Inside of the square brackets, two variables are multiplied by an asterisk (`*`). For example, `[4x*y]` indicates that the coefficients of both of the off-diagonal terms of ***Q***, corresponding to the variables `x` and `y` in the model are 2, since `4x*y` equals `2x*y + 2x*y`. Each pair of off-diagonal terms of ***Q*** is specified only once. ILOG CPLEX automatically creates both off-diagonal entries of ***Q***. Diagonal terms in ***Q*** (that is, terms with an exponent of 2) are indicated by the caret (`^`) followed by 2. For example, `4x^2` indicates that the coefficient of the diagonal term of ***Q*** corresponding to the variable `x` in the model is 4.

For example, this problem

Minimize $a + b + 1/2(a^2 + 4ab + 7b^2)$

subject to $a + b \geq 10$ *and* $a, b \geq 0$

in LP format looks like this:

```
Minimize
obj: a + b + [ a^2 + 4 a * b + 7 b^2 ]/2
Subject To
c1: a + b >= 10
End
```

18. This rule is of interest only to advanced users. It is possible to include pools of lazy constraints and user defined cuts in an LP file. A pool of lazy constraints or of user defined cuts must not contain any quadratic constraints. For more about these concepts, see *User-Cut and Lazy-Constraint Pools* on page 377 in the *ILOG CPLEX User's Manual*.

MPS File Format

MPS format, long established on mainframe LP systems, has become a widely accepted standard for defining LP problems. In contrast to the ILOG CPLEX LP format, MPS format is a *column-oriented* format: problems are specified by column (variable) rather than by row (constraint).

ILOG CPLEX Extensions to MPS Format

Historically, MPS format (including CPLEX MPS format for CPLEX version 2.1 and earlier releases) included restrictions, such as requiring input fields to occupy fixed columnar positions and limiting all names to a length of 8 characters or fewer. In CPLEX version 3.0 and subsequent releases, these restrictions were relaxed. The current ILOG CPLEX MPS format is actually an extended version of the historical MPS format. To allow for these extensions, certain practices which were accepted in MPS files for older CPLEX releases and other systems are no longer permitted. For example, since ILOG CPLEX no longer requires fixed columnar positions, blank spaces are interpreted as delimiters. Older MPS files containing names with embedded spaces therefore become unreadable. To maintain compatibility with earlier versions as well as MPS files from other systems, ILOG CPLEX provides an MPS file conversion utility which translates older files into the newer ILOG CPLEX MPS format. The section *Converting File Formats* on page 140 in the *ILOG CPLEX User's Manual* explains how to use the file conversion utility.

Records in MPS Format

MPS data files are analogous to a deck of computer input cards: each line of the MPS file represents a single card record. Records in an MPS data file consist of two types: indicator records and data records. The records contain fields delimited by blank spaces.

Indicator Records

Indicator records separate the individual sections of the MPS file. Each indicator record contains a single word that begins in the first column. There are seven kinds of indicator records, each corresponding to sections of the MPS file. They are listed in Table 1.

Table 1 *Indicator records*

Section name/indicator record	Purpose
`NAME`	specifies the problem name; unlike other indicator records, the name record contains data
`ROWS`	specifies name and sense for each constraint
`COLUMNS`	specifies the name assigned to each variable (column) and the nonzero constraint coefficients corresponding to that variable
`RHS`	specifies the names of righthand side vectors and values for each constraint (row)
`RANGES`	specifies constraints that are restricted to lie in the interval between two values; interval endpoints are also specified
`BOUNDS`	specifies the limits within which each variable (column) must remain
`ENDATA`	signals the end of the data; always the last entry in an MPS file

Each section of the MPS file except the `RANGES` and `BOUNDS` sections is mandatory. If no `BOUNDS` section is present, all variables have their bounds set from 0 (zero) to $+\infty$ (positive infinity). Failure to include an `RHS` section causes ILOG CPLEX to generate a warning message and set all righthand side values to 0 (zero). Variables and constraints must be declared in the `ROWS` and `COLUMNS` sections before they are referenced in the `RHS`, `RANGES`, and `BOUNDS` sections.

Data Records

Data records contain the information that describes the LP problem. Each data record comprises six fields, as in Table 2. The fields must be separated by white space (that is, blank space, tab, etc.), and the first field must begin in column 2 or beyond. Not all fields are used within each section of the input file.

