation marks.

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

The Callable Library 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 ILOG CPLEX parameter file:

```
CPLEX Parameter File Version 10.0
CPX_PARAM_EPPER          3.450000000000000e-06
CPX_PARAM_OBJULIM       1.23456789012345e+05
CPX_PARAM_PERIND        1
CPX_PARAM_SCRIND        1
CPX_PARAM_WORKDIR       "tmp"
```

Parameter Table

The CPLEX parameters and their types, options, and default values are listed in the following table. The Callable Library name for each parameter is listed first, followed by the Concert Technology name, followed by the name in the Interactive Optimizer. Some CPLEX parameters are not used in the Concert Technology Library, and in those cases, no Concert Technology Library name appears.

Parameter Name	Code	Type	Value [Symbolic Constants]
CPX_PARAM_ADVIND AdvInd advance	1001	int	0 Do not use advanced start information 1 CPLEX will use an advanced basis supplied by the user 2 CPLEX will crush an advanced basis or starting vector supplied by the user Default: 1
<p>Description: Advanced start indicator. If set to 1 or 2, this parameter indicates that CPLEX should use advanced starting information when optimization is initiated.</p> <p>For MIP models, settings 1 and 2 are currently identical. Both will cause CPLEX to continue with a partially explored MIP tree if one is available. If tree exploration has not yet begun, settings 1 or 2 indicate that CPLEX should use a loaded MIP start, if available.</p> <p>For continuous models solved with simplex, setting 1 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.</p> <p>Setting 2 can be particularly useful for solving fixed MIP models, where a start vector but no corresponding basis is available.</p> <p>For continuous models solved with the barrier algorithm, settings 1 or 2 will continue optimization from the last available barrier iterate.</p>			
CPX_PARAM_AGGCUTLIM AggCutLim mip limits aggforcut	2054	int	Any nonnegative integer Default: 3
<p>Description: Constraint aggregation limit for cut generation. Limits the number of constraints that can be aggregated for generating flow cover and mixed integer rounding cuts.</p>			
CPX_PARAM_AGGFILL AggFill preprocessing fill	1002	int	Any nonnegative integer Default: 10
<p>Description: Preprocessing aggregator fill. Limits variable substitutions by the aggregator. If the net result of a single substitution is more nonzeros than this value, the substitution is not made.</p>			

Parameter Name	Code	Type	Value [Symbolic Constants]
CPX_PARAM_AGGIND AggInd preprocessing aggregator	1003	int	-1 Automatic (1 for LP, infinite for MIP) 0 Do not use any aggregator Any positive integer Default: -1
<p>Description: 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. If set to a positive value, the aggregator is applied the specified number of times or until no more reductions are possible.</p>			
CPX_PARAM_BARALG BarAlg barrier algorithm	3007	int	0 Default setting 1 Infeasibility-estimate start 2 Infeasibility-constant start 3 Standard barrier Default: 0
<p>Description: 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. 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.</p>			
CPX_PARAM_BARCOLNZ BarColNz barrier colnonzeros	3009	int	0 Dynamically calculated or, any positive integer Default: 0
<p>Description: Barrier column nonzeros. 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.</p>			
CPX_PARAM_BARCROSSALG BarCrossAlg barrier crossover	3018	int	-1 No crossover 0 Automatic: let CPLEX choose 1 Primal crossover 2 Dual crossover Default: 0
<p>Description: Barrier crossover algorithm. Determines which, if any, crossover is performed at the end of a barrier optimization. This parameter also applies when CPLEX uses the barrier algorithm 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.</p>			

Parameter Name	Code	Type	Value [Symbolic Constants]
CPX_PARAM_BARDISPLAY BarDisplay barrier display	3010	int	0 No progress information 1 Normal setup and iteration information 2 Diagnostic information Default: 1
Description: Barrier display information. Determines the level of barrier progress information to be displayed.			
CPX_PARAM_BAREPCOMP BarEpComp barrier convergetol	3002	double	Any positive number $\geq 1e^{-12}$ Default: $1e^{-8}$
Description: Convergence tolerance for LP and QP problems. For problems with quadratic constraints (QCP), see CPX_PARAM_BARQCPEPCOMP. 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 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.			
CPX_PARAM_BARGROWTH BarGrowth barrier limits growth	3003	double	1.0 or greater Default: $1e^{12}$
Description: Barrier growth limit. 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.			
CPX_PARAM_BARITLIM BarItLim barrier limits iterations	3012	int	0 No Barrier iterations or, any positive integer Default: 2 100 000 000
Description: Barrier iteration limit. Sets the number of barrier iterations before termination. When set to 0, no barrier iterations occur, but problem “setup” occurs and information about the setup is displayed (such as Cholesky factor statistics).			

Parameter Name	Code	Type	Value [Symbolic Constants]
CPX_PARAM_BARMAXCOR BarMaxCor barrier limits corrections	3013	int	-1 Automatic: let CPLEX choose 0 None or, any positive integer Default: -1
<p>Description: Barrier maximum correction limit. Sets the maximum number of centering corrections done on each iteration. An explicit value greater than 0 may improve the numerical performance of the algorithm at the expense of computation time.</p>			
CPX_PARAM_BAROBJRNG BarObjRng barrier limits objrange	3004	double	Any nonnegative number Default: $1e^{20}$
<p>Description: Barrier objective range. Sets the maximum absolute value of the objective function. The barrier algorithm looks at this limit to detect unbounded problems.</p>			
CPX_PARAM_BARORDER BarOrder barrier ordering	3014	int	0 Automatic: let CPLEX choose 1 Approximate minimum degree (AMD) 2 Approximate minimum fill (AMF) 3 Nested dissection (ND) Default: 0
<p>Description: 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.</p>			
CPX_PARAM_BARQCPEPCOMP BarQCPEpComp set bar qcpconvergetol	3020	double	Any positive number $\geq 1e^{-12}$ Default: $1e^{-7}$
<p>Description: Convergence tolerance for quadratically constrained problems (QCP). For LPs and for QPs (that is, when all the constraints are linear) see CPX_PARAM_BAREPCOMP. 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 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.</p>			

Parameter Name	Code	Type	Value [Symbolic Constants]
CPX_PARAM_BARSTARTALG BarStartAlg barrier startalg	3017	int	1 Dual is 0 2 Estimate dual 3 Average of primal estimate, dual 0 4 Average of primal estimate, estimate dual Default: 1
Description: Barrier starting point algorithm. Sets the algorithm to be used to compute the initial starting point for the barrier optimizer.			
CPX_PARAM_BARTHREADS BarThreads barrier limits threads	3016	int	0 Determined by global thread default >0 upper limit on threads for Parallel Barrier Default 0
Description: Barrier thread limit. Determines the maximum number of parallel processes (threads) that will be invoked by the parallel barrier optimizer. The default value of 0 means that the limit will be determined by the value of CPX_PARAM_THREADS, the global thread limit parameter. A positive value will override the value found in CPX_PARAM_THREADS.			
CPX_PARAM_BBINTERVAL BBInterval mip strategy bbinterval	2039	int	0 Best estimate node always selected or, any positive integer Default: 7
Description: MIP strategy best bound interval. When you set this parameter to best estimate node selection, the bbinterval is the interval at which the best bound node, instead of the best estimate node, is selected from the tree. A bbinterval of 0 means to never select the best bound node. A bbinterval of 1 means always to select the best bound node, and is thus equivalent to nodeselect 1. Higher values of bbinterval mean that the best bound node will be selected less frequently; experience has shown it to be beneficial to occasionally select the best bound node, and therefore the default bbinterval is 7.			
CPX_PARAM_BNDSTRENIND BndStrenInd preprocessing boundstrength	2029	int	-1 Automatic: let CPLEX choose 0 Do not apply bound strengthening 1 Apply bound strengthening Default: -1
Description: Bound strengthening indicator. Used when solving mixed integer programs. Bound strengthening tightens the bounds on variables, perhaps to the point where the variable can be fixed and thus removed from consideration during branch & cut.			

Parameter Name	Code	Type	Value [Symbolic Constants]
CPX_PARAM_BRDIR BrDir mip strategy branch	2001	int	-1 [CPX_BRDIR_DOWN] Down branch selected first 0 [CPX_BRDIR_AUTO] Automatic: let CPLEX choose 1 [CPX_BRDIR_UP] Up branch selected first Default: 0
Description: MIP branching direction. Used to decide which branch, the up or the down branch, should be taken first at each node.			
CPX_PARAM_BTOL BtTol mip strategy backtrack	2002	double	Any number from 0.0 to 1.0 Default: 0.9999
Description: Backtracking tolerance. 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 (see CPX_PARAM_CUTUP and CPX_PARAM_CUTLO) 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 it seems a promising candidate for finding an integer feasible solution.			
CPX_PARAM_CLIQUES Cliques mip cuts cliques	2003	int	-1 Do not generate clique cuts 0 Automatic: let CPLEX choose 1 Generate clique cuts moderately 2 Generate clique cuts aggressively 3 Generate clique cuts very aggressively Default: 0
Description: MIP cliques indicator. Determines whether or not clique cuts should be generated for the problem. Setting the value to 0, the default, indicates that the attempt to generate cliques should continue only if it seems to be helping.			

