The MeCoSim quick application developing reference guide

Purpose

This guide aims to contain a quick reference guide about how to develop MeCoSim applications. Readers of this guide are supposed to be familiar with P systems, modeling, MeCoSim, databases and (at least) also are supposed to have read the MeCoSim user manual.

MeCoSim application structure

To design a MeCoSim application, it is necessary to input some information into different tabs of the Application Configuration File, which is an MS Excel / Libre Office Calc file; we will simply call this file the MeCoSim file from now on in what concerns to this document.

The aforementioned information must be input according to the MeCoSim application structure, which is:

A MeCoSim application can handle input and output data, which is displayed into tables contained in tabs. Values for initial parameters can be obtained from input tables, and contents of output tables have to be calculated by running queries against an inner MeCoSim database. Also, charts can be obtained from output tables.

Application Info (AppInfo tab)

To define a new application in MeCoSim, the first task to accomplish is to fill in the Application Info data, concretely in the AppInfo tab of the MeCoSim file. The following fields have to be filled in that tab:

- App Id [natural, mandatory]: a natural identifier for the application. This field is deprecated and must be set to 1.
- App Name [string, mandatory]: a unique name for the application.
- Version [string, optional]: a version id for the application.
- Version Date [string, optional]: a version date for the application.
- Ec2 file path [string, mandatory]: a default path for the ec2 file. Simply set it to *C:*\.
- P-Lingua File Path [string, mandatory]: a default path for the pli file. Simply set it to *C:*\.
- Simulation cycles [natural, mandatory]: the number of cycles to simulate.
- Simulations by cycle [natural, mandatory]: the number of simulations to perform.
- Steps by cycle [natural, mandatory]: the number of steps by cycle to execute.

- Mode [string, mandatory]: a string value specifying if the application interface is intended to model designers or end users. Allowed values are:
    - Designer: model designers.
    - End-User: end users.
- Version type [string, optional]: a string value to specify the maturity of the application version. Set this field to Release.
- Save parameters? [boolean formula, optional]: a field to specify if the initial parameters values have to be saved in a separated file.
- Show Log Console [boolean formula, optional]: a field to specify if a Log Console must be displayed in the MeCoSim Graphic User Interface (GUI).

Tabs hierarchy (TabsHierarchy tab)

In MeCoSim, an application is composed of a hierarchical structure (tree) of tabs; the root node of the tree corresponds to MeCoSim; this tab is defined by default.

There exists a parent-child relationship between tabs according to the following rules:

a) A tab can have zero, one, or more child tabs associated to it.
b) If a tab has zero child tabs, that tab is called a 'leaf' tab.
c) The set of child tabs of a given parent tab is enumerable, so every child tab is assigned a 'place order' according to the order in which it is defined in the MeCoSim file.
d) The first element of the tree to be defined in the MeCoSim file is always a child tab of the implicit tab representing MeCoSim; this first tab represents the application that is being defined in the MeCoSim file; this tab is called the 'application tab' and is mandatory.
e) Any other child tab of the MeCoSim tab is 'invisible'; its contents exists but are not displayed in the MeCoSim GUI.

To define a new tab into the aforementioned tree structure, some information has to be provided in the MeCoSim file, concretely in the TabsHierarchy tab (do not be confused here careful reader: a tab in the MeCoSim file is an Excel/Calc tab like TabsHierarchy, and in that tab you can define tabs for the MeCoSim tree structure). When defining a new tab, the following fields have to be filled in the TabsHierarchy tab of the MeCoSim file:

- Tab Id [natural, mandatory]: a unique correlative natural identifier for the tab, according to the following rules:
    - MeCoSim tab has id = 0.
    - The application tab has id = 1.
- Tab Name [string, required]: a unique name for the tab.
- Tab Parent Id [natural, required]: parent tab identifier.