Table 2 *Fields of a data record in MPS file format*

	Field 1	**Field 2**	**Field 3**	**Field 4**	**Field 5**	**Field 6**
Contents	Indicator	Name	Name	Value	Name	Value

Any ASCII character (32 through 126) is legal, but names must contain no embedded blanks. In addition, names over 255 characters are truncated. CPLEX issues an error message if truncation causes the names to lose their uniqueness. Numeric fields can be at most 25 characters long.

If the first character in Field 3 or 5 is a dollar sign ($), the remaining characters in the record are treated as a comment. Another method for inserting comments is to place an asterisk (*) in column 1. Everything on such a line is treated as a comment.

Values may be defined with decimal or exponential notation and may utilize 25 characters. In exponential notation, plus (+) and minus (-) signs must precede the exponent value. If an exponent value is missing where one is expected, it is assigned a value of 0 (zero).

ROWS Section

In the `ROWS` section, each row of the problem is specified with its name and sense, one row per record.

Field 1 contains a single letter designating the sense of each row. Acceptable values are:

- `N` indicates a free row.
- `G` indicates a greater-than-or-equal-to row.
- `L` indicates a less-than-or-equal-to row.
- `E` indicates an equality row.

Field 2 contains a character identifier, maximum length of 255 characters, specifying the name of the row.

Fields 3-6 are not used in the `ROWS` section.

If more than one free row is specified, the first one is used as the objective function and the others are discarded.

The `ROWS` section of our example looks like this:

```
ROWS
 N  obj
 L  c1
 L  c2
```

COLUMNS Section

In the `COLUMNS` section, all the columns of the constraint matrix are specified with their name and all of the nonzero elements. Multiple records may be required to completely specify a given column.

Field 1: Blank

Field 2: Column identifier

Field 3: Row identifier

Field 4: Value of matrix coefficient specified by Fields 2 and 3

Field 5: Row identifier (optional)

Field 6: Value of matrix coefficient specified by Fields 2 and 5 (optional)

After a matrix element is specified for a column, all other nonzero elements in that same column should be specified.

The `COLUMNS` section of our example looks like this:

```
COLUMNS
    x1        obj                 -1   c1                  -1
    x1        c2                   1
    x2        obj                 -2   c1                   1
    x2        c2                  -3
    x3        obj                 -3   c1                   1
    x3        c2                   1
```

RHS Section

In the `RHS` section, the nonzero righthand-side values of the constraints are specified.

Field 1: Blank

Field 2: RHS identifier

Field 3: Row identifier

Field 4: Value of RHS coefficient specified by Field 2 and 3

Field 5: Row identifier (optional)

Field 6: Value of RHS coefficient specified by Field 2 and 5 (optional)

Several RHS vectors can exist. The name of each RHS vector appears in Field 2. However, only the first RHS vector is selected when a problem is read. Additional RHS vectors are discarded.

The `RHS` section of our example looks like this:

```
RHS
    rhs       c1                   20   c2                   30
```

RANGES Section

In the `RANGES` section, RHS range values to be applied to constraints may be specified.

Field 1: Blank

Field 2: Righthand side range vector identifier

Field 3: Row identifier

Field 4: Value of the range applied to row specified by Field 3

Field 5: Row identifier (optional)

Field 6: Value of the range applied to row specified by Field 5 (optional)

The effect of specifying a righthand side range depends on the sense of the specified row and whether the range has a positive or negative coefficient. Table 3 specifies how range values are interpreted. For a given row, `rhs` is the righthand side value and `range` is the corresponding range value.

Table 3 *How range values are interpreted in data records of MPS files*

Row type	Range value sign	Resulting rhs upper limit	Resulting rhs lower limit
G	+ or -	rhs + |range|	rhs
L	+ or -	rhs	rhs - |range|
E	+	rhs + range	rhs
E	-	rhs	rhs + range

The name of each range vector appears in Field 2. More than one range vector can be specified within an MPS file. However, only the first range vector is selected when a problem is read. Additional range vectors are discarded.

In our example, there are no ranged rows, but suppose we want to add the following constraint to our problem:

```
x1 - 3x2 + x3 >= 15
```

Instead of explicitly adding another row to the problem, we can represent this additional constraint by modifying row 2 of the example to make it a ranged row in this way:

```
15 <= x1 - 3x2 + x3 <= 30
```

The `RANGES` section of the MPS file to support this modification looks like this:

```
RANGES
    rhs       c2                   15
```

The name of each range vector appears in Field 2. However, only the first range vector is selected when a problem is read. Additional range vectors are discarded.

