
Introduction

Welcome to SecWin, a comprehensive program security pack for Clarion for Windows developers.

Using this package you will be able to provide advanced security features to your end users simply and effectively. Whether your requirements are simple, such as simple logging on and program access control, or complex, stretching across multiple EXE's and multiple program areas with individual access controlled, at runtime, on a screen by screen and control by control basis, you will find that implementation is simple. Full use is made of Templates to assist you, but handcoding is also fully explained and supported.

Note: <u>If you are upgrading from a previous version of SecWin, skip to the end of this document for a list of new features otherwise you will miss out !!</u>

Tip : New in this version is the SecWiz utility template. Use this to add SecWin to your apps in seconds !!! See the template guide for more details. This feature is only available in the registered version.

CW 1.0 Note: As from this version, CW 1.0 is no longer supported directly. The Security file created is however still compatible with the SecWin10.Dll used by CW 1.0 developers. CW 1.0 developers can continue to use SecWin Version 1.7.

CW 2.0 Note: This manual has been written with CW 1.5 in mind. If you are using CW 2.0 the please consult the very end of this document for details relevant to you.

WARNING : There is a bug in CW 1.0 which can cause corrupt applications if due care is not taken. When upgrading SecWin, or any templates, **ALWAYS** de-register the old template before replacing it with the new one. Don't forget to Register the new template. This bug may or may not be in CW 1.5, but follow the above procedure to be safe.

In this documentation you will find:

SecWin Installation Guide

This covers installation of the SecWin package, Distribution and Licensing of SecWin components, using SecWin in your Applications and information for installing SecWin protected applications on end-user computers.

***<u>Note to users in a hurry</u>*:** If you're like me you want the quickest path to an immediate result. The first three sections of the Installation Guide get you up and running with all you need to have a application implementing SecWin. I also **strongly** recommend reading the User Guide for an explanation of concepts and general options. The security is trivial to implement, but like all programming, no product can aid poor design.

SecWin Template Guide

This is an in-depth reference to the SecWin Templates and all of their powerful options. This section includes registering your templates in the Clarion for Windows template registry, as well as a look at each of the available templates, what they are for, and when to use them.

SecWin Technical Reference

This is an in-depth reference of all of the functions exported by the SecWin Dll. This section would be most useful for people implementing SecWin features in handcoded modules. The functions are arranged alphabetically, and include source code examples.

SecWin User Guide

Last, but not least is the SecWin User Guide. This guide is of a more informal nature and explains the more conceptual features of the package. Methods of implementing more sophisticated security systems, implementing handcoded solutions, using the multi language features and conceptually defining your required security are covered here.

Documentation for the runtime screens inside the SecWin DLL is also provided here.

SecWin Installation Guide

This covers general developer information

Contents

Installation of SecWin

The following files are distributed, and required by SecWin. The examples given assume you are running a Dos Command Line Shell. You may however use any method to move the files from the distribution disk into your Clarion Windows directories. The examples further assume you are copying from the A drive to the C drive, and that your Clarion for Windows directory is called CW15 or CW20. Please modify the instructions to suit your particular setup.

The installation process has been greatly simplified. Simply use one of the following 2 methods to Install SecWin.

For Clarion for Windows version 1.5 users

1) I received SecWin as a ZIP file, via the Internet or Compuserve etc.
a) Go to a Dos Command Prompt Window.
b) Change directory to the CW15 directory
c) Copy the SecWin.Zip file into this directory.
d) Unzip the file using the -d switch. ie PKUNZIP -d SecWin20.Zip

2) I received SecWin on a floppy disk, CD etc
a) Go to a Dos Command Prompt Window.
b) Change directory to the CW15 directory
c) Type `XCopy a:\secwin\*.* /s` where a: is the drive with the SecWin disk.

For Clarion for Windows version 2.0 users

1) I received SecWin as a ZIP file, via the Internet or Compuserve etc.
a) Go to a Dos Command Prompt Window.
b) Change directory to the CW20 directory
c) Copy the SecWin.Zip file into this directory.
d) Unzip the file using the -d switch. ie PKUNZIP -d SecWn220.Zip

2) I received SecWin on a floppy disk, CD etc
a) Go to a Dos Command Prompt Window.
b) Change directory to the CW20 directory
c) Type `XCopy a:\secwin\*.* /s` where a: is the drive with the SecWin disk.

Using SecWin in your Apps

This is a quick guide to the minimum steps necessary to add SecWin to an existing, or new application. After each step is a reference (in brackets) to a fuller explanation of the step.

1) Register the SecWin template. (**Template Guide - Registering the SecWin Template**.) You only need to register the template the first time that you use SecWin.

2) Run the CRESEC15 utility to create a DsSW.TPS file in your Windows Directory. (Installation Guide - The CRESEC Utility.) CW 2.0 users can run the CRESEC20 utility instead.

3) Add the Activate Security features template extension to the applications Global Properties. (**Template Guide - Activating SecWin features**.)

If you have the registered version
4) Run the SecWiz utility template to add all the basic SecWin features (or just some of them) to your application. (**Template Guide - SecWiz Utility**)

Otherwise
4) Add at least one instance of the User Login Here template extension to one of the applications functions. The main function is a good candidate. (Template Guide - User Login Here.) This creates a Security Area.

5) Make sure that supervisors can run the ds_OperatorBrowse and ds_ChangePassword functions from within every security area. Do this by adding these functions to your main menu. You can use the Code templates to do this. (Template Guide - Calling Operator Browse and Template Guide - Calling Change Password.)

Optionally:

6) Add more instances of the User Login here template, making sure that Operator Browse is still visible.

7) Add instances of the User Screen Security template to functions in the application (Template Guide - User Screen Security.)

8) Add calls to ds_CurrentName, ds_CurrentLogin and ds_CurrentLevel to show the current user on, for example, the status line. You can use the StatusBar Code template to do this.

9) Add calls to ds_ChangeLogin and ds_LockScreen on your main menu. You can use the relevant code templates to do this.

Using SecWin in a multi DLL project

This section discusses the implementation of SecWin in an application that uses one, or more, DLL's, in addition to the root EXE. I will refer to three parts of the application, namely :
1) The **EXE**. This is the program that the user runs, which calls one or more DLL's.

2) The **Root DLL**. This is the DLL that contains all the global data declarations, file structures, global functions etc. This DLL must be referenced by all the other DLL's, and the EXE.

3) **Other DLL's**. These are optional. They fit in somewhere between the EXE and the Root DLL. If you wish to implement SecWin in these DLL's then they must reference the Root Dll.

To add SecWin to the EXE, follow the instructions set out in the section above (Using SecWin in your Apps). The only additional step is to click the "SecWin data defined in another DLL" check box on. You must still fill in an Application name.

To Add SecWin to a Root DLL, follow the instructions set out in the section above (Using SecWin in your Apps). Additionaly click the "Export SecWin data defined in this DLL" check box on. You may fill in an application name, but it is not used. You will also need to run the "Import Queue

Handling" Utility template. You can do this by :
1) Open the application in the normal fashion. (This is the root DLL app).
2) Click on the Application Menu. (In CW 1.0 this is the Utilities Menu)
3) Choose Utility Template.
4) Choose Import Queue Handling.
In addition you need to add two variables to the global data area of the DLL. They are

AppNameDesc	String
AppNum	Long

Important Note : You must add these two variables to the Global Data area, not to a Global embed point, or the varaibles will not be exported properly.

To add SecWin to an Other DLL, follow the instructions set out in the section above (Using SecWin in your Apps). The only additional step is to click the "SecWin data defined in another DLL" check box on. You may fill in an application name, but it is not used.

Note : There is a comprehensive example and documentation of this process distributed with this SecWin package. The example is called MultiDLL.

32 Bit SecWin

The 16 bit version of SecWin is absolutly free. A 32 bit version of SecWin, suitable for use with 32 bit Clarion applications, is available from Custom Business Software, Topspeed Corporation, and their respective dealers. In the US or Canada you can contact Topspeed at 800-354-544 The current price of 32 bit SecWin is US$99.99.

Installing a SecWin Protected App on a User's System

When installing an Application that uses SecWin on another system then the following points should be considered. I will refer to the destination computer as the **Client Machine**.

1) You will need to install one of the following file sets on the client machine depending on the version of Clarion that you are using. All of the files can be located in your **CW\BIN** or **\CW15\BIN** directories.
Clarion 1.0 : **SECWIN10.DLL**, **CWTPS.DLL** and **CWRUN10.DLL**
Clarion 1.5 : **SECWIN16.DLL**, **CWINTL16.DLL**, **CWTPS16.DLL** and **CWRUN16.DLL**
Clarion 2.0 : **SECWN216.DLL, CW2TPS16.DLL** and **CW2RUN16.DLL**

2) You will need to ensure that the file **DsSW.TPS** exists in your client's **WINDOWS** directory, or somewhere else pointed to by an INI file. For more information on INI files see the section below on the CRESEC utility.

You can either distribute a DsSW.Tps file, or distribute **CRESEC.EXE** which is capable of creating the file in the correct place. See below for more information on the CRESEC utility.

Running CRESEC on a machine that already has a copy of DsSW.TPS has no effect.

3) You may want to stamp the DsSW file with an Application PIN number. See the User Guide for more information on PIN numbers.

