GETARCHVAL

GETARCHVAL action

Function

Reading the value of a specified historical value in the given time.

Declaration

```
 GETARCHVAL valueIdent_Real, archIdent, timeExpression_TmA [STATUS
[isDataIdent_Bool], [archFlagsIdent_Int], [archivInstance_Int]] [VALID |
INVALID]
```

Parameters

valueIdent_Real	out	A reference to one value of the historical value, reference to an object or an item identifier of the structured variable type object (*Note:* values of object or item must be archived). **Warning:** If the parameter is the reference to an object archived several times, there is not specified which one of the historical objects is to be used.
archIdent	in	A reference to one value of the historical value, reference to an object or an item identifier of the structured variable type object.
timeExpression_TmA	in	Expression of the *Absolute* time type.
isDataIdent_Bool	out	Identifier of the *Boolean* type - attribute of successful reading from the archive, optional parameter.
archFlagsIdent_Int	out	Identifier of the *Int* type: archive flags, optional parameter.
archivInstance_Int	in	Optional identifier of the *Int* type - identification of the archive instance. If the parameter is not defined, the value 0 will replace it.

Description

The action reads values from the archive for the given time. It writes the result (value) to the variable *ValueIdent_Real*. The read value matches reading a time interval, which has identical begin time and end time, and a time step is 0.

If the parameter *archIdent* is a reference to an object of Historical value type, the action performance is described above. If the parameter is a reference to an object (not of the Historical value type) or a structured variable item that is not of the ***Object*** type, the system will attempt to find an object of the Historical value type that archives values of the object (item).
If the parameter *archIdent* is a reference to a structured variable item that is of the ***Object*** type, the item "points" to an object in the system. If the object is of the Historical value type, the action will read data from it. If it is not, the system will attempt to find an object of the Historical value type that archives values of the object.

The result of reading (value) will be assigned to the variable `valueIdent_Real`.

If the identifier *isDataIdent_Bool* is specified, the action will assign it the following value:

- **True** - data was read,
- **False** - no data.

If the identifier *isDataIdent_Bool* is not used and no data was read, the action doesn't modify the value of the identifier *valueIdent_Real*.

If the identifier `archFlagsIdent_Int` is specified, archive flags for a value, which were read, are saved into the identifier (archive flags are defined as the sum of the following constants):

- **1** - (ArcStart) - value stored into the database at the moment of the D2000 Archiv start
- **2** - (ArcStop) - value stored into the database at the moment of the D2000 Archiv stop
- **4** - (ArcBlock) - value stored into the database at the beginning of blocking of the archiving (by the stop condition of archiving configured in the D2000 CNF)
- **8** - (ArcUnBlock) - value stored into the database at the end of blocking of the archiving (by the start condition of archiving configured inthe D2000 CNF)
- **16** - (ArcDeleted) - value from the archive database that was deleted by an user. The action ignores the deleted value (as if it were not in the archive)
- **32** - (ArcUsermodify) - value in the archive that was modified by an user
- **64** - (ArcOldVal) - old value that was read from a communication station
- **128** - (ArcProcessModify) - value that was modified by a process (not by an user: ESL Script, D2000 VBApi, ...)

- **256** (ArcLoadData) - obsolete: value was obtained from OS/2 database SQL Gupta via On-line archive database import
- **512** (ArcMonoTime) - value is stored with monotonous time
- **1024** (ArcK) - value of periodic archive is generated during reading as a copy of previous value

The logical AND (&), that may be also applied into integer operands, is used to test archive flags.
For example: If the condition *archFlagsIdent_Int & 4 = 4* is true, then the flag of a value that was read from the archive database by means of the action **GETARCHVAL** is has **ArcBlock** flag (blocking of the archiving).

The value of the parameter *archivInstance_Int* defines the instance of archive which executes the request. If the parameter is not defined (or the value is 0), the active instance of archive will execute the request.

For versions newer than 21.0.70, you can specify a VALID or INVALID clause. This clause can be used to read the last valid (VALID) or invalid (INVALID) value for a given time (ignoring all following values which are not VALID or INVALID).

Note

- If the particular archive process is not running, the action generates the error _ERR_ARCHIV_NOT_RUNNING.
- If an an object of the Historical value is archived:
  - periodically, reading a value out of the period is unsuccessful
  - according to a filter, value for the required time is to be derived from the last stored value in the archive.

Example

Reading of archived value:

```
REAL _value
TIME _bt
BOOL _bIsArchData

; calculation of current minute begin
_bt := SysTime
_bt := _bt - %ModTime(_bt, 60)
 ; reading of value and active instance of archive
 GETARCHVAL _value, H.Sec, _bt STATUS _bIsArchData,,0
; has the value been read?
IF _bIsArchData THEN
; processing of value which has been read
 ENDIF
```

Related pages:

Script actions