BOUNDS Section

In the `BOUNDS` section, bound values for variables may be specified.

Field 1: Type of bound. Acceptable values are:

- `LO` Lower bound
- `UP` Upper bound
- `FX` Fixed value (upper and lower bound the same)
- `FR` Free variable (lower bound -∞ and upper bound +∞)
- `MI` Minus infinity (lower bound = -∞)
- `PL` Plus infinity (upper bound = +∞)

Field 2: Bound identifier

Field 3: Column identifier to be bounded

Field 4: Value of the specified bound

Fields 5 and 6 are not used in the `BOUNDS` section.

In our example, the `BOUNDS` section looks like this:

```
BOUNDS
 UP BOUND     x1                   40
```

If no bounds are specified, ILOG CPLEX assumes a lower bound of 0 (zero) and an upper bound of +∞. If only a single bound is specified, the unspecified bound remains at 0 or +∞, whichever applies, with one exception. If an upper bound of less than 0 is specified and no other bound is specified, the lower bound is automatically set to -∞. ILOG CPLEX deviates slightly from a convention used by some MPS readers when it encounters an upper bound of 0 (zero). Rather than automatically set this variable's lower bound to -∞, ILOG CPLEX accepts both a lower and upper bound of 0, effectively fixing that variable at 0.
ILOG CPLEX resets the lower bound to -∞ only if the upper bound is less than 0. A warning message is issued when this exception is encountered.

More than one bound vector may exist. The name of each bound vector appears in Field 2. However, only the first bound vector is selected when a problem is read. Additional bound vectors are discarded.

Complete Example of MPS File Format

```
NAME          example2.mps
ROWS
 N  obj
 L  c1
 L  c2
COLUMNS
    x1        obj                 -1   c1                  -1
    x1        c2                   1
    x2        obj                 -2   c1                   1
    x2        c2                  -3
    x3        obj                 -3   c1                   1
    x3        c2                   1
RHS
    rhs       c1                  20   c2                  30
BOUNDS
 UP BOUND     x1                  40
ENDATA
```

Special Records in MPS Files: ILOG CPLEX Extensions

ILOG CPLEX extends the MPS standard in several ways. The following sections document these extensions:

- *Objective Sense and Name in MPS Files* on page 20
- *Integer Variables in MPS Files* on page 21
- *Special Ordered Sets (SOS) in MPS Files* on page 23
- *Quadratic Objective Information in MPS Files* on page 25
- *Quadratically Constrained Programs (QCP) in MPS Files* on page 27
- *Indicator Constraints in MPS Files* on page 27
- *User Defined Cuts in MPS Files* on page 28
- *Lazy Constraints in MPS Files* on page 29

Objective Sense and Name in MPS Files

ILOG CPLEX extends the MPS standard by allowing two additional sections: `OBJSEN` and `OBJNAME`. They may be specified after the `NAME` section. `OBJSEN` sets the objective function sense, and `OBJNAME` selects an objective function from among the free rows within the file. If neither of these sections appears in the MPS file, ILOG CPLEX assumes that the problem is a minimization and that the objective function is the first free row encountered in the `ROWS` section. If these options are used, they must appear in order and as the first and second sections after the `NAME` section. The values for `OBJSENSE` can be `MAX` or `MIN`.

Here is an example of these optional sections:

```
NAME          example.mps
OBJSENSE
  MAX
OBJNAME
  rowname
```

Integer Variables in MPS Files

If you use the ILOG CPLEX mixed integer optimizer, then you may restrict any or all variables to integer values. ILOG CPLEX accepts two commonly used ways of extending the MPS file format to include integer variables: in the `COLUMNS` section or in the `BOUNDS` section.

In the first way, integer variables are identified within the `COLUMNS` section of the MPS file by marker lines. A marker line is placed at the beginning and end of a range of integer variables. Multiple sets of marker lines are allowed. Integer marker lines have a field format consisting of Fields 2 through 4.

Field 2: Marker name

Field 3: `'MARKER'` (including the single quotation marks)

Field 4: Keyword `'INTORG'` and `'INTEND'` to mark beginning and end respectively (including the single quotation marks)

Fields 5 and 6 are ignored.

The marker name must differ from the preceding and succeeding column names.

If no bounds are specified for the variables within markers, bounds of 0 (zero) and 1 (one) are assumed.