The CRESEC.EXE utility

Version note : If you are using CW 1.5 then your version of Cresec is called CRESEC15, if you are using CW 2.0 then your version of Cresec is called CRESEC20.EXE . Please substitute this into the following discussion where applicable.

SecWin makes use of a data file to store user names and access rights etc. This file is called **DsSW.TPS** and is stored in the Windows directory. For security reasons the SecWin DLL is incapable of creating DsSW.TPS. If it could then users could bypass the security by deleting DsSW. In order to prevent this you can distribute an empty copy of DsSW (created by yourself using CRESEC) or you may distribute the CRESEC.EXE utility.

If users delete the DsSW file then none of the protected programs will run. The file will need to be restored from a backup or recreated by an authorised person, before the affected applications will run again.

The CRESEC utility will not overwrite an existing DsSW file. By ensuring that only authorised people have access to CRESEC, or to the blank copy of DsSW, you allow the user to create a new file if necessary, but remove any unauthorised attempts to breach the security by deleting the DsSW file.

Clarion 1.5 Users : For your convenience CRESEC15.EXE has also been included with SecWin. This makes use of the CW15 Dlls instead of the CW10 Dlls. You may use, and distribute, either program. They are both functionally identical.

Network users : You can override the placement of the DsSW file by adding a section to the Win.Ini file called [SecWin]. In this section set the placement by setting a variable called path. This setting is subject to there being no setting in the SecWin.Ini file.

Example : In the Win.Ini file.
```
[Secwin]
path=z:\netsec
```

You can also specify the placement of the DsSW file by adding a file called SecWin.Ini in the Application Directory. This file has the same section and path as the Win.Ini file. This setting has priority over the Win.Ini file.

Example : In the SecWin.Ini file, in the Application Directory
```
[Secwin]
path=z:\netsec
```

If the path in the SecWin.Ini file is set to "Here" then you can place the DsSW file in the application directory, without knowing the path of that directory. This is to assist developers where the application path may be different for different machines.

Example : In the SecWin.Ini file, in the Application Directory
```
[Secwin]
path=Here
```

Distribution

CW Version 1.5 :You will need to, and are free to, distribute the **SECWIN16.DLL** with your completed applications. You will also need to distribute the **CWINTL16.DLL, CWTPS16.DLL** and the **CWRUN16.DLL**.

CW Version 2.0 : You will need to, and are free to, distribute the **SECWN216.DLL** with your completed applications. You will also need to distribute the **CW2INT16.DLL, CW2TPS16.DLL** and the **CW2RUN16.DLL**.

You are free to distribute the **CRESEC.EXE, CRESEC15.EXE** or **CRESEC20.EXE** programs should you want to.
See the section entitled 'Installing a SecWin Protected App on a User's System' for more information.

License

All the files distributed as part of the SecWin package are copyrighted and remain the property of Custom Business Software. This excludes EmpRpt.App and EmpRpt15.App files.

You are licensed to use 16 bit versions of SecWin (ie SecWin10 and SecWin16) on as many computers as you please. There is no charge for SecWin. Licenses for 32 bit versions of SecWin are available from Custom Business Software, Topspeed Corporation, and their respective dealers. If you are developing 32 bit applications on more than one computer at a time then you will require a license for each computer. We are extremely flexible regarding licensing agreements so please contact us if you require more than one license.

You are free to distribute the SecWin DLLs with any of your Applications without extra charge.

You may distribute the 16 bit SecWin package (ie SecWin20.Zip) to anyone, provided that the following 2 conditions are met:
1) The entire package must be distributed together.
2) You may not charge for the package, except for nominal copying and postage fees.

You may not distribute any part of the 32 bit version of SecWin, except for the DLL, at all.
The 32 bit version of SecWin is available only from Custom Business Software, Topspeed Corporation, and their appointed dealers.

Limitation of Liability

Custom Business Software and its employees assume no liability for the use of this package. Our sole liability rests with fixing any software errors that may exist in the distributed SecWin files. In certain cases, due to bugs in software supplied to us, it may be impossible to correct errors in SecWin.

Copyright

Address	: PO Box 18156 , Wynberg, 7824, South Africa		
Phone (local)	: (021) 797 9468	Fax	: (021) 797 4324
Phone (intl)	: +27 21 797 9468	Fax	: +27 21 797 4324
eMail	: datasoft@iaccess.za		
CompuServe	: 100076,656		

SecWin Template Guide

This guide describes the templates provided with SecWin.

Contents

Registering the SecWin template

1) Choose **Template Registry** from the **Setup** menu.
2) Click on **Register**.
3) ***Clarion 1.5 users*** : Choose **SecWin15.Tpl** from the **CW15\Template** directory.
   Clarion 2.0 users : Choose **SecWin20.Tpl** from the **CW20\Template** directory.

Clarion 1.5 and Clarion 2.0 users should note that the template class is still called SecWin10, even in Clarion 1.5 or Clarion 2.0. This is to assist in the porting of CW 1.0 applications to CW 1.5. and CW 1.5 applications to CW 2.0.

Extension : Activate SecWin features

Visible Effect to your App
None

Purpose
This is a Global Extension template which needs to be added to the Global area in order for the SecWin features to work.

To Include in your App
1) Open your App in the usual fashion.
2) Choose **Global Properties** from the **Edit** menu. (You can also click on the **Global** button to get here)
3) Click on **Extensions.**
4) Click on **Add**
5) Choose **Activate SecWin Features**
6) Fill in the Application name. This name must be unique for different applications. Do not include the version number on this line.
7) If this is a Multiple DLL project, and this is DLL containing all the file structures and global variables then check the 'Export SecWin data defined in this DLL' check box.
8) If this is a Multiple DLL project, and this is not the DLL mentioned in step 7 above, then check the 'SecWin data defined in another DLL' check box.
9) If you wish to make use of the Application Pin number feature then enter a Pin number here. You can read more about Pin numbers in the User Guide in the section entitled "Application PIN numbers".
10) Close all the windows and return to the main application tree screen.

Advanced
This template extension adds the following code to your application:

1) It adds the **SecMap10.Clw** or **SecMap15.Clw** file to your global map.
2) It adds the **SecEqu10.Clw** or **SecEqu15.Clw** file, containing the SecWin equates, to your global data area.
3) It adds a global variable, called **AppNameDesc**, which contains the unique name that you entered above. You may read this variable in your application, but don't write to it. If you checked on the 'SecWin data defined in another DLL' then this variable is declared as External.
4) It creates a global queue, called **AppNumQueue**, which contains the number returned by the Login function. This number is used by the other SecWin functions. It is stored in a queue so that multiple password areas within your application can be supported. Every call to ds_LoginText (using the **User Login Here** extension) adds a value to this queue, and removes it when the user logs out (also in the User Login here extension.) The value is stored in a variable called **AppNum**.

Extension : Set Default Font

Visible Effect to your App
SecWin screen will use the font you specify instead of it's usual font.

Purpose
This is a Global Extension template which can be added to the Global area of your Application. It allows you to specify the font that will be used by SecWin. It is there primarily so that SecWin screens use the same font as your program, thus resulting in a smoother user interface.

To Include in your App
1) Open your App in the usual fashion.
2) Choose **Global Properties** from the **Edit** menu. (You can also click on the **Global** button to get here)
3) Click on **Extensions.**
4) Click on **Add**
5) Choose **Set Default Font**
6) Fill in the Font name, size, color and style that you want. If you leave all these items blank then the user will be prompted with a Font Dialog at the beginning of the program. Valid sizes are from to 12 points. Set the size to 0 to leave the font size unchanged.

Advanced
This template extension adds the following code to your application:

1) A call to ds_SetDefaultFont() is added to the beginning of your code passing the relevant font details.

Extension : User Login Here

Visible Effect to your App
The user must enter a valid user code and password here before the screen will open.

Purpose
Use this extension wherever you want password protection. In the usual case, where the user logs in before running the Application, use this template extension on your 'main' function.

You can include any number of these extensions throughout your program. Each time one is encountered the user will be required to enter a valid user code and password in order to continue. In each section you can reset which users are operators, and which are supervisors.

When the function, in which you use this extension, is completed then the user is automatically logged out. The user then returns to the level, and accesses, they had before entering the function.

To include in a function in your App
1) Open the App in the normal fashion.
2) Point to the function you wish to password protect.
3) Click on the **Properties** button.
4) Click on **Extensions**.
5) Click on **Add**.
6) Choose **User Login Here** from the list of available extensions.
7) Enter the name of the area you are protecting.
8) Click on **"Allow Case Insensitive User Password"** if you want the user password to be case insensitive for this login.
9) Click on **"Make Login Optional to End User"** if you want this login to be optional. See the User Guide for more information on optional logins.
10) Click on **"Three Tries"** if you want the function to only allow 3 attempts before returning.
11) Click on **"Allow automatic logins from other Exe's"** if you want to be able to log in to this application using the login from another application. This option should only be clicked on for the first login screen.
12) Click on **"Allow default login values"** to pre-set the default login code and password. If this code and password is valid then this login will be accepted, otherwise the fields on the scren will be primed with these values. You must use quotes to enter fixed strings here, or put valid variable names here.
13) Clicking on **"Don't show screen if default fails"** will cause the Login function to automatically return 0 (ie Deny access) if the Default codes mentioned in 12 are invalid. This also applies to the automatic login mentioned in step 11 above.
14) Close the screens and return to the main tree view.

