Edelwise™

ceWISE Programming Guide

Title: User Manual for ceWISE option.

Type of documentation: Manual

Date of creation: 02/09/2007

This technical description replaces all previous versions.

Document history

Date	Name	Revision	Modification
2/9/2007	edelweiss	A	First version

1 TABLE OF CONTENTS

2 WHAT IS THE CEWISE PROFESSIONAL INTERFACE?

The ceWISE Professional Interface is an optional component for the beWISE Professional product. It is implemented as a DLL and it allows C programs to access beWISE variables in a similar manner as VB programs. This interface enables C applications to interchange data with other C programs or VB (VB6 and .NET) programs.

Since the ceWISE Interface is an add-on for the beWISE Professional product it supports many features available in the beWISE Professional product like

- a total of 5,120 concurrent beWISE variables
- a maximum string length of 256 characters
- functions to create, read, write and delete variables dynamically

3 IMPORTANT CONSIDERATIONS

3.1 Before Installation

You need to have the beWISE PROFESSIONAL or beWISE LAN product installed and properly licensed before you can use the ceWISE interface.

You also need a C compiler and linker in order to use the ceWISE C interface.

3.2 After Installation

During installation all sample programs and all ceWISE files will be installed in the directory

`C:\Program Files\Edelwise\VC.Option\`

It is recommended that you copy the `ceWISEpro.dll`, which comprises the C interface, into the system32 directory of your computer or change the PATH in order to make sure that the ceWISEpro.dll can be found.

4 GETTING STARTED WITH THE SAMPLE PROGRAMS

This product is shipped with two C sample programs that demonstrate the use of the interface. This chapter explains how to get started with the sample programs.

4.1 Compiling the Sample Files

After installation of the ceWISE interface the C sample source code can be found in
`C:\Program Files\Edelwise\ VC.Option\`

The directory contains the following files:

`Christl.c`	example that incrementes one variable (goes with KARL from „Counting“)
`Claudia.c`	example that generates random values (goes with KARL from „Listening“)
`Cdel.c`	example that shows how to delete variables
`ceWISE.h`	include file for the C interface
`ceWISEpro.lib`	library for the C interface
`ceWISEpro.dll`	DLL for the C interface; this DLL comprises the interface
`make.bat`	compiles and links the sample programs

To compile and link the two examples you just have to double-click the `make.bat` file.

NOTE:
The sample programs are written for the Microsoft C complier and linker. You may need to modify the sample programs and make.bat file if you are using another compiler or linker.

5 USING THE ceWISE DLL WITH YOUR C PROGRAMS

In order to use the ceWISE interface functions in your C programs the following needs to be considered:

5.1 Source Code

Include the `ceWISE.h` file in you C programs

```
#include "ceWISE.h"
```

Make sure to call the `ceWISEinit()` function before using any other ceWISE function.

```
main ()
{

ceWISEinit();           // initialize the Edelwise environment
...
```

5.2 Linking

You need to link your application with the `ceWISEpro.lib`

e.g. `cl c2vbpro.c /link ceWISEpro.lib`

5.3 Execution

You need to make sure that the **`ceWISEpro.dll`** is in your path. It can be in the current directory, anywhere else in the PATH or in the SYSTEM32 directory (preferred).

6 THE ceWISE API

6.1 ceWISEinit ()

```
int  ceWISEinit(void);
```

Initializes the ceWISEpro interface.

This function needs to be called before any other ceWISEpro functions can be called.

After a reboot or login the very first application creating a ceWISE variable (C programs) or beWISE variable (VB programs) will load the initialize the shared memory used for inter-task communication. This process is transparent to the programmer, however this is the reason why loading the very first application requires more time (1 to 3 seconds, depending on processor speed) then subsequent application starts.

Return Value

Value	Description
< 0	an error has occurred; the interface could not be initialized properly
>= 0	the interface has been initialized properly

```
#include <stdio.h>
#include "ceWISE.h"

main ()
{
int i, ret;
long value = 0;

ceWISEinit();              // initialise the Edelwise environment
```

6.2 ceWISEexit()

```
int  ceWISEexit(void);
```

Frees ceWISEpro interface resources.

This function should only be called upon termination of the application. It frees all resources that the application has allocated during initialization of the ceWISEpro interface.

```
    ceWISEexit();         // free Edelwise environment resources
```


6.3 ceWISEcreate()

```
int  ceWISEcreate(char *ceWISEname,unsigned short ceWISEtype);
```

Creates a variable.

This function creates (if it does not exist) or finds (if it does exist) a variable in the shared memory (ceWISE variable) and returns its handle. If the variable already exists then the ceWISEtype parameter is ignored. In this case the actual type of the variable could be different from the assumed type.