Tables (TablesConfig tab)

Tables are not mandatory, but in order to define a table, it must be associated to a tab, according to the following rules:

a) A table can only be associated to a leaf tab.
b) A table cannot be associated directly to the application tab; consequently, the application tab must have at least a child leaf tab where the table can be placed.
c) If a table is associated directly to an invisible leaf tab, or some ancestor of that leaf tab is invisible, then the table is also invisible in the MeCoSim GUI.

There exist two kinds of tables:

a) Input tables: the MeCoSim user can place some required input data to run the simulation into input tables; this data is converted into initial parameters values automatically by MeCoSim prior to the simulation execution.
b) Output tables: the simulation results can be displayed in output tables in the MeCoSim GUI; additionally, charts can be created from output tables.

When defining a new table, the following fields have to be filled in the TablesConfig tab of the MeCoSim file:

- Table Id [natural, required]: a unique correlative natural identifier for the table so that the first defined table has id = 1.
- Table Name [string, required]: a unique name for the table.
- Tab Id [natural, required]: the Tab Id corresponding to the leaf tab defined in TabsHierarchy where the table has to be placed.
- Columns [natural, required]: columns number for the table, equal or greater than 1
- Init Rows [natural, required]: initial rows number for the table, equal or greater than 1, according to the following recommendations:
    - For input tables, its value should be calculated beforehand and then fixed in the corresponding Init Rows field.
    - For output tables, any value is valid, so that the value is typically set to 1.
- Save To File [boolean formula, required]: this field specifies if the data contained in the table has to be saved into the scenario file.
- Input / Output [string, required]: a string value to specify if the table is an input or output one; the possible values for this field are:
    - *Input*: for input tables.
    - *Output*: for output tables.
- Output Graphic [string, optional]: a string value to specify if a chart has to be created from the output table data, according to the following rules:
    - This field requires the Input / Output field be set to *Output*.
    - The kind of chart that can be created from an output table depends on the table columns number, with only 2, 3 or 4 columns number being valid when defining a chart.
    - For each columns number there exists a default chart than can be obtained placing the value Default in the Output Graphic field.

- if the *Default* value is not placed in the Output Graphic field, then the name of the chart to be obtained must be explicity given, according to the following table (the red column names specifies the name of the chart and * means default chart for the corresponding columns number; detailed information about charts can be found into its specific section, see below):

Columns number	PieChart	LineChart	BarChart	StackedBarChart	MultiLineChart	MultiBarChart
2	Yes (*)	No	No	No	No	No
3	No	Yes (*)	Yes	No	No	No
4	No	No	No	Yes (*)	Yes	Yes

- External View [boolean formula, optional]: this field specifies if the calculated chart from the output table has to be displayed also in a separate window (by default charts display in the MeCoSim window), according to the following rules:
  - This field requires the Input / Output field be set to *Output*.
  - This field requires the Output Graphic field be set to a valid value.
  - The possible values for the External View field are:
    - *No value*: default behavior (equals to FALSE); do not use this, it is always recommended to define a value explicitly.
    - *FALSE*: do not use a separate window.
    - *TRUE*: use a separate window.
- Row addition [boolean formula, optional]: this field specifies if the defined table can be modified by adding new rows into it. Default value for this field is *FALSE*. Setting this value to *TRUE* enables the corresponding functionality in the MeCoSim GUI, which is useful for instance when reusing existing ec2 files containing information for less rows than the ones defined in the MeCoSim file.
- Row deletion [boolean formula, optional]: this field specifies if the defined table can be modified by erasing existing rows from it. Default value for this field is *FALSE*. Setting this value to *TRUE* enables the corresponding functionality in the MeCoSim GUI, which is useful for instance when reusing existing ec2 files containing information for more rows than the ones defined in the MeCoSim file.

Table Configuration (TC_XX tabs)

For every specified table in the TablesConfig tab, a new tab has to be created in the MeCoSim file by the application designer (that means you, careful reader of this document, and please remember again avoiding confusion with the term ‘tab’). This tab is called the ‘table config’ tab for the aforementioned table and it is used to specify information regarding to columns of that table (TC stands for just ‘table config’).

