

**BoxSoft**  
Corporation

## **Super Passcode - Documentation**

Copyright © 1989-2014, BoxSoft Corporation, All Rights Reserved.

# Table of Contents

Foreword	0
<b>Part I Getting Started</b>	<b>3</b>
1 Introduction.....	3
2 RTFM Warning!!!.....	7
3 Installation.....	8
<b>Part II Template Usage</b>	<b>12</b>
1 Adding Passcode to your Applications.....	12
2 Add a Configuration File to your Dictionary.....	13
3 Add a backdoor password variable to your Global Data.....	15
4 Global Extension Template.....	16
5 DemoProtect Extension Template.....	29
6 NewPasscode Code Template.....	30
7 MachineCode Code Template.....	31
8 Trial versus Demo.....	32
9 Performing Tasks with 15-digit Passcodes.....	33
10 Client/Server Passcodes.....	35
11 Assigning Passcodes.....	36
12 NEW: Automating ShowPass.....	38
<b>Part III Appendices</b>	<b>40</b>
1 Example Programs.....	40
2 Reference Section.....	42
3 Project Defines.....	44
4 Troubleshooting.....	46
5 Contacting Technical Support.....	47
6 License Agreement.....	48
<b>Index</b>	<b>49</b>

# 1 Getting Started

## 1.1 Introduction

The BoxSoft Super Passcode Templates security system for Clarion (ABC) is designed to help you to prevent software piracy of your developed applications. It does this by using unique passcodes that are dependent upon the system date. You can optionally use the user's company name and/or the machine's BIOS (you must use at least one of these methods).

The codes can expire at the end of the current period, or at one full period from the entry date, or they can be permanent. The system also optionally allows a "trial period" before requiring a Passcode to be entered, and you can use a "demo company name" to embarrass them into upgrading.

The user will need to enter a new Passcode in the following situations:

1. They are running the package for the first time.
2. The old passcode has expired.
3. They have changed their business name in the configuration file.
4. They have significantly changed their system (e.g.: a new motherboard), so that the machine's BIOS changes.
5. You are using the "trial before passcode" period feature, and the trial period has expired.

The passcode is a 6-digit number that is calculated based upon a complex checksum and mathematical algorithm. Don't ask to have it explained, as it's been tweaked almost beyond understanding (and beyond hacking). What is important is that it generates unique codes for each situation. If you are interested in deciphering the logic, all of the source code is open for your perusal.

Because you are the developer, you will be allowed to access these codes on-site with a 15-digit backdoor password. When performing this task, the various passcodes are not explicitly labeled. This makes it difficult for the user looking over your shoulder to understand or to remember the codes.

If you are not on-site, there is a special application included with this package that allows you to determine the new passcode over the phone, quickly and easily. This utility is called "ShowPass".

As a new feature for version 5.0, you can optionally use a 15-digit passcode instead of the 6 digit one. (The fifteen digits are grouped in threes, separated by dashes, which makes the number easier to read and enter.) Even though the code is 15-digits, the "unlock" portion of the code is still six digits. The other nine are comprised of a check digit, a task digit, and seven data digits. It's up to you to specify the check task and data digits in your ShowPass program, and to respond to them when a valid passcode is entered. More on this later in the documentation.

Another new feature in 5.0 is support for Servers and Clients. In this case, the "Server" is a program that is run on one machine (or more, if you desire). It verifies that the passcode is valid, and updates the "PasscodeType" and "PasscodeClientDate" fields in the configuration file. The "Clients" merely check these two fields to verify that the passcode has been validated recently.

There are six different types of passcodes:

1. Permanent
2. Yearly
3. Monthly
4. Daily
5. Date Reset
6. Upgrade Demo

The first is used to give the user permanent access to the system. Providing they don't change their name in the configuration file or significantly reconfigure their machine, then they will never need to enter another passcode.

The next three are expiring codes. They will give the user access as long as the specific period doesn't change. The yearly and monthly codes might be used for systems that are licensed or rented to the user on a yearly or monthly basis. The monthly code might also be used to provide temporary demo access. Finally, the monthly and daily codes might be used to allow short-term access while awaiting payment from the user.