Advanced
This template extension adds the following code to your application:

1) At the **Procedure Setup** Embed point it calls the ds_LoginText function. if the login fails then it does an immediate return. If the login is successful then the value returned is stored in AppNum and added to the AppNumQueue.
2) At the **End of Procedure** embed point the user is logged out, and the previous value of AppNum is restored.

Extension : User Screen Security

Visible Effect to your App
At runtime a supervisor can press a hotkey (Ctrl F8 by default), on this screen, and get a list of operators, along with the list of restrictions. By using the mouse he can toggle access rights to individual operators. Only users designated as **Operators** for this area of the App will be displayed on the list.

Purpose
Include this extension in a function if you wish to restrict operators, on an individual basis, from all, or part of, the function. This will automatically check each operator as he enters the function to check if access to the screen is permitted, and also to check if access to controls on the screen are allowed. You can specify up to 7 other controls, apart from the actual access to the screen, that the operator can be restricted from. Controls that the operator does not have access to will be hidden.

To include in a function in your App
1) Open the App in the normal fashion.
2) Point to the function you wish to protect.
3) Click on the **Properties** button.
4) Click on **Extensions**.
5) Click on **Add**.
6) Choose **User Screen Security** from the list of available extensions.
7) Enter the name of the screen you are protecting. If you omit this name then the function name

will be used. These screen names must be unique within the application.
8) Enter the equate labels (including the ?) for up to 7 additional controls that you wish to protect. Under each equate label is a button containing more information about the equate. Click on the button to see the other prompts. These prompts include the following:

Name : This is the name that will appear on the browse screen that the supervisor will see when he presses Ctrl F8. These names are limited to 6 characters.
Action : Hide or Disable. This instructs the generator to either hide or disable the control if the user does not have access to it. Menu options cannot be hidden, they must be disabled.
Equivalent Equate : This allows you to restrict two controls together. ie the user either has access to both controls or neither control. This is especially useful where you are using a toolbar and you want to equate a menu item with a toolbar button. The action set above applies to both controls.

9) Close the screens and return to the main tree view.

Advanced
This template extension adds the following code to your application:

1) A local variable, called **ThisAllowed**, is declared to hold the result of the access query.
2) At the Procedure Setup embed point ds_Allowed is called to check on the access rights of the user who is currently logged in. If the user is an Operator, and does not have access rights then a message is displayed, and the function returns.
3) At the Before Accept embed point the DS_SECURITYKEY is alerted. The controls to which the user does not have access are hidden.
4) At the Accept Loop Before Event Handling embed point, the keycode is checked to see if the hotkey was pressed. If it was then ds_SetAccess is called. If the user is not a supervisor then ds_SetAccess will do an immediate return.
5) At the Window Event Gain Focus embed point ds_Allowed is called. This is to check if the user changed (ie by calling ds_ChangeLogin) while the window did not have the focus.
6) At the End of Accept Loop embed point the controls which the user does not have access to are hidden or disabled.

Extension : User Workgroup Filtering

Visible effect to your App
Allows you to use the CWG variable in your browse filtering.

Purpose
This allows you to filter records from the browse based on the current user's workgroup. You can set the User's workgroup on the User Details Form, which is called from the Browse Users Screen. This template will create a variable called CWG (a long) which stores the current users workgroup. You can then use this varaible in your browse filtering.

To include in your App
1) Open the App in the normal fashion.
2) Point to the browse function you wish to filter.
3) Click on the **Properties** button.
4) Click on **Extensions**.
5) Click on **Add**.
6) Choose **User Workgroup Filtering** from the list of available extensions.
7) Click on the **OK** button.
8) Add your desired filter in the normal fashion.

Advanced
This app adds the following code to your procedure.
1) A variable called CWG is defined as a short.
2) The CWG variable is 'Bound' just before the Accept loop starts.
3) A call to ds_CurrentWorkGroup is placed in the GainFocus Window Event handler. The result of this call is stored in CWG.

Note : No actual filtering is done by this template. You are free to implement filtering in the normal fashion. This template simply defines, binds, and primes the CWG variable for your use.

Code : Call Current

Visible effect to your App
Allows the user to call the Current Name, Current Login or Current Level function and place the result on the status bar.

Purpose
By using this template is is easy to see, on the status bar, the currently logged in user. You can use any one of the user name, user login code or user level. You can also prefix your own text or variable before the result of the function call.

To include in your App
1) Go to the Properties screen of your Main Menu procedure.
2) Choose **Embeds**
3) Choose **After Opening Window**
4) Choose the **Call_Current** code template
5) Enter the section of the status bar where you want the result to go. ie 1 or 2 or 3 etc. Note that you must ensure that the status bar is formatted correctly and has the section that you ask for.
6) Enter any text that you would like to appear on the status bar before the user details. For example you could put 'User Name : '. You must use quotes if you want the literal text to appear, otherwise the template assumes you are using one of your own variables.
7) Choose from the radio button selection which of the functions you want.
8) If you have a call to the **Call Change Login** code template in the procedure then you should repeat steps 4 through 7 for the embed point containing the Call Change Login code template. Make sure that the **Call Current** template comes after the **Call Change Login** template.

Advanced
This code template simply calls one of ds_CurrentName, ds_CurrentLogin or ds_CurrentLevel and places the result on the status bar. If ds_CurrentLevel is chosen then the answer is interpreted before placing on the status bar.

Code : Calling Change Login

Visible effect to your App
Allows the user to call the Change Login Screen.

Purpose
This code template is designed to be attached to a menu item or button. It calls the ds_ChangeLogin function.

To include in your App
1) Add a control somewhere in your program. Usually this control would take the form of a menu item in your main menu, but it could also be a button, or a toolbar button.
2) Click on **Actions** for the control.
3) Choose **Embeds**
4) Choose **Control Event Handling , after generated code, Accepted**.
5) Choose the **Call_ChangeLogin** code template.
6) Close the windows in the normal fashion.

Advanced
This code template simply calls the ds_ChangeLogin function with the AppNum passed as a parameter.

Code : Calling Change Password

Visible effect to your App
Allows the user to call the Change Password Screen.

Purpose
This code template is designed to be attached to a menu item or button. It calls the ds_ChangePassword function.

To include in your App
1) Add a control somewhere in your program. Usually this control would take the form of a menu item in your main menu, but it could also be a button, or a toolbar button.
2) Click on **Actions** for the control.
3) Choose **Embeds**
4) Choose **Control Event Handling , after generated code, Accepted**.
5) Choose the **Call_ChangePassword** code template.
6) Close the windows in the normal fashion.

Advanced
This code template simply calls the ds_ChangePassword function with the AppNum passed as a parameter.

Code : Calling Lock Screen

Visible effect to your App
Allows the user to call the Lock Screen Screen.

Purpose
This code template is designed to be attached to a menu item or button. It calls the ds_LockScreen function.

To include in your App
1) Add a control somewhere in your program. Usually this control would take the form of a menu item in your main menu, but it could also be a button, or a toolbar button.
2) Click on **Actions** for the control.
3) Choose **Embeds**
4) Choose **Control Event Handling , after generated code, Accepted**.
5) Choose the **Call_LockScreen** code template.
6) Close the windows in the normal fashion.

Advanced
This code template simply calls the ds_LockScreen function with the AppNum passed as a parameter.

Code : Calling Operator Browse

Visible effect to your App
Allows the user to call the Operator Browse Screen.

Purpose
This code template is designed to be attached to a menu item or button. It calls the ds_OperatorBrowse function with the appropriate parameters.

To include in your App
1) Add a control somewhere in your program. Usually this control would take the form of a menu item in your main menu, but it could also be a button, or a toolbar button. This control should be visible from all the security areas within your App, so you may need to add more than one control.
2) Click on **Actions** for the control.
3) Choose **Embeds**
4) Choose **Control Event Handling , after generated code, Accepted**.
5) Choose the **Call_OperatorBrowse** code template.
6) Close the windows in the normal fashion.

Advanced
This code template simply calls the ds_OperatorBrowse function with the AppNum variable passed as the parameter.

Code : Calling SetPin

Visible effect to your App
None

Purpose
This code template stamps the DSSW file with an Application PIN number. Pin numbers enhance application security be ensuring that only correctly "stamped" DSSW files will be considered valid for the application. A full description of the PIN number concept, and when you would use this function, can be found in the User Guide in the section entitled Application PIN numbers.

To include in your App
1) Remember this function would not be used inside the Application you are intending to protect.
2) Add a control somewhere in your program. This control could be a menu item in your main menu, or a button, or a toolbar button.
2) Click on **Actions** for the control.
3) Choose **Embeds**
4) Choose **Control Event Handling , after generated code, Accepted**.
5) Choose the **Call_SetPin** code template.
7) Fill in the Application name and number that you wish to stamp into the file, using single quotes around the application name. If you wish to use a variable instead then fill in the variable name here, without any quotes.
6) Close the windows in the normal fashion.

Utility : Import Queue Handling

Visible effect to your App
None.

Purpose
This template is used in mutil DLL projects. It should only be used in Root DLL's of such projects.

This utility adds three functions to your DLL. They are qAdd, qGet and qDelete respectivly. These are needed so that other DLL's, and the Exe can access your AppNum Queue.