The name of a ‘table config’ tab for a given table is specified according to the following rules:

a) Get the table name from the Table Name field in the TablesConfig tab of the MeCoSim file, for instance, let's say that this name is just *NAME*.
b) Create a new tab for the table with the name *TC_NAME*.

For a given 'table config' tab regarding to a table defined in the TablesConfig tab of the MeCoSim file, information related to the table columns has to be specified, according to the following rules:

- Get the table columns number from the TablesConfig tab; let's say it is N.
- For each column (from 1 to N), in the 'table config' tab, fill in the following fields:
    - Column Id [natural, required]: a unique correlative natural identifier for the column so that the first defined column has id = 1.
    - Column Name [string, required]: a unique name for the column.
    - Default Value [string, optional]: a default value for the column cells.
    - Editable [boolean formula, required]: this field specifies if the column cells can be changed in the MeCoSim GUI; set it always to *TRUE*.
    - Tooltip [string, optional]: this field specifies the caption that has to be shown when the mouse hovers over a column cell in the MeCoSim GUI, so that the text reads: 'tooltip = cell value'; a good choice for a tooltip field is to use just the column name.
    - Graphic Role [string, optional]: when a chart is obtained from an output table, this field specifies the graphic role of the column when obtaining the chart; this means f.i. to specify which column defines the X axis values and which one defines de Y axis values, when plotting data in a bi-dimensional (x, y) chart; the Graphic Role is specified according to the following rules:
        - The Graphic Role field is only required when the output table is being used to generate a chart.
        - In this case, each column belonging to the output table must be assigned a different Graphic Role value, so that empty Graphic Roles values are not allowed.
        - The only valid values for the Graphic Role are: *Category*, *Subcategory*, *Series* and *Data*.
        - The column that is assigned the *Data* Graphic Role must contain numerical values.
        - Depending of the output table columns number, only the following Graphic Roles assignments are valid:
            - 2 columns: *Category*, *Data*.
            - 3 columns: *Category*, *Series*, *Data*.
            - 4 columns: *Category*, *Subcategory*, *Series*, *Data*.
        - The meaning of the Graphic Roles assignments for the different charts follows (dear careful reader, at this point maybe it is recommended to have a quick look at the charts specific section in this document):
            - PieChart
                - Category: X axis values.
                - Data: Y axis values.

- LineChart / BarChart
    - Series: series values.
    - Category: X axis values.
    - Data: Y axis values.
- StackedBarChart
    - Series: series values.
    - Category: X axis values.
    - Subcategory: sub X axis values.
    - Data: Y axis values.
- MultiLineChart / MultiBarChart
    - Subcategory: multi-chart generating values.
    - Series: series values.
    - Category: X axis values.
    - Data: Y axis values.

Charts

In this section, MeCoSim generated charts are considered in detail. Before reviewing the aforementioned chart types, the following concepts are introduced as a matter of convenience:

- Data series: a group of related data entries; for instance if available data correspond to average temperatures in 50 different cities of 17 different regions in 12 different months of a year, data series can be identified as follows:
    - 50 different data series, each one corresponding to average temperatures per month for a given city.
    - 17 different data series, each one corresponding to average temperatures per month for a given region.
    - 12 different data series, each one corresponding to average temperatures per city for a given month.
    - 12 different data series, each one corresponding to average temperatures per region for a given month.
- Categorical data: a grouping of data into discrete groups; for instance, in the previous example, different groups can be identified:
    - cities
    - regions
    - months
    - for temperatures: grouping into level: cold, medium, hot (calculated grouping); in this case, cold, medium and hot are categories and concrete temperatures per level define subcategories
    - for months: grouping into seasons (calculated grouping); in this case, seasons are categories and the concrete months per season define subcategories

At this point let's introduce the chart types mentioned above (while reading this section it is recommended to have a look at the generated charts by the Broadcasting application).