Parameter Name	Code	Type	Value [Symbolic Constants]
CPX_PARAM_CLOCKTYPE ClockType clocktype	1006	int	1 CPU time 2 Wall clock time (total physical time elapsed) Default: 1
<p>Description: Computation time reporting. Determines how computation times are measured on UNIX platforms. Computation time on Windows systems is always measured as wall clock time. Small variations in measured time on identical runs may be expected on any computer system under either setting of this parameter.</p>			
CPX_PARAM_COEREDIND CoeRedInd preprocessing coeffreduce	2004	int	0 Do not use coefficient reduction 1 Reduce only to integral coefficients 2 Reduce all potential coefficients Default: 2
<p>Description: Coefficient reduction setting Determines how coefficient reduction is used. Coefficient reduction improves the objective value of the initial (and subsequent) LP relaxations solved during branch & cut by reducing the number of non-integral vertices.</p>			
CPX_PARAM_COLREADLIM ColReadLim read variables	1023	int	Any integer from 0 to 268 435 450 Default: 60 000
<p>Description: Variable (column) read limit. 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.</p>			
CPX_PARAM_CONFLICTDISPLAY ConflictDisplay display conflict	1074	int	0 no display 1 summary display 2 detailed display Default: 1
<p>Description: Conflict information display Determines what CPLEX reports when the conflict refiner is working.</p>			

Parameter Name	Code	Type	Value [Symbolic Constants]
CPX_PARAM_COVERS Covers mip cuts covers	2005	int	-1 Do not generate cover cuts 0 Automatic: let CPLEX choose 1 Generate cover cuts moderately 2 Generate cover cuts aggressively 3 Generate cover cuts very aggressively Default: 0
<p>Description: MIP covers indicator. Determines whether or not cover cuts should be generated for the problem. Setting the value to 0, the default, indicates that the attempt to generate covers should continue only if it seems to be helping.</p>			
CPX_PARAM_CRAIND CraInd simplex crash	1007	int	<p>LP Primal: 0 Ignore objective coefficients during crash -1 or 1 Alternate ways of using objective coefficients</p> <p>LP Dual: 1 Default starting basis 0 or -1 Aggressive starting basis</p> <p>QP Primal: -1 Slack basis 0 Ignore Q terms and use LP solver for crash 1 Ignore objective and use LP solver for crash</p> <p>QP Dual: -1 Slack basis 0 or 1 Use Q terms for crash Default: 1</p>
<p>Description: Simplex crash ordering. Determines how CPLEX orders variables relative to the objective function when selecting an initial basis.</p>			
CPX_PARAM_CUTLO CutLo mip tolerances lowercutoff	2006	double	Any number Default: $-1e^{+75}$
<p>Description: Lower cutoff. When the problem is a maximization problem, the lower cutoff parameter is used to cut off any nodes that have an objective value at or below the lower cutoff value. When a mixed integer optimization problem is continued, the larger of these values and the updated cutoff found during optimization are used during the next mixed integer optimization. A too-restrictive value for the lower cutoff parameter may result in no integer solutions being found.</p>			

Parameter Name	Code	Type	Value [Symbolic Constants]
CPX_PARAM_CUTPASS CutPass mip limits cutpasses	2056	int	-1 None 0 Automatic: let CPLEX choose Positive values give number of passes to perform Default: 0
Description: 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.			
CPX_PARAM_CUTSFACOR CutsFactor mip limits cutsfactor	2033	double	Any nonnegative number Default: 4.0
Description: Row multiplier factor for cuts. Limits the number of cuts that can be added. The number of rows in the problem with cuts added is limited to CUTSFACOR times the original number of rows. If the problem is presolved, the original number of rows is that from the presolved problem. A CUTSFACOR of 1.0 or less means that no cuts will be generated. Because cuts can be added and removed during the course of optimization, CUTSFACOR may not correspond directly to the number of cuts seen during the node log or in the summary table at the end of optimization.			
CPX_PARAM_CUTUP CutUp mip tolerances uppercutoff	2007	double	Any number Default: $1e^{+75}$
Description: Upper cutoff. Cuts off any nodes that have an objective value at or above the upper cutoff value, when the problem is a minimization problem. When a mixed integer optimization problem is continued, the smaller of these values and the updated cutoff found during optimization are used during the next mixed integer optimization. A too-restrictive value for the upper cutoff parameter may result in no integer solutions being found.			
CPX_PARAM_DATACHECK DataCheck read datacheck	1056	int bool	0 [CPX_OFF/false] Off (do not check) 1 [CPX_ON/true] On (check) Default: 0
Description: Data consistency checking indicator. When this parameter is set to CPX_ON, the routines CPXcopy____, CPXread____ and CPXchg____ 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.			

Parameter Name	Code	Type	Value [Symbolic Constants]
CPX_PARAM_DEPIND DepInd preprocessing dependency	1008	int	-1 Automatic: let CPLEX choose 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 Default: -1 automatic
<p>Description: Dependency indicator. Determines whether to activate the dependency checker. If on, the dependency checker searches for dependent rows during preprocessing. If off, dependent rows are not identified.</p>			
CPX_PARAM_DISJ CUTS DisjCuts mip cuts disjunctive	2053	int	-1 Do not generate disjunctive cuts 0 Automatic: let CPLEX choose 1 Generate disjunctive cuts moderately 2 Generate disjunctive cuts aggressively 3 Generate disjunctive cuts very aggressively Default: 0
<p>Description: MIP disjunctive cuts indicator. Determines whether or not disjunctive cuts should be generated for the problem. Setting the value to 0, the default, indicates that the attempt to generate disjunctive cuts should continue only if it seems to be helping.</p>			
CPX_PARAM_DIVETYPE DiveType mip strategy dive	2060	int	0 Automatic: let CPLEX choose 1 Traditional dive 2 Probing dive 3 Guided dive Default: 0
<p>Description: 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 directs CPLEX never to perform probing dives, 2 always to probe, 3 spend more time exploring potential solutions that are similar to the current incumbent. Setting 2, always to probe, is helpful for finding integer solutions.</p>			

Parameter Name	Code	Type	Value [Symbolic Constants]
CPX_PARAM_DPRIIND DPriInd simplex dgradient	1009	int	0 [CPX_DPRIIND_AUTO] Automatic: let CPLEX choose 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_STEEPOSTART] Steepest-edge pricing, unit initial norms 5 [CPX_DPRIIND_DEVEX] devex pricing Default: 0
Description: Dual simplex pricing algorithm. The default pricing (0) usually provides the fastest solution time, but many problems benefit from alternate settings.			
CPX_PARAM_EPAGAP EpAGap mip tolerances absmipgap	2008	double	Any nonnegative number Default: 1e ⁻⁰⁶
Description: Absolute mipgap tolerance. 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 the EpAGap parameter, the mixed integer optimization is stopped.			
CPX_PARAM_EPGAP EpGap mip tolerances mipgap	2009	double	Any number from 0.0 to 1.0 Default: 1e ⁻⁰⁴
Description: Relative mipgap tolerance. Sets a relative tolerance on the gap between the best integer objective and the objective of the best node remaining. When the value $\frac{ bestnode - bestinteger }{(1e-10 + bestinteger)}$ falls below the value of the EpGap 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 mipgap tolerance to 0.05.			
CPX_PARAM_EPINT EpInt mip tolerances integrality	2010	double	Any number from 0.0 to 1.0 Default: 1e ⁻⁰⁵
Description: Integrality tolerance. 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 roundoff 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.			

Parameter Name	Code	Type	Value [Symbolic Constants]
CPX_PARAM_EPLIN (n/a) EpLin not available in Interactive Optimizer	2068	double	Any positive value greater than zero Default: 1e ⁻³
<p>Description: Epsilon used in linearization. Sets the epsilon (degree of tolerance) used in linearization in Concert Technology. Not applicable in the Callable Library. Not available in the Interactive Optimizer. This parameter controls how strict inequalities are managed during linearization. In other words, it provides an epsilon for determining when two values are not equal during linearization. For example, when x is a numeric variable (that is, an instance of <code>IloNumVar</code>), $x < a$ becomes $x \leq a - \text{eplin}$. Similarly, $x \neq a$ becomes $\{(x < a) \ \ (x > a)\}$ which is linearized automatically for you in Concert Technology as $\{(x \leq a - \text{eplin}) \ \ (x \geq a + \text{eplin})\}$. 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 a Concert Technology 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.</p>			
CPX_PARAM_EPMRK EpMrk simplex tolerances markowitz	1013	double	Any number from 0.0001 to 0.99999 Default: 0.01
<p>Description: Markowitz tolerance. Influences pivot selection during basis factoring. Increasing the Markowitz threshold may improve the numerical properties of the solution.</p>			
CPX_PARAM_EPOPT EpOpt simplex tolerances optimality	1014	double	Any number from 1e ⁻⁹ to 1e ⁻¹ Default: 1e ⁻⁰⁶
<p>Description: Optimality tolerance. Influences the reduced-cost tolerance for optimality. This parameter governs how closely CPLEX must approach the theoretically optimal solution.</p>			

Parameter Name	Code	Type	Value [Symbolic Constants]
CPX_PARAM_EPPER EpPer simplex perturbation	1015	double	Any positive number $\geq 1e^{-8}$ Default: $1e^{-6}$
<p>Description: 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. This parameter can be set to a smaller value if the default value creates too large a change in the problem.</p>			
CPX_PARAM_EPRELAX EpRelax feasopt tolerance	2073	double	Any nonnegative value Default $1e^{-6}$
<p>Description: Relaxation for <code>feasOpt</code>. This parameter controls the amount of relaxation for the routine <code>CPXfeasopt</code> in the Callable Library or the method <code>feasOpt</code> in Concert Technology. In the case of a MIP, it serves the purpose of the absolute gap for the <code>feasOpt</code> model in Phase I (the one to minimize relaxation). Using this parameter, you can implement other stopping criteria as well. To do so, first call <code>feasOpt</code> 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 <code>feasOpt</code> again. Since the solution from the first call already matches this parameter, Phase I will terminate immediately in this second call to <code>feasOpt</code>, and Phase II will start. In the case of an LP, this parameter controls the lower objective limit (<code>CPX_PARAM_OBJLLIM</code>) for Phase I of <code>feasOpt</code> and is thus relevant only when the primal optimizer is in use.</p>			
CPX_PARAM_EPRHS EpRHS simplex tolerances feasibility	1016	double	Any number from $1e^{-9}$ to $1e^{-1}$ Default: $1e^{-06}$
<p>Description: Feasibility tolerance. The feasibility tolerance specifies the degree to which a problem's basic variables may violate their bounds. 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 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.</p>			