To include in your App
1) Open the application in the normal fashion. (This is the root DLL app).
2) Click on the Application Menu.
3) Choose UtilityTemplate.
4) Choose Import Queue Handling.

Advanced
The three functions added are all extremely simple. They simply access a local queue by either adding, fetching (getting) or deleting an item. Only the first item in the queue can be accessed - so it behaves similarly to a stack.

Utility : SecWiz

Visible effect to your App
All basic SecWin features are added to your application automatically. This includes a Login screen, Screen security for all other screens, and Menu options to call the basic support functions.

Purpose
This template vastly reduces the time required to add basic SecWin security to an existing application. It automatically (optionally) performs the following functions;

Adds a User Login Here extension template to the Main procedure.

Adds a User Screen Security template to all your Browses, Forms and Windows. Insert, Change, Delete, Ok and Print buttons are all added to the list of controls wich can be limited individually.

Adds calls to OperatorBrowse, ChangePassword, LockScreen and ChangeLogin to the main menu. These menu options can be moved (or removed) later if you wish.

The only other step required to add basic SecWin security to your App, is to add the Global Extension Template "Activate Security".

To include in your App
1) Open the application in the normal fashion.
2) Click on the Application Menu.
3) Choose UtilityTemplate.
4) Choose SecWiz.
5) Follow the on-screen promptings about which features you would like.

Note : Running the SecWiz template multiple times does no harm to your App. In fact if you expand your app after running SecWiz for the first time, by adding more screens, then the easiest way to add SecWin features to the new screens is to re-run SecWiz.

Note : The SecWiz template is only available in the registered version.

SecWin Technical Reference

This guide describes the functions exported by the SecWin DLL.

Contents

ds_Allowed (ApplicationNumber,FunctionName,Override)

Parameters

ApplicationNumber long
This is the Application number returned by ds_LoginText.

Function Name string
This is the name of the function to which the user is requesting access. This parameter is not case sensitive.

Override long
This is an attribute that contains any overrides that the program may use to override the users access levels. This is not currently used and should be set to 0.

Purpose

This function enables the application to get the access rights for a specific user for a specific part of the program (usually a screen.) The function is called using the ApplicationNumber as well as the FunctionName. The function returns a long (although currently only the low 8 bits are supported), in which each bit signifies that access has been granted or denied to a particular section. If the bit is set the access has been granted, if it is not set then access has been denied.

These accesses only apply to users designated as 'Operators' on the Operator Browse screen. You cannot limit the access of an user designated as a 'Supervisor'.

Returns

A long which can be bitwise tested determine the access rights that the user has been granted. At the moment it is only possible to use the low 8 bits of the returned long. These access rights are set by calling ds_SetAccess

See Also

ds_LoginText
ds_SetAccess
User Guide : Set Access Screen
User Guide : User Levels

Example

```
ViewVideo               Procedure

window    WINDOW('Caption'),AT(-1,2,185,92),SYSTEM,RESIZE,MDI
           BUTTON('Delete'),AT(68,28,,),USE(?DeleteButton)
           BUTTON('Change'),AT(127,35,,),USE(?ChangeButton)
           BUTTON('Cancel'),AT(74,53,,),USE(?CancelButton)
           BUTTON('Ok'),AT(17,25,,),USE(?OkButton)
           BUTTON('Print'),AT(19,51,,),USE(?PrintButton)
         END

ThisAllowed             long

 CODE

 ThisAllowed = ds_allowed(Application Number,'ViewVideo',0)
 if ~band(ThisAllowed,1) then return.
 OPEN(window)
 WindowOpened=True
 if ~band(ThisAllowed,2) then hide(?OkButton).
 if ~band(ThisAllowed,4) then hide(?ChangeButton).
 if ~band(ThisAllowed,8) then hide(?DeleteButton).
 if ~band(ThisAllowed,16) then hide(?PrintButton).
 alert(DS_SECURITYKEY)
 ACCEPT
```

ds_ChangeLogin(ApplicationNumber)

Parameters
ApplicationNumber   long
This is the Application number returned by ds_LoginText.

Purpose
This allows another user to Login without quitting the application. The login screen will behave using the same options as was last passed to ds_LoginText.

Returns
Nothing

Example
```
 code
 ds_ChangeLogin(AppNum)
```

ds_CurrentLevel(ApplicationNumber)

Parameters
ApplicationNumber   long
This is the Application number returned by ds_LoginText.

Purpose
Allows your application to interogate the DLL for the level of the user who is currently logged on. This can be useful for displaying the level of the current user on the status bar etc.

Returns
A long containing the level of the user currently logged on. The returned value will be either DS_SUPERVISOR or DS_OPERATOR.

Example

```
currentlevel   long
 code
 currentlevel = ds_CurrentLevel(AppNum)
```

ds_Current Name(ApplicationNumber)

Parameters
ApplicationNumber long
This is the Application number returned by ds_LoginText.

Purpose
Allows your application to interogate the DLL for the name of the user who is currently logged on. This can be useful for displaying the name of the current user on the status bar etc.

Returns
A string containing the name of the user currently logged on. The maximum length of the string is 40 characters.

Example

```
currentname   string(40)
 code
 currentname = ds_CurrentName(AppNum)
```

ds_Current Login(ApplicationNumber)

Parameters
ApplicationNumber long
This is the Application number returned by ds_LoginText.

Purpose
Allows your application to interogate the DLL for the Login Code of the user who is currently logged on. This can be useful for displaying the login of the current user on the status bar etc.

Returns
A string containing the login code of the user currently logged on. The maximum length of the string is 8 characters.

Example

```
currentlogin   string(8)
 code
 currentlogin = ds_CurrentLogin(AppNum)
```

ds_Current WorkGroup(ApplicationNumber)

Parameters
ApplicationNumber long
This is the Application number returned by ds_LoginText.

Purpose
Allows your application to interogate the DLL for the Workgroup of the user who is currently logged on. The users' Workgroup is set on the users form by a Supervisor. Workgroups are a way of grouping users for browse filter purposes.

Returns
A short containing a number from 0 to 32767. The number returned is the number entered in on the User form.

Example

```
CWG         short
 code
 CWG = ds_CurrentWorkGroup(AppNum)
```

ds_ChangePassword ( )

Parameters
None

Purpose
Allows the user who is currently logged in to change his or her password. The user will be required to enter his existing password, and then his new password twice. There is only one password per user, regardless of the number of applications or security areas.

Returns
Nothing

Example

```
 code
 ds_ChangePassword
```

ds_LockScreen (ApplicationNumber)

Parameters
ApplicationNumber long
This is the Application number returned by ds_LoginText.

Purpose
Allows the user who is currently logged in to lock the application while being away from the terminal temporarily. The screen is blacked out and the user must enter their password before the application will continue. Note that the machine is not locked as the Windows task manager is still active. The application can also be terminated using the task manager.

Returns
Nothing

Example

```
 code
 ds_LockScreen(AppNum)
```

ds_Login (SecurityAreaName)

This is exactly the same as calling ds_LoginOpt(SecurityAreaName,ds_Default). It is included for compatibility with older Apps. See the documentation on ds_LoginOpt for more information on this function.

ds_LoginOpt (SecurityAreaName,Options)

This is exactly the same as calling ds_LoginText(SecurityAreaName,Options,'',''). It is included for compatibility with older Apps. See the documentation on ds_LoginText for more information on this function.

ds_LogInText (SecurityAreaName,Options,DefL,DefP)

Parameters

SecurityAreaName string

This is the name of the area being protected. Each name must be unique, even across applications, so by convention, this name is normally made up of the application name and the area name, separated by a pipe (|) symbol. The maximum length of this parameter is 43 characters, and it is not case sensitive.

Options long

This parameter governs the behavior of the Login function. Multiple options can be added together. Possible values are :

DS_DEFAULT (0) : Default behavior. A login is definitly required, and the password is case sensitive. Automatic logins are not allowed, and unlimited login tries are accepted.

DS_OPTIONALON : This indicates that this login is optional, and is governed by your user at runtime. If <u>any</u> user has been given access to this area, then a login is required. If no user has been assigned access then no login screen is presented. When inside an optional area all users have supervisor status ie no Screen Security is in place. A user can change the optional status of the area by entering the area, then opening the Browse Users screen. They can then assign a supervisor to the area, and from then on the area is not optional, until all users have been assigned 'No Access' again.

DS_CASESENSITIVITYOFF : If this flag is present then the User password is not case sensitive.

DS_THREETRIES : Limits the number of login attempts to 3 before the program automatically aborts the login attempt.

DS_USECURRENTLOGON : Allows this application to accept automatic login attempts, using the ds_Run function .

DS_AUTOLOGON : Allows this application to accept automatic login attempts, using the DefL and DefP parameters

DS_DONTSHOWSCREEN : If this flag is set, and the DS_AUTOLOGON flag is set, and the attempted automatic login fails, then the function will immediately return 0, without opening the login screen. If this flag is not set, and an automatic login attempt fails, then the normal login screen will be opened.

DefL string

This is the default Login code. In order for this to be used by this function, the DS_AUTOLOGON option flag must be set.

DefP   string
This is the default Password. In order for this to be used by this function, the DS_AUTOLOGON option flag must be set.

Purpose
Allows a user to login to the program, or program area. A window will be opened allowing the user to enter a user code and a password.
If there are no users defined for this application then one of the following two actions will occur.