These expiring passcodes can expire either one full period from today (e.g.: one month from today), or at the end of the current period (e.g.: at this end of this year). For example, it is August 1, 1995 and you enter the yearly passcode. If the code is to expire one full period from today, then the passcode will expire on August 1, 1996; otherwise, it will expire on January 1, 1996.

The "Date Reset" code takes a little explanation. With many protection systems that are date dependant, the user may change the system date back in time to allow access to a system that is expired. With the BoxSoft Passcode facility, this date change trick will not allow the user to access the system.

The user, however, may have accidentally changed the system date to some ridiculous future date. If this is the case, you will want to reset the passcode date check back to today's date, and not the accidental future date. Otherwise, any code that you assign will last until the end of that future period. That is the purpose of the Date Reset code.

The final passcode, "Upgrade Demo", is for those of you distributing demo copies of your software. You can give out copies with a funny name (e.g.: "\*\*\* Your Business Name \*\*\*") embedded in the company name field of the configuration file. Until the user pays you, they are not allowed to change the name. Entering the "Upgrade Demo" code causes Passcode to return a different value. The templates allow you to call an update form when the upgrade code is entered. After entering the new company name, they will have to enter the proper passcode for that name.

### **Extension Templates**

The "Passcode" extension template contains the majority of the passcode system. It is populated into your global area, and it adds code to your MAP, source to your Setup Program embed, an entry to your project, and it generates a file called xxxx\$PCD.CLW, where "xxxx" are the first four characters of your APP name.

The "DemoProtect" extension template is used to protect your configuration update form. If you are using the "Demo Company" support, you will want to prevent the user from changing various fields in the form. This template can protect one or more ranges of controls, by disabling, hiding and/or setting read-only.

### **Code Templates**

The "NewPasscode" code template lets your user force the entry of a new passcode. There are a couple of times when you would want to do this. If your user was running on a "monthly" code, and you received their payment, then you could call them and have them force passcode entry so that they could enter the permanent passcode. You could also force passcode entry if they had an invalid date and they needed to enter the date reset code.

The "MachineCode" code template calls the PasscodeMachine() function to display and/or return the user's unique code. This code is used as the seed for all of the other passcodes. You may want to print this on a registration/order form after the user enters their company name. They could mail or fax this to you, then you can send them their proper passcode after the trial period.

### **ABC and Legacy Template Chains**

This documentation pertains to both the ABC and Legacy (a.k.a. "Clarion") Super Template sets. In some situations we've implemented features in ABC that are not in Legacy, primarily because the old template chain was to be phased out. Due to customer pressures, however, Soft Velocity decided to reinstate support for the Legacy/Clarion chain.

Some of the Super Template features that are only in the ABC chain would be very difficult to implement in the legacy chain. However, we'll attempt to do this wherever it seems feasible to us. We apologize if this causes you any inconvenience. Please feel free to contact us if there's a particular feature in ABC that you would like to see in the Legacy chain, and we'll see if your needs can be accommodated.

For more information, see the following topics:

- Add a Configuration File to your Dictionary
- Add a backdoor password variable to your Global Data
- Passcode Global Extension
- DemoProtect Extension Template
- NewPasscode Code Template
- MachineCode Code Template
- Trial versus Demo
- Performing Tasks with 15-digit Passcodes
- Client/Server Passcodes
- Assigning Passcodes
- Automating ShowPass
- Example Programs
- Reference Section
- Contacting Technical Support

License Agreement

## 1.2 RTFM Warning!!!

It is very important that you read this documentation. If you follow the instructions step-by-step, then the usage is very simple. It is almost *impossible* if you try to do it on your own!

## 1.3 Installation

### Installation Directory Structure

NOTE: As of version 6.6, we've changed our installation to the defacto "3rdParty" directory structure. (In Clarion 7 this is actually the "Accessory" directory.) Your old CLARIONx\SUPER directory has been renamed to CLARIONx\SUPER-OLD.

