IBM ILOG CPLEX V12.1

Parameters Reference Manual

Legal notices

Copyright notice

Trademarks

IBM, the IBM logo, ibm.com, WebSphere, ILOG, the ILOG design, and CPLEX are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at *http://www.ibm.com/legal/copytrade.shtml*

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

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

Other company, product, or service names may be trademarks or service marks of others.

Further notices

Additional registered trademarks

Python® is a registered trademark of the Python Software Foundation.

MATLAB® is a registered trademark of The MathWorks, Inc.

Acknowledgement of use: dtoa routine of the gdtoa package

ILOG acknowledges use of the `dtoa` routine of the `gdtoa` package, available at

`http://www.netlib.org/fp/`.

The author of this software is David M. Gay.

Table of contents

For technical support

Explains prerequisites and procedure for IBM technical support.

In this section

Contacting IBM Support
Contains information on how to obtain technical support from IBM worldwide, should you encounter any problems in using IBM products.

Contacting IBM Support

IBM Software Support Handbook

This guide contains important information on the procedures and practices followed in the service and support of your IBM products. It does not replace the contractual terms and conditions under which you acquired specific IBM Products or Services. Please review it carefully. You may want to bookmark the site so you can refer back as required to the latest information. The "IBM Software Support Handbook" can be found on the web at *http://www14.software.ibm.com/webapp/set2/sas/f/handbook/home.html*.

Accessing Software Support

When calling or submitting a problem to IBM Software Support about a particular service request, please have the following information ready:

- IBM Customer Number
- The machine type/model/serial number (for Subscription and Support calls)
- Company name
- Contact name
- Preferred means of contact (voice or email)
- Telephone number where you can be reached if request is voice
- Related product and version information
- Related operating system and database information
- Detailed description of the issue
- Severity of the issue in relationship to the impact of it affecting your business needs

Contact by Web

Open service requests is a tool to help clients find the right place to open any problem, hardware or software, in any country where IBM does business. This is the starting place when it is not evident where to go to open a service request.

Service Request (SR) tool offers Passport Advantage clients for distributed platforms online problem management to open, edit and track open and closed PMRs by customer number. Timesaving options: create new PMRs with prefilled demographic fields; describe problems yourself and choose severity; submit PMRs directly to correct support queue; attach troubleshooting files directly to PMR; receive alerts when IBM updates PMR; view reports on open and closed PMRs. You can find information about assistance for SR at *http://www.ibm.com/software/support/help-contactus.html*

System Service Request (SSR) tool is similar to Electronic Service request in providing online problem management capability for clients with support offerings in place on System

i, System p, System z, TotalStorage products, Linux, Windows, Dynix/PTX, Retail, OS/2, Isogon, Candle on OS/390 and Consul z/OS legacy products.

IBMLink SoftwareXcel support contracts offer clients on the System z platform the IBMLink online problem management tool to open problem records and ask usage questions on System z software products. You can open, track, update, and close a defect or problem record; order corrective/preventive/toleration maintenance; search for known problems or technical support information; track applicable problem reports; receive alerts on high impact problems and fixes in error; and view planning information for new releases and preventive maintenance.

Contact by phone

If you have an active service contract maintenance agreement with IBM , or are covered by Program Services, you may contact customer support teams by telephone. For individual countries, please visit the Technical Support section of the *IBM Directory of worldwide contacts*.

Parameters Reference Manual

The behavior of IBM® 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 CPLEX® Component Libraries and the Interactive Optimizer. It also explains how to read and write parameter settings of the C API to a file.

In this section

Accessing parameters
Identifies the accessors for parameters in the different APIs and introduces sets of parameters.

Parameter names
Explains the naming conventions of CPLEX® parameters.

Correspondence of parameters
Associates parameters available in the Callable Library with those available in Concert Technology.

Saving parameter settings to a file
Describes PRM files.

Topical list of parameters
The following lists offer you access to the documentation of CPLEX® parameters, organized by topics.

List of CPLEX parameters
Presents the entire list of parameters

Accessing parameters

The following methods set and access parameters for objects of the class `IloCplex` in C++ and Java or the class `Cplex` in the .NET API:

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

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`.

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

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

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

- In the C++ API, use the member functions of an instance of the class `IloCplex::ParameterSet`.
- In the Java API, use the methods of an object of the class `IloCplex.ParameterSet`.
- In the .NET API, use the methods of the class `Cplex.ParameterSet`.
- In the Python API, use the methods of the class `Cplex.ParameterSet`.

Documentation about CPLEX® parameters specific to the Python API is available as online help inside a Python session. A brief introduction to CPLEX® parameters is available in the

topic Using CPLEX parameters in the CPLEX Python API in the tutorial about Python in *Getting Stated with CPLEX*.

Likewise, documentation about CPLEX® parameters specific to the CPLEX® connector for The MathWorks MATLAB is available as online help inside a MATLAB session.

Parameter names

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

- For the C API, these constants are capitalized and start with `CPX_PARAM_`; for example, `CPX_PARAM_ITLIM`. They are used as the second argument in all parameter routines (except `CPXsetdefaults`, which does not require them).
- For C++ applications, the parameters are defined in nested enumeration types for Boolean, integer, floating-point, and string parameters. The `enum` names use mixed (lower and upper) case letters and must be prefixed with the class name `IloCplex::` for scope. For example, `IloCplex::ItLim` is the `IloCplex` equivalent of `CPX_PARAM_ITLIM`.
- For Java applications, the parameters are defined as final static objects in nested classes 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, the parameters follow the usual conventions for capitalizing attributes and defining scope within a namespace.
- For Python applications, the names of parameters resemble the names in the CPLEX® Interactive Optimizer, modified for the syntax of a Python application. For example, the command in the Interactive Optimizer `set mip cuts mcfcut` looks like this in a Python application:`cplex.parameters.mip.cuts.set(mcfcut)`.

An integer that serves as a reference number for each parameter is shown in the reference page of each parameter. That integer reference number corresponds to the value that each symbolic constant represents, as found in the `cplex.h` header file of the Callable Library (C API), 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 CPLEX® .

Correspondence of parameters

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

- *epsilon used in linearization* `(EpLin)`, the parameter specifying the tolerance to use in linearization in the object oriented APIs (C++, Java, .NET), is not applicable in the C API, nor in the Python API.
- *MIP callback switch between original model and reduced, presolved model* `(CPX_PARAM_MIPCBREDLP)`, the parameter indicating whether to use the reduced or original model in MIP callbacks, has no equivalent in the object oriented APIs (C++, Java, .NET) nor in the Python API, nor in the MATLAB connector.
- Logging output is controlled by a parameter in the C API (`CPX_PARAM_SCRIND`), but when using the object oriented APIs, you control logging by configuring the output channel:
  - `IloCplex::out` in C++

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

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

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

    For example, to turn off output to the screen, use `cplex.set_results_stream(None)`.
- The parameter `IloCplex::RootAlg` in the C++ API corresponds to these parameters in the C API:
  - *MIP starting algorithm*: `CPX_PARAM_STARTALG`
  - *algorithm for continuous problems*: `CPX_PARAM_LPMETHOD`
  - *algorithm for continuous quadratic optimization*: `CPX_PARAM_QPMETHOD`
- The parameter `IloCplex::NodeAlg` in the C++ API corresponds to the parameter *MIP subproblem algorithm* `CPX_PARAM_SUBALG` in the C API.

Saving parameter settings to a file

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

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

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

When you write a parameter file from the C API, 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, CPLEX® ignores the rest of the line.

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

Here is an example of a correct CPLEX® parameter file:

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

Topical list of parameters

The following lists offer you access to the documentation of CPLEX® parameters, organized by topics.

In this section

Simplex
Lists parameters of interest to users of the simplex optimizers.

Barrier
Lists parameters of interest to users of the barrier optimizer.

MIP
Lists topics of interest to users of the MIP optimizer.

MIP general
Lists parameters of general interest to users of the MIP optimizer.

MIP strategies
Lists parameters controlling MIP strategies.

MIP cuts
Lists parameters controlling cuts.

MIP tolerances
Lists parameters setting MIP tolerances.

MIP limits
Lists parameters setting MIP limits.

Solution polishing
Lists parameters controlling starting conditions for solution polishing

Solution pool
Lists parameters controlling the solution pool.

Network
Lists parameters of interest to users of the network flow optimizer.

Parallel optimization
Lists parameters controlling parallel optimization.

Sifting
Lists parameters of interest to users of the sifting optimizer.

Preprocessing: aggregator, presolver
Lists parameters related to preprocessing.

Tolerances
Lists parameters setting tolerances.

Limits
Lists parameters setting general limits.

Display and output
Lists parameters controlling screen displays, logs, and files.

Simplex

Selecting the *algorithm for continuous problems*
advanced start switch
lower objective value limit
upper objective value limit
dual simplex pricing algorithm
primal simplex pricing algorithm
simplex crash ordering
Markowitz tolerance
optimality tolerance
perturbation constant
simplex perturbation switch
simplex perturbation limit
relaxation for FeasOpt
feasibility tolerance
simplex maximum iteration limit
memory reduction switch
numerical precision emphasis
simplex pricing candidate list size
sifting subproblem algorithm
simplex iteration information display
simplex singularity repair limit

Barrier

advanced start switch

barrier algorithm

barrier starting point algorithm

barrier crossover algorithm

sifting subproblem algorithm

barrier ordering algorithm

barrier display information

barrier growth limit

barrier column nonzeros

barrier iteration limit

barrier maximum correction limit

barrier objective range

convergence tolerance for LP and QP problems

convergence tolerance for QC problems

memory reduction switch

numerical precision emphasis

MIP

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

- *MIP general*
- *MIP strategies*
- *MIP cuts*
- *MIP tolerances*
- *MIP limits*

MIP general

advanced start switch

MIP emphasis switch

MIP repeat presolve switch

relaxed LP presolve switch

indefinite MIQP switch

bound strengthening switch

memory reduction switch

numerical precision emphasis

MIP callback switch between original model and reduced, presolved model

MIP node log display information

MIP node log interval

node storage file switch

MIP strategies

MIP starting algorithm

MIP subproblem algorithm

MIP variable selection strategy

MIP strategy best bound interval

MIP branching direction

backtracking tolerance

MIP dive strategy

MIP heuristic frequency

local branching heuristic

MIP priority order switch

MIP priority order generation

MIP node selection strategy

node presolve switch

MIP probing level

RINS heuristic frequency

feasibility pump switch

MIP cuts

constraint aggregation limit for cut generation

row multiplier factor for cuts

MIP cliques switch

MIP covers switch

MIP disjunctive cuts switch

MIP flow cover cuts switch

MIP flow path cut switch

MIP Gomory fractional cuts switch

MIP GUB cuts switch

MIP implied bound cuts switch

MCF cut switch

MIP MIR (mixed integer rounding) cut switch

MIP zero-half cuts switch

pass limit for generating Gomory fractional cuts

candidate limit for generating Gomory fractional cuts

type of cut limit

number of cutting plane passes

MIP tolerances

backtracking tolerance

lower cutoff

upper cutoff

absolute objective difference cutoff

relative objective difference cutoff

absolute MIP gap tolerance

relative MIP gap tolerance

integrality tolerance

relaxation for FeasOpt

MIP limits

MIP integer solution limit

pass limit for generating Gomory fractional cuts

candidate limit for generating Gomory fractional cuts

constraint aggregation limit for cut generation

type of cut limit

row multiplier factor for cuts

number of cutting plane passes

MIP node limit

time spent probing

frequency to try to repair infeasible MIP start

MIP strong branching candidate list limit

MIP strong branching iterations limit

limit on nodes explored when a subMIP is being solved

tree memory limit

Solution polishing

absolute MIP gap before starting to polish a feasible solution

relative MIP gap before starting to polish a feasible solution

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

nodes to process before starting to polish a feasible solution

time before starting to polish a feasible solution

Solution pool

solution pool intensity

solution pool replacement strategy

maximum number of solutions generated for solution pool by populate

maximum number of solutions kept in solution pool

absolute gap for solution pool

relative gap for solution pool

Network

network optimality tolerance

network primal feasibility tolerance

simplex network extraction level

network simplex iteration limit

network simplex pricing algorithm

network logging display switch

Parallel optimization

parallel mode switch

global default thread count

Sifting

sifting subproblem algorithm

sifting information display

upper limit on sifting iterations

Preprocessing: aggregator, presolver

symmetry breaking

preprocessing aggregator fill

preprocessing aggregator application limit

bound strengthening switch

coefficient reduction setting

dependency switch

presolve dual setting

presolve switch

linear reduction switch

limit on the number of presolve passes made

node presolve switch

relaxed LP presolve switch

MIP repeat presolve switch

primal and dual reduction type

Tolerances

convergence tolerance for LP and QP problems

convergence tolerance for QC problems

backtracking tolerance

lower cutoff

upper cutoff

absolute MIP gap tolerance

absolute MIP gap before starting to polish a feasible solution

relative MIP gap tolerance

relative MIP gap before starting to polish a feasible solution

integrality tolerance

epsilon used in linearization

Markowitz tolerance

optimality tolerance

network optimality tolerance

feasibility tolerance

relaxation for FeasOpt

absolute objective difference cutoff

relative objective difference cutoff

perturbation constant

absolute gap for solution pool

relative gap for solution pool

Limits

memory available for working storage

global default thread count

optimizer time limit

variable (column) read limit

constraint (row) read limit

nonzero element read limit

QP Q-matrix nonzero read limit

Display and output

messages to screen switch

tuning information display

barrier display information

simplex iteration information display

sifting information display

MIP node log display information

MIP node log interval

network logging display switch

clock type for computation time

conflict information display

data consistency checking switch

precision of numerical output in MPS and REW file formats

directory for working files

write level for MST, SOL files

List of CPLEX parameters

Presents the entire list of parameters

In this section

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

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

preprocessing aggregator fill
Limits variable substitutions by the aggregator.

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.

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.

barrier column nonzeros
Used in the recognition of dense columns.

barrier crossover algorithm
Decides which, if any, crossover is performed at the end of a barrier optimization.

barrier display information
Sets the level of barrier progress information to be displayed.

convergence tolerance for LP and QP problems
Sets the tolerance on complementarity for convergence.

barrier growth limit
Used to detect unbounded optimal faces.

barrier iteration limit
Sets the number of barrier iterations before termination.

