CODE MAGUS

strucenv: Structured Environment Variables, User Guide and API Reference Version 1

CML00066-01

Code Magus Limited (England reg. no. 4024745)
Number 6, 69 Woodstock Road
Oxford, OX2 6EY, United Kingdom
`www.codemagus.com`

Business Partner IBM™

August 16, 2016

Contents

1 Introduction

1.1 Overview

The Code Magus Limited `strucenv` library provides an interface for programs to retrieve the value of a structured environment variable (SEV). A SEV is an environment variable that resolves to a value obtained from a specific cache or store of values and is defined by a configuration file. The structure of the name of the SEV identifies it as a SEV; names the store; and lastly names the actual variable (see section 1.2 on page 2).

It is most usually called indirectly, as explained in section 4 on page 8, where an application uses and links to the Code Magus Limited `osmods` library and then calls the generic function `getenv()` to resolve the value of either a system or a structured environment variable with out having to know which it is dealing with.

However, the `strucenv` library may be called directly, as explained in section 4 on page 8, where in some circumstances it may not be possible to call the `osmods` library.

This manual also details the available stores and retrieval modules that are currently available. These are:

1. Application Parameter Store. This store uses the `applparms` [1] library to retrieve the value of a structured environment variable as explained in section 5 on page 10.

1.2 SEV Format

The format of a structured environment variable reference or name is as follows:

```
${SEV:storename:variablename}
```

where

- SEV - This literal with the trailing colon ':' identifies the environment variable reference as a structured environment variable.
- storename - This identifier names the structured environment variable store. It resolves to a configuration file (see section 2.1 on page 3) that defines the attributes of the store.
- variablename - This identifier names the variable that is required to be resolved from the store.

For example, an environment variable reference of `${SEV:runparms:run_date}` is a structured environment variable reference to the variable `run_date` within the `runparms` store.

2 SEV Configuration File

2.1 Naming and Locating

The system environment variable `CODEMAGUS_SEVPATH` defines a template to which the store name, folded to upper case, is applied to generate the fully qualified name of the store configuration file. The template is a path name and must include a ‘`%s`’ as a substitution place holder for the name of the structured environment variable store. If the environment variable is blank or not defined then the default template is ‘`%s.sev`’.

For example if a structured environment variable reference is `SEV:cmlstore:myvar` and the value of the environment variable `CODEMAGUS_SEVPATH` is
‘`c:\\CodeMagus\\Catalogs\\%s.sev`’
then the store configuration file name will be
‘`c:\\CodeMagus\\Catalogs\\CMLSTORE.sev`’

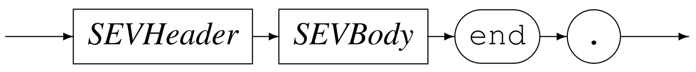
2.2 Syntax

The structured environment variable configuration file identifies the store and the module and its entry point that is loaded and called to resolve a structured environment variable reference.

2.2.1 Configuration File

The configuration file is made up of a header and a body portion and is terminated with the end keyword and a full stop.

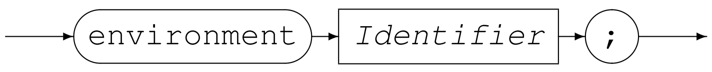
SEVConfiguration



2.2.2 Configuration header

The header names the configuration.

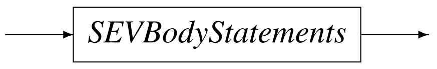
SEVHeader



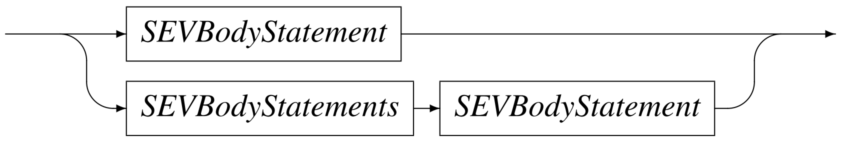
2.2.3 Configuration body

The body of the configuration is made of various statements as listed in the syntax diagram and explained below.