Once you've finished running the installation program, you should see the following structure under your C55 or Clarion6 directory:

```
C:\CLARION6, C:\C55, etc.
+-LibSrc      STA*.INC      (ABC headers)
+-3rdParty
| +-Bin ST_*.HLP, ST_*.CNT, STAB_CNV.DLL
| +-Template  STA?*.TPL, STA*.TPW  (ABC chain)
| |          STC?*.TPL, STC*.TPW  (Clarion chain)
| +-LibSrc    STA*.INC, STA*.CLW, STA*.TRN  (ABC chain)
| |          STC*.INC, STC*.CLW, STC*.TRN  (Clarion chain)
+-Images
| `--Super   *.ICO, *.CUR, *.WMF, *.GIF
+-Docs
| `--Super   *.PDF  (Documentation)
`--Vendor
   `--Super
      +-QBE
      | +-Examples
      | | +-ABC *.DCT, *.APP, *.TPS (Examples) (ABC chain)
      | | `--Clarion *.DCT, *.APP, *.TPS (Examples) (Clarion chain)
      | `--Source
      | | +-ABC *.TXD, *.TXA, *.DCT (Source) (ABC chain)
      | | `--Clarion *.TXD, *.TXA, *.DCT (Source) (Clarion chain)
      +-Tagging
      | +-Examples
      | | +-ABC *.DCT, *.APP, *.TPS (Examples) (ABC chain)
      | | `--Clarion *.DCT, *.APP, *.TPS (Examples) (Clarion chain)
      | `--Source
      | | +-ABC *.TXD, *.TXA, *.DCT (Source) (ABC chain)
      | | `--Clarion *.TXD, *.TXA, *.DCT (Source) (Clarion chain)
      `--Etc.
      +- . . .
`--SUPER-OLD      (ABC headers)
   `-- . . .
```

For Clarion 7 and later versions it should look like this (note the two trees):

```
C:\Program Files\SoftVelocity\Clarion 7
`--Accessory
   +-Bin          ST_*.HLP, ST_*.CNT, STAB_CNV.DLL
   +-Template
   | `--Win       STA?*.TPL, STA*.TPW          (ABC chain)
   |              STC?*.TPL, STC*.TPW          (Clarion chain)
   |              STMH*.TPW                    (Shared)
   +-LibSrc
   | `--Win       STA*.INC, STA*.CLW, STA*.TRN  (ABC chain)
   |              STC*.INC, STC*.CLW, STC*.TRN  (Clarion chain)
   +-Images
   | `--Super    *.ICO, *.CUR, *.WMF, *.GIF
   `--Docs
      `--Super   *.PDF  (Documentation)
```

```

"My Documents" or "Shared Data" (depending on OS)
^-Clarion 7\Accessory
  ^-Super
    +-QBE
      | +-Examples
      | | +-ABC          *.DCT, *.APP, *.TPS (Examples)      (ABC chain)
      | | ^-Clarion     *.DCT, *.APP, *.TPS (Examples)      (Clarion chain)
      | | ^-Source
      | | +-ABC          *.TXD, *.TXA, *.DCT (Source)         (ABC chain)
      | | ^-Clarion     *.TXD, *.TXA, *.DCT (Source)         (Clarion chain)
    +-Tagging
      | +-Examples
      | | +-ABC          *.DCT, *.APP, *.TPS (Examples)      (ABC chain)
      | | ^-Clarion     *.DCT, *.APP, *.TPS (Examples)      (Clarion chain)
      | | ^-Source
      | | +-ABC          *.TXD, *.TXA, *.DCT (Source)         (ABC chain)
      | | ^-Clarion     *.TXD, *.TXA, *.DCT (Source)         (Clarion chain)
    ^-Etc.
    +- . . .