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

barrier objective range
Sets the maximum absolute value of the objective function.

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.

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

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

MIP strategy best bound interval
Sets the best bound interval for MIP strategy.

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

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

backtracking tolerance
Controls how often backtracking is done during the branching process.

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

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

coefficient reduction setting
Decides how coefficient reduction is used.

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

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

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

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

lower cutoff
Sets lower cutoff tolerance.

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.

row multiplier factor for cuts
Limits the number of cuts that can be added.

upper cutoff
Sets the upper cutoff tolerance.

data consistency checking switch
Decides whether data should be checked for consistency.

dependency switch
Decides whether to activate the dependency checker.

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

MIP dive strategy
Controls the MIP dive strategy.

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

type of cut limit
Sets a limit for each type of cut.

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

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

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

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

Markowitz tolerance
Influences pivot selection during basis factoring.

optimality tolerance
Influences the reduced-cost tolerance for optimality.

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.

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

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

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

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

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

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

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

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

pass limit for generating Gomory fractional cuts
Limits the number of passes for generating Gomory fractional cuts.

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

MIP heuristic frequency
Decides how often to apply the periodic heuristic.

MIP implied bound cuts switch
Decides whether or not to generate implied bound cuts for the problem.

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

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

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

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

memory reduction switch
Directs CPLEX® that it should conserve memory where possible.

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

MIP node log display information
Decides what CPLEX® reports to the screen during mixed integer optimization (MIP).

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

MIP node log interval
Controls the frequency of node logging when the MIP display parameter (`CPX_PARAM_MIPDISPLAY`, `MIPDisplay`) is set higher than 1 (one).

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

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

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

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

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

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

network logging display switch
Decides what CPLEX® reports to the screen during network optimization.

network optimality tolerance
Specifies the optimality tolerance for network optimization.

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

simplex network extraction level
Establishes the level of network extraction for network simplex optimization.

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

network simplex pricing algorithm
Specifies the pricing algorithm for network simplex optimization.

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

node storage file switch
Used when working memory (`CPX_PARAM_WORKMEM`, `WorkMem`) has been exceeded by the size of the tree.

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

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

numerical precision emphasis
Emphasizes precision in numerically unstable or difficult problems.

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

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

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

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

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

simplex perturbation switch
Decides whether to perturb problems.

simplex perturbation limit
Sets the number of degenerate iterations before perturbation is performed.

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

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

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

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

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

time spent polishing a solution (deprecated)
Deprecated parameter

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

primal simplex pricing algorithm
Sets the primal simplex pricing algorithm.

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

presolve switch
Decides whether CPLEX® applies presolve during preprocessing.

linear reduction switch
Decides whether linear or full reductions occur during preprocessing.

limit on the number of presolve passes made
Limits the number of presolve passes that CPLEX® makes during preprocessing. When this parameter is set to a nonzero value, invokes CPLEX® presolve to simplify and reduce problems.

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

simplex pricing candidate list size
Sets the maximum number of variables kept in the list of pricing candidates for the simplex algorithms.

MIP probing level
Sets the amount of probing on variables to be performed before MIP branching.

time spent probing
Limits the amount of time in seconds spent probing.

indefinite MIQP switch
Decides whether CPLEX® will attempt to reformulate a MIQP or MIQCP model that contains only binary variables.

QP Q-matrix nonzero read limit
Specifies a limit for the number of nonzero elements to read for an allocation of memory in a model with a quadratic matrix.

primal and dual reduction type
Decides whether primal reductions, dual reductions, both, or neither are performed during preprocessing.

simplex refactoring frequency
Sets the number of iterations between refactoring of the basis matrix.

relaxed LP presolve switch
Decides whether LP presolve is applied to the root relaxation in a mixed integer program (MIP).

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

frequency to try to repair infeasible MIP start
Limits the attempts to repair an infeasible MIP start.

MIP repeat presolve switch
Decides whether to re-apply presolve, with or without cuts, to a MIP model after processing at the root is otherwise complete.

RINS heuristic frequency
Decides how often to apply the relaxation induced neighborhood search (RINS) heuristic.

algorithm for continuous problems
Controls which algorithm is used to solve continuous models or to solve the root relaxation of a MIP.

algorithm for continuous quadratic optimization
Sets which algorithm to use when the C routine `CPXqpopt` (or the command `optimize` in the Interactive Optimizer) is invoked.

MIP starting algorithm
Sets which continuous optimizer will be used to solve the initial relaxation of a MIP.

constraint (row) read limit
Specifies a limit for the number of rows (constraints) to read for an allocation of memory.

scale parameter
Decides how to scale the problem matrix.

messages to screen switch
Decides whether or not results are displayed on screen in an application of the C API.

sifting subproblem algorithm
Sets the algorithm to be used for solving sifting subproblems.

sifting information display
Sets the amount of information to display about the progress of sifting.

upper limit on sifting iterations
Sets the maximum number of sifting iterations that may be performed if convergence to optimality has not been reached.

simplex iteration information display
Sets how often CPLEX® reports about iterations during simplex optimization.

simplex singularity repair limit
Restricts the number of times CPLEX® attempts to repair the basis when singularities are encountered during the simplex algorithm.

absolute gap for solution pool
Sets an absolute tolerance on the objective value for the solutions in the solution pool.

maximum number of solutions kept in solution pool
Limits the number of solutions kept in the solution pool

relative gap for solution pool
Sets a relative tolerance on the objective value for the solutions in the solution pool.

solution pool intensity
Controls the trade-off between the number of solutions generated for the solution pool and the amount of time or memory consumed.

solution pool replacement strategy
Designates the strategy for replacing a solution in the solution pool when the solution pool has reached its capacity.

MIP strong branching candidate list limit
Controls the length of the candidate list when CPLEX® uses variable selection as the setting for strong branching.

MIP strong branching iterations limit
Controls the number of simplex iterations performed on each variable in the candidate list when CPLEX® uses variable selection as the setting for strong branching.

limit on nodes explored when a subMIP is being solved
Restricts the number of nodes explored when CPLEX® is solving a subMIP.

symmetry breaking
Decides whether symmetry breaking reductions will be automatically executed, during the preprocessing phase, in a MIP model.

global default thread count
Sets the default number of parallel threads that will be invoked by any CPLEX® parallel optimizer.

optimizer time limit
Sets the maximum time, in seconds, for a call to an optimizer. This time limit applies also to the conflict refiner.

tree memory limit
Sets an absolute upper limit on the size (in megabytes, uncompressed) of the branch-and-cut tree.

tuning information display
Specifies the level of information reported by the tuning tool as it works.

tuning measure
Controls the measure for evaluating progress when a suite of models is being tuned.

tuning repeater
Specifies the number of times tuning is to be repeated on reordered versions of a given problem.

tuning time limit
Sets a time limit per model and per test set (that is, suite of models) applicable in tuning.

MIP variable selection strategy
Sets the rule for selecting the branching variable at the node which has been selected for branching.

directory for working files
Specifies the name of an existing directory into which CPLEX® may store temporary working files.

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.

write level for MST, SOL files
Sets a level of detail for CPLEX® to write a file in MST or SOL format.

MIP zero-half cuts switch
Decides whether or not to generate zero-half cuts for the problem.

advanced start switch

Purpose
Advanced start switch

Syntax

C Name	CPX_PARAM_ADVIND (int)
C++ Name	AdvInd (int)
Java Name	AdvInd (int)
.NET Name	AdvInd (int)
OPL Name	advind
InteractiveOptimizer	advance
Identifier	1001

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

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

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

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

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

Values

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

constraint aggregation limit for cut generation

Purpose
Constraint aggregation limit for cut generation

Syntax

C Name	CPX_PARAM_AGGCUTLIM (int)
C++ Name	AggCutLim (int)
Java Name	AggCutLim (int)
.NET Name	AggCutLim (int)
OPL Name	aggcutlim
Interactive Optimizer	mip limits aggforcut
Identifier	2054

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

Values
Any nonnegative integer; **default**: 3

preprocessing aggregator fill

Purpose
Preprocessing aggregator fill

Syntax

C Name	CPX_PARAM_AGGFILL (int)
C++ Name	AggFill (int)
Java Name	AggFill (int)
.NET Name	AggFill (int)
OPL Name	aggfill
Interactive Optimizer	preprocessing fill
Identifier	1002

Description
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.

Values

Any nonnegative integer; **default**: 10

preprocessing aggregator application limit

Purpose
Preprocessing aggregator application limit

Syntax

```
C Name                  CPX_PARAM_AGGIND (int)

C++ Name                AggInd (int)

Java Name               AggInd (int)

.NET Name               AggInd (int)

OPL Name                aggind

Interactive Optimizer   preprocessing aggregator

Identifier              1003
```

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

Values

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

barrier algorithm

Purpose
Barrier algorithm

Syntax

C Name	CPX_PARAM_BARALG (int)
C++ Name	BarAlg (int)
Java Name	BarAlg (int)
.NET Name	BarAlg (int)
OPL Name	baralg
Interactive Optimizer	barrier algorithm
Identifier	3007

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

Values

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

barrier column nonzeros

Purpose
Barrier column nonzeros

Syntax

C Name	CPX_PARAM_BARCOLNZ (int)
C++ Name	BarColNz (int)
Java Name	BarColNz (int)
.NET Name	BarColNz (int)
OPL Name	barcolnz
Interactive Optimizer	barrier colnonzeros
Identifier	3009

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

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

barrier crossover algorithm

Purpose
Barrier crossover algorithm

Syntax

C Name	CPX_PARAM_BARCROSSALG (int)
C++ Name	BarCrossAlg (int)
Java Name	BarCrossAlg (int)
.NET Name	BarCrossAlg (int)
OPL Name	barcrossalg
Interactive Optimizer	barrier crossover
Identifier	3018

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

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

barrier display information

Purpose
Barrier display information

Syntax

C Name	CPX_PARAM_BARDISPLAY (int)
C++ Name	BarDisplay (int)
Java Name	BarDisplay (int)
.NET Name	BarDisplay (int)
OPL Name	bardisplay
Interactive Optimizer	barrier display
Identifier	3010

Description
Sets the level of barrier progress information to be displayed.

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

convergence tolerance for LP and QP problems

Purpose
Convergence tolerance for LP and QP problems

Syntax

C Name	CPX_PARAM_BAREPCOMP (double)
C++ Name	BarEpComp (double)
Java Name	BarEpComp (double)
.NET Name	BarEpComp (double)
OPL Name	barepcomp
Interactive Optimizer	barrier convergetol
Identifier	3002

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

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

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

See also
For problems with quadratic constraints (QCP), see *convergence tolerance for QC problems*

barrier growth limit

Purpose
Barrier growth limit

Syntax

C Name	CPX_PARAM_BARGROWTH (double)
C++ Name	BarGrowth (double)
Java Name	BarGrowth (double)
.NET Name	BarGrowth (double)
OPL Name	bargrowth
Interactive Optimizer	barrier limits growth
Identifier	3003

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

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

barrier iteration limit

Purpose
Barrier iteration limit

Syntax

C Name	CPX_PARAM_BARITLIM (int)
C++ Name	BarItLim (int)
Java Name	BarItLim (int)
.NET Name	BarItLim (int)
OPL Name	baritlim
Interactive Optimizer	barrier limits iterations
Identifier	3012

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

Values

Value	Meaning
0	No barrier iterations
2100000000	**default**
Any positive integer	Number of barrier iterations before termination

barrier maximum correction limit

Purpose
Barrier maximum correction limit

Syntax

C Name	CPX_PARAM_BARMAXCOR (int)
C++ Name	BarMaxCor (int)
Java Name	BarMaxCor (int)
.NET Name	BarMaxCor (int)
OPL Name	barmaxcor
Interactive Optimizer	barrier limits corrections
Identifier	3013

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

Values

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

barrier objective range

Purpose
Barrier objective range

Syntax

C Name	CPX_PARAM_BAROBJRNG (double)
C++ Name	BarObjRng (double)
Java Name	BarObjRng (double)
.NET Name	BarObjRng (double)
OPL Name	barobjrng
Interactive Optimizer	barrier limits objrange
Identifier	3004

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

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

barrier ordering algorithm

Purpose
Barrier ordering algorithm

Syntax

C Name	CPX_PARAM_BARORDER (int)
C++ Name	BarOrder (int)
Java Name	BarOrder (int)
.NET Name	BarOrder (int)
OPL Name	barorder
Interactive Optimizer	barrier ordering
Identifier	3014

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

Values

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

convergence tolerance for QC problems

Purpose
Convergence tolerance for quadratically constrained problems

Syntax

C Name	CPX_PARAM_BARQCPEPCOMP (double)
C++ Name	BarQCPEpComp (double)
Java Name	BarQCPEpComp (double)
.NET Name	BarQCPEpComp (double)
OPL Name	barqcpepcomp
Interactive Optimizer	barrier qcpconvergetol
Identifier	3020

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

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

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

For LPs and for QPs (that is, when all the constraints are linear) see *convergence tolerance for LP and QP problems* CPX_PARAM_BAREPCOMP, BarEpComp.

barrier starting point algorithm

Purpose
Barrier starting point algorithm

Syntax

C Name	CPX_PARAM_BARSTARTALG (int)
C++ Name	BarStartAlg (int)
Java Name	BarStartAlg (int)
.NET Name	BarStartAlg (int)
OPL Name	barstartalg
Interactive Optimizer	barrier startalg
Identifier	3017

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

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

MIP strategy best bound interval

Purpose
MIP strategy best bound interval

Syntax

C Name	CPX_PARAM_BBINTERVAL (int)
C++ Name	BBInterval (int)
Java Name	BBInterval (int)
.NET Name	BBInterval (int)
OPL Name	bbinterval
Interactive Optimizer	mip strategy bbinterval
Identifier	2039

Description
Sets the best bound interval for MIP strategy.

