# FUNCTION

FUNCTION calls a satellite program which uses the TeLiTab data format or Solution (also in other knowledge bases) and returns an output value

# Syntax

FUNCTION Process(OutputArg, Arg,Arg1,Arg2,...,Argn)

**Arguments**

- Process is the name of the satellite program **without .EXE**.
  - If Process is provided **without** quotes (so *Process* instead of *"Process"*) the process is considered to be **a command-line executable program** located in the ..\Applic directory (see also Directory structure.
  - If Process is provided **with** quotes (so *"Process"*) the process is considered to be a **knowledge base**, possibly augmented with a Solution to be executed for a given set of input.
- OutputArg is the parameter name of the result(s)
- Arg are the required input arguments to succesfully run the process

# Remarks

1. Input is passed to the external program through the file Process.EPI (External Process Input). Output is passed from the external program to Quaestor through a file with the name Process.EPO (External Process Output)
2. The location of the satellite program should be the Applications directory (See Options>Files). The location on which the EPI/EPO files are stored is the current Report directory (See Options>Files). See also Directory structure
3. Quaestor uses FUNCTION in a similar way as any other special function. This means that it is possible to recalculate any of the input parameters if the result is known or when it is possible to draft a system of equations in which (one of) the unknowns are input parameters in one or more FUNCTION calls.
4. Convergence can not be guaranteed because this depends on the process in the function(s) (the programs and knowledge bases). This capability is very attractive since it allows the use of (existing) codes in iterative design loops.
5. For the exchange of data with satellite programs, output is read from the .EPO file per value, i.e. the FUNCTION is used as any other standard function since it generates one single value, as described above.
6. If the source code of the program is available this can be implemented with varying effort, depending on the software architecture. However, for programs for which no access to source code is available the EPI/EPO IO method is more difficult to implement since each program requires an intermediate layer in the form of a dedicated program which translates the existing IO to Telitab IO. Another problem is that the dedicated translator may not work anymore with a new release of the program, which again means additional programming effort.To enable Quaestor to run programs with their own dedicated IO, an instruction set is available which can be used to create arbitrary input files as well as instructions to parse output for specific values. The basic assumption is still that program IO is dealt with in the form of plain ASCII files, however, no specific structure or naming conventions of these files is required. In this case it is best to use a combination of the functions: GET$, PUT$, TEMPLATE$ and PARSE#.

# Examples

If more than one value has to be read from the output, it is not efficient to run the input each time again. It is more efficient to compare the current input with the latest .EPI file and to decide whether or not to rerun the program.

In this case it is more logical to assign the total output of a program run to a string parameter. This parameter represents a Telitab set which can be used as argument for a variety of parameters and Quaestor functions. In this case, a parameter has to be assigned to a specific values of the result. In example:

A = Results_of_Process#.A.3

with:

Results_of_Process# = FUNCTION Process("",Arg1,Arg2,...,Argn)

the **third** element (A.3) of the table column with header "A" (Results_of_Process#.A) is selected and assigned to parameter A.

Results_of_Process$ has become a pointer to the Telitab output set of program *Process*. <u>*Process* will only be invoked if one of the input arguments obtains a new value</u>. This is a simple and reliable method which also applies for error trapping, but the obvious disadvantage is that existing programs have to be adapted to deal with Telitab IO (reading a Process.EPI file, running and producing a Process.EPO file containg the output).

To avoid modifying programs, use a combination of the functions: GET$, PUT$, TEMPLATE$ and PARSE#. In the GET$ function the approach is described.

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