```

To prevent conflicts between old Super Template files and same-named files in our new directory structure, the new installers attempt to delete the old files. If it encounters problems, then an error will be reported during the installation. Then you must delete any of the following files from the old directories, if they also exist in the new directory structure:

```

C:\CLARION6, C:\C55, etc.
+-LibSrc          STAB*.CLW, STCL*.CLW, STAM*.CLW, STCM*.CLW,
                  STAB*.TRN, STCL*.TRN, STAM*.TRN, STCM*.TRN,
                  STDEBUG.*
+-Template        STAB*.TP?, STCL*.TP?, STAM*.TPW, STCM*.TPW,
                  STGROUPS.TPW, STDEBUG.TPW
^-Bin             ST_*.HLP, ST_*.CNT, ST_*.GID

```

For example, you can use a tool like the indispensable Beyond Compare ([www.scootersoftware.com](http://www.scootersoftware.com)) to investigate the contents of C:\Clarion6\LibSrc and C:\Clarion6\3rdParty\LibSrc. View only files matching the mask ST\*.\* and hide all "orphans", which will show the files that exist in both directories. Delete the files from C:\Clarion6\LibSrc, and then do the same for the Template and Bin directories.

## **Filenames and Product Abbreviations**

- Super Template filenames generally start with the letters "ST". That's about all you can go on most of the time. (Our image files don't follow this convention, but they are sequestered in the Image\Super subdirectory.)
- The next two letters are usually AB (for the ABC chain) or CL (for the Clarion/legacy chain). One exception is Super Stuff (MH), which uses AM, CM and MH. Also, if both the ABC and Clarion chain share a TPW, then the AB/CL are skipped and it goes on to the product abbreviation. (Again, Super Stuff is an exception, as it uses MH for the shared files.)
- There are several TPWs that are shared by multiple Super Templates: STGROUPS.TPW, STABABC.TPW, STBLDEXP.TPW, STDEBUG.TPW
- The last four characters:
  - For TPL files, the last four letters are an underscore, followed by one of the following

suffices. The exceptions are STABAEQB.TPL and ST?M\_STF.TPL.

- For TPW files, the last four letters may match one of these in its entirety, or be followed by additional characters denoting the special purpose files.
- Super Stuff (MH) is an exception, in that it uses STcMxxxx, where "c" is the chain of A or C, and "xxxx" denotes the special purpose.

<b>AEQB</b>	Super QuickBooks-Export (i.e. Accounting-Export QuickBooks)
<b>BRW/BW</b>	Super Browse
<b>DIA</b>	Super Dialer
<b>FF</b>	Super Field-Filler
<b>IE</b>	Super Import-Export
<b>INV</b>	Super Invoice
<b>LIM</b>	Super Limiter
<b>PCD</b>	Super Passcode
<b>QBE</b>	Super QBE
<b>SEC</b>	Super Security
<b>TAG</b>	Super Tagging
<b>MH/STF</b>	Super Stuff (MH) (a.k.a. <i>The "MikeHanson" Templates</i> )

### Update the Redirection File

The installation program is able to update your redirection file automatically. If you decline the option during the installation, then you will have to edit the redirection file yourself. The three things that must be found are the Templates, LibSrc and Images. For example, you might make the following changes to the the \*.\* entry in Clarion 6:

```
*.* = .; %ROOT%\examples; %ROOT%\libsrc; %ROOT%\images; %ROOT%\template; %ROOT%
\3rdParty\template; %ROOT%\3rdParty\libsrc; %ROOT%\3rdParty\images\super
```

In Clarion 7 and above it will be more like this:

```
*.* = %ROOT%\Accessory\images; %ROOT%\Accessory\resources; %ROOT%\Accessory\libsrc\win; %
ROOT%\Accessory\template\win; %ROOT%\Accessory\images\Super
```

There are \*.RED examples in the SUPER\DOC directory.

### Register the Templates

Clarion allows you to have multiple template sets accessible in the same application. It does this with the Template Registry. To use a Super Template, you must register it first. The installation program attempts to do this for you, but in case it fails, or if your registry becomes corrupted, then you must register them manually.