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

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

Values

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

See also
MIP node selection strategy

bound strengthening switch

Purpose
Bound strengthening switch

Syntax

C Name	CPX_PARAM_BNDSTRENIND (int)
C++ Name	BndStrenInd (int)
Java Name	BndStrenInd (int)
.NET Name	BndStrenInd (int)
OPL Name	bndstrenind
Interactive Optimizer	preprocessing boundstrength
Identifier	2029

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

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

MIP branching direction

Purpose
MIP branching direction

Syntax

C Name	CPX_PARAM_BRDIR (int)
C++ Name	BrDir (int)
Java Name	BrDir (int)
.NET Name	BrDir (int)
OPL Name	brdir
Interactive Optimizer	mip strategy branch
Identifier	2001

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

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

backtracking tolerance

Purpose
Backtracking tolerance

Syntax

C Name	CPX_PARAM_BTTOL (double)
C++ Name	BtTol (double)
Java Name	BtTol (double)
.NET Name	BtTol (double)
OPL Name	bttol
Interactive Optimizer	mip strategy backtrack
Identifier	2002

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

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

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

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

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

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

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

See also
upper cutoff, lower cutoff

MIP cliques switch

Purpose
MIP cliques switch

Syntax

C Name	CPX_PARAM_CLIQUES (int)
C++ Name	Cliques (int)
Java Name	Cliques (int)
.NET Name	Cliques (int)
OPL Name	cliques
Interactive Optimizer	mip cuts cliques
Identifier	2003

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

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

clock type for computation time

Purpose
Clock type for computation time

Syntax

C Name	CPX_PARAM_CLOCKTYPE (int)
C++ Name	ClockType (int)
Java Name	ClockType (int)
.NET Name	ClockType (int)
OPL Name	clocktype
Interactive Optimizer	clocktype
Identifier	1006

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

The default setting 0 (zero) allows CPLEX® to choose wall clock time when other parameters invoke parallel optimization and to choose CPU time when other parameters enforce sequential (not parallel) optimization.

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

coefficient reduction setting

Purpose
Coefficient reduction setting

Syntax

C Name	CPX_PARAM_COEREDIND (int)
C++ Name	CoeRedInd (int)
Java Name	CoeRedInd (int)
.NET Name	CoeRedInd (int)
OPL Name	coeredind
Interactive Optimizer	preprocessing coeffreduce
Identifier	2004

Description
Decides how coefficient reduction is used. Coefficient reduction improves the objective value of the initial (and subsequent) LP relaxations solved during branch and cut by reducing the number of non-integral vertices.

Value	Meaning
0	Do not use coefficient reduction
1	Reduce only to integral coefficients
2	Reduce all potential coefficients; **default**

variable (column) read limit

Purpose
Variable (column) read limit

Syntax

C Name	CPX_PARAM_COLREADLIM (int)
C++ Name	ColReadLim (int)
Java Name	ColReadLim (int)
.NET Name	ColReadLim (int)
Interactive Optimizer	read variables
Identifier	1023

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

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

Values
Any integer from 0 to 268 435 450; **default**: 60 000.

conflict information display

Purpose
Conflict information display

Syntax

C Name	CPX_PARAM_CONFLICTDISPLAY (int)
C++ Name	ConflictDisplay (int)
Java Name	ConflictDisplay (int)
.NET Name	ConflictDisplay (int)
OPL Name	conflictdisplay
Interactive Optimizer	conflict display i
Identifier	1074

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

Values

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

MIP covers switch

Purpose
MIP covers switch

Syntax

C Name	CPX_PARAM_COVERS (int)
C++ Name	Covers (int)
Java Name	Covers (int)
.NET Name	Covers (int)
OPL Name	covers
Interactive Optimizer	mip cuts covers
Identifier	2005

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

Values

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

simplex crash ordering

Purpose
Simplex crash ordering

Syntax

C Name	CPX_PARAM_CRAIND (int)
C++ Name	CraInd (int)
Java Name	CraInd (int)
.NET Name	CraInd (int)
OPL Name	craind
Interactive Optimizer	simplex crash
Identifier	1007

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

Values

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

lower cutoff

Purpose
Lower cutoff

Syntax

C Name	CPX_PARAM_CUTLO (double)
C++ Name	CutLo (double)
Java Name	CutLo (double)
.NET Name	CutLo (double)
OPL Name	cutlo
Interactive Optimizer	mip tolerances lowercutoff
Identifier	2006

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

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

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

number of cutting plane passes

Purpose
Number of cutting plane passes

Syntax

C Name	CPX_PARAM_CUTPASS (int)
C++ Name	CutPass (int)
Java Name	CutPass (int)
.NET Name	CutPass (int)
OPL Name	cutpass
Interactive Optimizer	mip limits cutpasses
Identifier	2056

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

Values

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

row multiplier factor for cuts

Purpose
Row multiplier factor for cuts

Syntax

C Name	CPX_PARAM_CUTSFACTOR (double)
C++ Name	CutsFactor (double)
Java Name	CutsFactor (double)
.NET Name	CutsFactor (double)
OPL Name	cutsfactor
Interactive Optimizer	mip limits cutsfactor
Identifier	2033

Description
Limits the number of cuts that can be added. The number of rows in the problem with cuts added is limited to `CutsFactor` times the original number of rows. If the problem is presolved, the original number of rows is that from the presolved problem.

A `CutsFactor` of 1.0 or less means that no cuts will be generated.

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

Values
Any nonnegative number; **default**: 4.0

upper cutoff

Purpose
Upper cutoff

Syntax

C Name	CPX_PARAM_CUTUP (double)
C++ Name	CutUp (double)
Java Name	CutUp (double)
.NET Name	CutUp (double)
OPL Name	cutup
Interactive Optimizer	mip tolerances uppercutoff
Identifier	2007

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

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

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

data consistency checking switch

Purpose
Data consistency checking switch

Syntax

C Name	CPX_PARAM_DATACHECK (int)
C++ Name	DataCheck (bool)
Java Name	DataCheck (bool)
.NET Name	DataCheck (bool)
OPL Name	datacheck
Interactive Optimizer	read datacheck
Identifier	1056

Description
Decides whether data should be checked for consistency. When this parameter is on, the routines `CPXcopy____`, `CPXread____` and `CPXchg____` of the C API perform extensive checking of data in their array arguments, such as checking that indices are within range, that there are no duplicate entries, and that values are valid for the type of data or are valid numbers. This checking is useful for debugging applications. When this checking identifies trouble, you can gather more specific detail by calling one of the routines in `check.c` .

Values

int	bool	Symbol	Meaning
0	false	CPX_OFF	Data checking off; do not check; **default**
1	true	CPX_ON	Data checking on

dependency switch

Purpose
Dependency switch

Syntax

C Name	CPX_PARAM_DEPIND (int)
C++ Name	DepInd (int)
Java Name	DepInd (int)
.NET Name	DepInd (int)
OPL Name	depind
Interactive Optimizer	preprocessing dependency
Identifier	1008

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

Values

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

MIP disjunctive cuts switch

Purpose
MIP disjunctive cuts switch

Syntax

C Name	CPX_PARAM_DISJCUTS (int)
C++ Name	DisjCuts (int)
Java Name	DisjCuts (int)
.NET Name	DisjCuts (int)
OPL Name	disjcuts
Interactive Optimizer	mip cuts disjunctive
Identifier	2053

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

Values

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

MIP dive strategy

Purpose
MIP dive strategy

Syntax

C Name	CPX_PARAM_DIVETYPE (int)
C++ Name	DiveType (int)
Java Name	DiveType (int)
.NET Name	DiveType (int)
OPL Name	divetype
Interactive Optimizer	mip strategy dive
Identifier	2060

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

Values

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

dual simplex pricing algorithm

Purpose
Dual simplex pricing algorithm

Syntax

C Name	CPX_PARAM_DPRIIND (int)
C++ Name	DPriInd (int)
Java Name	DPriInd (int)
.NET Name	DPriInd (int)
OPL Name	dpriind
Interactive Optimizer	simplex dgradient
Identifier	1009

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

Values

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

See also
candidate limit for generating Gomory fractional cuts, *MIP Gomory fractional cuts switch*, *pass limit for generating Gomory fractional cuts*

type of cut limit

Purpose
Type of cut limit

Syntax

C Name	CPX_PARAM_EACHCUTLIM (int)
C++ Name	EachCutLim (int)
Java Name	EachCutLim (int)
.NET Name	EachCutLim (int)
OPL Name	eachcutlim
Interactive Optimizer	mip limit eachcutlimit
Identifier	2102

Description
Sets a limit for each type of cut.

This parameter allows you to set a uniform limit on the number of cuts of each type that CPLEX® generates. By default, the limit is the largest integer supported by a given platform; that is, there is no effective limit by default.

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

A setting of 0 (zero) means no cuts.

This parameter does **not** influence the number of Gomory cuts. For means to control the number of Gomory cuts, see also the fractional cut parameters:

- *candidate limit for generating Gomory fractional cuts*: `CPX_PARAM_FRACCAND`, `FracCand`;
- *MIP Gomory fractional cuts switch*: `CPX_PARAM_FRACCUTS`, `FracCuts`;
- *pass limit for generating Gomory fractional cuts*: `CPX_PARAM_FRACPASS`, `FracPass`.

Values

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

absolute MIP gap tolerance

Purpose
Absolute MIP gap tolerance

Syntax

C Name	CPX_PARAM_EPAGAP (double)
C++ Name	EpAGap (double)
Java Name	EpAGap (double)
.NET Name	EpAGap (double)
OPL Name	epagap
InteractiveOptimizer	mip tolerances absmipgap
Identifier	2008

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

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

relative MIP gap tolerance

Purpose
Relative MIP gap tolerance

Syntax

C Name	CPX_PARAM_EPGAP (double)
C++ Name	EpGap (double)
Java Name	EpGap (double)
.NET Name	EpGap (double)
OPL Name	epgap
Interactive Optimizer	mip tolerances mipgap
Identifier	2009

Description
When the value

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

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

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

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

integrality tolerance

Purpose
Integrality tolerance

Syntax

C Name	CPX_PARAM_EPINT (double)
C++ Name	EpInt (double)
Java Name	EpInt (double)
.NET Name	EpInt (double)
OPL Name	epint
Interactive Optimizer	mip tolerances integrality
Identifier	2010

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

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

However, in some models, computer 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 .

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

epsilon used in linearization

Purpose
Epsilon used in linearization

Syntax

C Name	CPX_PARAM_EPLIN but not applicable in the C API
C++ Name	EpLin (double)
Java Name	EpLin (double)
.NET Name	EpLin (double)
Interactive Optimizer	not available in the Interactive Optimizer
Identifier	2068

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

Not applicable in the C API.

Not available in the Interactive Optimizer.

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

```
x < a
```

becomes

```
x <= a-eplin .
```

Similarly, `x!=a`

becomes

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

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

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

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

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

Markowitz tolerance

Purpose
Markowitz tolerance

Syntax

C Name	CPX_PARAM_EPMRK (double)
C++ Name	EpMrk (double)
Java Name	EpMrk (double)
.NET Name	EpMrk (double)
OPL Name	epmrk
Interactive Optimizer	simplex tolerances markowitz
Identifier	1013

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

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

optimality tolerance

Purpose
Optimality tolerance

Syntax

C Name	CPX_PARAM_EPOPT (double)
C++ Name	EpOpt (double)
Java Name	EpOpt (double)
.NET Name	EpOpt (double)
OPL Name	epopt
Interactive Optimizer	simplex tolerances optimality
Identifier	1014

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

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

perturbation constant

Purpose
Perturbation constant

Syntax

C Name	CPX_PARAM_EPPER (double)
C++ Name	EpPer (double)
Java Name	EpPer (double)
.NET Name	EpPer (double)
OPL Name	epper
Interactive Optimizer	simplex perturbation
Identifier	1015

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

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

relaxation for FeasOpt

Purpose
Relaxation for feasOpt

Syntax

C Name	CPX_PARAM_EPRELAX (double)
C++ Name	EpRelax (double)
Java Name	EpRelax (double)
.NET Name	EpRelax (double)
OPL Name	eprelax
Interactive Optimizer	feasopt tolerance
Identifier	2073

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

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

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

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

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

See also
lower objective value limit

feasibility tolerance

Purpose
Feasibility tolerance

Syntax

C Name	CPX_PARAM_EPRHS (double)
C++ Name	EpRHS (double)
Java Name	EpRHS (double)
.NET Name	EpRHS (double)
OPL Name	eprhs
Interactive Optimizer	simplex tolerances feasibility
Identifier	1016

Description
Specifies the feasibility tolerance, that is, the degree to which values of the basic variables calculated by the simplex method may violate their bounds. 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.