- PieChart
  - Pie charts (or circle graphs) are used to show percentages; the circle of pie charts represents 100%; the circle is subdivided into slices representing data values so that size of each slice shows what part of the 100% it represents.
  - Additional info can be found at: http://office.microsoft.com/en-us/excel-help/available-chart-types-HA010342187.aspx?CTT=1#BMpiecharts
- LineChart
  - Line charts (or line graphs) can be used to plot changes in numerical variables usually over time or over a set of variables which values change over time (like temperature, pressure, etc.).
  - Line charts have a vertical axis (Y axis) and a horizontal axis (X axis), so that numerical variables values are plotted in the Y axis and time-changing variables values are plotted in the X axis; time-changing variables values defines categories (and subcategories if required).
  - Each pair (x, y) is plotted with a dot; dots are often connected by a polygonal line.
  - Line charts also accept plotting several data series in the same chart, that is, different sets of pairs (x, y) can be plotted simultaneously; in this case, each set of pairs (x, y) is said to correspond to a variable value s (the series variable) belonging to another set of data, so that the tuples (x, y, s) are plotted in the following way: for each data series, its pairs (x, y) are plotted independently (with respect to other data series) using dots and connecting lines if required, and finally all the dots and connecting lines are plotted together in the same chart using different colors, line styles, etc.
  - Additional info can be found at: http://office.microsoft.com/en-us/excel-help/available-chart-types-HA010342187.aspx?CTT=1#BMpiecharts
- BarChart
  - Bar charts (or bar graphs) are charts with rectangular bars with lengths proportional to the values that they represent; the bars can be plotted vertically or horizontally; a vertical bar chart is sometimes called a column bar chart.
  - Bar charts provide a visual presentation of categorical data; in a column bar chart, categories appear along the horizontal axis; the height of the bar corresponds to the value of each category (which is numerical).
  - MeCoSim plots vertical (column) bar charts, but not horizontal bar charts.
  - Bar charts also accept plotting several data series in the same chart in a similar way to line charts.
  - Additional info can be found at: http://office.microsoft.com/en-us/excel-help/available-chart-types-HA010342187.aspx?CTT=1#BMlinecharts
- StackedBarChart
  - Stacked bar charts are a variant of bar charts so that they show the relationship of individual items to the whole.

    - Stacked column charts can be used when there exist multiple data series and for a given category (and subcategory if required) the total of the data values for the different series has to be calculated and emphasized.
    - Additional info can be found at: http://office.microsoft.com/en-us/excel-help/available-chart-types-HA010342187.aspx?CTT=1#BMcolumncharts
- MultiLineChart
    - A multi-line chart is composed of a set of line charts that MeCoSim can automatically generate.
    - In order to do this, data is required to be distributed along categories, subcategories and series.
    - For each subcategory value x', a line chart is generated allowing to plot the different tuples (x, y, s) (category, data, series).
- MultiBarChart
    - A multi-bar chart is composed of a set of bar charts that MeCoSim can automatically generate in similar way to multi-line charts.

Parameters (SimulationParams tab)

Models can be parametric or not. When a model is parametric, there exist initial parameters in the model whose values have to be provided each time a simulation is performed. Values for initial parameters can be provided as global variables in the model file itself, but it is better to set the values in the MeCoSim file, concretely in the SimulationParams tab.

For each parameter the following fields have to be filled:

- Param Name [identifier, required]: the parameter name. It must be a valid P-Lingua identifier.
- Param Value [numeric expression, required]: the parameter value. It must be a valid numeric expression, which is composed by numbers, mathematical operators, input tables' data access, predefined functions and previous defined parameters.
- Index 1, Index 2, Index 3, Index 4 [index, optional]: up to 4 indexes can be optionally defined. In order to define an index, all the previous indexes must be also defined. Each index must be a valid numeric expression or a range with the form: [initial_value..final_value], satisfying that initial_value <= final_value.