1. Load Clarion, then select the "Setup / Template Registry" pulldown menu option.
2. Press the [Register] button.
3. Select C:\CLARION\3rdParty\Template\ST\_\*.TPL (ABC) or ST\_\*.TPL (Clarion). The directory name may not exactly match your system.

Assuming this all went without a hitch, you're ready to start using the templates.

## 2 Template Usage

### 2.1 Adding Passcode to your Applications

The SuperPasscode templates are actually a combination of templates and functions. The templates should handle all interfacing with the functions, so you should never have to call them yourself. If you are adventurous, though, you can certainly give it a try. (For more information on this, see Function Reference.)

To use the Passcode features, you must perform the following steps:

1. Add a Configuration File to your Dictionary
2. Add a backdoor password variable to your Global Data
3. Add the Passcode Global Extension template

Once you've done this, you can proceed to add one or more of the other templates:

DemoProtect Extension Template

NewPasscode Code Template

MachineCode Code Template

## 2.2 Add a Configuration File to your Dictionary

You must add a Configuration File to your dictionary. This will be a file containing a single record to store the company name, entered passcode values, and various other support fields. The names of the file and fields are not important, so you can use our suggested names or make your own. You can add these fields to an existing configuration file, or create a new file just for this purpose. This file will contain only one record, so it does not need any keys. At the very minimum it must include the following fields:

Passcode	LONG	@N06	Required
PassType	STRING(1)	@S1	Required for Server/Client
PassDate	LONG	@D1	Required
PassCliDate	LONG	@D1	Required for Server/Client
PassExpiry	LONG	@D1	Optional
PassTrial	LONG	@D1	Optional
Company	STRING(40)	@S40	Optional

If you wish, you can copy the definition from the example dictionary

```
SUPER\EXAMPLES\PASSCODE\TEST.DCT.
```

The file should use `!Glo:Lock` (note the leading exclamation point) as an "Owner Name" and it should be encrypted. (Without an Owner Name and Encryption, your system will not be very well protected.) If you don't wish to use `Glo:Lock`, then you may use another variable name or a constant value.

Some file drivers don't yet support this, so there is an additional encryption feature for the date fields. (The other fields do not really require protection.) This is not a foolproof method, though, so it is preferable to use a file driver that supports encryption.

If you already have a Config file, you may wish to add these fields along with this owner name and encryption. We suggest that you combine the two files, as it saves file handles.

The `Company` field contains the user's company name. If you are not planning on using the company name to make the passcodes unique, then this field is optional. If you are planning on using the "Demo Company" support, then you must specify this field. (See the section on *Trial versus Demo*.)

The `Passcode` field holds the user's Passcode itself. If this value is zero or invalid, the Passcode system will ask for a new code when the user starts their application.

The `PassType` field contains the most recent type of code entered and validated. This could be [P]ermanent, [Y]early, [M]onthly, [D]aily, [T]rial, [X]-ExpiredTrialContinuingButRestricted, [C]-DemoCompany, or [ ]-Blank (i.e. invalid passcode). There are equates for each of these generated in your global data section.

The `PassDate` field remembers the latest date that the Passcode was checked. This means that the user may not change their date back to trick the Passcode system. If the user accidentally changes the date back and you need to revert it to the proper date, then you can have them enter the "Date Reset" Passcode.

The `PassCliDate` field is used for communication between the Server and Client (if you're using that feature). The Server updates this field with the current date when it validates the passcode. The Client expects this field to contain a "recent" date for it to proceed.

The `PassExpiry` field is used if you wish to have passcodes grant access for full periods from the date of entry, rather than just to the end of the current period. If this setting is omitted, then the passcodes

will always expire at the end of the current period (e.g.: month, year).

The `PassTrial` field is used to provide a trial period before the system requires a Passcode from the user. (See the section on Trial versus Demo.)