Parameter Name	Code	Type	Value [Symbolic Constants]
CPX_PARAM_FEASOPTMODE FeasOptMode feasopt mode	1084	int	0 MinSum CPX_FEASOPT_MIN_SUM Minimize the sum of all required relaxations in first phase only 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 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 Default: 0
<p>Description: Mode of FeasOpt</p> <p>Determines 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.</p> <p>Values of this parameter indicate two aspects to ILOG CPLEX: (1) whether to stop in phase one or continue to phase two and (2) how to measure the relaxation (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).</p>			
CPX_PARAM_FLOWCOVERS FlowCovers mip cuts flowcuts	2040	int	-1 Do not generate flow cover cuts 0 Automatic: let CPLEX choose 1 Generate flow cover cuts moderately 2 Generate flow cover cuts aggressively Default: 0
<p>Description: MIP flow cover cuts indicator.</p> <p>Determines whether or not to generate flow cover cuts for the problem. Setting the value to 0, the default, indicates that the attempt to generate flow cover cuts should continue only if it seems to be helping.</p>			

Parameter Name	Code	Type	Value [Symbolic Constants]
CPX_PARAM_FLOWPATHS FlowPaths mip cuts pathcut	2051	int	-1 Do not generate flow path cuts 0 Automatic: let CPLEX choose 1 Generate flow path cuts moderately 2 Generate flow path cuts aggressively Default: 0
Description: MIP flow path cut indicator. Determines whether or not flow path cuts should be generated for the problem. Setting the value to 0, the default, indicates that the attempt to generate flow path cuts should continue only if it seems to be helping.			
CPX_PARAM_FRACCAND FracCand mip limits gomorycand	2048	int	Any positive integer Default: 200
Description: Candidate limit for generating Gomory fractional cuts. Limits the number of candidate variables for generating Gomory fractional cuts.			
CPX_PARAM_FRACCUTS FracCuts mip cuts gomory	2049	int	-1 Do not generate Gomory fractional cuts 0 Automatic: let CPLEX choose 1 Generate Gomory fractional cuts moderately 2 Generate Gomory fractional cuts aggressively Default: 0
Description: MIP Gomory fractional cuts indicator. Determines whether or not Gomory fractional cuts should be generated for the problem. Setting the value to 0, the default, indicates that the attempt to generate Gomory fractional cuts should continue only if it seems to be helping.			
CPX_PARAM_FRACPASS FracPass mip limits gomorypass	2050	int	0 Automatic: let CPLEX choose or, any positive integer Default: 0
Description: Pass limit for generating Gomory fractional cuts. Limits the number of passes for generating Gomory fractional cuts. At the default setting of 0, CPLEX decides. The parameter is ignored if the Gomory fractional cut parameter, CPX_PARAM_FRACCUTS, is set to a nonzero value.			

Parameter Name	Code	Type	Value [Symbolic Constants]
CPX_PARAM_GUBCOVERS GUBCovers mip cuts gubcovers	2044	int	-1 Do not generate GUB cuts 0 Automatically determined 1 Generate GUB cuts moderately 2 Generate GUB cuts aggressively Default: 0
<p>Description: MIP GUB cuts indicator. Determines whether or not to generate GUB cuts for the problem. Setting the value to 0, the default, indicates that the attempt to generate GUB cuts should continue only if it seems to be helping.</p>			
CPX_PARAM_HEURFREQ HeurFreq mip strategy heuristicfreq	2031	int	-1 None 0 Automatic: let CPLEX choose or, any positive integer Default: 0
<p>Description: MIP heuristic frequency. Determines how often to apply the periodic heuristic. Setting the value to -1 turns off the periodic heuristic. Setting the value to 0, 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 HEURISTICFREQ to 20 dictates that the heuristic be called at node 0, 20, 40, 60, etc.</p>			
CPX_PARAM_IMPLBD ImplBd mip cuts implied	2041	int	-1 Do not generate implied bound cuts 0 Automatically determined 1 Generate implied bound cuts moderately 2 Generate implied bound cuts aggressively Default: 0
<p>Description: MIP implied bound cuts indicator. Determines whether or not to generate implied bound cuts for the problem. Setting the value to 0, the default, indicates that the attempt to generate implied bound cuts should continue only if it seems to be helping.</p>			
CPX_PARAM_INTSOLLIM IntSolLim mip limits solutions	2015	int	Any positive integer Default: 2 100 000 000
<p>Description: MIP solution limit. Sets the number of MIP solutions to be found before stopping.</p>			

Parameter Name	Code	Type	Value [Symbolic Constants]
CPX_PARAM_ITLIM ItLim simplex limits iterations	1020	int	Any nonnegative integer Default: 2 100 000 000
<p>Description: Simplex maximum iteration limit. Sets the maximum number of 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.</p>			
CPX_PARAM_LBHEUR LBHeur mip strategy lbheur	2063	int bool	0 [CPX_OFF/false] off (default) 1 [CPX_ON/true] (on) Apply local branching heuristic to new incumbent Default: 0 off
<p>Description: Local branching heuristic. This parameter lets you control 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.</p>			
CPX_PARAM_LPMETHOD RootAlg lpmethod	1062	int	0 [CPX_ALG_AUTOMATIC] Automatic: let CPLEX choose 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) Default: 0
<p>Description: Algorithm for linear optimization. Determines which algorithm is used when <code>CPXlpopt</code> (or <code>optimize</code> in the Interactive Optimizer) is invoked. 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, concurrent optimization algorithm is selected by the automatic setting. The Automatic setting may be expanded in the future so that CPLEX chooses the algorithm based on additional problem characteristics.</p>			

Parameter Name	Code	Type	Value [Symbolic Constants]
CPX_PARAM_MEMORYEMPHASIS MemoryEmphasis emphasis memory	1082	int bool	0 [CPX_OFF/false] (off) Do not conserve memory. 1 [CPX_ON/true] (on) Conserve memory where possible. Default: 0
<p>Description: Reduces use of memory. This parameter lets you indicate to 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.</p>			
CPX_PARAM_MIPCBREDLP (available only in Callable Library)	2055	int	0 [CPX_OFF] (off) Use original problem. 1 [CPX_ON] (on) Use presolved problem. Default: 1 (on)
<p>Description: MIP callback original or reduced, presolved model indicator. Controls whether your callback accesses vectors of the original model (off) or vectors of the reduced, presolved model (on, default). Advanced routines to control MIP callbacks (such as CPXgetcallbacklp, CPXsetheuristiccallbackfunc, CPXsetbranchcallbackfunc, CPXgetbranchcallbackfunc, CPXsetcutcallbackfunc, CPXsetincumbentcallbackfunc, CPXgetcallbackinfos, CPXcutcallbackadd, CPXcutcallbackaddlocal, and others) consider the setting of this parameter and access the original model or the reduced, presolved model accordingly.</p> <p>In the C++, Java, and .NET 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 Callable Library.</p>			
CPX_PARAM_MIPDISPLAY MIPDisplay mip display	2012	int	0 No display 1 Display integer feasible solutions 2 Display nodes under CPX_PARAM_MIPInterval 3 Same as 2 with information about node cuts 4 Same as 3 with LP subproblem information at root 5 Same as 4 with LP subproblem information at nodes Default: 2
<p>Description: MIP node log display information. Determines what CPLEX reports to the screen during mixed integer optimization. The amount of information displayed increases with increasing values of this parameter. A setting of 0 causes no node log to be displayed until the optimal solution is found. A setting of 1 displays an entry for each integer feasible solution found. Each entry contains the objective function value, the node count, the number of unexplored nodes in the tree, and the current optimality gap. A setting of 2 also generates an entry for every n-th node (where n is the setting of the MIP INTERVAL parameter). A setting of 3 additionally generates an entry for every nth node giving the number of cuts added to the problem for the previous INTERVAL nodes. A setting of 4 additionally generates entries for the LP root relaxation according to the set simplex display setting. A setting of 5 additionally generates entries for the LP subproblems, also according to the set simplex display setting.</p>			

Parameter Name	Code	Type	Value [Symbolic Constants]
CPX_PARAM_MIPEMPHASIS MIPEmpphasis emphasis mip	2058	int	0 [CPX_MIPEMPHASIS_BALANCED] Balance optimality and feasibility 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 Default: 0
<p>Description: MIP emphasis indicator. With the default setting of <code>BALANCED</code>, 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 <code>FEASIBILITY</code>, 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 <code>OPTIMALITY</code>, less effort may be applied to finding feasible solutions early. With the setting <code>BESTBOUND</code>, 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 <code>HIDDENFEAS</code>, the MIP optimizer works hard to find high quality feasible solutions that are otherwise very difficult to find, so consider this setting when the <code>FEASIBILITY</code> setting has difficulty finding solutions of acceptable quality.</p>			
CPX_PARAM_MIPINTERVAL MIPInterval mip interval	2013	int	Any positive integer Default: 100
<p>Description: MIP node log interval. Controls the frequency of node logging when <code>CPX_PARAM_MIPDISPLAY</code> is set higher than 1.</p>			
CPX_PARAM_MIPORDIND MIPOrdInd mip strategy order	2020	int bool	0 [CPX_OFF/false] (off) Do not use order information 1 [CPX_ON/true] (on) Use order information if it exists Default: 1
<p>Description: MIP priority order indicator. When set to on, uses the priority order (if it exists) for the next mixed integer optimization.</p>			

Parameter Name	Code	Type	Value [Symbolic Constants]
CPX_PARAM_MIPORDTYPE MIPordType mip ordertype	2032	int	0 Do not generate a priority order 1 [CPX_MIPORDER_COST] Use decreasing cost 2 [CPX_MIPORDER_BOUNDS] Use increasing bound range 3 [CPX_MIPORDER_SCALED COST] Use increasing cost per coefficient count Default: 0
Description: MIP priority order generation. Used to select the type of generic priority order to generate when no priority order is present.			
CPX_PARAM_MIPTHREADS MIPThreads mip limits threads	2014	int	0 Determined by global thread default >0 Upper limit on threads for Parallel MIP Default: 0
Description: MIP thread limit Determines the maximum number of parallel processes (threads) that will be invoked by the Parallel MIP optimizer. The default value of 0 means that the limit will be determined by the value of CPX_PARAM_THREADS, the global thread limit parameter. A positive value will override the value found in CPX_PARAM_THREADS.			
CPX_PARAM_MIRCUTS MIRCuts mip cuts mircut	2052	int	-1 Do not generate MIR cuts 0 Automatic: let CPLEX choose 1 Generate MIR cuts moderately 2 Generate MIR cuts aggressively Default: 0
Description: MIP MIR (mixed integer rounding) cut indicator. Determines whether or not to generate MIR cuts for the problem. Setting the value to 0, the default, indicates that the attempt to generate MIR cuts should continue only if it seems to be helping.			
CPX_PARAM_MPSSLONGNUM MPSLongNum output mpsslong	1081	int bool	0 [CPX_OFF/false] (off) Use limited MPS precision 1 [CPX_ON/true] (on) Use full-precision Default: 1 On
Description: MPS and REW file format precision of numeric output. Determines the precision of numeric 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.			