In the following example, column x4 is an integer variable and looks like this in the `COLUMNS` section of an MPS file, according to this first way of treating integer variables:

```
NAME
ROWS
 N  obj
 L  c1
 L  c2
 E  c3
COLUMNS
    x1        obj                 -1   c1                  -1
    x1        c2                   1
    x2        obj                 -2   c1                   1
    x2        c2                  -3   c3                   1
    x3        obj                 -3   c1                   1
    x3        c2                   1
    MARK0000  'MARKER'                 'INTORG'
    x4        obj                 -1   c1                  10
    x4        c3                -3.5
    MARK0001  'MARKER'                 'INTEND'
RHS
    rhs       c1                  20   c2                  30
BOUNDS
 UP BOUND     x1                  40
 LO BOUND     x4                   2
 UP BOUND     x4                   3
ENDATA
```

In the second way of treating integer variables, integer variables are declared in the `BOUNDS` section with special bound types in Field 1. The acceptable special bound types appear in Table 4.

Table 4 *Special bound types for handling integer variables in MPS files*

Type	Purpose	Special Considerations
`BV`	Binary variable	Field 4 must be 1.0 or blank
`LI`	Integer lower bound	Field 4 is the lower bound value and must be an integer
`SC`	Semi-continuous variable	Field 4 is the upper bound and must be specified
`UI`	Integer upper bound	Field 4 is the upper bound value and must be an integer

To specify general integers with no upper bounds, use `LI` with the value 0.0.

For example, column x4 is an integer variable declared in the `BOUNDS` section of an MPS file, according to this second way of treating integer variables:

```
NAME
ROWS
 N  obj
 L  c1
 L  c2
 E  c3
COLUMNS
    x1        obj                 -1   c1                 -1
    x1        c2                   1
    x2        obj                 -2   c1                  1
    x2        c2                  -3   c3                  1
    x3        obj                 -3   c1                  1
    x3        c2                   1
    x4        obj                 -1   c1                 10
    x4        c3                -3.5
RHS
    rhs       c1                  20   c2                 30
BOUNDS
 UP BOUND     x1                  40
 LI BOUND     x4                   2
 UI BOUND     x4                   3
ENDATA
```

Special Ordered Sets (SOS) in MPS Files

If you use the ILOG CPLEX mixed integer optimizer (that is, the MIP optimizer), then you may define special ordered sets (SOS) in MPS format.

The convention for SOS uses set declaration lines and member declaration lines, both of which begin in column 2 or beyond. In a set declaration line, columns 2 and 3 specify S1 or S2. Optionally, the name of a set is specified in column 4. In a member declaration line, column 5 or beyond specifies a variable name. Note that in an MPS file, the SOS section must follow the BOUNDS section.

If weighting information is to be provided, it is after the member name in a member declaration line.

In the following example, an SOS section is placed after the BOUNDS section:

```
NAME
ROWS
 N  obj
 L  c1
 L  c2
 E  c3
COLUMNS
    x1        obj                -1   c1                 -1
    x1        c2                  1
    x2        obj                -2   c1                  1
    x2        c2                 -3   c3                  1
    x3        obj                -3   c1                  1
    x3        c2                  1
    x4        obj                -1   c1                 10
    x4        c3               -3.5
RHS
    rhs       c1                 20   c2                 30
BOUNDS
 UP BOUND     x1                 40
 LI BOUND     x4                  2
 UI BOUND     x4                  3
SOS
 S1 set1
    x1                 10000
    x2                 20000
    x4                 40000
    x5                 50000
ENDATA
```

'MARKER' Lines for SOS in MPS Files

`'MARKER'` lines are used to delimit SOS in the `COLUMNS` section of an MPS file, much like using integer markers. (The single quotation mark before and after the term is necessary.) The names of the sets are specified in the second field, starting in column 4 or beyond. Names of sets must be unique. The `'MARKER'` lines must come in pairs of an `'SOSORG'` and `'SOSEND`' surrounding the columns that are in the SOS. Optionally, in Field 1 of a `'MARKER' . . . 'SOSORG'` line, either S1 or S2 may be specified to indicate the type of the SOS. An SOS `'MARKER'` line without an S1 or S2 indicator is assumed to denote an S1 set. Members of an SOS may or may not be integer or binary variables.

There is no requirement that there be a constraint that all members of an SOS sum to 1.0 (nor is any such constraint implicit). However, providing such a constraint in your formulation may be desirable as it may strengthen the LP relaxation of the mixed integer problem, as for example in the case of an S1 set consisting of binary variables.

