RUN

RUN action

Function

The action runs a specified external program.

Declaration

```
RUN "program name" [HIDE]  SYNC/ASYNC [paramExpr_Str] [TIMEOUT
timeOutExprStr_Int]
```

or

```
intIdent_Int := RUN "program name" [HIDE] SYNC [paramExpr_Str] [TIMEOUT
timeOutExpr_Int]
```

```
RUNEX _toExec [HIDE] SYNC/ASYNC [paramExpr_Str] [TIMEOUT
timeOutExprStr_Int]
```

or

```
intIdent_Int := RUNEX  _toExec [HIDE] SYNC [paramExpr_Str] [TIMEOUT
timeOutExpr_Int]
```

Parameters

"program name"	in	Executable program name (optionally in quotation marks).
intIdent_Int	out	Identifier - program return code.
HIDE	in	Optional parameter - hide the window of the program you want to run.
SYNC	in	Synchronous program run (waiting for termination).
ASYNC	in	Asynchronous program run.
paramExpr_Str	in	`Expression of String type, the result value of which will create parameters of the running program.`
TIMEOUT timeoutExpr_Int	in	Expression of the *Int* type -the maximum program running time [s].
_toExec	in	Identifier of the text type.

Description

The RUN action will run the program determined by text string "*program name*". RUNEX action will run the program determined by value of identifier of text type *_toExec*.
In the first type of the declaration, it is able to specify whether the program will be run synchronously (**SYNC**), or asynchronously (**ASYNC**). The second type is always asynchronous. After the program termination, its return code is assigned to the identifier of the *Int* type.

If the maximal program running time is limited by the keyword **TIMEOUT** and the program is not to be terminated in this time limit, the program is to be terminated and the program and the return value (for the first type) gets the value _ERR_TIME_OUT.

In some cases, the parameter *program name* must be created on the basis of text operations. It is recommended to use RUNEX action to run program determined by value of text identifier *_toExec.*
The same result is reached by RUN action but the program name must be an empty text string and whole command is to be declared as parameter using the expression *paramExpr_Str* (see the example below).
For **Linux/Raspberry PI** it is necessary to enter separate "program name" and parameters, e.g.
RUN "/usr/bin/ncftpget" SYNC _paramStr
If the "program name" is empty or contains parameters, the program will not start.

If RUN action is used with the return code and starting the program failed, return code (*intIdent_Int*) will have an invalid value.

Note on "wildcard" parameters for Linux/Raspberry PI
It should be noted that the expansion of asterisks and similar special characters is performed by a shell (e.g. bash). Therefore, the following command will not work:

RUNEX "/usr/bin/rm" SYNC "/tmp/*.txt"

It is necessary to create an auxiliary shell script (e.g. */opt/d2000/bin/mydel.sh*), which contains the called command and the parameter $* (in the bash syntax it represents all parameters with which the script is called), ie. the script will contain two lines:

#!/usr/bin/env bash
/usr/bin/rm $@

Do not forget to set the rights of the script so that the user under which D2000 is running (by default *d2000*) can run this script, i.e.
chmod 755 /opt/d2000/bin/mydel.sh
Such a script can be called and the shell will expand the parameters when called:

RUNEX "/opt/d2000/bin/mydel.sh" SYNC "/tmp/*.txt"

Another example for copying files to a remote server using ncftpput. The script */opt/d2000/bin/ncftpput.sh* will contain the lines:

```
#!/usr/bin/env bash
/usr/bin/ncftpput $@
```

Call from script::

```
 _parameter := "-F -u myuser -p mypassword ftpserver /some/server/path /some/local/path/*.zip"
 _retCode := RUN "/opt/d2000/bin/ncftpput.sh" HIDE SYNC _parameter TIMEOUT 60
```

Example

The following example copies the file *d:\d2000.v70\prefix\sqlback\syscfg.db* into the directory *c:\archive* using standard facilities of the operating system (shell will run the command copy).

```
 ; Copying the file

; Source file
TEXT _copySrc = "d:\d2000.v70\prefix\sqlback\syscfg.db"
; Target directory
TEXT _copyDst = " c:\archive"
; Parameters for the command copy
TEXT _switch = "/Y "
; Return code
INT _ret

_ret := RUN "cmd /C copy " SYNC _switch + _copySrc + _copyDst

; or whole command declared as expression

; _ret := RUN "" SYNC "cmd /C copy + _switch + _copySrc + _copyDst

IF _ret # _ERR_NO_ERROR THEN
 ; error
ENDIF
END
```

Related pages:

Script actions