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

mode of FeasOpt

Purpose
Mode of FeasOpt

Syntax

C Name	CPX_PARAM_FEASOPTMODE (int)
C++ Name	FeasOptMode (int)
Java Name	FeasOptMode (int)
.NET Name	FeasOptMode (int)
OPL Name	feasoptmode
Interactive Optimizer	feasopt mode
Identifier	1084

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

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

Values

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

MIP flow cover cuts switch

Purpose
MIP flow cover cuts switch

Syntax

C Name	CPX_PARAM_FLOWCOVERS (int)
C++ Name	FlowCovers (int)
Java Name	FlowCovers (int)
.NET Name	FlowCovers (int)
OPL Name	flowcovers
Interactive Optimizer	mip cuts flowcovers
Identifier	2040

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

Values

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

MIP flow path cut switch

Purpose
MIP flow path cut switch

Syntax

C Name	CPX_PARAM_FLOWPATHS (int)
C++ Name	FlowPaths (int)
Java Name	FlowPaths (int)
.NET Name	FlowPaths (int)
OPL Name	flowpaths
Interactive Optimizer	mip cuts pathcut
Identifier	2051

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

Values

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

feasibility pump switch

Purpose
Feasibility pump switch

Syntax

C Name	CPX_PARAM_FPHEUR (int)
C++ Name	FPHeur (int)
Java Name	FPHeur (int)
.NET Name	FPHeur (int)
OPL Name	fpheur
Interactive Optimizer	mip strategy fpheur
Identifier	2098

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

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

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

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

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

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

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

Values

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

candidate limit for generating Gomory fractional cuts

Purpose
Candidate limit for generating Gomory fractional cuts

Syntax

C Name	CPX_PARAM_FRACCAND (int)
C++ Name	FracCand (int)
Java Name	FracCand (int)
.NET Name	FracCand (int)
OPL Name	fraccand
Interactive Optimizer	mip limits gomorycand
Identifier	2048

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

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

MIP Gomory fractional cuts switch

Purpose
MIP Gomory fractional cuts switch

Syntax

C Name	CPX_PARAM_FRACCUTS (int)
C++ Name	FracCuts (int)
Java Name	FracCuts (int)
.NET Name	FracCuts (int)
OPL Name	fraccuts
Interactive Optimizer	mip cuts gomory
Identifier	2049

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

Values

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

pass limit for generating Gomory fractional cuts

Purpose
Pass limit for generating Gomory fractional cuts

Syntax

C Name	CPX_PARAM_FRACPASS (int)
C++ Name	FracPass (int)
Java Name	FracPass (int)
.NET Name	FracPass (int)
OPL Name	fracpass
Interactive Optimizer	mip limits gomorypass
Identifier	2050

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

Values

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

MIP GUB cuts switch

Purpose
MIP GUB cuts switch

Syntax

C Name	CPX_PARAM_GUBCOVERS (int)
C++ Name	GUBCovers (int)
Java Name	GUBCovers (int)
.NET Name	GUBCovers (int)
OPL Name	gubcovers
Interactive Optimizer	mip cuts gubcovers
Identifier	2044

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

Values

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

MIP heuristic frequency

Purpose
MIP heuristic frequency

Syntax

C Name	CPX_PARAM_HEURFREQ (int)
C++ Name	HeurFreq (int)
Java Name	HeurFreq (int)
.NET Name	HeurFreq (int)
OPL Name	heurfreq
Interactive Optimizer	mip strategy heuristicfreq
Identifier	2031

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

Values

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

MIP implied bound cuts switch

Purpose
MIP implied bound cuts switch

Syntax

C Name	CPX_PARAM_IMPLBD (int)
C++ Name	ImplBd (int)
Java Name	ImplBd (int)
.NET Name	ImplBd (int)
OPL Name	implbd
Interactive Optimizer	mip cuts implied
Identifier	2041

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

Values

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

MIP integer solution limit

Purpose
MIP integer solution limit

Syntax

C Name	CPX_PARAM_INTSOLLIM (int)
C++ Name	IntSolLim (int)
Java Name	IntSolLim (int)
.NET Name	IntSolLim (int)
OPL Name	intsollim
Interactive Optimizer	mip limits solutions
Identifier	2015

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

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

Values
Any positive integer strictly greater than zero; zero is **not** allowed; **default**: 2100000000.

See also
maximum number of solutions generated for solution pool by populate

simplex maximum iteration limit

Purpose
Simplex maximum iteration limit

Syntax

C Name	CPX_PARAM_ITLIM (int)
C++ Name	ItLim (int)
Java Name	ItLim (int)
.NET Name	ItLim (int)
OPL Name	itlim
Interactive Optimizer	simplex limits iterations
Identifier	1020

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

Values
Any nonnegative integer; **default**: 2100000000.

local branching heuristic

Purpose
Local branching heuristic

Syntax

C Name	CPX_PARAM_LBHEUR (int)
C++ Name	LBHeur (bool)
Java Name	LBHeur (bool)
.NET Name	LBHeur (bool)
OPL Name	lbheur
Interactive Optimizer	mip strategy lbheur
Identifier	2063

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

Values

Value	bool	Symbol	Meaning
0	false	CPX_OFF	Local branching heuristic is off; **default**
1	true	CPX_ON	Apply local branching heuristic to new incumbent

MCF cut switch

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

Syntax

C Name	CPX_PARAM_MCFCUTS (int)
C++ Name	MCFCuts (int)
Java Name	MCFCuts (int)
.NET Name	MCFCuts (int)
OPL Name	
Interactive Optimizer	mip cuts mcfcut
Identifier	2134

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

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

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

Values

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

memory reduction switch

Purpose
Reduces use of memory

Syntax

C Name	CPX_PARAM_MEMORYEMPHASIS (int)
C++ Name	MemoryEmphasis (bool)
Java Name	MemoryEmphasis (bool)
.NET Name	MemoryEmphasis (bool)
OPL Name	memoryemphasis
Interactive Optimizer	emphasis memory
Identifier	1082

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

Values

Value	bool	Symbol	Meaning
0	false	CPX_OFF	Off; do not conserve memory; **default**
1	true	CPX_ON	On; conserve memory where possible

MIP callback switch between original model and reduced, presolved model

Purpose
MIP callback switch between original model and reduced, presolved model

Syntax

```
C Name                  CPX_PARAM_MIPCBREDLP (int)

C++ Name                MIP callback reduced LP parameter not available in
this API

Java Name               not available

.NET Name               not available

Interactive Optimizer   not available

Identifier              2055
```

Description
Controls whether your callback accesses node information of the original model (off) or node information of the reduced, presolved model (on, default). Advanced routines to control MIP callbacks (such as `CPXgetcallbacklp` , `CPXsetheuristiccallbackfunc` , `CPXsetbranchcallbackfunc` , `CPXgetbranchcallbackfunc` , `CPXsetcutcallbackfunc` , `CPXsetincumbentcallbackfunc` , `CPXgetcallbacksosinfo` , `CPXcutcallbackadd` , `CPXcutcallbackaddlocal` , and others) consider the setting of this parameter and access the original model or the reduced, presolved model accordingly.

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

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

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

Values

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

MIP node log display information

Purpose
MIP node log display information

Syntax

C Name	CPX_PARAM_MIPDISPLAY (int)
C++ Name	MIPDisplay (int)
Java Name	MIPDisplay (int)
.NET Name	MIPDisplay (int)
OPL Name	mipdisplay
Interactive Optimizer	mip display
Identifier	2012

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

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

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

  Each entry contains:

  - the value of the **objective function**;
  - the **node count**;
  - the **number of unexplored nodes** in the tree;
  - the current **optimality gap**.
- A setting of 2 also generates an entry for every **n-th node** (where n is the setting of the *MIP node log interval* parameter).
- A setting of 3 additionally generates an entry for every n-th node giving the number of **cuts** added to the problem for the previous `MIPInterval` number of nodes, plus an entry for each successfully processed **MIP start**.
- A setting of 4 additionally generates entries for the **LP root relaxation** according to the setting of the parameter to control the *simplex iteration information display* (`SimDisplay`, `CPX_PARAM_SIMDISPLAY`).

♦ A setting of 5 additionally generates entries for the **LP subproblems**, also according to the setting of the parameter to control the *simplex iteration information display* (`SimDisplay`, `CPX_PARAM_SIMDISPLAY`).

Values

Value	Meaning
0	No display until optimal solution has been found
1	Display integer feasible solutions
2	Display integer feasible solutions plus an entry for every n-th node; **default**
3	Display integer feasible solutions, every n-th node entry, number of cuts added, and information about the processing of each successful MIP start
4	Display integer feasible solutions, every n-th node entry, number of cuts added, information about the processing of each successful MIP start, and information about the LP subproblem at root
5	Display integer feasible solutions, every n-th node entry, number of cuts added, information about the processing of each successful MIP start, and information about the LP subproblem at root and at nodes

See also
MIP node log interval, *simplex iteration information display*, *network logging display switch*, and *messages to screen switch*

MIP emphasis switch

Purpose
MIP emphasis switch

Syntax

C Name	CPX_PARAM_MIPEMPHASIS (int)
C++ Name	MIPEmphasis (int)
Java Name	MIPEmphasis (int)
.NET Name	MIPEmphasis (int)
OPL Name	mipemphasis
Interactive Optimizer	emphasis mip
Identifier	2058

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

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

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

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

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

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

Values

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

MIP node log interval

Purpose
MIP node log interval

Syntax

C Name	CPX_PARAM_MIPINTERVAL (int)
C++ Name	MIPInterval (int)
Java Name	MIPInterval (int)
.NET Name	MIPInterval (int)
OPL Name	mipinterval
Interactive Optimizer	mip interval
Identifier	2013

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

Values
Any positive integer; **default**: 100.

See also
MIP node log display information

MIP priority order switch

Purpose
MIP priority order switch

Syntax

C Name	CPX_PARAM_MIPORDIND (int)
C++ Name	MIPOrdInd (bool)
Java Name	MIPOrdInd (bool)
.NET Name	MIPOrdInd (bool)
OPL Name	mipordind
Interactive Optimizer	mip strategy order
Identifier	2020

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

Values

Value	bool	Symbol	Meaning
	false	CPX_OFF	Off: do not use priority order
	true	CPX_ON	On: use priority order, if it exists; **default**

MIP priority order generation

Purpose
MIP priority order generation

Syntax

C Name	CPX_PARAM_MIPORDTYPE (int)
C++ Name	MIPOrdType (int)
Java Name	MIPOrdType (int)
.NET Name	MIPOrdType (int)
OPL Name	mipordtype
Interactive Optimizer	mip ordertype
Identifier	2032

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

Values

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

MIP dynamic search switch

Purpose
MIP dynamic search switch

Syntax

C Name	CPX_PARAM_MIPSEARCH (int)
C++ Name	MIPSearch (int)
Java Name	MIPSearch (int)
.NET Name	MIPSearch (int)
OPL Name	mipsearch
Interactive Optimizer	mip strategy search
Identifier	2109

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

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

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

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

Values

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

MIQCP strategy switch

Purpose
MIQCP strategy switch

Syntax

C Name	CPX_PARAM_MIQCPSTRAT (int)
C++ Name	MIQCPStrat (int)
Java Name	MIQCPStrat (int)
.NET Name	MIQCPStrat (int)
OPL Name	miqcpstrat
Interactive Optimizer	mip strategy miqcpstrat
Identifier	2110

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

This parameter controls how MIQCPs (that is, mixed integer programs with one or more constraints including quadratic terms) are solved. For more detail about the types of quadratically constrained models that CPLEX® solves, see Identifying a quadratically constrained program (QCP) in the *CPLEX User's Manual*.

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

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

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

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

Values

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

MIP MIR (mixed integer rounding) cut switch

Purpose
MIP MIR (mixed integer rounding) cut switch

Syntax

C Name	CPX_PARAM_MIRCUTS (int)
C++ Name	MIRCuts (int)
Java Name	MIRCuts (int)
.NET Name	MIRCuts (int)
OPL Name	mircuts
Interactive Optimizer	mip cuts mircut
Identifier	2052

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

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

precision of numerical output in MPS and REW file formats

Purpose
Precision of numerical output in MPS and REW file formats

Syntax

C Name	CPX_PARAM_MPSLONGNUM (int)
C++ Name	MPSLongNum (bool)
Java Name	MPSLongNum (bool)
.NET Name	MPSLongNum (bool)
Interactive Optimizer	output mpslong
Identifier	1081

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

Value	bool	Symbol	Meaning
0	false	CPX_OFF	Off: use limited MPS precision
1	true	CPX_ON	On: use full-precision; **default**

See also
MPS file format: industry standard

network logging display switch

Purpose
Network logging display switch

Syntax

C Name	CPX_PARAM_NETDISPLAY (int)
C++ Name	NetDisplay (int)
Java Name	NetDisplay (int)
.NET Name	NetDisplay (int)
OPL Name	netdisplay
Interactive Optimizer	network display
Identifier	5005

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

Value	Symbol	Meaning
0	CPXNET_NO_DISPLAY_OBJECTIVE	No display
1	CPXNET_TRUE_OBJECTIVE	Display true objective values
2	CPXNET_PENALIZE_OBJECTIVE	Display penalized objective values; **default**

network optimality tolerance

Purpose
Optimality tolerance for network optimization

Syntax

C Name	CPX_PARAM_NETEPOPT (double)
C++ Name	NetEpOpt (double)
Java Name	NetEpOpt (double)
.NET Name	NetEpOpt (double)
OPL Name	netepopt
Interactive Optimizer	network tolerances optimality
Identifier	5002

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

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

network primal feasibility tolerance

Purpose
Feasibility tolerance for network primal optimization

Syntax

C Name	CPX_PARAM_NETEPRHS (double)
C++ Name	NetEpRHS (double)
Java Name	NetEpRHS (double)
.NET Name	NetEpRHS (double)
OPL Name	neteprhs
Interactive Optimizer	network tolerances feasibility
Identifier	5003

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

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

simplex network extraction level

Purpose
Simplex network extraction level

Syntax

C Name	CPX_PARAM_NETFIND (int)
C++ Name	NetFind (int)
Java Name	NetFind (int)
.NET Name	NetFind (int)
OPL Name	netfind
Interactive Optimizer	network netfind
Identifier	1022

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

Values

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

network simplex iteration limit

Purpose
Network simplex iteration limit

Syntax

C Name	CPX_PARAM_NETITLIM (int)
C++ Name	NetItLim (int)
Java Name	NetItLim (int)
.NET Name	NetItLim (int)
OPL Name	netitlim
Interactive Optimizer	network iterations
Identifier	5001

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

Values
Any nonnegative integer; **default**: 2100000000.

network simplex pricing algorithm

Purpose
Network simplex pricing algorithm

Syntax

C Name	CPX_PARAM_NETPPRIIND (int)
C++ Name	NetPPriInd (int)
Java Name	NetPPriInd (int)
.NET Name	NetPPriInd (int)
OPL Name	netppriind
Interactive Optimizer	network pricing
Identifier	5004

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

Values

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

MIP subproblem algorithm

Purpose
MIP subproblem algorithm

Syntax

C Name	CPX_PARAM_SUBALG (int)
C++ Name	NodeAlg (int)
Java Name	NodeAlg (int)
.NET Name	NodeAlg (int)
OPL Name	nodealg
Interactive Optimizer	mip strategy subalgorithm
Identifier	2026

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

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

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

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

For MIQCP (integer constraints and positive semi-definite quadratic terms among the constraints), only the Barrier optimizer is implemented, and therefore no settings other than 0 (Automatic) and 4 (Barrier) are permitted.

Values

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

node storage file switch

Purpose
Node storage file switch

Syntax

C Name	CPX_PARAM_NODEFILEIND (int)
C++ Name	NodeFileInd (int)
Java Name	NodeFileInd (int)
.NET Name	NodeFileInd (int)
OPL Name	nodefileind
Interactive Optimizer	mip strategy file
Identifier	2016

Description
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 working directory parameter (CPX_PARAM_WORKDIR, WorkDir), and CPLEX® actively manages which nodes remain in memory for processing.