1) If security is not optional then you will be asked to either assign an existing user to this area as a Supervisor, or to add a new user as a Supervisor. If you add a new user then the initial password is the login code.

2) If security is optional then no window will be opened and the user will automatically be given the Supervisor level.

If the DS_USECURRENTLOGON flag is set then the function will attempt to log the user in using the same login code and password of the calling program. The calling program uses the ds_Run function (or template) to run this program.

If the DS_AUTOLOGON flag is set then the function will attempt to log the user in using the DefL and DefP parameters.

If either of these attempts fail, then the normal login screen will be displayed. If the DS_DONTSHOWSCREEN option flag is set then the function won't open the screen, but will return 0.

Returns
The Application Number (long).
This number is used by other functions so it must be stored, a global, non-threaded long is the perfect storage vessel. If the number returned is 0 then the login was unsuccessful and appropriate action should be taken.

Advanced
If you have multiple login areas within one application then store the application number (AppNum) in a queue, adding the new AppNum to the top of the queue when successfully logging in, and removing it when logging out. This ensures that AppNum always points to the current user area.

Calling ds_LoginText automatically logs out the existing user for this instance of the area,if one exists. In other words calling ds_LoginText(SecurityAreaName,x) causes an implicit ds_Logout(ApplicationNumber) - ie a logout of the same area - to be called before ds_LoginText. If no user is logged in then ds_Logout ignores the request.

See Also
ds_Logout

Example, for applications with one login area

```
AppNum       long

 code
 AppNum = ds_LoginText('Video.Exe',DS_OPTIONALON |
                          +DS_CASESENSITIVITYOFF,'','')
 if AppNum = 0
   ! login was unsuccessful
   return
 .
```

Example, for applications with multiple login areas

```
AppNumQueue    Queue
AppNum           long
               .
 code
 AppNum = ds_LoginText('Video.Exe',DS_DEFAULT)
 if AppNum = 0
   ! login was unsuccessful
   return
 .
 Add(AppNumQueue,1)
 !< Function code goes here >
 ds_Logout(AppNum)
 delete (AppNumQueue)
 get (AppNumQueue,1)
 return
```

ds_LogOut (ApplicationNumber)

Parameters
ApplicationNumber   long
This is the Application number returned by ds_LoginText.

Purpose
Allows a user to logout of an application, or application area. The user will have to re-enter his login code and password before he can reenter the part of the application protected by the password.

It is not necessary to LogOut if the program is being terminated, but is considered good behaviour. Future versions of SecWin may require logging out.

Returns
0 if successful. 1 if unsuccessful.
If the function returns 1, then the user was not logged in to the application.

See Also
ds_LoginText

Example

```
AppNum      long
Result      long

 code
 AppNum = ds_LoginText('Video.Exe',ds_DEFAULT)
 if AppNum = 1
   ! login was unsuccessful
   return
 .

 < function code goes here >

 result = ds_LogOut (AppNum)
 return
```

ds_OperatorBrowse (ApplicationNumber)

Parameters
ApplicationNumber long
This is the Application number returned by ds_LoginText.

Purpose
Allows you to Browse the users for this Application or Security area. Only Supervisors have access to this screen. A Supervisor must use this function to add, edit or remove other Supervisors or Operators.

Returns
Nothing

See Also
ds_LoginText
User Guide : Operator Browse Screen
User Guide : User Levels

Example

```
AppNum     long

 code
 ds_Browse(AppNum)
```

ds_Run (ApplicationNumber)

Parameters
ApplicationNumber long
This is the Application number returned by ds_LoginText.

Purpose
Allows you to launch another SecWin enabled EXE, performing an automatic login using the currently logged in user.

In other words, App A can use this function to run App B, at the same time passing App B the login code and password of the current user. App B will automatically log the person in.

In order for this to occur App B **MUST** have the login option DS_AUTOLOGON bit set. On the Login User Here template this option is referred to as "Allow automatic login's from other Exe's".

Returns
A string which needs to be included as a parameter in the Clarion "Run" statement.

See Also

ds_LoginText
User Guide : Operator Browse Screen
User Guide : User Levels

Example

```
AppNum     long

 code
 run('ProgName ProgParam ' & ds_Run(AppNum))
```

ds_Security (Action)

Note : This function is now redundant. It's documentation is included for completeness sake only. From Version 1.6 it has no effect.

Parameters
Action   long
This is one of the following equates. You may not use more than one equate at a time.
DS_OPTIONALON
DS_OPTIONALOFF

Purpose
This instructs SecWin to turn certain features on and off at will. For example by default applications using SecWin require at least one user code, and users have to login using a valid user code and password.

If the security is set to DS_OPTIONALON, and no users are declared for the application area, then no login is required. This setting is Application wide, regardless of login areas.

If the security is set to DS_CASESENSITIVEOFF then user passwords are case insensitive, but only for this application.

Returns
Nothing

Example

```
AppNum      long

 code
 ds_Security(DS_OPTIONALON)
 AppNum = ds_Login('Video.Exe',ds_DEFAULT)
```

ds_SetAccess (ApplicationNumber,ScreenName,Options)

Parameters
ApplicationNumber   long
This is the Application number returned by ds_LoginText.

ScreenName   string
This is the name of the screen or function. This name must be unique within an application.

Options   string
This string contains all the access options for this function. Each item in the string is separated by a pipe (|) character. There may be up to 8 items in the string. Each item must be 6, or fewer, characters long. These items then appear on the Set Access screen in different columns.

Purpose
This screen is normally called from within a specific function in order to change the access levels for the different operators, for that screen. Only Supervisors have access to this function. The access levels are retrieved using the ds_Allowed function. ds_SetAccess is normally invoked by pressing a hotkey. This hotkey is normally stored in the DS_SECURITYKEY equate. By default this key is Ctrl-F8. The function can however be called in almost any manner, for example by using a button on the window.

The Options parameter string contains all of the access options for the calling function, as you want them to be displayed on the ds_SetAccess browse box.

Returns
Nothing

See Also
ds_LoginText
ds_Allowed
User Guide : The Set Access Screen

Example

```
ViewVideo                       procedure

 code

 < normal opening window code etc goes here >

 alert(DS_SECURITYKEY)
 accept
   if keycode() = DS_SECURITYKEY
     ds_SetAccess (AppNum,'ViewVideo','Access|Delete|Print')
   .
 < rest of accept handling goes here >
 .
```

ds_SetLanguage (LanguageFileName)

Parameters
LanguageFileName string
This is the name of the language file to use. If this name is invalid, or the file cannot be found, then SecWin will default to English.

Purpose
Allows you to adjust the text which appears on the SecWin screens. This is used to translate the native English of the SecWin screens into other languages.

Returns
Nothing

See also
User Guide : Internationalisation support

Example

```
AppNum     long

 code
 ds_SetLanguage('Spanish.Irf')
```

ds_SetLogo (LogoFileName, CallbackFunctionAddress)

Parameters
LogoFileName string
This is the name of the logo file to use.

CallbackFunctionAddress long
This parameter is reserved for future use. You should set this to 0.

Purpose
Allows you to change the logo that appears on all of the SecWin screens.

Returns
Nothing

Example

```
AppNum     long

 code

 ds_SetLogo('c:\logo.ico',0) ! use logo in root directory
 ds_SetLogo('logo.ico',0) ! use logo in current directory
```

SecWin User Guide

This guide describes the runtime use of the SecWin DLL.

Contents

Introduction

SecWin provides password protection for your Windows applications. It also allows you to restrict users, at runtime, from viewing all or part of certain screens and from accessing certain controls on the screen. These two methods of security are referred to here as Password Security and Screen Security. This User Guide documents these two security options, as well as the Screens included in the DLL. There is also a section for adding SecWin functionality to hand code.

In addition to this, this section discusses some of SecWins more advanced features, from a design point of view. This is a good section to read to find out more general information on a specific topic.

Password Security

You can choose to password protect an entire program with one login and password, or divide the program into multiple areas, each area requiring its own login. In the case of multiple logins the user would use the same user code and password for all the areas.

For the sake of clarity I will refer to **security areas** in the documentation that follows. If your program has only one login point then it is in effect one security area. Equally you may choose to not password protect the main program, but protect only certain areas of the program, such as report generation. In that case the program may have one or more different security areas.

Security areas are EXE independant. That means that if you have three EXE applications, each of which has one security area, then SecWin would treat that the same as one EXE with three areas.

Multiple applications using the SecWin DLL will share a common list of users and passwords. Thus a user has the same login code and password to all the EXE's that he has access to.

In order to use password protection in any function in the program you must include the **User Login Template Extension** (or a hand coded equivalent) in that function. The first time you use this extension template you will also need to include the **Activate Security Template Extension** in the **Global Properties** for the Application.

Screen Security

The other form of security that SecWin offers is the ability to limit access to certain screens, and certain controls, on a user by user basis. This is called screen security.

In order to use screen security, the user must have logged in to a security area. In this way SecWin knows who is using the program, and is therefore able to manage access restrictions.

Only Supervisors can impose or modify screen security, and only on Operators. (see the next section for a fuller discussion about Supervisors and Operators).

To modify the screen access rights of an operator a supervisor must go to the screen in question and press the security hot key (**Ctrl F8** by default). The **Set Access Screen** will appear to allow the access rights to be modified.

In order to use screen security on a particular screen you must use the **Screen Security Template Extension** (or a hand coded equivalent) in that function.