## 2.3 Add a backdoor password variable to your Global Data

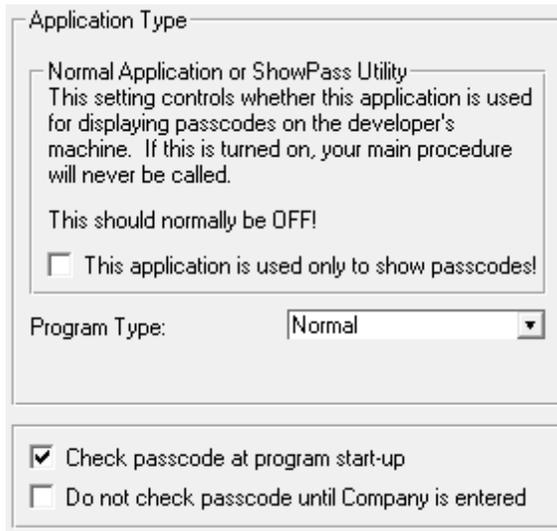
To protect your configuration file and provide a backdoor password, you need a new global variable. Add the following backdoor password and encryption variable to the Global Data section of your application (press [Global], [Data], [Insert]):

```
Glo:Lock          STRING(15)
```

## 2.4 Global Extension Template

The prompts for the Passcode global extension template are as follows:

### "General" Tab



Application Type

Normal Application or ShowPass Utility  
This setting controls whether this application is used for displaying passcodes on the developer's machine. If this is turned on, your main procedure will never be called.

This should normally be OFF!

This application is used only to show passcodes!

Program Type:

Check passcode at program start-up  
 Do not check passcode until Company is entered

**Normal Application or ShowPass Utility** - There are two ways to determine passcodes: on-site and off-site. When you do it on-site, pressing [Alt-F9] from the "Enter Passcode" window will take you to the "ShowPass" window, then entering the backdoor password will show you the appropriate codes.

When you are determining passcodes off-site (the customer calls you or sends their machine code by fax or mail), then you will need a "ShowPass" utility. In this case, you'll create an empty APP (the main procedure will never be called), and turn this option on. Depending on the setting here, there will be numerous Passcode prompts disabled or enabled. (Remember that there is a SHOWPASS.APP in the SUPER\EXAMPLES\PASSCODE directory.)

**Program Type** - If you are not creating a ShowPass utility, then you can specify the program type here.

**Normal** - This program will validate passcodes for itself, and prompt the user for passcode entry as required.

**Passcode Server** - This program will validate passcodes for a group of machines, and will prompt the user for passcode entry as required. It updates the PassType and PassCliDate fields in the Configuration file to communicate the current passcode status with the Client machines.

**Passcode Client** - This program will check the Configuration file to ensure that a Passcode Server has validated the code recently. If not, then it will simply not proceed (unless it's an extended but restricted trial). With the Client setting, you must also specify the number of "Grace Days". That is, the number of days that the Client will wait for the Server to update the passcode's authenticity.

**Check passcode at program start-up** - Turn this OFF if you want to call the Passcode check at some time other than program startup. If this is off, the Passcode will never be validated unless you

call it yourself.

**Do not check passcode until Company is entered** - If you are using the "Demo Company" feature, then you may want to prevent prompting for a Passcode until the user manually changes the company name from the "Demo Company".

### **"Config File" Tab**

**Passcode Server/Config File** - This is the file in which the calculated passcode will be stored. If you are creating a client/server system, then the server will store the passcode in this location, and various other bits of information in the client file. This allows you to have multiple servers, with each storing it's own passcode in a separate file.

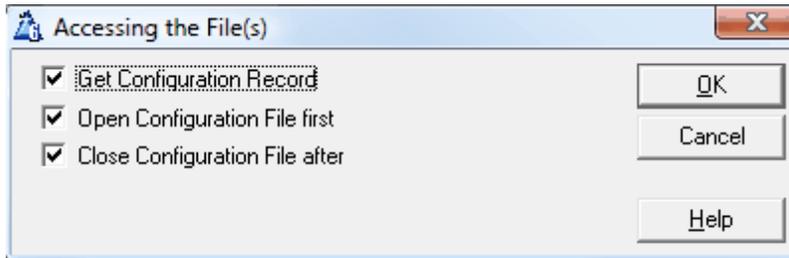
By default, the Configuration record is fetched with SET(File)/Access:File.TryNext(). If this is inappropriate, then you can write your own procedure or function to perform this task. Enter the name of that procedure here, along with the method that it uses to communicate an error condition. If you are using multiple passcode servers, you could have a multi-record config file, with each server storing it's passcode in a different record. This may be preferable to having a separate file on each server's local drive.