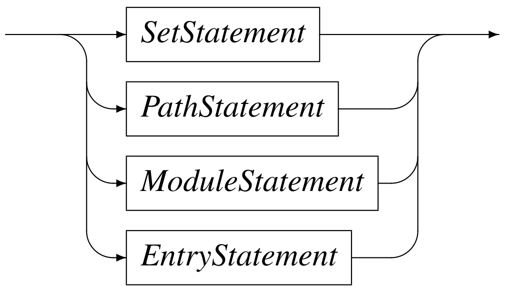
SEVBody



SEVBodyStatements



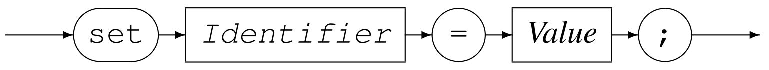
SEVBodyStatement



2.2.4 *Set* statement

This statement sets a system environment variable to `Value` for use later in the configuration file or in the interface module.

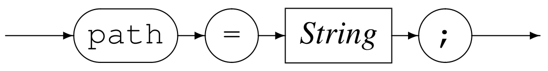
SetStatement



2.2.5 *Path* statement

This statement identifies the path to the store interface module.

PathStatement



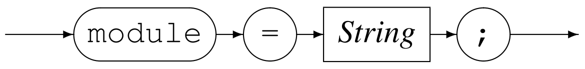
Examples are:

```
path "c:\CodeMagus\bin\"
```

2.2.6 *Module* statement

This statement identifies the module that is the interface to the store. It implements all the methods required for accessing the store of variables. This is a DLL or shared object program that adheres to calling conventions and function names as shown in section 4 on page 8.

ModuleStatement



Examples are:

```
module "cmlapsem.dll"
```

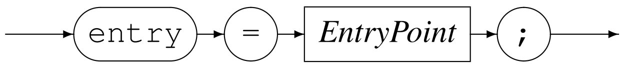
or . . .

```
module ${CMLAPSEM_MODULE}
```

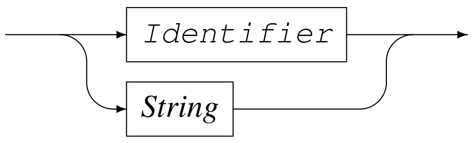
2.2.7 *Entry* statement

This statement identifies the entry point in the store interface module. The entry point is usually an identifier but may be a string (especially if the name is held in an environment variable reference.

EntryStatement



EntryPoint



Examples are:

```
entry cmlapsem_entry
```

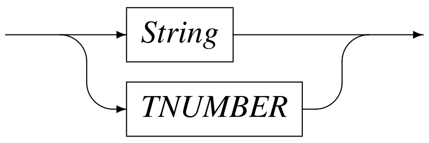
or . . .

```
entry ${CMLAPSEM_ENTRY}
```

2.2.8 *Value* types

A *Value* can be either a `String` or a `Number`.

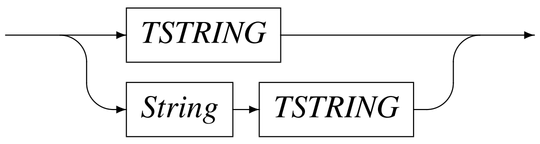
Value



2.2.9 *String* type

The *String* type is a concatenation of one or more Strings.

String



3 Indirect Retrieval Method

The indirect method offers the application developer the ability to use structured environment variables with minimal or no coding changes.

3.1 Application Source Code and Linking

The application must include the header file `osmods.h` and link to the `osmods` and supporting libraries for the relevant platform. Examples of the link statements are:

For Linux and Unix:

```
-L path_to_libraries -losmods -lenvvar -lstrucenv -lhashtab -ldl
```

For Windows using Microsoft Visual C:

```
/LIBPATH path_to_libraries osmods.lib envvar.lib strucenv.lib hashtab.lib
```

3.2 Resolving a reference

The inclusion of `osmods` into an application means that the function `getenv()` is mapped to the `osmods` library. The application need only call the `getenv()` function to resolve any variable and `osmods` will resolve the reference from the system or the structured environment variable library automatically.

4 Direct Retrieval Method

The direct method is discouraged as a general programming interface and is only of importance to library developers (for example `osmods`).

4.1 Application Source Code and Linking

The calling application must include the header file `strucenv.h` and link to the `strucenv` and supporting libraries for the relevant platform. Examples of the link statements are:

For Linux and Unix:

```
-L path_to_libraries -lstrucenv -lhashtab -ldl
```

For Windows using Microsoft Visual C:

```
/LIBPATH path_to_libraries strucenv.lib hashtab.lib
```

4.2 Resolving a Reference

With the direct method an application calls the `strucenv` library directly. If this is the first reference to a store then the store is opened at the same time.