Regarding to the way in which MeCoSim generates parameters and their corresponding values (assuming that i,j,k,l are the values for indexes 1 to 4 and no ranges are used), we have:

- If not index is provided: name = value;
- Index 1 provided: name{i} = value;
- Index 1, Index 2 provided: name{i,j} = value;
- Index 1, Index 2, Index 3 provided: name{i,j,k} = value;
- Index 1, Index 2, Index 3, Index 4 provided: name{i,j,k,l} = value;

When generating parameters initial values, if some index t (t = 1,2,3,4) is defined as a range with the form [initial_t..final_t], then a loop is performed in the following way:

- The current value for the index t goes from initial_t to final_t with step 1.
- In each step, the following assignation is generated: name{i,..,t,..,l} = expr{i,..,t,..,l}, with expr{i,..,t,..,l} being a valid numeric expression where the current value on index t can be accessed with the expression $t$.

As different indexes can correspond to different ranges, nested loops can be performed.

Regarding to valid numerical expressions, they can be formed in the following way:

- Numbers (when a number is just the value of a parameter in the Param Value field, the number should be entered preceded by a quote symbol to prevent parsing errors).
- P-Lingua operators like *, /, %, +, -, ^.
- P-Lingua exponential notation.
- Special expressions and functions.

Special expressions and functions are:

Expression / Function	Meaning
@cycles	number of cycles to simulate
@sims	number of simulations per cycle
@steps	number of steps per cycle
<@r,ID>	number of rows of table with id ID
<@c,ID>	number of columns of table with id ID
<ID,i,j>	returns value in the cell (i,j) of table with id ID
<@random, expr>	returns a random value between 1 and expr according to an uniform random distribution
<@max,expr1,expr2>	returns max value between expr1 and expr2
<@func,min,expr1,expr2>	returns min value between expr1 and expr2
<@ncdf,mu,sigma,x>	returns value according to a normal cumulative distribution function of parameters mu, sigma, x
<@func,ceil,expr>	returns ceil of exp
<@func,floor,expr>	returns floor of expr
<@func,abs,expr>	returns absolute value of expr
<@func,exp,expr>	returns exponential base e of expr
<@func,log,expr>	returns logarithm base e of expr
<@func,eq,expr1,expr2>	returns 1 if expr1 = expr2; 0 otherwise
<@func,g,expr1,expr2>	returns 1 if expr1 > expr2; 0 otherwise
<@func,if,expr,exp-true,exp-false>	returns expr-true if expr > 0.0; expr-true otherwise

Text Parameters (TextParams tab)

Text Parameters are used for some advances features. Their current status is experimental, so they are not covered in this guide. For additional information about text parameters, please contact the MeCoSim developing team (lvalencia@us.es).

Getting Simulation Results (SimulationResults tab)

In MeCoSim, simulation results are stored in an inner database. In what follows, we will call this database the MeCoSim database. Simulation results are stored in a table inside that database. In what follows we will call this table the MeCoSim results table. The structure of this table is:

Column name	Data type (Java)	Meaning
simulation	Integer	simulation number
cycle	Integer	cycle number (per simulation)
step	Integer	step number (per cycle)
membraneID	Integer	P-Lingua membraneID
environmentID	String	P-Lingua environmentID
labelID	String	P-Lingua labelID
object	String	object name
multiplicity	Integer	object multiplicity

Each row of the MeCoSim results table stores the *multiplicity* of a given *object* inside a given membrane of the P system, specified by the unique identifier *membraneID*, which also has a corresponding associated pair (label, environment) represented by the columns *labelID* and *environmentID* respectively, after performing a computation *step* of a given *cycle* of a given *simulation*. Consequently, this table stores:

- Information equivalent to a full trace of a computation for the related P system, corresponding to a simulation and, moreover, for the different simulations executed by MeCoSim when dealing with probabilistic models.
- The P system initial configuration for a given simulation, stored at cycle 1, step 0.
- Only information for objects with multiplicity different from zero.
- An implicit environment value called *Environment* in the *environmentID* column for P systems models where not environment is defined (for instance, SN P systems).