Parameter Name	Code	Type	Value [Symbolic Constants]
CPX_PARAM_NETDISPLAY NetDisplay network display	5005	int	0 [CPXNET_NO_DISPLAY_OBJECTIVE] No display 1 [CPXNET_TRUE_OBJECTIVE] Display true objective values 2 [CPXNET_PENALIZE_OBJECTIVE] Display penalized objective values Default: 2
Description: Network logging display indicator. Settings 1 and 2 differ only during Phase I. Setting 2 shows monotonic values, whereas 1 usually does not.			
CPX_PARAM_NETEPOPT NetEpOpt network tolerances optimality	5002	double	Any number from $1e^{-11}$ to $1e^{-1}$ Default: $1e^{-6}$
Description: Optimality tolerance for CPXNETprimopt. The optimality tolerance specifies the amount a reduced cost may violate the criterion for an optimal solution.			
CPX_PARAM_NETEPRHS NetEprHS network tolerances feasibility	5003	double	Any number from $1e^{-11}$ to $1e^{-1}$ Default: $1e^{-6}$
Description: Feasibility tolerance for CPXNETprimopt. The feasibility tolerance specifies the degree to which a problem's flow value 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.			
CPX_PARAM_NETFIND NetFind network netfind	1022	int	1 [CPX_NETFIND_PURE] Extract pure network only 2 [CPX_NETFIND_REFLECT] Try reflection scaling 3 [CPX_NETFIND_SCALE] Try general scaling Default: 2
Description: Simplex network extraction level. Establishes the level of network extraction for network simplex optimizations. The default value is suitable for recognizing commonly used modeling approaches when representing a network problem within an LP formulation.			
CPX_PARAM_NETITLIM NetItLim network iterations	5001	int	Any nonnegative integer Default: 2 100 000 000
Description: Network simplex iteration limit. Sets the maximum number of iterations to be performed before the algorithm terminates without reaching optimality.			

Parameter Name	Code	Type	Value [Symbolic Constants]
CPX_PARAM_NETPPIIND NetPpriInd network pricing	5004	int	0 [CPXNET_PRICE_AUTO] Automatic: let CPLEX choose 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 Default: 0
Description: Network Simplex pricing algorithm. The default (0) shows best performance for most problems, and currently is equivalent to 3.			
CPX_PARAM_NODEFILEIND NodeFileInd mip strategy file	2016	int	0 No node file 1 Node file in memory and compressed 2 Node file on disk 3 Node file on disk and compressed Default: 1
Description: Node storage file indicator. Used when working memory, (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 128 MBytes, 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 a directory named by the parameter CPX_PARAM_WORKDIR (WorkDir), and CPLEX actively manages which nodes remain in memory for processing.			
CPX_PARAM_NODELIM NodeLim mip limits nodes	2017	int	Any nonnegative integer Default: 2 100 000 000
Description: MIP node limit. Sets the maximum number of nodes solved before the algorithm terminates without reaching optimality. When this parameter is set to 0 (zero), CPLEX does not create any nodes, but it does solve the root node LP relaxation and repeatedly apply cuts and resolve this LP.			

Parameter Name	Code	Type	Value [Symbolic Constants]
CPX_PARAM_NODESEL NodeSel mip strategy nodeselect	2018	int	0 [CPX_NODESEL_DFS] Depth-first search 1 [CPX_NODESEL_BESTBOUND] Best-bound search 2 [CPX_NODESEL_BESTEST] Best-estimate search 3 [CPX_NODESEL_BESTEST_ALT] Alternative best-estimate search Default: 1
<p>Description: MIP node selection strategy. Used to set the rule for selecting the next node to process when 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.</p>			
CPX_PARAM_NUMERICALEMPHASIS NumericalEmphasis emphasis numerical	1083	int bool	0 [CPX_OFF/false] (default) 1 [CPX_ON/true] Exercise extreme caution in computation Default: 0 Off
<p>Description: Emphasizes precision in numerically unstable or difficult problems. This parameter lets you indicate to CPLEX that it should emphasize precision in numerically difficult or unstable problems, with consequent performance trade-offs in time and memory.</p>			
CPX_PARAM_NZREADLIM NzReadLim read nonzeros	1024	int	Any integer from 0 to 268 435 450 Default: 250 000
<p>Description: Nonzero element read limit. 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.</p>			
CPX_PARAM_OBJDIF ObjDif mip tolerances objdifference	2019	double	Any number Default: 0.0
<p>Description: Absolute objective difference cutoff. 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.</p>			

Parameter Name	Code	Type	Value [Symbolic Constants]
CPX_PARAM_OBLLIM ObjLLim simplex limits lowerobj	1025	double	Any number Default: $-1e^{+75}$
Description: Lower objective value limit. Setting a lower objective function limit causes CPLEX to halt the optimization process once the minimum objective function value limit has been exceeded. This limit applies only during Phase II of the simplex algorithm.			
CPX_PARAM_OBJULIM ObjULim simplex limits upperobj	1026	double	Any number Default: $1e^{+75}$
Description: Upper objective value limit. Setting an upper objective function limit causes CPLEX to halt the optimization process once the maximum objective function value limit has been exceeded. This limit applies only during Phase II of the simplex algorithm.			
CPX_PARAM_PERIND PerInd simplex perturbation	1027	int bool	0 [CPX_OFF/false] Off 1 [CPX_ON/true] On Default: 0
Description: Simplex perturbation indicator. Setting this parameter to 1 causes all problems to be automatically perturbed as optimization begins. A setting of 0 allows CPLEX to determine dynamically, during solution, whether progress is slow enough to merit a perturbation. The situations in which a setting of 1 helps are rare and restricted to problems that exhibit extreme degeneracy.			
CPX_PARAM_PERLIM PerLim simplex limits perturbation	1028	int	0 Determined automatically or, any positive integer Default: 0
Description: Simplex perturbation limit. Sets the number of degenerate iterations before perturbation is performed.			
CPX_PARAM_POLISHTIME PolishTime mip limit polishtime	2066	double	Any nonnegative value in seconds. Default: 0 (zero) seconds
Description: Time spent polishing a solution. This parameter lets you indicate to CPLEX how much time in seconds to spend after a normal mixed integer optimization in polishing a solution. Default is zero, no polishing time.			

Parameter Name	Code	Type	Value [Symbolic Constants]
CPX_PARAM_PPRIIND PPriInd simplex pgradient	1029	int	-1 [CPX_PPRIIND_PARTIAL] Reduced-cost pricing 0 [CPX_PPRIIND_AUTO] Hybrid reduced-cost & devex pricing 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 Default: 0
Description: Primal Simplex pricing algorithm. The default pricing (0) usually provides the fastest solution time, but many problems benefit from alternative settings.			
CPX_PARAM_PREDUAL PreDual preprocessing dual	1044	int	-1 Off 0 Automatic: let CPLEX choose 1 On Default: 0
Description: Presolve dual setting. Determines whether CPLEX Presolve should pass the primal or dual linear programming problem to the linear programming optimization algorithm. By default, CPLEX chooses automatically. If the PreDual 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.			
CPX_PARAM_PREIND PreInd preprocessing presolve	1030	int bool	0 [CPX_OFF/false] Off (do not use presolve) 1 [CPX_ON/true] On (use presolve) Default: 1
Description: Presolve indicator. When set to 1, invokes the CPLEX Presolve to simplify and reduce problems.			
CPX_PARAM_PRELINEAR PreLinear preprocessing linear	1058	int	0 Only linear reductions 1 Full reductions Default: 1
Description: Linear reduction indicator. 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 guarantees, for example, that users can add their own custom cuts to the presolved model.			

Parameter Name	Code	Type	Value [Symbolic Constants]
CPX_PARAM_PREPASS PrePass preprocessing numpass	1052	int	-1 Determined automatically 0 Do not use Presolve; other reductions may still occur or, any positive integer Default: -1
<p>Description: Limit on the number of Presolve passes made. When set to a nonzero value, invokes CPLEX Presolve to simplify and reduce problems. When 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 should continue only if it seems to be helping. When set to zero, CPLEX does not apply Presolve, but other reductions may occur, depending on settings of other parameters and specifics of your model.</p>			
CPX_PARAM_PRESLVND PreslvNd mip strategy presolvenode	2037	int	-1 No node presolve 0 Automatic: let CPLEX choose 1 Force node presolve 2 Perform probing on integer-infeasible variables Default: 0
<p>Description: Node presolve selector. Indicates whether node presolve should be performed at the nodes of a mixed integer programming solution. Node presolve can significantly reduce solution time for some models. The default setting is generally effective at determining whether to apply node presolve, although runtimes can be reduced for some models by turning node presolve off.</p>			
CPX_PARAM_PRICE LIM PriceLim simplex pricing	1010	int	0 Determined automatically or, any positive integer Default: 0
<p>Description: Simplex pricing candidate list size. Sets the maximum number of variables kept in the pricing candidate list.</p>			
CPX_PARAM_PROBE Probe mip strategy probe	2042	int	-1 No probing 0 Automatic: let CPLEX choose 1 Moderate probing level 2 Aggressive probing level 3 Very aggressive probing level Default: 0
<p>Description: MIP probing level. Determines 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.</p>			

Parameter Name	Code	Type	Value [Symbolic Constants]
CPX_PARAM_PROBETIME ProbeTime mip limit probetime	2065	double	Any nonnegative number Default: 1e+75
Description: Time spent probing Limits the amount of time in seconds spent probing.			
CPX_PARAM_QPMAKEPSDIND QPmakePSDInd preprocessing qpmakepsd	4010	int bool	0 [CPX_OFF/false] Off 1 [CPX_ON/true] On Default: On
Description: Indefinite MIQP indicator. Determines 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 will also attempt to tighten an already PSD matrix for better numeric behavior. The default setting of 1 means yes, CPLEX should attempt to reformulate, but you can turn it off if necessary; most models should benefit from the default setting.			
CPX_PARAM_QPMETHOD RootAlg qpmethod	1063	int	0 [CPX_ALG_AUTOMATIC] Automatic: let CPLEX choose 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) Default: 0
Description: Algorithm for continuous quadratic optimization. Determines which algorithm is used when CPXqpopt (or 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.			
CPX_PARAM_QPNZREADLIM QPNzReadLim read qpnnonzeros	4001	int	Any integer from 0 to 268 435 450 Default: 5 000
Description: QP Q matrix nonzero read limit. 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.			