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

See also

directory for working files
memory available for working storage

MIP node limit

Purpose
MIP node limit

Syntax

C Name	CPX_PARAM_NODELIM (int)
C++ Name	NodeLim (int)
Java Name	NodeLim (int)
.NET Name	NodeLim (int)
OPL Name	nodelim
Interactive Optimizer	mip limits nodes
Identifier	2017

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

Values
Any nonnegative integer; **default**: 2100000000.

MIP node selection strategy

Purpose
MIP node selection strategy

Syntax

C Name	CPX_PARAM_NODESEL (int)
C++ Name	NodeSel (int)
Java Name	NodeSel (int)
.NET Name	NodeSel (int)
OPL Name	nodesel
Interactive Optimizer	mip strategy nodeselect
Identifier	2018

Description
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.

Values

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

numerical precision emphasis

Purpose
Numerical precision emphasis

Syntax

C Name	CPX_PARAM_NUMERICALEMPHASIS (int)
C++ Name	NumericalEmphasis (bool)
Java Name	NumericalEmphasis (bool)
.NET Name	NumericalEmphasis (bool)
OPL Name	numericalemphasis
Interactive Optimizer	emphasis numerical
Identifier	1083

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.

Values

Value	bool	Symbol	Meaning
0	false	CPX_OFF	Do not emphasize numerical precision; **default**
1	true	CPX_ON	Exercise extreme caution in computation

nonzero element read limit

Purpose
Nonzero element read limit

Syntax

C Name	CPX_PARAM_NZREADLIM (int)
C++ Name	NzReadLim (int)
Java Name	NzReadLim (int)
.NET Name	NzReadLim (int)
Interactive Optimizer	read nonzeros
Identifier	1024

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

Values
Any integer from 0 to 268 435 450; **default**: 250 000.

absolute objective difference cutoff

Purpose
Absolute objective difference cutoff

Syntax

C Name	CPX_PARAM_OBJDIF (double)
C++ Name	ObjDif (double)
Java Name	ObjDif (double)
.NET Name	ObjDif (double)
OPL Name	objdif
Interactive Optimizer	mip tolerances objdifference
Identifier	2019

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

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

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

Values
Any number; **default**: 0.0.

See also

relative objective difference cutoff

lower objective value limit

Purpose
Lower objective value limit

Syntax

C Name	CPX_PARAM_OBJLLIM (double)
C++ Name	ObjLLim (double)
Java Name	ObjLLim (double)
.NET Name	ObjLLim (double)
OPL Name	objllim
Interactive Optimizer	simplex limits lowerobj
Identifier	1025

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

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

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

upper objective value limit

Purpose
Upper objective value limit

Syntax

C Name	CPX_PARAM_OBJULIM (double)
C++ Name	ObjULim (double)
Java Name	ObjULim (double)
.NET Name	ObjULim (double)
OPL Name	objulim
Interactive Optimizer	simplex limits upperobj
Identifier	1026

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

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

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

parallel mode switch

Purpose
Parallel mode switch

Syntax

C Name	CPX_PARAM_PARALLELMODE (int)
C++ Name	ParallelMode (int)
Java Name	ParallelMode (int)
.NET Name	ParallelMode (int)
OPL Name	parallelmode
Interactive Optimizer	parallel
Identifier	1109

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

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

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

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

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

Deterministic and sequential optimization
Parallel MIP optimization can be opportunistic or deterministic.

Parallel barrier optimization is only deterministic.

Concurrent optimization is only opportunistic.

Interaction with the threads parameter

Settings of this parallel mode parameter interact with settings of the *global default thread count* parameter (`Threads`, `CPX_PARAM_THREADS`) as summarized in the tables:

- *Interaction of Callbacks with Threads and Parallel Mode Parameters: No Callbacks or only Informational Callbacks in Application*
- *Interaction of Callbacks with Threads and Parallel Mode Parameters: Only Query Callbacks in Application*
- *Interaction of Callbacks with Threads and Parallel Mode Parameters: Control Callbacks in Application*

The default (automatic) setting of the parallel mode parameter allows CPLEX® to choose between deterministic and opportunistic mode depending on the threads parameter. If the threads parameter is set to its automatic setting (the default), CPLEX® chooses deterministic mode.

If the threads parameter is set to one, CPLEX® runs sequentially in deterministic mode in a single thread.

Otherwise, if the threads parameter is set to a value greater than one, CPLEX® chooses opportunistic mode.

Callbacks and MIP optimization

If callbacks other than informational callbacks are used for solving a MIP, the order in which the callbacks are called cannot be guaranteed to remain deterministic, not even in deterministic mode. Thus, to make sure of deterministic runs when the parallel mode parameter is at its default setting, CPLEX® will revert to sequential solving of the MIP in the presence of query callbacks, diagnostic callbacks, or control callbacks.

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

Determinism vs opportunism

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

Values

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

See also: *global default thread count*: `CPX_PARAM_THREADS`, `Threads`

simplex perturbation switch

Purpose
Simplex perturbation switch

Syntax

C Name	CPX_PARAM_PERIND (int)
C++ Name	PerInd (bool)
Java Name	PerInd (bool)
.NET Name	PerInd (bool)
OPL Name	perind
Interactive Optimizer	simplex perturbation
Identifier	1027

Description
Decides whether to perturb problems.

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

Values

Value	bool	Symbol	Meaning
0	false	CPX_OFF	Automatic: let CPLEX choose; **default**
1	true	CPX_ON	Turn on perturbation from beginning

simplex perturbation limit

Purpose
Simplex perturbation limit

Syntax

C Name	CPX_PARAM_PERLIM (int)
C++ Name	PerLim (int)
Java Name	PerLim (int)
.NET Name	PerLim (int)
OPL Name	perlim
Interactive Optimizer	simplex limits perturbation
Identifier	1028

Description
Sets the number of degenerate iterations before perturbation is performed.

Values

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

absolute MIP gap before starting to polish a feasible solution

Purpose
Absolute MIP gap before starting to polish a feasible solution

Syntax

C Name	CPX_PARAM_POLISHAFTEREPAGAP (double)
C++ Name	PolishAfterEpAGap (double)
Java Name	PolishAfterEpAGap (double)
.NET Name	PolishAfterEpAGap (double)
OPL Name	
Interactive Optimizer	mip polishafter absmipgap
Identifier	2126

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

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

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

See also
absolute MIP gap tolerance

relative MIP gap before starting to polish a feasible solution

Purpose
Relative MIP gap before starting to polish a solution

Syntax

C Name	CPX_PARAM_POLISHAFTEREPGAP (double)
C++ Name	PolishAfterEpGap (double)
Java Name	PolishAfterEpGap (double)
.NET Name	PolishAfterEpGap (double)
OPL Name	
Interactive Optimizer	mip polishafter mipgap
Identifier	2127

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

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

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

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

See also
relative MIP gap tolerance

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

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

Syntax

C Name	CPX_PARAM_POLISHAFTERINTSOL (int)
C++ Name	PolishAfterIntSol (int)
Java Name	PolishAfterIntSol (int)
.NET Name	PolishAfterIntSol (int)
OPL Name	polishafterintsol
Interactive Optimizer	mip polishafter solutions
Identifier	2129

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

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

Values
Any positive integer strictly greater than zero; zero is **not** allowed; **default**: 2 100 000 000

See also
MIP integer solution limit

nodes to process before starting to polish a feasible solution

Purpose
Nodes to process before starting to polish a feasible solution

Syntax

C Name	CPX_PARAM_POLISHAFTERNODE (int)
C++ Name	PolishAfterNode (int)
Java Name	PolishAfterNode (int)
.NET Name	PolishAfterNode (int)
OPL Name	polishafternode
Interactive Optimizer	mip polishafter nodes
Identifier	2128

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

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

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

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

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

Values
Any nonnegative integer; **default**: 2 100 000 000

See also
MIP node limit

time before starting to polish a feasible solution

Purpose
Time before starting to polish a feasible solution

Syntax

C Name	CPX_PARAM_POLISHAFTERTIME (double)
C++ Name	PolishAfterTime (double)
Java Name	PolishAfterTime (double)
.NET Name	PolishAfterTime (double)
OPL Name	
Interactive Optimizer	mip polishafter time
Identifier	2130

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

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

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

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

See also
clock type for computation time

time spent polishing a solution (deprecated)

Purpose
Time spent polishing a solution (deprecated)

Syntax

C Name	CPX_PARAM_POLISHTIME (double)
C++ Name	PolishTime (double)
Java Name	PolishTime (double)
.NET Name	PolishTime (double)
OPL Name	polishtime
Interactive Optimizer	mip limit polishtime
Identifier	2066

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

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

- *absolute MIP gap before starting to polish a feasible solution*
- *relative MIP gap before starting to polish a feasible solution*
- *MIP integer solutions to find before starting to polish a feasible solution*
- *nodes to process before starting to polish a feasible solution*
- *time before starting to polish a feasible solution*
- *optimizer time limit*

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

maximum number of solutions generated for solution pool by populate

Purpose
Maximum number of solutions generated for the solution pool by populate

Syntax

C Name	CPX_PARAM_POPULATELIM (int)
C++ Name	PopulateLim (int)
Java Name	PopulateLim (int)
.NET Name	PopulateLim (int)
OPL Name	populatelim
Interactive Optimizer	mip limits populate
Identifier	2108

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

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

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

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

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

- *relative gap for solution pool*: `SolnPoolGap`, `CPX_PARAM_SOLNPOOLGAP`
- *absolute gap for solution pool*: `SolnPoolAGap`, `CPX_PARAM_SOLNPOOLAGAP`
- *MIP node limit*: `NodeLim`, `CPX_PARAM_NODELIM`
- *optimizer time limit*: `TiLim`, `CPX_PARAM_TILIM`

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

primal simplex pricing algorithm

Purpose
Primal simplex pricing algorithm

Syntax

C Name	CPX_PARAM_PPRIIND (int)
C++ Name	PPriInd (int)
Java Name	PPriInd (int)
.NET Name	PPriInd (int)
OPL Name	ppriind
Interactive Optimizer	simplex pgradient
Identifier	1029

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

Values

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

presolve dual setting

Purpose
Presolve dual setting

Syntax

C Name	CPX_PARAM_PREDUAL (int)
C++ Name	PreDual (int)
Java Name	PreDual (int)
.NET Name	PreDual (int)
OPL Name	predual
Interactive Optimizer	preprocessing dual
Identifier	1044

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

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

Values

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

presolve switch

Purpose
Presolve switch

Syntax

C Name	CPX_PARAM_PREIND (int)
C++ Name	PreInd (bool)
Java Name	PreInd (bool)
.NET Name	PreInd (bool)
OPL Name	preind
Interactive Optimizer	preprocessing presolve
Identifier	1030

Description
Decides whether CPLEX® applies presolve during preprocessing. When set to 1 (one), the default, this parameter invokes the CPLEX® presolve to simplify and reduce problems.

Values

Value	bool	Symbol	Meaning
0	false	CPX_OFF	Do not apply presolve
1	true	CPX_ON	Apply presolve; **default**

linear reduction switch

Purpose
Linear reduction switch

Syntax

C Name	CPX_PARAM_PRELINEAR (int)
C++ Name	PreLinear (int)
Java Name	PreLinear (int)
.NET Name	PreLinear (int)
OPL Name	prelinear
Interactive Optimizer	preprocessing linear
Identifier	1058

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

Values

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

limit on the number of presolve passes made

Purpose
Limit on the number of presolve passes made

Syntax

C Name	CPX_PARAM_PREPASS (int)
C++ Name	PrePass (int)
Java Name	PrePass (int)
.NET Name	PrePass (int)
OPL Name	prepass
Interactive Optimizer	preprocessing numpass
Identifier	1052

Description
Limits the number of presolve passes that CPLEX® makes during preprocessing. When this parameter is set to a nonzero value, invokes CPLEX® presolve to simplify and reduce problems.

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

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

When this parameter is set to zero, CPLEX® does not apply presolve, but other reductions may occur, depending on settings of other parameters and specifics of your model.

Values

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

node presolve switch

Purpose
Node presolve switch

Syntax

C Name	CPX_PARAM_PRESLVND (int)
C++ Name	PreslvNd (int)
Java Name	PreslvNd (int)
.NET Name	PreslvNd (int)
OPL Name	preslvnd
Interactive Optimizer	mip strategy presolvenode
Identifier	2037

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

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

simplex pricing candidate list size

Purpose
Simplex pricing candidate list size

Syntax

C Name	CPX_PARAM_PRICELIM (int)
C++ Name	PriceLim (int)
Java Name	PriceLim (int)
.NET Name	PriceLim (int)
OPL Name	pricelim
Interactive Optimizer	simplex pricing
Identifier	1010

Description
Sets the maximum number of variables kept in the list of pricing candidates for the simplex algorithms.

Values

Value	Meaning
0	Automatic: let CPLEX choose; **default**
Any positive integer	Number of pricing candidates

MIP probing level

Purpose
MIP probing level

Syntax

C Name	CPX_PARAM_PROBE (int)
C++ Name	Probe (int)
Java Name	Probe (int)
.NET Name	Probe (int)
OPL Name	probe
Interactive Optimizer	mip strategy probe
Identifier	2042

Description
Sets the amount of probing on variables to be performed before MIP branching. Higher settings perform more probing. Probing can be very powerful but very time-consuming at the start. Setting the parameter to values above the default of 0 (automatic) can result in dramatic reductions or dramatic increases in solution time, depending on the model.

Values

Value	Meaning
-1	No probing
0	Automatic: let CPLEX choose; **default**
1	Moderate probing level
2	Aggressive probing level
3	Very aggressive probing level

time spent probing

Purpose
Time spent probing

Syntax

C Name	CPX_PARAM_PROBETIME (double)
C++ Name	ProbeTime (double)
Java Name	ProbeTime (double)
.NET Name	ProbeTime (double)
OPL Name	probetime
Interactive Optimizer	mip limit probetime
Identifier	2065

Description
Limits the amount of time in seconds spent probing.