4.2.1 Synopsis

```
char *strucenv_getenv(const char *sev, int flags);
```

The value of the SEV `sev` is returned as a pointer to a NULL terminated string. The string should be copied and saved locally if the application wishes to retain the value for any length of time. The flags allow various run time options to be set, see appendix A on page 13.

4.2.2 Return Value

On success a pointer to the value of the SEV is returned, otherwise `NULL` is returned and an error message can be obtained by calling `strucenv_error`.

4.3 Closing a SEV Store

With the direct method the store can be explicitly closed in order to release any resources held by it. The store name must be passed on this call. If the store is not explicitly closed it will be closed by the `atexit()` function on platforms that support it.

4.3.1 Synopsis

```
void strucenv_close(const char *storename);
```

4.3.2 Return Value

There is no return value from this call.

4.4 Obtaining Error Information

With the direct method, if there is an error during any processing, the last error that occurred can be retrieved using this function. The error message is available until another error overwrites it and as such should never be called after a successful call to the library.

4.4.1 Synopsis

```
char *strucenv_error(void);
```

4.4.2 Return Value

This routine returns a pointer to the last error to occur on a call to the `strucenv` library or `NULL` if there is no outstanding error.

5 `applparms` Based Store

5.1 Configuration Details

A structured environment variable configuration file that describes a store that uses `applparms` to resolve the structured environment variables requires the following configuration settings:

- Retrieval method module. This must be `cmlapsem.so` for Unix and Linux platforms and `cmlapsem.dll` for Windows platforms.
- Retrieval method module entry point. This must be `cmlapsem_init`.
- The application parameter configuration file name must be set via a system environment variable as follows:

  ```
  set applparms_apd = "SAMPLE.apd";
  ```

  This names `SAMPLE.apd` as the `applparms` configuration file.
- The application parameter processing mode must be set via a system environment variable as follows:

  ```
  set applparms_mode = "ForeGround";
  ```

  This value may only be `ForeGround` to force the application parameter user interface to interact with the user. If any other value is specified or it is not specified then the processing is assumed to be back ground and there is no ability for the user to change or supply any values for the variables. In back ground mode all the variables defined by the `applparms` configuration file are assumed to have a value and if, at run time, they do not the `applparms` library will return an error and thus a NULL value for the structured environment variable that the application is attempting to resolve.

5.2 Example SEV Configuration File

```
-- File: SAMPLE.sev
--
-- This Structured Environment Variable Configuration File (SEV) is a definition
-- of how to retrieve the value of a variable from a store defined by a Code
-- Magus Ltd. applparms definition.
-- It names the applparms definition and the module that will be used to
-- retrieve the values.
--
-- $Author: hayward $
-- $Date: 2014/09/23 12:30:32 $
-- $Id: SAMPLE.sev,v 1.2 2014/09/23 12:30:32 hayward Exp $
-- $Name:  $
```

```
-- $Revision: 1.2 $
-- $State: Exp $
--
-- $Log: SAMPLE.sev,v $
-- Revision 1.2  2014/09/23 12:30:32  hayward
-- Allow the "default" keyword on a set command.
-- This means that the environment variable will
-- only be set if it is not already set.
--
-- Revision 1.1  2010/05/17 15:01:36  hayward
-- Write documentation and update code
-- in places that showed problems or
-- inconsistent naming.
--
environment sample;

-- The applparms_apd environment variable defines which APD file to use for
-- retrieving the values of the variables it defines.
-- For DNGG00 we use the same names APD file.

   set applparms_apd = "SAMPLE.apd";

-- The applparms_mode environment variable defines whether the applparms UI will
-- wait for user interaction or attempt to continue without it. If set to
-- "BackGround" the UI will not wait for user interaction. In this case if any
-- of the parameters defined by the APD file do not have a value an error
-- condition will be raised.

   set default applparms_mode = "BackGround";

-- Define the program module whose methods will be called to load and retrieve
-- the applparms values.

   path = ${CODEMAGUS_HOME}"/lib/";
   module = "cmlapsem.so";
   entry = cmlapsem_init;

end.
```