Parameter Name	Code	Type	Value [Symbolic Constants]
CPX_PARAM_REDUCE Reduce preprocessing reduce	1057	int	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: 3
Description: Primal and dual reduction type. Determines whether primal reductions, dual reductions, both, or neither are performed during preprocessing.			
CPX_PARAM_REINV ReInv simplex refactor	1031	int	0 Determined automatically or, any integer from 1 to 10,000 Default: 0
Description: Simplex refactoring frequency. Sets the number of iterations between refactoring of the basis matrix.			
CPX_PARAM_RELAXPREIND RelaxPreInd preprocessing relax	2034	int	-1 Automatic: let CPLEX choose 0 [CPX_OFF] Off (do not use presolve on initial relaxation) 1 [CPX_ON] On (use presolve on initial relaxation) Default: -1(automatic)
Description: Relaxed LP presolve indicator. Determines whether LP presolve is applied to the root relaxation in a mixed integer program. 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.			
CPX_PARAM_RELOBJDIF RelObjDif mip tolerances reobjdifference	2022	double	Any number from 0.0 to 1.0 Default: 0.0
Description: Relative objective difference cutoff. 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 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 RELOBJDIFFERENCE and OBJDIFFERENCE are nonzero, the value of OBJDIFFERENCE is used.			

Parameter Name	Code	Type	Value [Symbolic Constants]
CPX_PARAM_REPAIRTRIES RepairTries mip limits repairtries	2067	int	-1 None: do not try to repair 0 Automatic: let CPLEX choose or, any positive integer indicates the frequency Default: 0
<p>Description: Try to repair infeasible MIP start. This parameter lets you indicate to 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.</p>			
CPX_PARAM_REPEATPRESOLVE RepeatPresolve preprocessing repeatpresolve	2064	int	-1 Automatic: let CPLEX choose (default) 0 Turn off represolve 1 Represolve without cuts 2 Represolve with cuts 3 Represolve with cuts and allow new root cuts Default: -1
<p>Description: Reapply presolve after processing the root node. This integer parameter tells CPLEX whether to re-apply presolve, with or without cuts, to a MIP model after processing at the root is otherwise complete.</p>			
CPX_PARAM_RINSHEUR RINSHeur mip strategy rinsheur	2061	int	-1 None 0 Automatic (default) or, any positive integer Default: 0
<p>Description: RINS heuristic frequency. Determines 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, 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.</p>			
CPX_PARAM_ROWREADLIM RowReadLim read constraints	1021	int	Any integer from 0 to 268 435 450 Default: 30 000
<p>Description: Constraint (row) read limit. 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.</p>			

Parameter Name	Code	Type	Value [Symbolic Constants]
CPX_PARAM_SCAIND ScaInd read scale	1034	int	-1 No scaling 0 Equilibration scaling 1 More aggressive scaling Default: 0
Description: Scale parameter. Indicates how to scale the problem matrix.			
CPX_PARAM_SCRIND (available only in Callable Library)	1035	int	0 [CPX_OFF] Off 1 [CPX_ON] On Default: 0
Description: Messages to screen indicator. Indicates whether or not results messages are displayed on screen in a Callable Library application. To turn off output to the screen, in a C++ application, use <code>cplex.setOut(env.getNullStream())</code> , where <code>cplex</code> is an instance of the class <code>IloCplex</code> and <code>env</code> is an instance of the class <code>IloEnv</code> , the environment. In a Java application, use <code>cplex.setOut(null)</code> . In a .NET application, use <code>Cplex.SetOut(Null)</code> .			
CPX_PARAM_SIFTALG SiftAlg sifting algorithm	1077	int	0 [CPX_ALG_AUTOMATIC] Automatic: let CPLEX choose 1 [CPX_ALG_PRIMAL] Primal Simplex 2 [CPX_ALG_DUAL] Dual Simplex 3 [CPX_ALG_NET] Network Simplex 4 [CPX_ALG_BARRIER] Barrier Default: 0
Description: Sifting subproblem algorithm Sets the algorithm to be used for solving sifting subproblems. The default automatic setting will typically use a mix of barrier and primal simplex.			
CPX_PARAM_SIFTDISPLAY SiftDisplay sifting display	1076	int	0 No display 1 Display major iterations 2 Display LP subproblem information within each sifting iteration Default: 1
Description: Sifting display information. Determines the amount of information to be displayed about the progress of sifting.			

Parameter Name	Code	Type	Value [Symbolic Constants]
CPX_PARAM_SIFTITLIM SiftItLim sifting iterations	1078	int	Any nonnegative integer Default: 2 100 000 000
Description: Upper limit on sifting iterations. Sets the maximum number of sifting iterations that may be performed if convergence to optimality has not been reached.			
CPX_PARAM_SIMDISPLAY SimDisplay simplex display	1019	int	0 No iteration messages until solution 1 Iteration info after each refactoring 2 Iteration info for each iteration Default: 1
Description: Simplex iteration display information. Determines how often CPLEX reports during simplex optimization.			
CPX_PARAM_SINGLIM SingLim simplex limits singularity	1037	int	Any nonnegative integer Default: 10
Description: Simplex singularity repair limit. 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.			

Parameter Name	Code	Type	Value [Symbolic Constants]
CPX_PARAM_STARTALG RootAlg mip strategy startalgorithm	2025	int	0 [CPX_ALG_AUTOMATIC] Automatic 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 Default: 0
<p>Description: MIP starting algorithm. Determines which continuous optimizer will be used to solve the initial relaxation of a MIP.</p> <p>The default Automatic setting (0) of this parameter currently selects the dual simplex optimizer for root relaxations 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.</p> <p>For MILP (integer constraints and otherwise continuous variables), all settings are permitted.</p> <p>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 and setting 6 (Concurrent) reverts to 4.</p> <p>For MIQCP (integer constraints and positive semi-definite quadratic terms among the constraints), only the Barrier optimizer is implemented, and therefore no settings other than 0 and 4 are permitted.</p>			
CPX_PARAM_STRONGCANDLIM StrongCandLim mip limits strongcand	2045	int	Any positive number Default: 10
<p>Description: MIP candidate list Controls the length of the candidate list when CPLEX uses the setting strong branching variable selection (set mip strategy variableselect 3).</p>			
CPX_PARAM_STRONGITLIM StrongItLim mip limits strongit	2046	int	0 Automatic: let CPLEX choose or any positive integer Default: 0
<p>Description: MIP simplex iterations Controls the number of simplex iterations performed on each variable in the candidate list when CPLEX uses the setting strong branching variable selection (set mip strategy variableselect 3). The default setting 0 chooses the iteration limit automatically.</p>			

Parameter Name	Code	Type	Value [Symbolic Constants]
CPX_PARAM_STRONGTHREADLIM StrongThreadLim mip limits strongthreads	2047	int	Any positive integer Default: 1
<p>Description: MIP parallel threads Controls the number of parallel threads used to perform strong branching. Note that this parameter does nothing if the MIP thread limit (set <code>mip limits threads</code>) is greater than 1. Note also that the global thread limit, <code>CPX_PARAM_THREADS</code>, does not affect this parameter.</p>			
CPX_PARAM_SUBALG NodeAlg mip strategy subalgorithm	2026	int	0 [CPX_ALG_AUTOMATIC] Automatic: let CPLEX choose 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 Default: 0
<p>Description: MIP subproblem algorithm. Determines which continuous optimizer will be used to solve the subproblems in a MIP, after the initial relaxation. The default Automatic setting (0) 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 Barrier optimizer is implemented, and therefore no settings other than 0 (Automatic) and 4 (Barrier) are permitted.</p>			
CPX_PARAM_SUBMIPNODELIM SubMIPNodeLim mip limits submipnodelim	2062	int	Any positive integer Default: 500
<p>Description: Limit on nodes explored when a subMIP is being solved. Restricts the number of nodes explored when CPLEX is solving a subMIP. CPLEX solves subMIPs when it builds a solution from a partial MIP start, when repairing an infeasible MIP start, when executing the relaxation induced neighborhood search (RINS) heuristic, when branching locally, or when polishing a solution.</p>			

Parameter Name	Code	Type	Value [Symbolic Constants]
CPX_PARAM_SYMMETRY Symmetry preprocessing symmetry	2059	int	-1 Automatic: let CPLEX determine the level of symmetry breaking 0 Off 1 CPLEX exerts a moderate level of symmetry breaking 2 CPLEX exerts an aggressive level of symmetry breaking 3 CPLEX exerts a very aggressive level of symmetry breaking Default: -1 (automatic)
<p>Description: Symmetry breaking. Determines whether symmetry breaking reductions will be automatically executed, during the preprocessing phase, in a MIP model.</p>			
CPX_PARAM_THREADS Threads threads	1067	int	Minimum: 1 Maximum: determined by license key and computer Default: 1
<p>Description: Global default thread count. Determines the default number of parallel processes (threads) that will be invoked by any CPLEX parallel optimizer. This provides a convenient way to control parallelism with a single parameter setting. The value in place for this parameter can be overridden for any particular CPLEX parallel optimizer by setting the appropriate thread limit (CPX_PARAM_BARTHREADS or CPX_PARAM_MIPTHREADS).</p>			
CPX_PARAM_TILIM TiLim timelimit	1039	double	Any nonnegative number Default: 1e ⁺⁷⁵
<p>Description: Global time limit. 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 <code>ClockType</code> parameter. 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 mipopt twice then there could be a total of (at most) 20 seconds of running time if each call consumes its maximum allotment.</p>			
CPX_PARAM_TRELIM TreLim mip limits treememory	2027	double	Any nonnegative number Default: 1e ⁺⁷⁵
<p>Description: Tree memory limit. Sets an absolute upper limit on the size (in megabytes) of the branch & cut tree. If this limit is exceeded, CPLEX terminates optimization.</p>			