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

indefinite MIQP switch

Purpose
Indefinite MIQP switch

Syntax

C Name	CPX_PARAM_QPMAKEPSDIND (int)
C++ Name	QPmakePSDInd (bool)
Java Name	QPmakePSDInd (bool)
.NET Name	QPmakePSDInd (bool)
OPL Name	qpmakepsdind
Interactive Optimizer	preprocessing qpmakepsd
Identifier	4010

Description
Decides whether CPLEX® will attempt to reformulate a MIQP or MIQCP model that contains only binary variables. When this feature is active, adjustments will be made to the elements of a quadratic matrix that is not nominally positive semi-definite (PSD, as required by CPLEX® for all QP and most QCP formulations), to make it PSD, and CPLEX® will also attempt to tighten an already PSD matrix for better numerical behavior. The default setting of 1 (one) means yes, CPLEX® should attempt to reformulate, but you can turn it off if necessary; most models should benefit from the default setting.

Values

Value	bool	Symbol	Meaning
0	false	CPX_OFF	Turn off attempts to make binary model PSD
1	true	CPX_ON	On: CPLEX attempts to make binary model PSD; **default**

QP Q-matrix nonzero read limit

Purpose
QP Q matrix nonzero read limit

Syntax

C Name	CPX_PARAM_QPNZREADLIM (int)
C++ Name	QPNzReadLim (int)
Java Name	QPNzReadLim (int)
.NET Name	QPNzReadLim (int)
Interactive Optimizer	read qpnonzeros
Identifier	4001

Description
Specifies a limit for the number of nonzero elements to read for an allocation of memory in a model with a quadratic matrix.

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

Values
Any integer from 0 to 268 435 450; **default**: 5 000.

primal and dual reduction type

Purpose
Primal and dual reduction type

Syntax

C Name	CPX_PARAM_REDUCE (int)
C++ Name	Reduce (int)
Java Name	Reduce (int)
.NET Name	Reduce (int)
OPL Name	reduce
Interactive Optimizer	preprocessing reduce
Identifier	1057

Description
Decides whether primal reductions, dual reductions, both, or neither are performed during preprocessing.

Values

Value	Symbol	Meaning
0	CPX_PREREDUCE_NOPRIMALORDUAL	No primal or dual reductions
1	CPX_PREREDUCE_PRIMALONLY	Only primal reductions
2	CPX_PREREDUCE_DUALONLY	Only dual reductions
3	CPX_PREREDUCE_PRIMALANDDUAL	Both primal and dual reductions; **default**

simplex refactoring frequency

Purpose
Simplex refactoring frequency

Syntax

C Name	CPX_PARAM_REINV (int)
C++ Name	ReInv (int)
Java Name	ReInv (int)
.NET Name	ReInv (int)
OPL Name	reinv
Interactive Optimizer	simplex refactor
Identifier	1031

Description
Sets the number of iterations between refactoring of the basis matrix.

Values

Value	Meaning
0	Automatic: let CPLEX choose; **default**
Integer from 1 to 10 000	Number of iterations between refactoring of the basis matrix

relaxed LP presolve switch

Purpose
Relaxed LP presolve switch

Syntax

C Name	CPX_PARAM_RELAXPREIND (int)
C++ Name	RelaxPreInd (int)
Java Name	RelaxPreInd (int)
.NET Name	RelaxPreInd (int)
OPL Name	relaxpreind
Interactive Optimizer	preprocessing relax
Identifier	2034

Description
Decides whether LP presolve is applied to the root relaxation in a mixed integer program (MIP). Sometimes additional reductions can be made beyond any MIP presolve reductions that were already done. By default, CPLEX® applies presolve to the initial relaxation in order to hasten time to the initial solution.

Value	Symbol	Meaning
-1		Automatic: let CPLEX choose; **default**
0	CPX_OFF	Off: do not use presolve on initial relaxation
1	CPX_ON	On: use presolve on initial relaxation

relative objective difference cutoff

Purpose
Relative objective difference cutoff

Syntax

C Name	CPX_PARAM_RELOBJDIF (double)
C++ Name	RelObjDif (double)
Java Name	RelObjDif (double)
.NET Name	RelObjDif (double)
OPL Name	relobjdif
Interactive Optimizer	mip tolerances relobjdifference
Identifier	2022

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

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

If both the relative objective difference and the *absolute objective difference cutoff* (`CPX_PARAM_OBJDIF`, `ObjDif`) are nonzero, the value of the absolute objective difference is used.

Values
Any number from 0.0 to 1.0; **default**: 0.0.

See also
absolute objective difference cutoff

frequency to try to repair infeasible MIP start

Purpose
Frequency to try to repair infeasible MIP start

Syntax

C Name	CPX_PARAM_REPAIRTRIES (int)
C++ Name	RepairTries (int)
Java Name	RepairTries (int)
.NET Name	RepairTries (int)
OPL Name	repairtries
Interactive Optimizer	mip limits repairtries
Identifier	2067

Description
Limits the attempts to repair an infeasible MIP start. This parameter lets you tell CPLEX® whether and how many times it should try to repair an infeasible MIP start that you supplied. The parameter has no effect if the MIP start you supplied is feasible. It has no effect if no MIP start was supplied.

Values

Value	Meaning
-1	None: do not try to repair
0	Automatic: let CPLEX choose; **default**
Any positive integer	Frequency to attempt repairs

MIP repeat presolve switch

Purpose
Reapply presolve after processing the root node

Syntax

C Name	CPX_PARAM_REPEATPRESOLVE (int)
C++ Name	RepeatPresolve (int)
Java Name	RepeatPresolve (int)
.NET Name	RepeatPresolve (int)
OPL Name	repeatpresolve
Interactive Optimizer	preprocessing repeatpresolve
Identifier	2064

Description
Decides whether to re-apply presolve, with or without cuts, to a MIP model after processing at the root is otherwise complete.

Values

Value	Symbol
-1	Automatic: let CPLEX choose; **default**
0	Turn off represolve
1	Represolve without cuts
2	Represolve with cuts
3	Represovle with cuts and allow new root cuts

RINS heuristic frequency

Purpose
RINS heuristic frequency

Syntax

C Name	CPX_PARAM_RINSHEUR (int)
C++ Name	RINSHeur (int)
Java Name	RINSHeur (int)
.NET Name	RINSHeur (int)
OPL Name	rinsheur
Interactive Optimizer	mip strategy rinsheur
Identifier	2061

Description
Decides how often to apply the relaxation induced neighborhood search (RINS) heuristic. This heuristic attempts to improve upon the best solution found so far. It will not be applied until CPLEX® has found at least one incumbent solution.

Setting the value to -1 turns off the RINS heuristic. Setting the value to 0 (zero), the default, applies the RINS heuristic at an interval chosen automatically by CPLEX® . Setting the value to a positive number applies the RINS heuristic at the requested node interval. For example, setting RINSHeur to 20 dictates that the RINS heuristic be called at node 0, 20, 40, 60, etc.

RINS is a powerful heuristic for finding high quality feasible solutions, but it may be expensive.

Values

Value	Meaning
-1	None: do not apply RINS heuristic
0	Automatic: let CPLEX choose; **default**
Any positive integer	Frequency to apply RINS heuristic

algorithm for continuous problems

Purpose
Solution algorithm for continuous problems

Syntax

C Name	CPX_PARAM_LPMETHOD (int)
C++ Name	RootAlg (int)
Java Name	RootAlg (int)
.NET Name	RootAlg (int)
OPL Name	rootalg
Interactive Optimizer	lpmethod
Identifier	1062

Description
Controls which algorithm is used to solve continuous models or to solve the root relaxation of a MIP. In the object-oriented APIs, you make this selection through the `RootAlg` parameter. In the C API and the Interactive Optimizer, there are separate parameters to control LP, QP, and MIP optimizers, depending on the problem type.

In all cases, the default setting is 0 (zero). The default setting means that CPLEX® will select the algorithm in a way that should give best overall performance.

For specific problem classes, the following details document the automatic settings. Note that future versions of CPLEX® could adopt different strategies. Therefore, if you select any nondefault settings, you should review them periodically.

Currently, the behavior of the automatic setting is that CPLEX® almost always invokes the dual simplex algorithm when it is solving an LP model from scratch. When it is continuing from an advanced basis, it will check whether the basis is primal or dual feasible, and choose the primal or dual simplex algorithm accordingly.

If multiple threads have been requested, the 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.

Values

Value	Symbol	Meaning
0	CPX_ALG_AUTOMATIC	Automatic: let CPLEX choose; **default**
1	CPX_ALG_PRIMAL	Primal simplex
2	CPX_ALG_DUAL	Dual simplex
3	CPX_ALG_NET	Network simplex
4	CPX_ALG_BARRIER	Barrier
5	CPX_ALG_SIFTING	Sifting
6	CPX_ALG_CONCURRENT	Concurrent (Dual, Barrier, and Primal)

algorithm for continuous quadratic optimization

Purpose
Algorithm for continuous quadratic optimization

Syntax

C Name	CPX_PARAM_QPMETHOD (int)
C++ Name	RootAlg (int)
Java Name	RootAlg (int)
.NET Name	RootAlg (int)
OPL Name	rootalg
Interactive Optimizer	qpmethod
Identifier	1063

Description
Sets which algorithm to use when the C routine `CPXqpopt` (or the command `optimize` in the Interactive Optimizer) is invoked.

Currently, the behavior of the Automatic setting is that CPLEX® invokes the Barrier Optimizer for continuous QP models. The Automatic setting may be expanded in the future so that CPLEX® chooses the algorithm based on additional problem characteristics.

Values

Value	Symbol	Meaning
0	CPX_ALG_AUTOMATIC	Automatic: let CPLEX choose; **default**
1	CPX_ALG_PRIMAL	Use the primal simplex optimizer.
2	CPX_ALG_DUAL	Use the dual simplex optimizer.
3	CPX_ALG_NET	Use the network optimizer.
4	CPX_ALG_BARRIER	Use the barrier optimizer.

MIP starting algorithm

Purpose
MIP starting algorithm

Syntax

C Name	CPX_PARAM_STARTALG (int)
C++ Name	RootAlg (int)
Java Name	RootAlg (int)
.NET Name	RootAlg (int)
OPL Name	rootalg
Interactive Optimizer	mip strategy startalgorithm
Identifier	2025

Description
Sets which continuous optimizer will be used to solve the initial relaxation of a MIP.

The default Automatic setting (0 zero) of this parameter currently selects the 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.

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

For MIQP (integer constraints and positive semi-definite quadratic terms in the objective), settings 5 (Sifting) and 6 (Concurrent) are **not** implemented; if you happen to choose them, setting 5 (Sifting) reverts to 0 (ero) and setting 6 (Concurrent) reverts to 4.

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 (zero) and 4 are permitted.

Values

Value	Symbol	Meaning
0	CPX_ALG_AUTOMATIC	Automatic: let CPLEX choose; **default**
1	CPX_ALG_PRIMAL	Primal Simplex
2	CPX_ALG_DUAL	Dual Simplex
3	CPX_ALG_NET	Network Simplex
4	CPX_ALG_BARRIER	Barrier
5	CPX_ALG_SIFTING	Sifting
6	CPX_ALG_CONCURRENT	Concurrent (Dual, Barrier, and Primal)

constraint (row) read limit

Purpose
Constraint (row) read limit

Syntax

C Name	CPX_PARAM_ROWREADLIM (int)
C++ Name	RowReadLim (int)
Java Name	RowReadLim (int)
.NET Name	RowReadLim (int)
Interactive Optimizer	read constraints
Identifier	1021

Description
Specifies a limit for the number of rows (constraints) to read for an allocation of memory.

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

Values
Any integer from 0 to 268 435 450; **default**: 30 000.

scale parameter

Purpose
Scale parameter

Syntax

C Name	CPX_PARAM_SCAIND (int)
C++ Name	ScaInd (int)
Java Name	ScaInd (int)
.NET Name	ScaInd (int)
OPL Name	scaind
Interactive Optimizer	read scale
Identifier	1034

Description
Decides how to scale the problem matrix.

Values

Value	Meaning
-1	No scaling
0	Equilibration scaling; **default**
1	More aggressive scaling

messages to screen switch

Purpose
Messages to screen switch

Syntax

C Name	CPX_PARAM_SCRIND (int)
C++ Name	screen indicator not available in this API
Java Name	screen indicator not available in this API
.NET Name	screen indicator not available in this API
Interactive Optimizer	screen indicator not available in this interface
Identifier	1035

Description
Decides whether or not results are displayed on screen in an application of the C API.

To turn off output to the screen, in a C++ application, where `cplex` is an instance of the class `IloCplex` and `env` is an instance of the class `IloEnv` , the environment, use `cplex.setOut(env.getNullStream())` .

In a Java application, use `cplex.setOut(null)` .

In a .NET application, use `Cplex.SetOut(Null)` .

Values

Value	Symbol	Meaning
0	CPX_OFF	Turn off display of messages to screen; **default**
1	CPX_ON	Display messages on screen

sifting subproblem algorithm

Purpose
Sifting subproblem algorithm

Syntax

C Name	CPX_PARAM_SIFTALG (int)
C++ Name	SiftAlg (int)
Java Name	SiftAlg (int)
.NET Name	SiftAlg (int)
OPL Name	siftalg
Interactive Optimizer	sifting algorithm
Identifier	1077

Description
Sets the algorithm to be used for solving sifting subproblems. The default automatic setting will typically use a mix of barrier and primal simplex.

Values

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

sifting information display

Purpose
Sifting information display

Syntax

C Name	CPX_PARAM_SIFTDISPLAY (int)
C++ Name	SiftDisplay (int)
Java Name	SiftDisplay (int)
.NET Name	SiftDisplay (int)
OPL Name	siftdisplay
Interactive Optimizer	sifting display
Identifier	1076

Description
Sets the amount of information to display about the progress of sifting.

Values

Value	Meaning
0	No display of sifting information
1	Display major iterations; **default**
2	Display LP subproblem information within each sifting iteration

upper limit on sifting iterations

Purpose
Upper limit on sifting iterations

Syntax

C Name	CPX_PARAM_SIFTITLIM (int)
C++ Name	SiftItLim (int)
Java Name	SiftItLim (int)
.NET Name	SiftItLim (int)
OPL Name	siftitlim
Interactive Optimizer	sifting iterations
Identifier	1078

Description
Sets the maximum number of sifting iterations that may be performed if convergence to optimality has not been reached.