User Levels

Users have one of three levels for a specific security area. Note that being a supervisor in one area does not imply that the user is a supervisor in all areas. He may be a supervisor in one area, and an operator in another, with no access at all to a third area.

Supervisor : The user has access to the entire security area, and access cannot be restricted to any part of the area. Only supervisors have access to the list of operators, and only supervisors are able to change the access rights of other users. There must be at least one supervisor defined for each security area.

Operator : The user has access to the security area, but this access can be limited by a supervisor. The access is limited on a screen by screen, and control by control, basis. An operator is not able to modify other user's access rights, or view/edit the list of users.

No Access : The user does not have access to this security area.

Workgroups

Workgroups are a method of restricting data visible to your users, in a browse,on a group basis.

For example let's say your users are divided into departments. Clients which appear on your Client Browse screen should be limited to the same department as the user. ie Manufacturing's clients should only be visible to manufacturing users, and Sales' clients should only be visible to Sales users.

SecWin allows you to give each user a "WorkGroup" number. What this number means is entirely up to you. The currently-logged-in user's workgroup number can be returned by a call to ds_CurrentWorkGroup. The workgroup is strictly restricted to the current login area, and the users workgroup number can differ from login area to login area.

Two methods of using WorkGroups are possible, depending on the applications requirements. Firstly, if the users belong to one, and only one, department then each number could simply be eqivalent to a department. For example Workgroup number 1 = Department 1, workgroup 2 department 2 etc. This affords a maximum number of about 32000 workgroups. It would be wise in this case to let Workgroup 0 signify all departments, ie all data is visible.

The second method is used if users can belong to more than one group. In this case each bit of the workgroup can signify a number. This implies a maximum of 15 workgroups.

Note : Although workgroups apply equally to Supervisors as they do to Operators, Supervisors are able to change their own workgroup number, and therefore have indirect access to all the workgroups.

Internationalisation

From version 2.0, SecWin incorporates features supplied by Pro Domus' CwIntl package to provide SecWin users with even more control over the SecWin screens. In essence SecWin now allows you to keep the text used by the SecWin screen in a text file. This means that translating SecWin screens into other languages is trivial.

First one or two bits of terminology.
A *language file* (note the small letters) is a text file containing a list of translations, from the original English into whatever language you like. The exact format of the file will be discussed later.

LanguageFile (note the big letters, and the lack of a space) is a SecWin.INI file parameter, containing a DOS file name (including drive and path) of a valid language file. This file normally uses the .IRF extension.

LanguagePath is a SecWin.Ini parameter that is used with the ds_SetLanguage function to change languages on the fly. It contains a valid drive and path.

Choosing a language
SecWin allows you to choose your language using two methods, the first is via the use of an INI file, and the second is via a new function (ds_SetLanguage) at runtime.

When your SecWin enable App is started SecWin looks for two parameters namely LanguageFile and LanguagePath. SecWin looks first in the Application directory, in the SecWin.Ini file, in the [SecWin] section. If either, or both are not there then it looks in the Windows directory, in the Win.Ini file, in the [SecWin] section. If LanguageFile is still not present then it defaults to a file called SecWin.Irf, in the Windows directory. If LanguagePath hasn't been found then it defaults to the Windows directory. If LanguagePath is set to 'Here' (no quotes) then it will be set to the application directory.

LanguageFile is a full path- and filename to a valid SecWin language file. The exact format of a valid SecWin language file is discussed a little later on. SecWin uses LanguageFile as the default language file containing the required language phrases. If this file does not exist, then SecWin uses the normal English phrases on the screen.

During the running of your application you can set, or change, the language file being used by calling the ds_SetLanguage function. This function takes one parameter, the name of a language file. This parameter is added to the LanguagePath parameter (discussed above), and then used as the current language file.

eg
Inside the SecWin.Ini file...
```
[SecWin]
LanguageFile=c:\Windows\Spanish.Irf
LanguagePath=c:\Lang
```

Using the above SecWin will default to using the Spanish.Irf language file in the Windows directory, on the C Drive.
At runtime the application could call the ds_SetLanguage function, like this...
```
ds_SetLanguage('Russian.Irf')
```

Which would change SecWin to use the Russian.Irf file, in the Lang subdirectory on the C Drive.

The last point to discuss is the structure of the Language file itself.

Format of a SecWin language file
The format of a SecWin language file is very simple. It is made even simpler by the fact that a sample language file (English.Irf) ships with SecWin, and thus simply copying this sample file and

translating the english terms is all you really need to do. Note that SecWin doesn't use the English.lrf file, it is only included as an example.

Inside the file there are a number of Sections, each one enclosed in a set of square brackets, like this... [engds_OperatorBrowse].
Inside each section are a number of field equates that identify each visible text field on the screen. On the right of the equals sign is the actual text that would be displayed. For example the following.

```
[engds_operatorbrowse]
BrowseWindow=Browse Users
?ListMsg=Browsing Records
?ListFormat1=First Name
?ListFormat2=Surname
?ListFormat3=Login Code
?ListFormat4=Level
?ListFormat5=Work Group
```

By replacing the text to the right of the equals sign, you can get the ds_OperatorBrowse screen to display anything you like.

At the top of the file is a section called [Intl], and this has a single entry called sLanguage. eg

```
[Intl]
sLanguage=eng
```

This is used to find the correct section later on. You'll have noticed above that the function name (eg ds_OperatorBrowse) is prefixed with this. This is a requirement of the way the support works. If you change the value in the Intl section, then you must change the prefix in all the other sections as well.

At the bottom of the file is a section called [enuMessages]. This section contains the messages that appear when SecWin needs to communicate with the user. You will need to translate these messages as well.

Application PIN numbers

One of the only current methods of compromising SecWin security is through the replacement of the DSSW security file by an empty DSSW file. Application PIN numbers prevent this from being effective by requiring that a DSSW file be "stamped" with a PIN number before the Application will accept it as valid.

The PIN number system introduces two new functions, and their associated templates. ds_SetPin is used to stamp a PIN number into a DSSW file, and ds_GetPin is used to read the number out of the file.

If you choose to make use of the PIN number system, then you will need some way of stamping the DSSW file with the pin number. This can be done in a number of ways, some of which are laid out belowl

1) Use another EXE program, over which 100% physical control can be maintained. It shoult not be accessable to un-authorised users, and can be treated as a physical key. The PIN number itself is hard-coded into the program, and can be run by anyone with access to it. An example of such a program ships with SecWin and is called HARDPIN.

2) Use the enclosed SETPIN program which ships with SecWin. This program requires that the PIN number be entered by the user. The user will then need to know the correct PIN number in order to properly stamp the DSSW file.

What about Btrieve ?

From version 2.0 support for the Btrieve driver has been added. This is primarily to support Wide Area Networks (WANS). However a few items should be considered before leaping on to the Btrieve wagon.

Firstly, and most obviously, you should use *either* the regular TopSpeed support, *or* Btrieve support - otherwise you will loose all the benefits of the central user database.

Secondly, you will need to license all the Btrieve DLL's yourself. None of the Btrieve files are distributed, or licensed for distribution, with SecWin. See your Clarion User Guide for more information about Btrieve Licensing.

Thirdly, SecWin currently only supports 16 bit Btrieve. Later versions of Clarion that support 32 bit Btrieve will allow 32 bit support to be added, but I cannot guarentee when it will occur, or indeed if it will occur at all.

Fourthly, Btrieve support is only included with the registered version of SecWin.

In conclusion, you should only use the Btrieve support if network performance is an absolute priority, if you are already using (and have licensed) the Btrieve DLL's, and if you don't require 32 bit support.

All about INI files

INI files are not required by SecWin, but they can be used to make installing SecWin enabled applications easier on user machines, as well as on networks. SecWin assumes that the INI files will be created by the developer or installer. SecWin does not create, or maintain, and INI files of it's own.

SecWin looks in 2 ini files for possible settings. The first is SecWin.INI in the application directory. If this does not exist, or if a setting is not present in the file, then it also looks in the Win.Ini file, located in the current Windows directory.

SecWin always looks for a section called [SecWin], regardless of the Ini file being used.

The following settings are looked for in the above files;

1) **Path**
This tells the SecWin DLL where to find the data file (usually DSSW.TPS) containing all the user's settings. If it is set to "Here" then the application directory will be used. If it doesn't exist then it will default to the current Windows directory.
```
eg   [Secwin]
    Path=Here
or   [SecWin]
    Path=c:\
```

2) **LanguageFile**
This contains the name of a valid SecWin language file that should be used, by default, by the Dll. If it doesn't exist then SecWin doesn't use any language file, but defaults to the built in text (which is in English). Language files are created by the developer. An example language file, English.lrf, ships with SecWin and can be found in your \cw15\LibSrc directory.
```
eg   [Secwin]
    LanguageFile=Spanish.Irf
```

3) **LanguagePath**
This contains a valid path name which points to a directory containing language files that will be loaded by the program, at runtime, using the ds_SetLanguage command. This setting is prefixed to the ds_SetLanguage parameter to create the complete filename of the required language file.

If it is set to "Here" then the application directory is used. If it doesn't exist then it defaults to the current Windows directory.

```
eg   [Secwin]
    LanguagePath=Here
or   [Secwin]
    LanguagePath=c:\
```

The Operator Browse Screen

This screen lists all the users that have been entered using SecWin. Alongside each user is the current access level for this particular security area. In order to modify the user levels you must ensure that supervisors have access to this screen from within the security area.