Parameter Name	Code	Type	Value [Symbolic Constants]
CPX_PARAM_VARSEL VarSel mip strategy variableselect	2028	int	-1 [CPX_VARSEL_MININFEAS] Branch on variable with minimum infeasibility 0 [CPX_VARSEL_DEFAULT] Branch variable automatically selected 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 Default: 0
<p>Description: MIP variable selection strategy. Sets the rule for selecting the branching variable at the node which has been selected for branching. The maximum infeasibility rule chooses the variable with the value furthest from an integer; 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 (1) 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) allows CPLEX to select the best rule based on the problem and its progress.</p>			
CPX_PARAM_WORKDIR WorkDir workdir	1064	string	Default: '.'
<p>Description: Directory for working files. 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.</p>			
CPX_PARAM_WORKMEM WorkMem workmem	1065	double	Any nonnegative number, in megabytes Default: 128.0
<p>Description: 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 before swapping to disk files. See also CPX_PARAM_WORKDIR, WorkDir.</p>			

Index

A

accessing parameters (how to)

Callable Library

- CPXgetdblparam **10**
- CPXgetintparam **10**
- CPXgetstrparam **10**
- CPXinfodblparam **10**
- CPXinfointparam **10**
- CPXinfostrparam **10**
- CPXsetdblparam **10**
- CPXsetdefaults **10**
- CPXsetintparam **10**
- CPXsetstrparam **10**

Concert Technology

- getDefault **9**
- getMax **9**
- getMin **9**
- getParam **9**
- setDefault **10**
- setParam **9**

AdvInd **13**

AggCutLim **13**

AggFill **13**

AggInd **14**

B

BarAlg **14**

BarColNz **14**

BarCrossAlg **14**

BarDisplay **15**

BarEpComp **15**

BarGrowth **15**

BarItLim **15**

BarMaxCor **16**

BarObjRng **16**

BarOrder **16**

BarQCPEpComp **16**

barrier

- algorithm **14**

- as algorithm for continuous quadratic optimization **39**

- as algorithm for linear optimization **29**

- as MIP starting algorithm **44**

- as MIP subproblem LP algorithm **45**

- as sifting subproblem algorithm **42**

- column nonzeros **14**

- convergence tolerance for LP and QP problems **15**

- convergence tolerance for QCP problems **16**

- crossover algorithm **14**

- directory for working files (out of core) **47**

- display information **15**

- global time limit **46**

- growth limit **15**

- iteration limit **15**

- maximum correction limit **16**

- objective range **16**

- ordering algorithm **16**

- starting point algorithm **17**

- thread limit **17**

BarStartAlg **17**
BarThreads **17**
BBInterval **17**
BndStrenInd **17**
BrDir **18**
BtTol **18**

C

Callable Library

CPX_PARAM_ADVIND **13**
CPX_PARAM_AGGCUTLIM **13**
CPX_PARAM_AGGFILL **13**
CPX_PARAM_AGGIND **14**
CPX_PARAM_BARALG **14**
CPX_PARAM_BARCOLNZ **14**
CPX_PARAM_BARCROSSALG **14**
CPX_PARAM_BARDISPLAY **15**
CPX_PARAM_BAREPCOMP **15**
CPX_PARAM_BARGROWTH **15**
CPX_PARAM_BARTLIM **15**
CPX_PARAM_BARMAXCOR **16**
CPX_PARAM_BAROBRJNG **16**
CPX_PARAM_BARORDER **16**
CPX_PARAM_BARQCPEPCOMP **16**
CPX_PARAM_BARSSTARTALG **17**
CPX_PARAM_BARTHREADS **17**
CPX_PARAM_BBINTERVAL **17**
CPX_PARAM_BNDSTRENIND **17**
CPX_PARAM_BRDIR **18**
CPX_PARAM_BTTOL **18**
CPX_PARAM_CLIQUES **18**
CPX_PARAM_CLOCKTYPE **19**
CPX_PARAM_COEREDIND **19**
CPX_PARAM_COLREADLIM **19**
CPX_PARAM_CONFLICTDISPLAY **19**
CPX_PARAM_COVERS **20**
CPX_PARAM_CRAIND **20**
CPX_PARAM_CUTLO **20**
CPX_PARAM_CUTPASS **21**
CPX_PARAM_CUTSFACOR **21**
CPX_PARAM_CUTUP **21**
CPX_PARAM_DATACHECK **21**
CPX_PARAM_DEPIND **22**
CPX_PARAM_DISJUNCTIONS **22**
CPX_PARAM_DIVETYPE **22**
CPX_PARAM_DPRIIND **23**
CPX_PARAM_EPAGAP **23**
CPX_PARAM_EPGAP **23**
CPX_PARAM_EPINT **23**
CPX_PARAM_EPLIN **24**
CPX_PARAM_EPMRK **24**
CPX_PARAM_EPOPT **24**
CPX_PARAM_EPPER **25**
CPX_PARAM_EPRELAX **25**
CPX_PARAM_EPRHS **25**
CPX_PARAM_FEASOPTMODE **26**
CPX_PARAM_FLOWCOVERS **26**
CPX_PARAM_FLOWPATHS **27**
CPX_PARAM_FRACCAND **27**
CPX_PARAM_FRACCUTS **27**
CPX_PARAM_FRACPASS **27**
CPX_PARAM_GUBCOVERS **28**
CPX_PARAM_HEURFREQ **28**
CPX_PARAM_IMPLBD **28**
CPX_PARAM_INTSOLLIM **28**
CPX_PARAM_ITLIM **29**
CPX_PARAM_LBHEUR **29**
CPX_PARAM_LPMETHOD **29**
CPX_PARAM_MEMORYEMPHASIS **30**
CPX_PARAM_MIPCBREDLP **30**
CPX_PARAM_MIPDISPLAY **30**
CPX_PARAM_MIPEMPHASIS **31**
CPX_PARAM_MIPINTERVAL **31**
CPX_PARAM_MIPORDIND **31**
CPX_PARAM_MIPORDTYPE **32**
CPX_PARAM_MIPTHREADS **32**
CPX_PARAM_MIRCUTS **32**
CPX_PARAM_MPSLONGNUM **32**
CPX_PARAM_NETDISPLAY **33**
CPX_PARAM_NETEPOPT **33**
CPX_PARAM_NETEPRHS **33**
CPX_PARAM_NETFIND **33**
CPX_PARAM_NETITLIM **33**
CPX_PARAM_NETPPRIIND **34**
CPX_PARAM_NODEFILEIND **34**
CPX_PARAM_NODELIM **34**
CPX_PARAM_NODESEL **35**
CPX_PARAM_NUMERICALEMPHASIS **35**
CPX_PARAM_NZREADLIM **35**

CPX_PARAM_OBJDIF **35**
 CPX_PARAM_OBJLLIM **36**
 CPX_PARAM_OBJULIM **36**
 CPX_PARAM_PERIND **36**
 CPX_PARAM_PERLIM **36**
 CPX_PARAM_POLISHTIME **36**
 CPX_PARAM_PPRIIND **37**
 CPX_PARAM_PREDUAL **37**
 CPX_PARAM_PREIND **37**
 CPX_PARAM_PRELINEAR **37**
 CPX_PARAM_PREPASS **38**
 CPX_PARAM_PRESLVND **38**
 CPX_PARAM_PRICELIM **38**
 CPX_PARAM_PROBE **38**
 CPX_PARAM_PROBETIME **39**
 CPX_PARAM_QPMAKEPSDIND **39**
 CPX_PARAM_QPMETHOD **39**
 CPX_PARAM_QPNZREADLIM **39**
 CPX_PARAM_REDUCE **40**
 CPX_PARAM_REINV **40**
 CPX_PARAM_RELAXPREIND **40**
 CPX_PARAM_RELOBJDIF **40**
 CPX_PARAM_REPAIRTRIES **41**
 CPX_PARAM_REPEATPRESOLVE **41**
 CPX_PARAM_RINSHEUR **41**
 CPX_PARAM_ROWREADLIM **41**
 CPX_PARAM_SCAIND **42**
 CPX_PARAM_SCRIND **42**
 CPX_PARAM_SIFTALG **42**
 CPX_PARAM_SIFTDISPLAY **42**
 CPX_PARAM_SIFTITLIM **43**
 CPX_PARAM_SIMDISPLAY **43**
 CPX_PARAM_SINGLIM **43**
 CPX_PARAM_STARTALG **44**
 CPX_PARAM_STRONGCANDLIM **44**
 CPX_PARAM_STRONGITLIM **44**
 CPX_PARAM_STRONGTHREADLIM **45**
 CPX_PARAM_SUBALG **45**
 CPX_PARAM_SUBMIPNODELIM **45**
 CPX_PARAM_SYMMETRY **46**
 CPX_PARAM_THREADS **46**
 CPX_PARAM_TILIM **46**
 CPX_PARAM_TRELIM **46**
 CPX_PARAM_VARSEL **47**
 CPX_PARAM_WORKDIR **47**