Values
Any nonnegative integer; **default**: 2100000000.

simplex iteration information display

Purpose
Simplex iteration information display

Syntax

C Name	CPX_PARAM_SIMDISPLAY (int)
C++ Name	SimDisplay (int)
Java Name	SimDisplay (int)
.NET Name	SimDisplay (int)
OPL Name	simdisplay
Interactive Optimizer	simplex display
Identifier	1019

Description
Sets how often CPLEX® reports about iterations during simplex optimization.

Values

Value	Meaning
0	No iteration messages until solution
1	Iteration information after each refactoring; **default**
2	Iteration information for each iteration

simplex singularity repair limit

Purpose
Simplex singularity repair limit

Syntax

C Name	CPX_PARAM_SINGLIM (int)
C++ Name	SingLim (int)
Java Name	SingLim (int)
.NET Name	SingLim (int)
OPL Name	singlim
Interactive Optimizer	simplex limits singularity
Identifier	1037

Description
Restricts the number of times CPLEX® attempts to repair the basis when singularities are encountered during the simplex algorithm. When this limit is exceeded, CPLEX® replaces the current basis with the best factorable basis that has been found.

Values
Any nonnegative integer; **default**: 10.

absolute gap for solution pool

Purpose
Absolute gap for solution pool

Syntax

C Name	CPX_PARAM_SOLNPOOLAGAP (double)
C++ Name	SolnPoolAGap (double)
Java Name	SolnPoolAGap (double)
.NET Name	SolnPoolAGap (double)
OPL Name	solnpoolagap
Interactive Optimizer	mip pool absgap
Identifier	2106

Description
Sets an absolute tolerance on the objective value for the solutions in the solution pool. Solutions that are worse (either greater in the case of a minimization, or less in the case of a maximization) than the objective of the incumbent solution according to this measure are not kept in the solution pool.

Values of the solution pool *absolute* gap (`SolnPoolAGap` or `CPX_PARAM_SOLNPOOLAGAP`) and the solution pool *relative* gap (*relative gap for solution pool*: `SolnPoolGap` or `CPX_PARAM_SOLNPOOLGAP`) may differ: For example, you may specify that solutions must be within 15 units by means of the solution pool absolute gap and also within 1% of the incumbent by means of the solution pool relative gap. A solution is accepted in the pool only if it is valid for both the relative and the absolute gaps.

The solution pool absolute gap parameter can also be used as a stopping criterion for the populate procedure: if populate cannot enumerate any more solutions that satisfy this objective quality, then it will stop. In the presence of both an absolute and a relative solution pool gap parameter, populate will stop when the smaller of the two is reached.

Values
Any nonnegative real number; **default**: 1.0e+75.

maximum number of solutions kept in solution pool

Purpose
Maximum number of solutions kept in the solution pool

Syntax

C Name	CPX_PARAM_SOLNPOOLCAPACITY (int)
C++ Name	SolnPoolCapacity (int)
Java Name	SolnPoolCapacity (int)
.NET Name	SolnPoolCapacity (int)
OPL Name	solnpoolcapacity
Interactive Optimizer	mip pool capacity
Identifier	2103

Description
Sets the maximum number of solutions kept in the solution pool. At most, `SolnPoolCapacity` solutions will be stored in the pool. Superfluous solutions are managed according to the strategy set by the *solution pool replacement strategy* parameter (`SolnPoolReplace`, `CPX_PARAM_SOLNPOOLREPLACE`).

The optimization (whether by MIP optimization or the populate procedure) will not stop if more than `SolnPoolCapacity` solutions are generated. Instead, stopping criteria can be specified by these parameters:

- *maximum number of solutions generated for solution pool by populate* (`PopulateLim`, `CPX_PARAM_POPULATELIM`)
- *relative gap for solution pool* (`SolnPoolGap`, `CPX_PARAM_SOLNPOOLGAP`)
- *absolute gap for solution pool* (`SolnPoolAGap`, `CPX_PARAM_SOLNPOOLAGAP`)
- *MIP node limit* (`NodeLim`, `CPX_PARAM_NODELIM`)
- *optimizer time limit* (`TiLim`, `CPX_PARAM_TILIM`)

The default value for `SolnPoolCapacity` is 2100000000, but it may be set to any nonnegative integer value. If set to zero, it will turn off all features related to the solution pool.

If you are looking for a parameter to control the number of solutions generated by the populate procedure, consider the parameter *maximum number of solutions generated for solution pool by populate*.

Values
Any nonnegative integer; 0 (zero) turns off all features of the solution pool; **default**: 2100000000.

relative gap for solution pool

Purpose
Relative gap for the solution pool

Syntax

C Name	CPX_PARAM_SOLNPOOLGAP (double)
C++ Name	SolnPoolGap (double)
Java Name	SolnPoolGap (double)
.NET Name	SolnPoolGap (double)
OPL Name	solnpoolgap
Interactive Optimizer	mip pool relgap
Identifier	2105

Description
Sets a relative tolerance on the objective value for the solutions in the solution pool. Solutions that are worse (either greater in the case of a minimization, or less in the case of a maximization) than the incumbent solution by this measure are not kept in the solution pool. For example, if you set this parameter to 0.01, then solutions worse than the incumbent by 1% or more will be discarded.

Values of the *absolute gap for solution pool* (SolnPoolAGap or CPX_PARAM_SOLNPOOLAGAP) and the *relative gap for solution pool* (SolnPoolGap or CPX_PARAM_SOLNPOOLGAP) may differ: For example, you may specify that solutions must be within 15 units by means of the solution pool absolute gap and within 1% of the incumbent by means of the solution pool relative gap. A solution is accepted in the pool only if it is valid for both the relative and the absolute gaps.

The solution pool relative gap parameter can also be used as a stopping criterion for the populate procedure: if populate cannot enumerate any more solutions that satisfy this objective quality, then it will stop. In the presence of both an absolute and a relative solution pool gap parameter, populate will stop when the smaller of the two is reached.

Values
Any nonnegative real number; **default**: 1.0e+75.

solution pool intensity

Purpose
Solution pool intensity

Syntax

C Name	CPX_PARAM_SOLNPOOLINTENSITY (int)
C++ Name	SolnPoolIntensity (int)
Java Name	SolnPoolIntensity (int)
.NET Name	SolnPoolIntensity (int)
OPL Name	solnpoolintensity
Interactive Optimizer	mip pool intensity
Identifier	2107

Description
Controls the trade-off between the number of solutions generated for the solution pool and the amount of time or memory consumed. This parameter applies both to MIP optimization and to the populate procedure.

Values from 1 (one) to 4 invoke increasing effort to find larger numbers of solutions. Higher values are more expensive in terms of time and memory but are likely to yield more solutions.

Effect
For MIP optimization, increasing the value of the parameter corresponds to increasing the amount of effort spent setting up the branch and cut tree to prepare for a subsequent call to the populate procedure.

For populate, increasing the value of this parameter corresponds, in addition, to increasing the amount of effort spent exploring the tree to generate more solutions. If MIP optimization is called before populate, populate will reuse the information computed and stored during MIP optimization only if this parameter has not been increased between calls. Similarly, if populate is called several times successively, populate will re-use the information computed and stored during previous calls to populate only if the solution pool intensity has not increased between calls. Therefore, it is most efficient **not** to change the value of this parameter between calls to MIP optimization and populate, nor between successive calls of populate. Increase the value of this parameter only if too few solutions are generated.

Settings
Its default value, 0 (zero), lets CPLEX® choose which intensity to apply. If MIP optimization is called first after the model is read, CPLEX® sets the intensity to 1 (one) for this call to MIP optimization and to subsequent calls of populate. In contrast, if populate is called directly after the model is read, CPLEX® sets the intensity to 2 for this call and subsequent calls of populate.

For value 1 (one), the performance of MIP optimization is not affected. There is no slowdown and no additional consumption of memory due to this setting. However, populate will quickly generate only a small number of solutions. Generating more than a few solutions with this setting will be slow. When you are looking for a larger number of solutions, use a higher value of this parameter.

For value 2, some information is stored in the branch and cut tree so that it is easier to generate a larger number of solutions. This storage has an impact on memory used but does not lead to a slowdown in the performance of MIP optimization. With this value, calling populate is likely to yield a number of solutions large enough for most purposes. This value is a good choice for most models.

For value 3, the algorithm is more aggressive in computing and storing information in order to generate a large number of solutions. Compared to values 1 (one) and 2, this value will generate a larger number of solutions, but it will slow MIP optimization and increase memory consumption. Use this value only if setting this parameter to 2 does not generate enough solutions.

For value 4, the algorithm generates all solutions to your model. Even for small models, the number of possible solutions is likely to be huge; thus enumerating all of them will take time and consume a large quantity of memory. In this case, remember to set the *maximum number of solutions generated for solution pool by populate* (`PopulateLim`, `CPX_PARAM_POPULATELIM`) to a value appropriate for your model; otherwise, the populate procedure will stop prematurely because of this stopping criterion instead of enumerating all solutions. In addition, a few limitations apply to this exhaustive enumeration, as explained in Enumerating all solutions in the *CPLEX User's Manual*.

Values

Value	Meaning
0	Automatic: let CPLEX choose; **default**
1	Mild: generate few solutions quickly
2	Moderate: generate a larger number of solutions
3	Aggressive: generate many solutions and expect performance penalty
4	Very aggressive: enumerate all practical solutions

solution pool replacement strategy

Purpose
Solution pool replacement strategy

Syntax

C Name	CPX_PARAM_SOLNPOOLREPLACE (int)
C++ Name	SolnPoolReplace (int)
Java Name	SolnPoolReplace (int)
.NET Name	SolnPoolReplace (int)
OPL Name	solnpoolreplace
Interactive Optimizer	mip pool replace
Identifier	2104

Description
Designates the strategy for replacing a solution in the solution pool when the solution pool has reached its capacity.

The value 0 (CPX_SOLNPOOL_FIFO) replaces solutions according to a first-in, first-out policy. The value 1 (CPX_SOLNPOOL_OBJ) keeps the solutions with the best objective values. The value 2 (CPX_SOLNPOOL_DIV) replaces solutions in order to build a set of diverse solutions.

If the solutions you obtain are too similar to each other, try setting SolnPoolReplace to 2.

The replacement strategy applies only to the subset of solutions created in the current call of MIP optimization or populate. Solutions already in the pool are not affected by the replacement strategy. They will not be replaced, even if they satisfy the criterion of the replacement strategy.

Values

Value	Symbol	Meaning
0	CPX_SOLNPOOL_FIFO	Replace the first solution (oldest) by the most recent solution; first in, first out; **default**
1	CPX_SOLNPOOL_OBJ	Replace the solution which has the worst objective
2	CPX_SOLNPOOL_DIV	Replace solutions in order to build a set of diverse solutions

MIP strong branching candidate list limit

Purpose
MIP strong branching candidate list limit

Syntax

C Name	CPX_PARAM_STRONGCANDLIM (int)
C++ Name	StrongCandLim (int)
Java Name	StrongCandLim (int)
.NET Name	StrongCandLim (int)
OPL Name	strongcandlim
Interactive Optimizer	mip limits strongcand
Identifier	2045

Description
Controls the length of the candidate list when CPLEX® uses strong branching as the way to select variables. For more detail about that parameter, see *MIP variable selection strategy*:

- `VarSel` in the C++, Java, or .NET API;
- `CPX_PARAM_VARSEL` in the C API;
- `set mip strategy variableselect 3` in the Interactive Optimizer.

Values
Any positive number; **default**: 10.

MIP strong branching iterations limit

Purpose
MIP strong branching iterations limit

Syntax

C Name	CPX_PARAM_STRONGITLIM (int)
C++ Name	StrongItLim (int)
Java Name	StrongItLim (int)
.NET Name	StrongItLim (int)
OPL Name	strongitlim
Interactive Optimizer	mip limits strongit
Identifier	2046

Description
Controls the number of simplex iterations performed on each variable in the candidate list when CPLEX® uses strong branching as the way to select variables. For more detail about that parameter, see *MIP variable selection strategy*:

- `VarSel` in the C++, Java, or .NET API;
- `CPX_PARAM_VARSEL` in the C API;
- `set mip strategy variableselect 3` in the Interactive Optimizer.

The default setting 0 (zero) chooses the iteration limit automatically.

Values

Value	Meaning
0	Automatic: let CPLEX choose; **default**
Any positive integer	Limit of the simplex iterations performed on each candidate variable

limit on nodes explored when a subMIP is being solved

Purpose
Limit on nodes explored when a subMIP is being solved

Syntax

C Name	CPX_PARAM_SUBMIPNODELIM (int)
C++ Name	SubMIPNodeLim (int)
Java Name	SubMIPNodeLim (int)
.NET Name	SubMIPNodeLim (int)
OPL Name	submipnodelim
Interactive Optimizer	mip limits submipnodelim
Identifier	2062

Description
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.

Values
Any positive integer; **default**: 500.

symmetry breaking

Purpose
Symmetry breaking

Syntax

C Name	CPX_PARAM_SYMMETRY (int)
C++ Name	Symmetry (int)
Java Name	Symmetry (int)
.NET Name	Symmetry (int)
OPL Name	symmetry
Interactive Optimizer	preprocessing symmetry
Identifier	2059

Description
Decides whether symmetry breaking reductions will be automatically executed, during the preprocessing phase, in a MIP model. The default level, -1, allows CPLEX® to choose the degree of symmetry breaking to apply. The value 0 (zero) turns off symmetry breaking. Levels 1 through 5 apply increasingly aggressive symmetry breaking.

Values

Value	Meaning
-1	Automatic: let CPLEX choose; **default**
0	Turn off symmetry breaking
1	Exert a moderate level of symmetry breaking
2	Exert an aggressive level of symmetry breaking
3	Exert a very aggressive level of symmetry breaking
4	Exert a highly aggressive level of symmetry breaking
5	Exert an extremely aggressive level of symmetry breaking

global default thread count

Purpose
Global default thread count

Syntax

C Name	CPX_PARAM_THREADS (int)
C++ Name	Threads (int)
Java Name	Threads (int)
.NET Name	Threads (int)
OPL Name	threads
Interactive Optimizer	threads
Identifier	1067

Description
Sets the default number of parallel threads that will be invoked by any CPLEX® parallel optimizer. Settings of this thread parameter interact with settings of the *parallel mode switch* (CPX_PARAM_PARALLELMODE, ParallelMode) as summarized in:

- **Table 1:** *Interaction of Callbacks with Threads and Parallel Mode Parameters: No Callbacks or only Informational Callbacks in Application*
- **Table 2:** *Interaction of Callbacks with Threads and Parallel Mode Parameters: Only Query Callbacks in Application*
- **Table 3:** *Interaction of Callbacks with Threads and Parallel Mode Parameters: Control Callbacks in Application*

For single threads, the parallel algorithms behave deterministically, regardless of thread and parallel mode parameter settings; that is, the algorithm proceeds sequentially in a single thread.

