SCENARIO$

SCENARIO$ enables the definition of a specific process of input and calculations

Syntax

SCENARIO$(InputPar, GoalPar)

Arguments

- InputPar can be more than one parameter seperated by comma's and are parameters defining input. The parameters should have an @INPUTSOLUTION and @SCENARIOCHOICE attribute;
- GoalPar can be more than one goal parameter seperated by comma's and the word "SUBGOAL" (see example). GoalPar are all parameters defining calculations. The parameters should have a @SCENARIOCHOICE attribute.
  Furthermore, GoalPar parameters should be either string or Telitab parameters initiating additional calculations. Because in addition to the identifiers of the calculation steps, GoalPars are (or can be used as) result containers;

By means of defining INPUTSOLUTIONS and SUBGOALS a calculation process is designed. INPUTSOLUTIONS are arranged sets of input (defined by the Knowledge Engineer) that will be saved in the dataset of the scenario. The SUBGOALS are the results or calculations steps during the calculation process. It enables the user to stop at serveral stages of the total process.

Remarks

- A scenario should only be used in combination with an object in the Dataset containing the @DBENTRY and @DBOBJECT attribute;
- A scenario is best used in combination with the Process Manager window;
- For the definition of scanario's in addition to the SCENARIO$ function attributes in the accompanying parameters are required.
  All required attributes are:

    a. @SCENARIO, in the dataslot of the top goal parameter of the scenario (TotalCostsScenario$ in the example below), indicating that this parameter should present the scenario progress;
    b. @SCENARIOCHOICES, in the dataslot of the all parameters (input and calculation steps) that are part of the scenario (exept the top goal parameter initiating the scenario), indicating you should have a "Now" or "Later" choice and "Modify" or "Done" choice during the process;
    c. @INPUTSOLUTION, in the dataslot of the parameters defining the required input steps, indicating it is an input step;

All optional functional attributes are:

    a. @NEWNOSHOW, in the dataslot of the top goal parameter of the scenario, indicating the scenario overview will not be shown when you restart the solution;
    b. @AUTOK, in the dataslot of the parameters defining the required calculation steps, indicating that calculation should restart automatically when a solution is restarted;
    c. @TELITAB, in the dataslot of the all parameters (input or calculation steps), indicating the result is in a Telitab format;

Additional the the optional attribute the all usual attributes can be used to manipulate layout (see Documentation of knowledge). Usefull additions are:

    a. Add color to scenario steps, otherwise all steps will appear in the color defined as input). Simply select the relevant parameter and choose a "Cell color" in the Slots & Properties window;
    b. Use the @ORDER:[number] attribute to give an order to the presentation of the scenario when finished, otherwise an alphabetical order is applied;

- All parameters used in the scenario function should be of the string type (string or TeLiTab) and should have the Determined by property **USL** (see Slots & Properties);
- Parameters used in the scenario function should be in or under the "Top Goals/Undifined" node of the knowledge base;
- By placing "SUBGOAL" between the SUBGOALS (see the example below) you indicated that the next goal will use the input and results of the previous SUBGOAL. You place it behind the goal you want to reuse the solution as part your following goal. Usually you want this so it is adviced to add "SUBGOAL" between the SUBGOALS at all times.

Why use scenario's

For several reasons you may consider scenario's instead of ordinary top goals or macro's:

1. Ordening the design / engineering / calculation process. With scenario's you are able to define clear steps and let these be used only while still being able to use the reasoning mechanisme (the Quaestor intelligence of find relations to calculate parameters) to finalise without compromissing to much;
2. In combination with the Process Manager window it reduces complexity to start, continu and/or redo these processes;
3. All data requested in the INPUTSOLUTION parameters are automatically stored in the dataset selected for the process;

Please note that Qnowledge is at the moment finalising a highly improved way of ordening your design / engineering / calculation process by means of a Taxonomy approach. SeeENTITY#() function for more detail.

In the example below more details on the SCENARIO$ functionality is explained;

Examples

Take the following scenario into account:

1. Give input you know that is always required (so no reasoning requried)
2. Perform a calculation;
3. Create a report;

TotalCostsScenario$ = SCENARIO$(RequiredInput#, TotalCostsCalculation#, "SUBGOAL", ReportGeneration#)In this relation TotalCostsScenario$

is the scenario parameter, this parameter will communicate about the progress of the scenario in words like "In progress" or "Completed".

All other parameters are either INPUTSOLUTIONS or SUBGOALS:

- RequiredInput# is an INPUTSOLUTIONS;
- TotalCostsCalculation# and ReportGeneration# are SUBGOALS.

Please note that you may add as may INPUTSOLUTIONS or SUBGOALS as you like.

As mentioned, all parameters require addtional atttributes to make the scenario function work properly. Attributes are placed in the dataslot of the parameter. You can get access to the dataslot by selecting "data" in the Frame Viewer.

An example for the above input solution:
RequiredInput# = "Required input finished..." + LEFT$(CarName$ + FuelInstallation$ + + Weight$ + STR$(BuildinCosts + FuelEconomy + AmountKm + Years), 0)

By using the LEFT$ function, you force Quaestor to request the parameters inside this function for the INPUTSOLUTION.

An example for the subgoals:

TotalCostsCalculation# = TELITAB#(0, CarName$, FuelInstallation$, Weight$, BuildinCosts, FuelEconomy, AmountKm, Years, TotalCosts)

and
ReportGeneration# = TELITAB#(0, Report$)

CarName$, FuelInstallation$, Weight$, BuildinCosts, FuelEconomy, AmountKm, Years, TotalCosts and Report$ can be any input or calculation.

You see that we still use the Quaestor reasoning for each separate solution. We only pre-discribe the order in which these (3) relations have to be analysed and make a difference between relations to ask input (an you want to save in the data set) and relations that calculate/create things.

Quick links: Functions overview | Attribute overview | Constants overview | Dimensions overview