You can also view the users' workgroup on this screen. For a fuller discussion of Workgroups see the previous section.

You may not set users as operators for this security area without having set at least one user as a supervisor.

From this screen you can Add, Edit and Delete operators using the operator Form screen.

The Operator Form Screen

This screen contains the details about the operators including their name, surname, login code and access level. The access level applies to this security area only, in all other areas new users will automatically be set to the level of No Access. You will need to enter each of those areas in turn in order to set the user to a different level.

If the user is set to the level of Operator, within this area, then you can also specify the starting screen access rights that the operator will have. This takes the form of a pop down list with the names of other defined operators. Thus you can equate the starting access rights of a new operator with the current access rights of another operator. The options **All Access** and **No Access** are also on this list.

When a new user is first defined his password is automatically set to the same as the login code. The user should obviously change this password after he has logged in.

The Set Access Screen

This screen will appear when a supervisor presses the security hotkey (**Ctrl F8** by default) on a screen that has included the **User Screen Security Template Extension**.

This screen will list the users currently defined in the security area as Operators. Next to each name will be up to 8 columns containing the words Yes or No. The columns are headed by the names of the different controls that may be restricted. The first column is usually headed by the word **Access** and this column controls access to the screen as a whole.
By using the mouse and double-clicking on the columns the supervisor can change the access rights to the window, or controls. These changes are automatically saved.

Adding SecWin to Source Code Procedures & Applications

You may find that you need to add SecWin features to applications or procedures that are hand-coded. The following section details the function calls required to implement SecWin by hand, without templates. Each section basically traces the steps that each template performs, and explains how to implement the equivalent of the template by hand. If an item is in italics then the word is meant to be replaced by your equivalent name. For information on using the templates in an Application, see the SecWin Template Reference. For more information, and more examples, about the use of the ds_ functions refer to the SecWin Technical Reference.

Extension : Activating SecWin features
This template is used to enable SecWin in the application. It creates some global variables, defines the global equates, prototypes the DLL procedures in the Map and adds SecWin to the project file. The following outline the steps necessary to handcode this template.

1) In the project, under External Libraries, add a reference to SecWin10.Lib CW15 users should add a reference to SecWin16.Lib.

2) In the Program Map include the following:
```
 Include('SecMap10.Clw') ! CW 1.5 users use SecMap15.Clw
```

3) In the Global Data area add the following declarations:
```
  AppNameDesc  string(20)
  AppNumQueue  Queue
  AppNum         long
               End
   Include ('SecEqu10.Clw') ! CW 1.5 users use SecEqu15.Clw
```

4) In the main module, just after the CODE statement put:
```
   AppNameDesc = 'ApplicationName' !replace with the name of your App
```

Extension : User Login Here
This template prompts the user for a login code and password before displaying the screen. It can be used multiple times in the same application.

Functions that call the ds_LoginText function must call the ds_Logout function at the end of the procedure. This template handles both Logging in and out. In all of the following code an AreaName is used. This basically identifies the area within the Application which the user is logging in to. Replace AreaName with the text string of your choice.

1) After the CODE statement for the procedure add the following:
```
  LocalRequest = GlobalRequest           ! Save GlobalRequest
 LocalResponse = RequestCancelled        ! In case the login fails
 AppNum = ds_LoginText(Clip(AppNameDesc) & ' | AreaName',DS_DEFAULT)
 If AppNum = 0 then
   If Message('Access Denied','Security',Icon:Exclamation).
   get(AppNumQueue,1)
   Return                                ! maybe return a value
 .
 Add(AppNumQueue,1)                      ! add the successful login
 GlobalRequest = LocalRequest            ! Restore GlobalRequest
```

2) At the end of the procedure, before the RETURN statement, add the following:
```
 If ds_LogOut(AppNum). !Throw away the value returned from Logout
 Delete(AppNumQueue)
 Get(AppNumQueue,1)
```

Extension : User Screen Security
This template is the most complex, and manages the Control by Control Screen security. In includes the facility to allow a Supervisor to change the User Access at runtime, as well as Hiding or Disabling controls that are restricted for this user.

1) In the Local Data Declaration (ie before the CODE statement) area add the following
```
ThisAllowed        long    ! contains the users access rights
```

2) After the CODE statement add the following.
```
 ! get and check users access rights to this function
 ThisAllowed = ds_Allowed(AppNum,'ScreenName',0)
 If Band(ThisAllowed,1) = 0
```

```
    If Message('Your access to this part of the program ' & |
     'has been restricted','Security' ,ICON:Exclamation).
    GlobalResponse = RequestCancelled
    Return
  .
```

Replace *ScreenName* with the name of the Screen.

3) Just before the ACCEPT command add the following:

```
 Alert(DS_SECURITYKEY)
 If ~Band(ThisAllowed,2) then Disable(?Equate1).
 If ~Band(ThisAllowed,4) then Hide(?Equate2).
```

Replace the *?Equates* with equates to the controls. In the above 2 lines 2 different bits were tested. One was the bit-value 2, and the other 4. You may also test the bit-values 8, 16, 32, 64 and 128. Use Disable and Hide interchangably as you wish. You may add as many of these lines as you need.

4) Repeat the code in step 3 (except for the ALERT command) just before the end of the accept loop. This is only necessary if the function uses DISABLE or HIDE in the rest of the procedure. This is most common in BROWSE and VIEW procedures.

5) Just after the ACCEPT command add the following:

```
 If Event() = Event:AlertKey
   If KeyCode() = DS_SECURITYKEY
     ds_SetAccess(AppNum,'ScreenName','RestrictString')
     SetKeyCode(0)
  . .
```

The ScreenName must be the same name as was used in Step 2. The restrict string is made up of up to eight short (6 characters or less) words which identify the controls being restricted. Each word is separated from the other by use of a pipe (|) character. Each word is associated with a bit-value, from the following. 1,2,4,8,16,32,64,128. The words, and the bits are arranged from right to left. The first word should always be ACCESS. Here are some examples:

```
Access|OK|Cancel|Delete
Access|Salary|Phone
```

6) For each equate that has restrictions, add the following code when processing the Event:Accepted event.

```
If ~band(ThisAllowed,n) then cycle.
```

The number *n* should be replaced with the relevant value, ie the same as the value in step 5 above.
This code should be before any other code. It eliminates the possibility of the control being accepted by using a shortcut method, for example the Keyboard, or the right mouse button.

Extension : User Workgroup Filtering
This is a simple template that created a variable called CWG (short) in your function, and primes it with the WorkGroup of the user who is currently logged on. You can use this variable in filter expressions to filter records based on the user's workgroup. For CW 1.5 users this variable is also 'bound'. To hand-code this template add the following code.

1) In the local data area declare a variable CWG as a short.
CWG short

2) Just before the Accept statement add the following :
Bind('CWG',CWG)

3) In the Window Event Handling, for the Event:GainFocus event add the following code :
CWG = ds_CurrentWorkGroup(AppNum)

4) Add your own code to the browse filter, based on the value of CWG, to filter out undesirable records.

Code : Call Current
This template calls on of the ds_Current functions and returns the result onto the status bar. If the ds_CurrentLevel function is called then the answer is interpreted before placing on the status bar. The usual place for this is in the embeds for the Main Menu. The correct embed to use is the After Window Opened embed. You should also use it after using the Call Change Login code template or after calling the ds_ChangeLogin function in code. For example

```
appframe{prop:statustext,3} = ds_CurrentName(AppNum)
appframe{prop:statustext,2} = ds_currentLogin(AppNum)
if ds_CurrentLevel = DS_SUPERVISOR
  appframe{prop:statustext,1} = 'Supervisor'
else
  appframe{prop:statustext,1} = 'Operator'
end
```

Code : Calling Change Login
This template calls the Change Login screen which allows the current user to logout and a new user to login without quitting the application. The usual place for this is in the Main Menu. Note that if you are using the Application Generator then you shouldn't use the 'Call Procedure' template. This would cause the procedure to be added to your Application as a todo. Use the following code in the source code or in an Embed. For example :

```
ds_ChangeLogin(AppNum)
```

Code : Calling Change Password
This template calls the Change Password screen which allows any user to change their password. The usual place for this is in the Main Menu. Note that if you are using the Application Generator then you shouldn't use the 'Call Procedure' template. This would cause the procedure to be added to your Application as a todo. Use the following code in the source code or in an Embed. For example.

```
ds_ChangePassword(AppNum)
```

Code : Calling Operator Browse
This template calls the Operator Browse Screen which allows Runtime access to the list of Supervisors, Operators and other Users. You can call it from anywhere in the application. The most usual place would be to put it on the Main Menu. Note that if you are using the Application Generator then you shouldn't use the 'Call Procedure' template. This would cause the procedure to be added to your Application as a todo. Use the following code in the source code, or in an Embed. For example

```
ds_OperatorBrowse(AppNum)
```

SecWin Version Update

This guide describes the changes to SecWin from version to version.

Contents

Introduction

This section of the documentation enables you to find changes from previous versions of SecWin easily. Please note that the documentation above has been updated, and does not require the following information. However if you have printed out a previous version of the SecWin manual then you will find this section of benefit to keeping your manual up-to-date.