When an output table is defined, its content is obtained by performing queries against the MeCoSim database table. Queries can be nested, satisfying that:

- Results of a given query can be sent to an output query and/or can be used as input data for another query.
- When a query B uses as input data results from another query A, we say that query A is the predecessor of query B or alternatively that query B is the successor of query A. This relationship can be represented as A -> B.
- When queries are successively nested, a predecessor-successor chain is formed with the form: Q1 -> Q2 -> … -> Qn.
- Multiples predecessor-successor chains can be defined.
- The leftmost predecessor of each chain is always the MeCoSim results table itself.

Queries are listed (meaning only named but not actually defined or specified) in the SimulationResults tab of the MeCoSim file. For each query the following fields have to be filled:

- Result Table Id [natural, required]: a unique correlative natural identifier for the query so that the first defined query has id = 1.
- Result Table Name [string, required]: a unique name for the query.
- Table Id [natural, required]: an output table identifier (as defined in TablesConfig tab) where the results of the query will be loaded into; if zero is specified in this field, the query results are not loaded into any output table.
- Referred Table Id [natural, required]: The predecessor query of the query that it has being defined; if the predecessor query is the MeCoSim results table, then zero must be specified in this field.

Building queries (SD_XX tabs)

For every query listed in the SimulationResults tab, a new tab has to be created in the MeCoSim file by the application designer (that means you careful reader, and please remember again avoiding confusion with the term 'tab'). This tab is called the 'query specification' tab for the aforementioned query and is used to actually build the query.

The name of a 'query specification' tab for a given table is specified according to the following rules:

a) Get the query name from the Result Table Name field in the SimulationResults tab of the MeCoSim file, for instance, let's say that this name is just *NAME*.
b) Create a new tab for the query with the name *SD_NAME*.

For a given 'query specification' tab regarding to a query listed in the SimulationResults tab of the MeCoSim file, some rows have to be filled in order to actually build the query. Each row allows defining a part of the query in a SQL-like style, so that some SQL basic background is advisable when building queries. Rows are linked to criteria. Each one of the rows / criteria allows getting data, filtering data or grouping data.

For each criterion the following fields have to be filled:

- Criteria Id [natural, required]: a unique correlative natural identifier for the criterion so that the first defined criterion has id = 1.
- Select/Where/Group [string, required]: a string field specifying the criterion type (and consequently its role/purpose in the query). Allowed criterion types are:
  - Select: the criterion corresponds to a value computed out from a column from the MeCoSim table or a column from a preceding query (possibly by using a formula) that is populated as a result (column) of the current query.
  - Auxiliary: similar to the previous criterion type, but in this case the criterion is not populated as a result of the query; it is used to compute other criterions instead.

    - WhereAux: the criterion is used to specify part of a condition to filter data when getting query results; WhereAux criteria are linked to build a final Where criterion.
    - Where: the criterion is used to specify the global filter when getting query results.
    - Group: the criterion is used to specify a part of a data grouping when getting query results (and also for setting an ordering in the results, in a similar way to the ORDER BY SQL clause).
- Criteria [string, required]: the criterion name. Valid values are:
    - A string with the name of a column in the MeCoSim table or a column in the result of the preceding query; additionally qualified names (as specified in the Qualified Name field, see below) can be used only for Group criteria.
    - The special value *formula* indicating that the criterion is computed out of an SQL formula; this value in incompatible when the field Select/Where/Group is set to *Group*.
- Formula [string, optional]: the name of the SQL formula to apply. This field only makes sense when the value *formula* is specified in the Criteria field. Some considerations regarding to the formula types follows:
    - In general, it is possible to use almost all the formulas provided by the HSQLDB database (all of them including two or less input arguments). For additional info, see: http://hsqldb.org/doc/2.0/guide/builtinfunctions-chapt.html
    - Also, it is possible to use operators like +, -, etc. as the function name.
    - When the field Select/Where/Group is set to *WhereAux* or *Where* only logical functions / operators *NOT*, *AND*, *OR* can be used.
- Referred Criteria Id [natural, optional]: a natural number corresponding to a previous defined criterion id. The value of this criterion will be used as a first argument to the formula which name is specified in the Criteria field, so that the Referred Criteria Id field only makes sense when the value *formula* is specified in the Criteria field. In order to reference a previous defined criterion, that criterion must be defined as an Auxiliary criterion type.
- Argument [string, optional]: the value of this field will be used as a second argument to the formula which name is specified in the Criteria field, so that the Referred Criteria Id field only makes sense when the value *formula* is specified in the Criteria field, and the specified formula in Formula field takes two arguments. Optionally, the argument order can be swapped when the Where condition field is set to *Reverse*. Some considerations regarding to this field are:
    - In general, this field corresponds to:
        - A string representing an Integer or String literal, so that this value has to be converted to the proper data type by MeCoSim; this requires the Where/Select condition type field to be set to the corresponding type.
        - A string representing String constants that do not have to be converted by MeCoSim because they are passed as such directly to the related function; this requires the Where/Select condition type field be set to *DirectString* (a function that uses this feature is CONVERT).