CPX_PARAM_WORKMEM **47**
 Cliques **18**
 ClockType **19**
 CoeRedInd **19**
 ColReadLim **19**
 Concert Technology
 AdvInd **13**
 AggCutLim **13**
 AggFill **13**
 AggInd **14**
 BarAlg **14**
 BarColNz **14**
 BarCrossAlg **14**
 BarDisplay **15**
 BarEpComp **15**
 BarGrowth **15**
 BarItLim **15**
 BarMaxCor **16**
 BarObjRng **16**
 BarOrder **16**
 BarQCPEpComp **16**
 BarStartAlg **17**
 BarThreads **17**
 BBInterval **17**
 BndStrenInd **17**
 BrDir **18**
 BtTol **18**
 Cliques **18**
 ClockType **19**
 CoeRedInd **19**
 ColReadLim **19**
 ConflictDisplay **19**
 Covers **20**
 CraInd **20**
 CutLo **20**
 CutPass **21**
 CutsFactor **21**
 CutUp **21**
 DataCheck **21**
 DepInd **22**
 DisjCuts **22**
 DiveType **22**
 DPriInd **23**
 EpAGap **23**
 EpGap **23**

EpInt **23**
 EpLin **24**
 EpMrk **24**
 EpOpt **24**
 EpPer **25**
 EpRHS **25**
 FeasOptMode **26**
 FlowCovers **26**
 FlowPaths **27**
 FracCand **27**
 FracCuts **27**
 FracPass **27**
 GUBCovers **28**
 HeurFreq **28**
 ImplBd **28**
 IntSolLim **28**
 ItLim **29**
 LBHeur **29**
 MemoryEmphasis **30**
 MIPDisplay **30**
 MIPEmphasis **31**
 MIPInterval **31**
 MIPOrdInd **31**
 MIPOrdType **32**
 MIPThreads **32**
 MIRCuts **32**
 MPSLongNum **32**
 NetDisplay **33**
 NetEpOpt **33**
 NetEpRHS **33**
 NetFind **33**
 NetItLim **33**
 NetPPriInd **34**
 NodeAlg **45**
 NodeFileInd **34**
 NodeLim **34**
 NodeSel **35**
 NumericalEmphasis **35**
 NzReadLim **35**
 ObjDif **35**
 ObjLLim **36**
 ObjULim **36**
 PerInd **36**
 PerLim **36**
 PolishTime **36**

PPriInd **37**
 PreDual **37**
 PreInd **37**
 PrePass **38**
 PreslvNd **38**
 PriceLim **38**
 Probe **38**
 ProbeTime **39**
 QPmakePSDInd **39**
 QPNzReadLim **39**
 Reduce **40**
 ReInv **40**
 RelaxPreInd **40**
 RelObjDif **40**
 RepairTries **41**
 RepeatPresolve **41**
 RINSHeur **41**
 RootAlg **29, 39, 44**
 RowReadLim **41**
 ScaInd **42**
 SiftAlg **42**
 SiftDisplay **42**
 SiftItLim **43**
 SimDisplay **43**
 SingLim **43**
 StrongCandLim **44**
 StrongItLim **44**
 StrongThreadLim **45**
 SubMIPNodeLim **45**
 Symmetry **46**
 Threads **46**
 TiLim **46**
 TreLim **46**
 VarSel **47**
 WorkDir **47**
 WorkMem **47**
 concurrent
 as algorithm for linear optimization **29**
 as MIP starting algorithm **44**
 ConflictDisplay **19**
 Covers **20**
 CPX_PARAM_ADVIND **13**
 CPX_PARAM_AGGCUTLIM **13**
 CPX_PARAM_AGGFILL **13**
 CPX_PARAM_AGGIND **14**

CPX_PARAM_BARALG **14**
 CPX_PARAM_BARCOLNZ **14**
 CPX_PARAM_BARCROSSALG **14**
 CPX_PARAM_BARDISPLAY **15**
 CPX_PARAM_BAREPCOMP **15**
 CPX_PARAM_BARGROWTH **15**
 CPX_PARAM_BARITLIM **15**
 CPX_PARAM_BARMAXCOR **16**
 CPX_PARAM_BAROBJRNG **16**
 CPX_PARAM_BARORDER **16**
 CPX_PARAM_BARQCPEPCOMP **16**
 CPX_PARAM_BARSTARTALG **17**
 CPX_PARAM_BARTHREADS **17**
 CPX_PARAM_BBINTERVAL **17**
 CPX_PARAM_BNDSTRENIND **17**
 CPX_PARAM_BRDIR **18**
 CPX_PARAM_BTTOL **18**
 CPX_PARAM_CLIQUES **18**
 CPX_PARAM_CLOCKTYPE **19**
 CPX_PARAM_COEREDIND **19**
 CPX_PARAM_COLREADLIM **19**
 CPX_PARAM_CONFLICTDISPLAY **19**
 CPX_PARAM_COVERS **20**
 CPX_PARAM_CRAIND **20**
 CPX_PARAM_CUTLO **20**
 CPX_PARAM_CUTPASS **21**
 CPX_PARAM_CUTSFACTOR **21**
 CPX_PARAM_CUTUP **21**
 CPX_PARAM_DATACHECK **21**
 CPX_PARAM_DEPIND **22**
 CPX_PARAM_DISJCUTS **22**
 CPX_PARAM_DIVETYPE **22**
 CPX_PARAM_DPRIIND **23**
 CPX_PARAM_EPAGAP **23**
 CPX_PARAM_EPGAP **23**
 CPX_PARAM_EPINT **23**
 CPX_PARAM_EPLIN **24**
 CPX_PARAM_EPMRK **24**
 CPX_PARAM_EPOPT **24**
 CPX_PARAM_EPPER **25**
 CPX_PARAM_EPRELAX **25**
 CPX_PARAM_EPRHS **25**
 CPX_PARAM_FEASOPTMODE **26**
 CPX_PARAM_FLOWCOVERS **26**
 CPX_PARAM_FLOWPATHS **27**

CPX_PARAM_FRACCAND **27**
 CPX_PARAM_FRACCUTS **27**
 CPX_PARAM_FRACPASS **27**
 CPX_PARAM_GUBCOVERS **28**
 CPX_PARAM_HEURFREQ **28**
 CPX_PARAM_IMPLBD **28**
 CPX_PARAM_INTSOLLIM **28**
 CPX_PARAM_ITLIM **29**
 CPX_PARAM_LBHEUR **29**
 CPX_PARAM_LPMETHOD **29**
 CPX_PARAM_MEMORYEMPHASIS **30**
 CPX_PARAM_MIPCBREDLP **30**
 CPX_PARAM_MIPDISPLAY **30**
 CPX_PARAM_MIPEMPHASIS **31**
 CPX_PARAM_MIPINTERVAL **31**
 CPX_PARAM_MIPORDIND **31**
 CPX_PARAM_MIPORDTYPE **32**
 CPX_PARAM_MIPTHREADS **32**
 CPX_PARAM_MIRCUTS **32**
 CPX_PARAM_MPSLONGNUM **32**
 CPX_PARAM_NETDISPLAY **33**
 CPX_PARAM_NETEPOPT **33**
 CPX_PARAM_NETEPRHS **33**
 CPX_PARAM_NETFIND **33**
 CPX_PARAM_NETITLIM **33**
 CPX_PARAM_NETPPRIIND **34**
 CPX_PARAM_NODEFILEIND **34**
 CPX_PARAM_NODELIM **34**
 CPX_PARAM_NODESEL **35**
 CPX_PARAM_NUMERICALEMPHASIS **35**
 CPX_PARAM_NZREADLIM **35**
 CPX_PARAM_OBJDIF **35**
 CPX_PARAM_OBJLLIM **36**
 CPX_PARAM_OBJJULIM **36**
 CPX_PARAM_PERIND **36**
 CPX_PARAM_PERLIM **36**
 CPX_PARAM_POLISHTIME **36**
 CPX_PARAM_PPRIIND **37**
 CPX_PARAM_PREDUAL **37**
 CPX_PARAM_PREIND **37**
 CPX_PARAM_PRELINEAR **37**
 CPX_PARAM_PREPASS **38**
 CPX_PARAM_PRESLVND **38**
 CPX_PARAM_PRICELIM **38**
 CPX_PARAM_PROBE **38**

CPX_PARAM_PROBETIME **39**
 CPX_PARAM_QPMAKEPSDIND **39**
 CPX_PARAM_QPMETHOD **39**
 CPX_PARAM_QPNZREADLIM **39**
 CPX_PARAM_REDUCE **40**
 CPX_PARAM_REINV **40**
 CPX_PARAM_RELAXPREIND **40**
 CPX_PARAM_RELOBJDIF **40**
 CPX_PARAM_REPAIRTRIES **41**
 CPX_PARAM_REPEATPRESOLVE **41**
 CPX_PARAM_RINSHEUR **41**
 CPX_PARAM_ROWREADLIM **41**
 CPX_PARAM_SCAIND **42**
 CPX_PARAM_SCRIND **42**
 CPX_PARAM_SIFTALG **42**
 CPX_PARAM_SIFTDISPLAY **42**
 CPX_PARAM_SIFTITLIM **43**
 CPX_PARAM_SIMDISPLAY **43**
 CPX_PARAM_SINGLIM **43**
 CPX_PARAM_STARTALG **44**
 CPX_PARAM_STRONGCANDLIM **44**
 CPX_PARAM_STRONGITLIM **44**
 CPX_PARAM_STRONGTHREADLIM **45**
 CPX_PARAM_SUBALG **45**
 CPX_PARAM_SUBMIPNODELIM **45**
 CPX_PARAM_SYMMETRY **46**
 CPX_PARAM_THREADS **46**
 CPX_PARAM_TILIM **46**
 CPX_PARAM_TRELIM **46**
 CPX_PARAM_VARSEL **47**
 CPX_PARAM_WORKDIR **47**
 CPX_PARAM_WORKMEM **47**
 CPXgetchparams **10**
 CPXgetparamname **10**
 CPXgetparamnum **10**
 CraInd **20**
 CutLo **20**
 CutPass **21**
 CutsFactor **21**
 CutUp **21**

D

DataCheck **21**
 DepInd **22**

DisjCuts **22**
 DiveType **22**
 DPriInd **23**

E

emphasis
 memory **30**
 MIP **31**
 numerical **35**
 EpAGap **23**
 EpGap **23**
 EpInt **23**
 EpLin **24**
 EpMrk **24**
 EpOpt **24**
 EpPer **25**
 EpRelax **25**
 EpRHS **25**
 epsilon linearity **24**