In this context, *sequential* means that the algorithm proceeds step by step, consecutively, in a predictable and repeatable order within a single thread. *Deterministic* means that repeated solving of the same model with the same parameter settings on the same computing platform will follow exactly the same solution path, yielding the same level of performance and the same values in the solution. Sequential execution is deterministic. In multithreaded computing, a deterministic setting requires synchronization between threads. *Opportunistic* entails less synchronization between threads and thus may offer better performance at the sacrifice of repeatable, invariant solution paths and values in repeated runs on multiple threads or multiple processors.

In the following tables, *maximum number of threads* means the **minimum** of these two values:

- the maximum number of threads licensed;
- number of CPUs, cores available.

Interaction of Callbacks with Threads and Parallel Mode Parameters: No Callbacks or only Informational Callbacks in Application

	Parallel Mode Auto	Parallel Mode Opportuniistic (-1)	Parallel Mode Deterministic (1)
Threads 0 Auto	Uses maximum number of threads; deterministic	Uses maximum number of threads; opportunistic	Uses maximum number of threads; deterministic
Threads 1	Uses one thread; sequential	Uses one thread; sequential	Uses one thread; sequential
Threads N > 1	Uses N threads; opportunistic	Uses N threads; opportunistic	Uses N threads; deterministic

Interaction of Callbacks with Threads and Parallel Mode Parameters: Only Query Callbacks in Application

	Parallel Mode Auto	Parallel Mode Opportunistic (-1)	Parallel Mode Deterministic (1)
Threads 0 Auto	Uses one thread; deterministic	Uses maximum number of threads; opportunistic	Uses maximum number of threads; deterministic
Threads 1	Uses one thread; sequential	Uses one thread; sequential	Uses one thread; sequential
Threads N > 1	Uses N threads; opportunistic	Uses N threads; opportunistic	Uses N threads; deterministic

Interaction of Callbacks with Threads and Parallel Mode Parameters: Control Callbacks in Application

	Parallel Mode Auto	Parallel Mode Opportunistic (-1)	Parallel Mode Deterministic (1)
Threads 0 Auto	Uses one thread; sequential	Uses one thread; sequential	Uses one thread; sequential
Threads 1	Uses one thread; sequential	Uses one thread; sequential	Uses one thread; sequential
Threads N > 1	Uses N threads; opportunistic	Uses N threads; opportunistic	Uses N threads; deterministic

Values

Value	Meaning
0	Automatic: let CPLEX decide; **default**
1	Sequential; single threaded
N	Uses N threads; N is limited by license and available processors

See also
parallel mode switch

optimizer time limit

Purpose
Optimizer time limit

Syntax

C Name	CPX_PARAM_TILIM (double)
C++ Name	TiLim (double)
Java Name	TiLim (double)
.NET Name	TiLim (double)
OPL Name	tilim
Interactive Optimizer	timelimit
Identifier	1039

Description
Sets the maximum time, in seconds, for a call to an optimizer. This time limit applies also to the conflict refiner.

The time is measured in terms of either CPU time or elapsed time, according to the setting of the *clock type for computation time* parameter (CPX_PARAM_CLOCKTYPE, ClockType).

The time limit for an optimizer applies to the sum of all its steps, such as preprocessing,crossover, and internal calls to other optimizers.

In a sequence of calls to optimizers, the limit is not cumulative but applies to each call individually. For example, if you set a time limit of 10 seconds, and you call mipopt twice then there could be a total of (at most) 20 seconds of running time if each call consumes its maximum allotment.

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

See also
clock type for computation time

tree memory limit

Purpose
Tree memory limit

Syntax

C Name	CPX_PARAM_TRELIM (double)
C++ Name	TreLim (double)
Java Name	TreLim (double)
.NET Name	TreLim (double)
OPL Name	trelim
Interactive Optimizer	mip limits treememory
Identifier	2027

Description
Sets an absolute upper limit on the size (in megabytes, uncompressed) of the branch-and-cut tree. If this limit is exceeded, CPLEX® terminates optimization.

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

tuning information display

Purpose
Tuning information display

Syntax

```
C Name                    CPX_PARAM_TUNINGDISPLAY (int)

C++ Name                  TuningDisplay (int)

Java Name                 TuningDisplay (int)

.NET Name                 TuningDisplay (int)

OPL Name                  tuningdisplay

Interactive Optimizer     tune display

Identifier                1113
```

Description
Specifies the level of information reported by the tuning tool as it works.

Use level 0 (zero) to turn off reporting from the tuning tool.

Use level 1 (one), the **default**, to display a minimal amount of information.

Use level 2 to display the minimal amount plus the parameter settings that the tuning tool is trying.

Use level 3 to display an exhaustive report of minimal information, plus settings that are being tried, plus logs.

Values

Value	Meaning
0	Turn off display
1	Display standard, minimal reporting; **default**
2	Display standard report plus parameter settings being tried
3	Display exhaustive report and log

tuning measure

Purpose
Tuning measure

Syntax

C Name	CPX_PARAM_TUNINGMEASURE
C++ Name	TuningMeasure
Java Name	TuningMeasure
.NET Name	TuningMeasure
OPL Name	tuningmeasure
Interactive Optimizer	tune measure
Identifier	1110

Description
Controls the measure for evaluating progress when a suite of models is being tuned.

Possible values are:

- CPX_TUNE_AVERAGE uses the mean average of time to compare different parameter sets over a suite of models.
- CPX_TUNE_MINMAX uses a minmax approach to compare the time of different parameter sets over a suite of models.

Values

Value	Meaning
CPX_TUNE_AVERAGE	mean time; **default**
CPX_TUNE_MINMAX	minmax time

tuning repeater

Purpose
Tuning repeater

Syntax

C Name	CPX_PARAM_TUNINGREPEAT (int)
C++ Name	TuningRepeat (int)
Java Name	TuningRepeat (int)
.NET Name	TuningRepeat (int)
OPL Name	tuningrepeat
Interactive Optimizer	tune repeat
Identifier	1112

Description
Specifies the number of times tuning is to be repeated on reordered versions of a given problem. The problem is reordered automatically by CPLEX® permuting its rows and columns. This repetition is helpful when only one problem is being tuned, as repeated reordering and re-tuning may lead to more robust tuning results.

This parameter applies to only one problem in a tuning session. That is, in the Interactive Optimizer, this parameter is effective only when you are tuning a single problem; in the Callable Library (C API), this parameter is effective only when you are tuning a single problem with the routine `CPXtuneparam` .

Values
Any nonnegative integer; **default**: 1 (one)

tuning time limit

Purpose
Tuning time limit

Syntax

C Name	CPX_PARAM_TUNINGTILIM (double)
C++ Name	TuningTiLim (double)
Java Name	TuningTiLim (double)
.NET Name	TuningTiLim (double)
OPL Name	tuningtilim
Interactive Optimizer	tune timelimit
Identifier	1113

Description
Sets a time limit per model and per test set (that is, suite of models) applicable in tuning.

For an overall time limit on tuning, use the global time limit parameter (*optimizer time limit* TiLim, CPX_PARAM_TILIM).

For an example of how to use these time limit parameters together, see Example: time limits on tuning in the Interactive Optimizer in the *CPLEX User's Manual*.

Values
Any nonnegative number; **default**: 10 000.

See also
optimizer time limit

MIP variable selection strategy

Purpose
MIP variable selection strategy

Syntax

C Name	CPX_PARAM_VARSEL (int)
C++ Name	VarSel (int)
Java Name	VarSel (int)
.NET Name	VarSel (int)
OPL Name	varsel
Interactive Optimizer	mip strategy variableselect
Identifier	2028

Description
Sets the rule for selecting the branching variable at the node which has been selected for branching.

The minimum infeasibility rule chooses the variable with the value closest to an integer but still fractional. The minimum infeasibility rule (-1) may lead more quickly to a first integer feasible solution, but is usually slower overall to reach the optimal integer solution.

The maximum infeasibility rule chooses the variable with the value furtherest from an integer. The maximum infeasibility rule (1 one) forces larger changes earlier in the tree.

Pseudo cost (2) variable selection is derived from pseudo-shadow prices.

Strong branching (3) causes variable selection based on partially solving a number of subproblems with tentative branches to see which branch is the most promising. This strategy can be effective on large, difficult MIP problems.

Pseudo reduced costs (4) are a computationally less-intensive form of pseudo costs.

The default value (0 zero) allows CPLEX® to select the best rule based on the problem and its progress.

Values

Value	Symbol	Meaning
-1	CPX_VARSEL_MININFEAS	Branch on variable with minimum infeasibility
0	CPX_VARSEL_DEFAULT	Automatic: let CPLEX choose variable to branch on; **default**
1	CPX_VARSEL_MAXINFEAS	Branch on variable with maximum infeasibility
2	CPX_VARSEL_PSEUDO	Branch based on pseudo costs
3	CPX_VARSEL_STRONG	Strong branching
4	CPX_VARSEL_PSEUDOREDUCED	Branch based on pseudo reduced costs

directory for working files

Purpose
Directory for working files

Syntax

C Name	CPX_PARAM_WORKDIR (string)
C++ Name	WorkDir (string)
Java Name	WorkDir (string)
.NET Name	WorkDir (string)
OPL Name	workdir
Interactive Optimizer	workdir
Identifier	1064

Description
Specifies the name of an existing directory into which CPLEX® may store temporary working files, such as for MIP node files or for out-of-core barrier files. The default is the current working directory.

Values
Any existing directory; **default**: '.'

memory available for working storage

Purpose
Memory available for working storage

Syntax

C Name	CPX_PARAM_WORKMEM (double)
C++ Name	WorkMem (double)
Java Name	WorkMem (double)
.NET Name	WorkMem (double)
OPL Name	workmem
Interactive Optimizer	workmem
Identifier	1065

Description
Specifies an upper limit on the amount of central memory, in megabytes, that CPLEX® is permitted to use for working memory before swapping to disk files, compressing memory, or taking other actions.

Values
Any nonnegative number, in megabytes; **default**: 128.0

See also
directory for working files

write level for MST, SOL files

Purpose
Write level for MST, SOL files

Syntax

C Name	CPX_PARAM_WRITELEVEL (int)
C++ Name	WriteLevel (int)
Java Name	WriteLevel (int)
.NET Name	WriteLevel (int)
OPL Name	
Interactive Optimizer	output writelevel
Identifier	

Description
Sets the level of detail for CPLEX® to write a solution to a file in SOL format or a MIP start to a file in MST format. CPLEX® writes information about a MIP start to a formatted file of type MST with the file extension `.mst`. CPLEX® writes information about a solution to a formatted file of type SOL with the file extension `.sol`. CPLEX® records the write level at which it created a file in that file, so that the file can be read back accurately later.

The default setting of this parameter is 0 (zero) AUTO; that is, let CPLEX® decide the level of detail. CPLEX® behaves differently, depending on whether the format is SOL or MST and on whether it is writing a solution or MIP start. For SOL files, AUTO resembles level 1 (one): CPLEX® writes all variables and their respective values to the file. For MST files, AUTO resembles level 2: CPLEX® writes discrete variables and their respective values to the file.

When the value of this parameter is 1 (one), CPLEX® writes **all** variables, both discrete and continuous, with their values.

When the value of this parameter is 2, CPLEX® writes values for **discrete** variables only.

When the value of this parameter is 3, CPLEX® writes values of **nonzero** variables only.

When the value of this parameter is 4, CPLEX® writes values of **nonzero discrete** variables only.

Treatment of nonzeros
With respect to levels 3 and 4, where **nonzero** values are significant, CPLEX® considers a value nonzero if the absolute value is strictly less than 1e-16. In the case of **SOL** files, CPLEX® applies this test to **primal** and **dual variable** values, that is, both x and pi variable values. In the case of **MST** files, CPLEX® applies this test only to x values.

Restrictions due to reduced file size

Levels 3 and 4 reduce the size of files, of course. However, this reduced file entails restrictions and may create surprising results when the file is re-used. Levels 3 and 4 are not equivalent to levels 1 and 2. Indeed, if a MIP start does not contain a value for a variable expected at level 3 or 4, then this variable will be fixed to 0 (zero) when that MIP start file is processed. Specifically, at level 3, if the MIP start does not specify a value for a variable of any type, or at level 4, if the MIP start does not specify a value for a discrete variable, such a variable will be fixed to 0 (zero). Consequently, the same MIP start written at level 1 or 2 may produce satisfactory solutions, but the reduced MIP start file, written at level 3 or 4, perhaps does not lead to solutions. This surprising situation arises typically in the case of model changes with the addition of new variables.

Values

Value	Symbol	Meaning
0	AUTO	Automatic: let CPLEX decide
1	CPX_WRITELEVEL_ALLVARS	CPLEX writes all variables and their values
2	CPX_WRITELEVEL_DISCRETEVARS	CPLEX writes only discrete variables and their values
3	CPX_WRITELEVEL_NONZEROVARS	CPLEX writes only nonzero variables and their values
4	CPX_WRITELEVEL_NONZERODISCRETEVARS	CPLEX writes only nonzero discrete variables and their values

MIP zero-half cuts switch

Purpose
MIP zero-half cuts switch

Syntax

C Name	CPX_PARAM_ZEROHALFCUTS (int)
C++ Name	ZeroHalfCuts (int)
Java Name	ZeroHalfCuts (int)
.NET Name	ZeroHalfCuts (int)
OPL Name	zerohalfcuts
Interactive Optimizer	mip cuts zerohalfcut
Identifier	2111

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

If you find that too much time is spent generating zero-half cuts for your model, consider setting this parameter to -1 (minus one) to turn off zero-half cuts.

If the dual bound of your model does not make sufficient progress, consider setting this parameter to 2 to generate zero-half cuts more aggressively.

Values

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

Index

C

N

O

P

Q

R