

SMARTS Global Environment Variables

The SMARTS server system makes it possible for you to specify global environment variables for the SMARTS address space. These variables are returned to a program issuing the 'getenv' function for any given environment variable and enable a system-wide specification of any given variable. If the same variable has been set in a local process using the 'putenv' function, this is returned and the global version of the variable is ignored.

- File Requirements
 - File Processing
 - Examples
-

File Requirements

For z/OS environments, the file containing these variables

- has a record format of F, FB, or VB.
- has a valid record length. Software AG recommends a record length small enough for editors to handle. A record length of 256 is sufficient for most needs.
- is identified on the ENVIRONMENT_VARIABLES configuration parameter of SMARTS.

For VSE environments, the file containing these variables

- is a member in a VSE library.
- is identified on the ENVIRONMENT_VARIABLES configuration parameter of SMARTS.

For other environments, see the documentation for the particular Software AG application product that uses SMARTS.

File Processing

The contents of the file are processed as follows:

1. Each record in the file is read. Only one global environment variable may be specified per line.
2. The start of the variable name is the first nonblank character in the record.
3. The end of the variable name is the next '=' sign or blank found following the first nonblank.
4. If there is no '=' sign or no data follows the '=' sign, the environment variable is defined but has no value; that is, it is a null-terminated string with null as the first character.
5. Comments are allowed and are specified by an asterisk (*) in column 1.

6. To establish the value, SMARTS searches from the end of the record to find the first nonblank. The data after the equals sign to this point is treated as the variable. It is not possible to specify comments on these lines. For this reason, Software AG recommends that you not use numbered datasets such as those produced by TSO/ISPF to avoid interference with the values assigned to environment variables.
7. If the string starts with an apostrophe and ends with an apostrophe, the apostrophes are omitted in the environment variable but all data between them (including blanks, apostrophes, etc.) form part of the environment variable.
8. If the string starts with the value "X'" (that is, the character X followed by an apostrophe) and ends with an apostrophe, the data between them is treated as a hexadecimal value and must therefore be 0 to 9 or A to F. Note that 'a' to 'f' are treated as invalid hexadecimal data.
9. If the same variable name is specified more than once, the last one in the file is the active value for the variable after initialization.

Examples

Following are a number of examples of global variables:

```
MyVariable=This is my variable string
```

A request to getenv for MyVariable returns a pointer to 'This is my variable string' (without apostrophes).

```
QuotedVariable='This is my quoted variable string  '
```

A request to getenv for QuotedVariable returns a pointer to 'This is my quoted variable string ' (without the apostrophes).

```
NullVariable=
```

A request to getenv for NullVariable returns a pointer to '' (without apostrophes).

```
HexVariable=X'AABBCCDD'
```

A request to getenv for HexVariable returns a pointer to the hexadecimal value 'AABBCCDD'.

```
NotHexVariable=x'AABBCCDD'
```

A request to getenv for NotHexVariable returns a pointer to the string 'AABBCCDD' (without the apostrophes).