Return Value

Value	Description
< 0	an error has occurred; the variable could not be created
>= 0	handle of the previously existing or newly created variable

Parameters

Name	Description
ceWiseName	C-string with the ceWISE variable name. This is the unique ID/name of the variable in the shared memory. The following rules apply: Maximum of 125 characters; only alphanumeric characters and digits and the underscore (_) are allowed in the name.
ceWiseType	Type of the variable beeing created. The following types are allowed: **ceWise_DOUBLE**, **ceWise_LONG** and **ceWise_STRING**

```
for ( i=0; i<100; i++)
     {
     char name[15];

     sprintf( name, "Data%02d", i );
     ew_graph[i] = ceWISEcreate( name, ceWise_LONG );
     }
```


6.4 ceWISEfind()

```
int  ceWISEfind(char *ceWISEname);
```

Finds a variable.

This function finds a variable in the shared memory (ceWISE variable) and returns its handle. If the variable does not exist an error code is returned.

Return Value

Value	Description
< 0	an error has occurred; the variable could not be found
>= 0	handle of the variable

Parameters

Name	Description
ceWiseName	C-string with the ceWISE variable name. This is the unique ID/name of the variable in the shared memory. The following rules apply: Maximum of 125 characters; only alphanumeric characters and digits and the underscore (_) are allowed in the name.

```
if ( (ew_del = ceWISEfind( argv[1] )) < 0 )
    {
    printf(">>> variable [%s] not found\n", argv[1]);
    exit(2);
    }
```

6.5 ceWISEdelete()

```
int  ceWISEdelete(int ceWISEhandle);
```

Deletes a variable.

This function deletes the variable in the shared memory (ceWISE variable) identified by ceWISEhandle.

Return Value

Value	Description
< 0	an error has occurred; the variable could not be found
= 0	the variable was deleted

Parameters

Name	Description
ceWiseHandle	Integer representing the handle of the variable.

```
int ew_del;
int ret;

if ( (ew_del = ceWISEfind( argv[1] )) < 0 )
          {
          printf(">>> variable [%s] not found\n", argv[1]);
          exit(2);
          }

ret = ceWISEdelete( ew_del );

printf("*** variable [%s] deleted with rc=(%d)\n", argv[1], ret);
```

6.6 ceWISEput()

```
int  ceWISEput(int ceWISEhandle,void *ceWISEdata);
```

Updates the value of a variable.

This function updates the value of the variable in the shared memory (ceWISE variable) identified by ceWISEhandle.

Return Value

Value	Description
< 0	an error has occurred; the variable could not be updated
>= 0	the variable was successfully updated

Parameters

Name	Description
ceWiseHandle	Integer representing the handle of the variable.

ceWiseData	Pointer to the C variable holding the value. The data type of the C variable has to correspond with the data type of the shared memory variable (ceWISE variable) as shown below:
	
	C type — *ceWISE type*
	`long *` *ceWiseData* — `ceWise_LONG`
	`double *` *ceWiseData* — `ceWise_DOUBLE`
	`char *` *ceWiseData* — `ceWise_STRING`
	
	It is the responsibility of the programmer to ensure that the provided C variable is of the proper type and, in case of strings, the string length does not exceed 256 byte.

6.7 ceWISEget()

```
int  ceWISEget(int ceWISEhandle,void *ceWISEdata);
```

Returns the value of a variable.

This function returns the value of the variable in the shared memory (ceWISE variable) identified by ceWISEhandle.

Return Value

Value	Description
< 0	an error has occurred; the variable could not be found
> 0	the value of the variable was successfully retrieved

Parameters

Name	Description
ceWiseHandle	Integer representing the handle of the variable.

ceWiseData	Pointer to the C variable where the value will be stored. The data type of the C variable has to correspond with the data type of the shared memory variable (ceWISE variable) as shown below:
	
	C type — *ceWISE type*
	`long *` *ceWiseData* — `ceWise_LONG`
	`double *` *ceWiseData* — `ceWise_DOUBLE`
	`char *` *ceWiseData* — `ceWise_STRING`
	
	It is the responsibility of the programmer to ensure that the provided C variable is of the proper type and, in case of strings, the character buffer is long enough to accommodate the returned value. The maximum length of a ceWISE STRING is 256+1 byte.

7 DISTRIBUTING AND INSTALLING RUNTIME FILES

Since the ceWISEpro Interface is an add-on to the beWISEpro product you need to include ALL items in your distribution package that are required runtime files for the beWISE product. For more information on that consult your beWISE documentation.

In addition you need to include the `ceWISEpro.dll` (the C interface DLL) in your distribution. The DLL needs to be copied into the system32 directory of the target computer.

8 TESTED ENVIRONMENTS

Currently all programs (source and binary files) have been tested in the following environments:

Compiling and Linking

- Microsoft 32-bit C/C++ Optimizing Compiler Version 12.00.8804 for 80x86
- Microsoft Incremental Linker Version 6.00.8447

Operating Systems

- Windows 2000 (SP3)
- Windows XP Professional (SP1 and SP2)