F

feasOpt
 relaxation **25**
 FeasOpt mode **26**
 FeasOptMode **26**
 FlowCovers **26**
 FlowPaths **27**
 FracCand **27**
 FracCuts **27**
 FracPass **27**

G

GUBCovers **28**

H

HeurFreq **28**
 heuristic
 local branching **29**

I

ImplBd **28**

infeasible

Barrier and **14**

repairing MIP start **41**

tolerance for network **33**

tolerance for simplex **25**

Interactive Optimizer

advance **13**

barrier algorithm **14**

barrier colnonzeros **14**

barrier convergetol **15**

barrier crossover **14**

barrier display **15**

barrier limits corrections **16**

barrier limits growth **15**

barrier limits iterations **15**

barrier limits objrange **16**

barrier limits threads **17**

barrier ordering **16**

barrier startalg **17**

clocktype **19**

display conflict **19**

emphasis memory **30**

emphasis numerical **35**

feasopt mode **26**

feasopt tolerance **25**

lpmethod **29**

mip cuts cliques **18**

mip cuts covers **20**

mip cuts disjunctive **22**

mip cuts flowcuts **26**

mip cuts gomory **27**

mip cuts gubcovers **28**

mip cuts implied **28**

mip cuts mircut **32**

mip cuts pathcut **27**

mip display **30**

mip emphasis **31**

mip interval **31**

mip lim submipnodes **45**

mip limit probetime **39**

mip limits aggforcut **13**

mip limits cutpasses **21**

mip limits cutsfactor **21**

mip limits gomorycand **27**

mip limits gomorypass **27**

mip limits nodes **34**

mip limits repair tries **41**

mip limits solutions **28**

mip limits strongcand **44**

mip limits strongit **44**

mip limits strongthreads **45**

mip limits threads **32**

mip limits treememory **46**

mip ordertype **32**

mip strategy backtrack **18**

mip strategy bbinterval **17**

mip strategy branch **18**

mip strategy dive **22**

mip strategy file **34**

mip strategy heuristicfreq **28**

mip strategy lbheur **29**

mip strategy nodelist **35**

mip strategy order **31**

mip strategy presolvenode **38**

mip strategy probe **38**

mip strategy rinsheur **41**

mip strategy startalgorithm **44**

mip strategy subalgorithm **45**

mip strategy variablesselect **47**

mip tolerances absmipgap **23**

mip tolerances integrality **23**

mip tolerances lowercutoff **20**

mip tolerances mipgap **23**

mip tolerances objdifference **35**

mip tolerances relobjdifference **40**

mip tolerances uppercutoff **21**

network display **33**

network iterations **33**

network netfind **33**

network pricing **34**

network tolerances feasibility **33**

network tolerances optimality **33**

output mpsnumber **32**

polish time **36**

preprocessing aggregator **14**

preprocessing boundstrength **17**

preprocessing coeffreduce **19**

- preprocessing dependency **22**
- preprocessing dual **37**
- preprocessing fill **13**
- preprocessing linear **37**
- preprocessing numpass **38**
- preprocessing presolve **37**
- preprocessing qpmakepsd **39**
- preprocessing reduce **40**
- preprocessing relax **40**
- preprocessing repeatpresolve **41**
- preprocessing symmetry **46**
- qpmethod **39**
- read constraints **41**
- read datacheck **21**
- read nonzeros **35**
- read qpnonzeros **39**
- read scale **42**
- read variables **19**
- set bar qcpconvergetol **16**
- sifting algorithm **42**
- sifting display **42**
- sifting iterations **43**
- simplex crash **20**
- simplex dgradient **23**
- simplex display **43**
- simplex limits iterations **29**
- simplex limits lowerobj **36**
- simplex limits perturbation **36**
- simplex limits singularity **43**
- simplex limits upperobj **36**
- simplex perturbation **25, 36**
- simplex pgradient **37**
- simplex pricing **38**
- simplex refactor **40**
- simplex tolerances feasibility **25**
- simplex tolerances markowitz **24**
- simplex tolerances optimality **24**
- threads **46**
- workdir **47**
- workmem **47**
- IntSolLim **28**
- ItLim **29**

L

- LBHeur **29**
- linearity, epsilon **24**
- local branching heuristic **29**
- local improvement **36**

M

- memory, conserving **30**
- MemoryEmphasis **30**
- MIP
 - absolute mipgap tolerance **23**
 - absolute objective difference cutoff **35**
 - backtracking tolerance **18**
 - barrier in infeasibility start **14**
 - branching direction **18**
 - callback **30**
 - candidate limit for generating Gomory fractional cuts **27**
 - candidate list **44**
 - cliques indicator **18**
 - constraint aggregation limit for cut generation **13**
 - covers indicator **20**
 - cutting plane passes **21**
 - directory for working files **47**
 - disjunctive cuts indicator **22**
 - dive strategy **22**
 - emphasis indicator **31**
 - flow cover cuts indicator **26**
 - flow path cut indicator **27**
 - Gomory fractional cuts indicator **27**
 - GUB cuts indicator **28**
 - heuristic for local branching **29**
 - heuristic frequency **28**
 - implied bound cuts indicator **28**
 - integrality tolerance **23**
 - local branching heuristic **29**
 - mixed integer rounding (MIR) cut indicator **32**
 - node log display information **30**
 - node log interval **31**
 - node selection strategy **35**
 - number of cutting plane passes **21**
 - parallel threads **45**
 - pass limit for generating Gomory fractional cuts **27**
 - preprocessing aggregator application limit **14**

- priority order generation **32**
- priority order indicator **31**
- re-apply presolve **41**
- relative mipgap tolerance **23**
- relaxation induced neighborhood search (RINS) heuristic **45**
- row multiplier factor for cuts **21**
- simplex iterations **44**
- solution limit **28**
- starting algorithm **44**
- strategy best bound interval **17**
- subnode limit **45**
- subproblem LP algorithm **45**
- subproblems and barrier **14**
- symmetry breaking cuts **46**
- thread limit **32**
- tolerances lower cutoff **20**
- tolerances upper cutoff **21**

MIP start

- repair tries **41**

MIPDisplay **30**

MIPEmphasis **31**

MIPInterval **31**

MIPOrdInd **31**

MIPOrdType **32**

MIPThreads **32**

MIQP

- indefiniteness indicator **39**

MIRCuts **32**

MPS file format

- numeric precision **32**

- output precision **32**

MPSLongNum **32**

N

NetDisplay **33**

NetEpOpt **33**

NetEpRHS **33**

NetFind **33**

NetItLim **33**

NetPPriInd **34**

network

- as algorithm for continuous quadratic optimization **39**

- as algorithm for linear optimization **29**

- as MIP starting algorithm **44**
- as MIP subproblem LP algorithm **45**
- as sifting subproblem algorithm **42**
- extraction level **33**
- feasibility tolerance **33**
- global time limit **46**
- logging display indicator **33**
- optimality tolerance **33**
- simplex iteration limit **33**
- simplex pricing algorithm **34**

NodeAlg **45**

NodeFileInd **34**

NodeLim **34**

NodeSel **35**

numerical emphasis **35**

numerical instability **35**

NumericalEmphasis **35**

NzReadLim **35**

O

ObjDif **35**

ObjLLim **36**

ObjULim **36**

P

parameter

- correspondence in APIs **11**

- methods to access **9**

- naming conventions **10**

- save settings in file **12**

PerInd **36**

PerLim **36**

polishing time **36**

PolishTime **36**

PPriInd **37**

PreDual **37**

PreInd **37**

PreLinear **37**

PrePass **38**

preprocessing **41**

PreslvNd **38**

presolving **41**

PriceLim **38**

Probe **38**
probe time **39**
ProbeTime **39**

Q

QCP
 convergence tolerance **16**
QP
 algorithm for continuous quadratic optimization **39**
 convergence tolerance **15**
 indefinite MIQP indicator **39**
 positive semi-definiteness and **39**
 Q matrix nonzero read limit **39**
 simplex crash ordering for **20**
QPmakePSDInd **39**
QPNzReadLim **39**

R

Reduce **40**
ReInv **40**
relaxation
 feasOpt **25**
RelaxPreInd **40**
RelObjDif **40**
repairing MIP start **41**
RepairTries **41**
RepeatPresolve **41**
REW file format
 numeric precision **32**
 output precision **32**
RINSHeur **41**
RootAlg **29, 39, 44**
RowReadLim **41**

S

ScaInd **42**
SiftAlg **42**
SiftDisplay **42**
sifting
 as algorithm for linear optimization **29**
 as MIP starting algorithm **44**
 as MIP subproblem LP algorithm **45**

 display information **42**
 subproblem algorithm **42**
 upper limit on iterations **43**

SiftItLim **43**

SimDisplay **43**

simplex

 as algorithm for linear optimization **29**
 as MIP starting algorithm **44**
 as MIP subproblem LP algorithm **45**
 as QP algorithm at root **39**
 as sifting subproblem algorithm **42**
 crash ordering **20**
 dual pricing algorithm **23**
 feasibility tolerance **25**
 global time limit **46**
 iteration display information **43**
 iterations in MIP **44**
 lower objective value limit **36**
 Markowitz tolerance **24**
 maximum iteration limit **29**
 network extraction level **33**
 network iteration limit **33**
 network pricing algorithm **34**
 optimality tolerance **24**
 perturbation constant **25**
 perturbation indicator **36**
 perturbation limit **36**
 pgradient indicator **37**
 pricing candidate list size **38**
 primal pricing algorithm **37**
 refactoring frequency **40**
 singularity repair limit **43**
 upper objective value limit **36**

SingLim **43**

StrongCandLim **44**

StrongItLim **44**

StrongThreadLim **45**

SubMIPNodeLim **45**

Symmetry **46**

T

Threads **46**

TiLim **46**

time limit

probing **39**
TreLim **46**

V

VarSel **47**

W

WorkDir **47**
WorkMem **47**