        - A natural number corresponding to a previous defined criterion id; this requires the Where/Select condition type field be set to *Reference*.
    - When the field Select/Where/Group is set to *WhereAux* or *Where* the Argument field always corresponds to a previous defined criterion id.
- Qualified Name [string, optional]: a string value specifying an alias for a calculated value from a formula, similarly to the AS clause in SQL. This field only makes sense when the value *formula* is specified in the Criteria field. Also, this field must be used when a formula result has to be used in a Group type criterion.
- Where/Select condition type [string, optional]: a string field to define a data type related to the defined criterion. The following rules apply:
    - When the Select/Where/Group field is set to the value *Select* or *Auxiliary*, this field is used to specify the data type of the second argument in a formula, so that the following cases are possible:
        - When the Criteria field is not set to the value *formula*
            - this field is not filled
        - When the Criteria field is set to the value *formula*
            - When the formula takes only one argument
                - this field is not filled
            - When the formula takes two arguments, the possible values for the field are:
                - *Integer*: the second argument is an Integer literal
                - *String*: the second argument is a String literal
                - *DirectString*: the second argument is a constant String
                - Reference: the second argument is a previous defined criterion
    - When the Select/Where/Group field is set to the value *WhereAux* or *Where*, this field is used to specify the data type of a literal involved in a simple condition, i. e. by using the = (Integer) or LIKE (String) operators, so that the following cases are possible:
        - When the Criteria field is not set to the value *formula*, the possible values for the field are:
            - *Integer*: the specified literal of Integer type (operator = will be applied in this case)
            - *String*: the specified literal is of String type (operator LIKE will be applied in this case)
        - When the Criteria field is set to the value *formula*
            - this field is not filled
    - When the Select/Where/Group field is set to the value *Group*
            - this field is not filled
- Where condition [string, optional]: a string field to specify a literal or an argument order condition according to the following situations:
    - When the Select/Where/Group field is set to the value *Select* or *Auxiliary*
        - When the Criteria field is not set to the value *formula*
            - this field is not filled
        - When the Criteria field is set to the value *formula*

        - When the formula takes only one argument
            - this field is not filled
        - When the formula takes two arguments, the possible values for the field are:
            - *Reverse*: the arguments order must be swapped
            - No value: the arguments order is not swapped
- When the Select/Where/Group field is set to the value *WhereAux* or *Where*
    - When the Criteria field is not set to the value *formula*, the possible values for this field are:
        - An integer literal (preceded by the quote symbol) when the specified column type in the Where/Select condition type field is Integer (operator = will be applied in this case)
        - A String literal (not preceded by the quote symbol) when the specified column type in the Where/Select condition type field is String (operator = will be applied in this case and special comparator characters like % are allowed)
    - When the Criteria field is set to the value *formula*
        - this field is not filled
- When the Select/Where/Group field is set to the value *Group*
    - this field is not filled

As a final remark, it is mandatory to specify a *Where* criterion when defining the first query of a query chain.