In the following example, the excerpt from the COLUMNS section of an MPS file defines an SOS Type 1 set consisting of x5 and x6. which may be continuous or integer variables.

```
 S1 NAME1      ‘MARKER’                  ‘SOSORG’
    x5         obj              -9       c1      5
    x5         c2               3
    x6         obj              -6       c1       8
    x6         c3               -4.5
    NAME2      ‘MARKER’                  ‘SOSEND’
```

The SOS `'MARKER'` lines can appear between integer `'MARKER'` lines (if all members of the SOS are integer), or integer `'MARKER'` lines can appear between SOS `'MARKER'` lines (if some members of the SOS are non-integer).

The MARKER format cannot accommodate overlapping SOSs. That is, a variable cannot be a member of two special ordered sets. Overlapping SOSs can, however, be specified by the ILOG CPLEX SOS format, documented in *Special Ordered Sets (SOS) in MPS Files* on page 23.

REFROW Section for SOS in MPS Files

A `REFROW` section may be included immediately before the `ROWS` section. It consists of exactly one record line with the name of the reference row starting in Field 2. The specified row must also be defined in the `ROWS` section. The nonzeros of the reference row are used as weights within an SOS. All weights within one SOS must be unique values. A `REFROW` section is optional; if no reference row is specified, the weighting values 1, . . . , n is given to the n members of an SOS in the order in which they are read. In other words, without specific reference row information, it is assumed that the user has ordered the SOS variables in ascending order with respect to some relevant criterion (for example, in importance, capacity, objective weighting, or cost).

Quadratic Objective Information in MPS Files

If you use the ILOG CPLEX barrier optimizer for quadratic programming problems (QPs), then you can specify quadratic objective coefficients in MPS format in a `QMATRIX` section.

Following the `BOUNDS` section, a `QMATRIX` section may be specified. Each line of this section defines one nonzero coefficient of the matrix ***Q***. Each line should contain two variable names (which must have been specified in the `COLUMNS` section) in Fields 2 and 3, followed by a nonzero coefficient value in Field 4. For each off-diagonal coefficient, two lines must appear: one for the lower triangular element, and one for the upper triangular element. ILOG CPLEX evaluates the quadratic part of the objective function as 0.5 x'***Q***x, when the coefficients of ***Q*** are specified in an MPS file.

For example, consider the following problem:

Minimize

$$a + b + 1/2 \quad (a^2 + 4ab + 7b^2)$$

subject to

$$a + b \geq 10$$

$$a, \quad b \geq 0$$

In MPS format, you may enter the problem in the following way:

```
NAME          problem
ROWS
 N  obj
 G  c1
COLUMNS
    a         obj                   1   c1                    1
    b         obj                   1   c1                    1
RHS
    rhs       c1                   10
QMATRIX
    a         a                     1
    a         b                     2
    b         a                     2
    b         b                     7
ENDATA
```

You can also enter the quadratic objective coefficients by using a QUADOBJ section. In this format, only the upper diagonal elements of the Q matrix are entered. For the same example, the input with a QUADOBJ section looks like this:

```
NAME          problem
ROWS
 N  obj
 G  c1
COLUMNS
    a         obj                   1   c1                    1
    b         obj                   1   c1                    1
RHS
    rhs       c1                   10
QUADOBJ
    a         a                     1
    a         b                     2
    b         b                     7
ENDATA
```

If you have a model with quadratic objective information in MPS format in a `QUADOBJ` section of the following form, you do not have to convert your file in order for ILOG CPLEX to make use of it.

```
varname1 varname2 value
```

ILOG CPLEX can read that file and interpret the `QUADOBJ` section correctly. However, the MPS file writers of ILOG CPLEX do not produce a `QUADOBJ` section themselves. Instead, they produce a `QMATRIX` section, as explained here.

Quadratically Constrained Programs (QCP) in MPS Files

As explained in the *ILOG CPLEX User's Manual* in *Solving Problems with Quadratic Constraints (QCP)* on page 225, ILOG CPLEX can solve problems with quadratic terms among the constraints if the Q matrix for the quadratic term is positive semi-definite and the quadratic function defines a convex region. ILOG CPLEX has extended the MPS format to accommodate QCP models.

The quadratic constraints of such a model are listed in the `ROWS` section, and their linear coefficients appear in the `COLUMNS` section, just the same as coefficients from the linear constraints.

The quadratic terms go in `QCMATRIX` sections, one `QCMATRIX` per quadratic constraint. `QCMATRIX` sections appear after the optional `SOS` section. They may appear either after or before the `QMATRIX` (objective) section.