Whereever possible references into the updated manuals have been provided.

Note : The features added for Version 2.0 are not avaliable directly to Clarion 1.0 users. You can however continue using SecWin 1.7 with Cw 1.0.

Version 2.0

Note : If you are using SecWin in a Multi-DLL project, then you will need to add the following two variables in the GLOBAL DATA section of your Root DLL. They must be added to the Data window, and not placed in the Global Data Embed so that they will be exported properly.

1) AppNameDesc	String
2) AppNum	Long

Note : If you upgrade an App from an earlier version of SecWin (prior to version 1.7) then you will probably get numerous Duplication Errors/Warnings when you re-compile the app. To remove them do the following -

1) Open the App
2) Click on Global
3) Click on Data
4) Delete the AppNumQueue and AppNameDesc variables.

First the bad news - the price of SecWin has increased from $ 59.99 to $ 99.99. I regret this, but unfortunately it is unavoidable. The good news is that SecWin has been signed up as a member of the TopSpeed Accessories program which means wider exposure (and hopefully sales) and this translates into continued improvement of the product. Users who have already registered will continue to receive their upgrades free of charge. On to the new features...

1) Support for automatic logging in to Exe's has been added. You can now Run an Exe, and automatically log the current user of the current Exe into the new Exe.
See also : Technical Reference | ds_Run, ds_LoginText
Template Reference | Run Code Template, User Login Here template

2) You can automatically log in to SecWin, passing the login code and password as parameters, instead of getting the familiar Login screen. This means you can write your own login screens, as well as doing automatic logins.
See also : Technical Reference | ds_LoginText
Template Reference | User Login Here template

3) Support for Application PIN numbers has been added. This gives the developer more control

over the creation of an acceptable DSSW.TPS file, and hence removes the "back door" possibility of deleting, and re-creating, the DSSW.TPS file.
See also : User Guide | PIN numbers explained
Technical reference | ds_SetPin , ds_UsePin
Template reference | Activate Security template, SetPin Code template

4) A new Wizard template has been added to the registered version to assist in the implementation of SecWin into an existing application. The entire process of adding SecWin features to your application has been automated. For more information see the SecWiz Utility template.
See also : Template Guide | SecWiz Utility template

5) Internationalisation support has been added by using by Pro Domus' CwIntl package. You can now translate the SecWin screens into your home language. SecWin even supports multiple languages at the same time for maximum compatibility with your own App. Thanks to the generosity of Pro Domus you do not require CwIntl in order to make full use of SecWin's International features.
See also : User Guide | Internationalisation
Technical Reference | ds_SetLanguage
Template Guide |

6) More network support has been added by allowing the DSSW.TPS file to be located in the application directory, without knowing in advance what that directory will be.
See also : Installation Guide | The CRESEC utility.

7) Support for Wide Area Networks has been added. SecWin now allows you to use the Btrieve driver for it's data files. Note that this feature is included only in the registered version of SecWin, and currently is only supported in the 16 bit library.
See also : User Guide | What about Btrieve ?

8) A new function, ds_SetLogo has been added. This allows you to replace the familiar SecWin logo with your own logo.
See also : Technical Reference | ds_SetLogo

9) More examples have been added. Along with the familiar EMPRPT example there are examples of Multi DLL and Multi EXE projects. There are also 2 Pin number examples called HardPin and SetPin. There is also a Internationalisation demo called Language.

10) More documentation has been added. Apart from the continued growth of this document, a comprehensive document included with the Multi DLL example gives a step by step illustration of creating Multi DLL projects, and implementing SecWin in such projects.

11) The "Change Password" screen now allows case-insensitive passwords. The same setting as that for the Login Screen are used.

12) The SecWin template now fully supports the use of interchangable compiling in 16 and 32 bit modes. The correct library files are used automatically. You no longer ned to manually add the SecWin libarary file to your project.

Version 1.6 & Version 1.7

Note : If you upgrade an App from an earlier version of SecWin then you will probably get numerous Duplication Errors/Warnings when you re-compile the app. To remove them do the following -

1) Open the App
2) Click on Global
3) Click on Data
4) Delete the AppNumQueue and AppNameDesc variables.

Version 1.7

1) Support for PowerBrowse template has been added. For example in the past pressing the **Del** key on a Power Browse screen worked, even if the DELETE button was disabled or hidden. This also applies to the Insert and Change buttons.

2) Login codes are now always case insensitive. There was a bug in earlier releases which made all login codes case sensitive.

3) The **Lock Screen** function now supports case insensitive passwords.

4) Users are no longer allowed a blank password, or one containing just spaces. The **Change Password** screen no longer accepts these as valid new passwords.

5) Some of the INI file support was not retro-added to the SecWin10.DLL. This has been fixed.

6) Cresec and Cresec15 have been updated to support the new INI file conventions.

7) A new option, "Three Tries" has been added to the user login template. This instructs the LoginText function to return after Three unsuccessful attempts.
See also : Template Guide | User Login here

8) When the "Change Login" function is called, then the controls in the window, and any other open windows are reset to the conditions available to the new user.

9) A new function has been added, ds_SetDefaultFont, to assist in getting the SecWin screens to look more like the host application. A new template has been added to support this function
See also : Technical Reference | ds_SetDefaultFont
Template Guide | Extension : Set Default Font

Version 1.6

1) Minor cosmetic bugs on the Operator Form have been fixed.

2) You can now change the **Login code** for a user on the Operator Form.

3) Support for WorkGroups has been added. These are provided for browse filtering purposes. A new function, ds_CurrentWorkGroup has been added. A new Extension template, Workgroup Filtering, has been added to implement this new feature on screens.
See also : Template Guide | Extension : User Workgroup Filtering
Technical Reference | ds_CurrentWorkGroup
User Guide | Workgroups

4) Support for Multiple DLL projects has been added. This includes a new Utility template, and new options on the 'Activate SecWin Features' template.
See also : Installation Guide | Using SecWin in a multi DLL project.
Template Guide | Utility : Import Queue Handling
Template Guide | Extension : Activate SecWin features

5) More support for network users has been added. You can now specify the location of the DSSW.TPS file from within the Application directory, using SecWin.Ini.
See also : Installation Guide | The CRESEC.EXE Utility

6) A new function ds_LoginOpt has been added. This allows you to override the behaviour of the Login screen. This allows password case insensitivity and optional logins. New options have been added to the User Login template to support this.
See also : Template Guide | Extension : User Login Here
Technical Reference | ds_LoginOpt

Version 1.5

1) Support for Clarion for Windows 1.5 has been added in the form of additional files, namely SecWin16.Lib and SecWin16.Dll. This Dll uses the Clarion Dll's that came with Clarion Version 1.5 for Windows. Thus if you are distributing a CW 1.5 application this cuts down considerably the number of Dll's required for distribution. There are also a number of support files, such as the SecWin15.Tpl which have been modified. There is also a new example application, EmpRpt15.App which demonstates using SecWin features in an Application.
See also : Installation Guide | Installing SecWin, Distribution, License

2) Support for networks has been improved by adding the [SecWin] section to the Win.Ini file. If the section exists in the Win.Ini file then SecWin will automatically check for the existance of a path setting to locate the DsSW.TPS file. If this setting is not present then the Dll defaults to the Windows directory as before.
See also : Installation Guide | The CRESEC utility.

3) Three new functions have been added to the DLL to provide information to the calling application. These are ds_CurrentName, ds_CurrentLogin and ds_CurrentLevel. A new code template, Call Current, has been added to support these functions.
See also : Technical Reference | ds_CurrentName , ds_CurrentLogin, ds_CurrentLevel.
Template Reference | Code : Call Current
4) A new function ds_ChangeLogin has been added to the DLL. This allows one user to logout and another login without closing the application. A new code template has been added to support calling this function.
See also : Technical Reference | ds_ChangeLogin
Template Reference | Code : Call Change Login

5) A new function ds_LockScreen has been added to the DLL. This allows the user to blank the screen at any time, and requires their password to be filled in before the application can procede. A new code template has been added to support calling this function.
See also : Technical Reference | ds_LockScreen
Template Reference | Code : Call Lock Screen

Important information for CW 2.0 beta users

The success of the CW 2.0 beta program has lead to requests for a version of SecWin supporting the new version. The additional files you will require are available, free, on request by contacting us using one of the methods mentioned in the Installation Guide above. A general release of SecWin for CW 2.0 will be made as soon as CW 2.0 is out of the beta stage.

At this time the above document is written specifically for CW 1.5, however the extreme compatibility between CW 1.5 and CW 2.0 has meant that very few changes have occured. The following points however should be made if you are using CW 2.0.

1) At this moment all Internationlisation features have been removed from SecWin for CW 2.0. This will hopefully be returning in the near future. The Internationalisation features remain obviously in the CW 1.5 version.

2) The various versions of CW and Secwin have brought about a small change in the names used. The SecWin16.DLL is now called the SecWn216.Dll and the SecWin32.Dll is renamed to SecWn232.Dll. The equivalent LIB file names have also changed.

3) The template file is now called SecWin20.Tpl, not SecWin15.Tpl. Other that the naming change, and the changed Dll name, the templates are identical.

4) Cresec15.Exe has been recompiled to use the CW2.0 Dll's, and is called Cresec20.Exe.

End of Document. This document last updated on 11 June 1996