5.3 Example `applparms` Configuration File

```
application SAMPLE;
-- This APD file defines the parameters used by the SAMPLE store and defines
-- various userids and passwords for remote access.

-- $Author: hayward $
-- $Date: 2010/06/02 16:16:55 $
-- $Id: SAMPLE.apd,v 1.1 2010/06/02 16:16:55 hayward Exp $
-- $Name:  $
-- $Revision: 1.1 $
-- $State: Exp $
```

```
--
-- $Log: SAMPLE.apd,v $
-- Revision 1.1  2010/06/02 16:16:55  hayward
-- Add to CVS.
--

   title "SAMPLE: Connection Parameters for Remote Access";
   description "This file defines the parameters (or variables) and "
      "their values that are required when accessing a remote system."
     ;

   set TODAY = ${DATE_YYYYMMDD};

   store ${CODEMAGUS_APDSTORE};

-- The interface defines the shared object or DLL program that will interact
-- with the user to ensure that all parameters have a value. This may be
-- different of Windows as to Unix/Linux.

   interface default;
   entry     none;

   parameter MVS_USERID
      title "MVS UserID";
      default NULL;
      options ;
      description "User ID to connect to MVS with."
         ;
      constraint "^[^ ]\+$";
   end

   parameter MVS_PASSWD
      title "MVS Password";
      default NULL;
      options secret;  -- Value must not be shown by the UI.
      description
         "Password required when connecting to MVS."
         ;
      constraint "^[^ ]\+$"; --- at least one character, no spaces.
   end
end.
```

A `strucenv` Header File

```
#ifndef STRUCENV_H
#define STRUCENV_H
  /* File: structenv.h
   *
   * This header describes the interface to the Code Magus Limited Environment
   * Variable store Module. The Environment Variable Store Module allows
   * applications to retrieve the value of an variable as defined by the store
   * the module reads from.
   *
   * Author: Stephen Donaldson [www.codemagus.com].
   *
   * Copyright (c) 2010 Code Magus Limited. All rights reserved.
   */

  /*
   * $Author: hayward $
   * $Date: 2010/05/17 15:01:36 $
   * $Id: strucenv.h,v 1.3 2010/05/17 15:01:36 hayward Exp $
   * $Name:  $
   * $Revision: 1.3 $
   * $State: Exp $
   *
   * $Log: strucenv.h,v $
   * Revision 1.3  2010/05/17 15:01:36  hayward
   * Write documentation and update code
   * in places that showed problems or
   * inconsistent naming.
   *
   * Revision 1.2  2010/05/06 08:41:33  hayward
   * Remove all references to osmods as this
   * module is built before it.
   *
   * Revision 1.1.1.1  2010/05/04 13:36:24  hayward
   * Import sources to CVS.
   *
   */
static char *cvs_strucenv_h =
   "$Id: strucenv.h,v 1.3 2010/05/17 15:01:36 hayward Exp $";

  /*
   * Constants and options:
   */

  /*
   * Exposed types and structures:
   */

  /*
   * Exported functions:
   */
   /* Function strucenv_getenv attemps to retrieve and return the value of the
    * qualified structured environment variable given in sev. If the value can
    * not be returned then NULL is returned.
    *
    * Any errors encounted in the processing are written to stderr. If more
    * diagnostic information is required then set CODEMAGUS_MSGLEVEL=VERBOSE or
    * TRACE. This diagnostic information is also written to stderr.
    */

char *strucenv_getenv(const char *sev, int flags);
#define STRUCENV_OPT_VERBOSE 0x80000000

  /* Function strucenv_close will free all allocated resources held for the
   * store. Once complete the store is closed and can not be accessed again
   * unless it is re-opened.
   * There is no return code from this routine.
   */

void strucenv_close(const char *storename);

  /* Function strucenv_error will return the last error that occurred within the
   * library when the return from another call to the library was not
   * successful. The error message is available until another error overwrites
   * it and as such should never be called after a successful call to the
   * library.
   */

char *strucenv_error(void);

  /*
   * Globals variables exported by library:
   */

#ifdef STRUCENV_INCLUDED_FROM_STRUCENV_C
   #define STRUCENV_EXTERN /* local */
#else
   #define STRUCENV_EXTERN extern
#endif


#undef STRUCENV_INCLUDED_FROM_STRUCENV_C
#undef STRUCENV_EXTERN

#endif /* STRUCENV_H */
```

References

[1] applparms: Application Parameters Library User Guide and Reference Version 1. CML Document CML00054-01, Code Magus Limited, January 2010. PDF.