The name of the constraint appears on the same line after `QCMATRIX`.

The quadratic terms of the quadratic expression must be given as a symmetric matrix. That is, if there is an entry for Q_{ij}, then there must be an identical entry for Q_{ji} when i is not equal to j. This requirement is the same as for the `QMATRIX` section, where any quadratic terms in the objective function are declared. The formats of the Q parts are the same.

Indicator Constraints in MPS Files

Indicator constraints provide a way for you to express relations among variables by identifying a binary (0-1) variable to control whether or not a given constraint is active. ILOG CPLEX has extended the MPS format to express indicator constraints. The constraints to be controlled by the binary variable are listed in the ROWS section; their linear coefficients appear in the COLUMNS section (that is, the same as coefficients from linear constraints). Only rows of types E, L, and G may be part of indicator constraints. In other words, a row of type N cannot appear as a constraint controlled by a binary variable in this sense (that is, an indicator constraint).

The binary variables that control the linear constraints are specified in the BOUNDS section or with MARKER lines (that is, like any other binary variable). The relationship between the binary variables and the constraints they control is specified in the INDICATORS section. The INDICATORS section follows any quadratic constraint section and any quadratic objective section. Each line of the INDICATORS section has a type field starting in column 2 or beyond; the type must be "IF" followed by the name of the row of the indicator constraint, the name of the binary variable, and finally the value 0 (zero) or 1 (one) to indicate when the constraint should be active.

Rows that appear in the INDICATORS section cannot be ranged rows. In other words, a row that appears in the RANGES section cannot appear also in the INDICATORS section.

Here is an example of an INDICATORS section:

```
NAME          ind1.mps
ROWS
 N  obj
 L  row2
 L  row4
 E  row1
 E  row3
COLUMNS
    x         obj                            -1
    x         row2                            1
    x         row4                            1
    x         row1                            1
    y         row4                            1
    z         row4                            1
    z         row3                            1
RHS
    rhs       row2                           10
    rhs       row4                           15
BOUNDS
 UI bnd       y                               1
INDICATORS
 IF row1      y                               1
 IF row3      y                               0
ENDATA
```

That declaration represents the following model:

```
Minimize
 obj: - x
Subject To
 row2: x <= 10
 row4: x + y + z <= 15
 row1: y = 1 -> x  = 0
 row3: y = 0 -> z  = 0
Bounds
 0 <= y <= 1
Binaries
 y
End
```

User Defined Cuts in MPS Files

The advanced feature user defined cuts can be declared in a special section following the `ROWS` section. The title of this section is `USERCUTS`. The order of sections must be `ROWS` `USERCUTS`. The format of the `USERCUTS` section is the same as the format of the `ROWS` section with this exception: the type must be one of E, L, or G; the row must not be ranged. For more information about user defined cuts, see *User-Cut and Lazy-Constraint Pools* on page 377 in the *ILOG CPLEX User's Manual*.

Lazy Constraints in MPS Files

The advanced feature lazy constraints can be declared in a special section following the `ROWS` and `USERCUTS` sections. The title of this section is `LAZYCONS`. The order of sections must be `ROWS USERCUTS LAZYCONS`. The format of the `LAZYCONS` section is the same as the format of the `ROWS` section with this exception: the type must be one of E, L, or G; the row must not be ranged. For more information about lazy constraints and an example of an MPS file extended to include them, see *User-Cut and Lazy-Constraint Pools* on page 377 in the *ILOG CPLEX User's Manual*.

NET File Format

The NET file format is an ASCII file format specific to ILOG CPLEX for network-flow problems. It is the recommended file format for representing pure network problems within CPLEX. This format is supported by Concert Technology, by the Callable Library, and by the Interactive Optimizer. In particular, it works with `CPXNETptr` objects (not `CPXLPptr` objects).

Comments

This is a free-format file; that is, line breaks or column positions are irrelevant to the interpretation of the file. The only exceptions to this convention are comments: anything from a backslash (\) character to the end of a line is a comment and does not contribute to the network specified by the file. Comments are allowed anywhere in the file.

Keywords

The NET format recognizes the following keywords in a file:

- `MAXIMIZE`
- `MINIMIZE`
- `NETWORK`
- `ENDNETWORK`
- `SUPPLY`
- `DEMAND`
- `ARCS`
- `BOUNDS`
- `OBJECTIVE`
- `INFINITY`
- `FREE`