**Client Config File (may be same as above)** - In most cases this will be the same file as the one above. If, however, you are using multiple servers, then this file will be the one that is shared by all machines, while the file above is specific to the server.

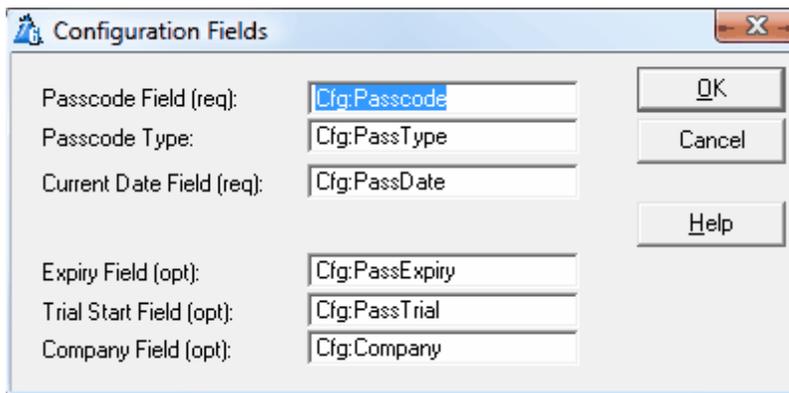
This is where you specify the file and field names that you setup in the Configuration File in your dictionary. Depending on your client/server specifications from the first tab, some of these settings may not appear.

**NOTE:** If you are using a Server/Client system, and you intend to have only one Server machine, then you can use one file for all the fields. If, however, you intend to have two or more Passcode Servers, then you must specify a separate Client Config file, which must be shared by all Servers and Clients. All the fields *except* the "Passcode" field must be in the shared file. Since the Passcode itself is machine specific, each Server machine must have its own version of the

file. It's suggested that you store this file on the local hard drive of each Passcode Server.



**Accessing the File(s)** - The Passcode template can optionally read your configuration records for you. If you wish for it to do this, then check the "Get Configuration Record" option. If this is checked, then you can optionally have it "Open the Configuration File first". If you are already doing this at an earlier point in your program, then you may want to turn this option off. Press [OK] when you are done with the Config File settings.



**Configuration Fields** - Some of these fields are required (like the Passcode field), while others are optional. Depending on whether you specify certain fields, the Passcode system may omit certain features.

**Passcode Field (required)** - This is where the passcode is stored. If you are running a multi-passcode-server system, then this field must be in the "Server" configuration file.

**Passcode Type (required for client/server, otherwise optional)** - This field records the type of passcode that was last entered (e.g. Permanent, Yearly, etc.) If you don't specify the field, it then the type will not be recorded.

**Current Date Field (required)** - This is the last date that the passcode was authenticated. In a client/server situation, it must be in the server's config file.

**Client Date Field (required for client/server)** - This field is only applicable in client/server mode. The server updates this field when it authenticates its passcode. The clients read this date when they start, to ensure that the server has checked-in "recently".

**Expiry Field (optional)** - if you specify the "Expiry Field", you are implicitly indicating that you wish passcodes to expire one full period from the date of entry, instead of at the end of the current period. When using client/server, this field must be in the server configuration file.

**Trial Start Field (optional)** - If you specify this field, then you are enabling the "Trial Before

Start" feature. The other settings for this are in a later tab.

**Company Field (optional)** - If you specify this field, then it will be included in the Passcode seed value. If you don't, then you will not be able to use the "Demo Version Support" (but you will still be able to use the "Trial before Passcode" support). There is a section later in the manual describing the benefits of "Trial" versus "Demo".

**"Backdoor" Tab**

Variable or String Constant  
 The backdoor can be a variable or a string constant. We suggest that you use a variable, as it makes it more difficult to hack the program.

If you are specifying a variable name, make sure that you prefix it with an exclamation point (ie: !Glo:Lock).

You must create this 15 character string yourself!!!

Backdoor:

---

Run-time Variable Initialization  
 Using a variable and initializing it at run-time is the best way to prevent hacking!

Set Backdoor Variable

First 8 characters:

Last 7 characters:

PassDate Encryption

ExpiryDate Encryption      TrialDate Encryption

These settings control the variable used for assigning passcodes at the customer site, as well as for encryption the Passcode date fields (Date, Trial and Expiry) in the configuration file. You can specify a numerical string constant for the backdoor (e.g.: '123456781234567'), but we strongly recommend that you use a variable (e.g.: !Glo:Lock). Remember to prefix the variable name with an exclamation point.

If you decide to use a variable, you can have the Passcode system initialize it at start time. This is more secure than assigning an "initial value" in the variable definition window. This variable will be assigned as two segments of eight and seven characters.

## "Trial Before" Tab

This feature enables you to allow the users a trial period before requiring that a passcode be entered.

You can only access these options if you have specified a "Trial Start Field" in the "Config File" options.

Trial Period Before Passcode

Number of days:

Continue trial period if invalid passcode attempted?  
 Continue with restricted trial after expired?

Initialize Trial Date Expression  
 You should call a function to check for existing "transactions", so that you can use that date rather than zero

Mention trial before expiration (each time program is

These settings control how long the user may run the program before needing to enter a Passcode. This is easier to administer, because the user can try the software for a month before bugging you for a Passcode. You can use this separately from the "Demo" feature, or in conjunction with it.

The first option is the number of days of trial. Each time the user starts the program, they will see a "trial mode" window appear, and they will have the option of entering a Passcode. If they try the Passcode before they are ready, then the next setting, "Continue trial period if invalid Passcode attempted?", controls whether they are locked out or whether they can continue the trial.

If you're concerned that the user may try to delete the configuration file to extend their trial, then you can specify an "Initialize Trial Date Expression". Write a function to access the first record (chronologically) from a file that contains dates (like a Invoice file). Then return this date from the function. That way, the trial always begins on the date of first use.

The "Trial Messages" button leads to the messages that appear when the Passcode is okay or expired. For each you can specify the title and message. In the case of the "Trial NOT Expired" message, you can also include a tilde (~) to indicate the number of trial days left. (This works only in the message field, not in the title.) You should also look at the section discussing Trial versus Demo model

## "Demo" Tab

A demo version of your program does not allow the user to change the company information until they enter the upgrade passcode.

You can use this only if you specify a "Company" field in the "Config File" settings.

Provide Demo Version Support

Demo Company Name:

UpdateCompany Proc.:

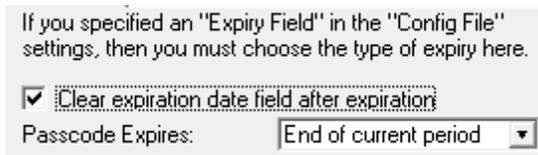
These settings are used if you want to distribute your application with a configuration file containing an

embarrassing company name like "\*\*\*\* Your Company Name \*\*\*\*" or "UNREGISTERED COMPANY". You can use the "DemoProtect" extension template on your configuration form to prevent the user from changing this demo name.

If you are using this feature, then there will be an additional "Upgrade" Passcode generated. If the user enters this code, then the "UpdateCompany Procedure" will automatically be called so that they can enter their actual company name. After that, the system will ask for another Passcode, using that new company name to determine the seed value.

**WARNING:** The Passcode seed value generated by "PasscodeMachine" is based upon the Configuration Company field. If you send them the "Upgrade" Passcode then the company name will change, along with the machine code seed value. Therefore, we suggest that you use the "Trial before Passcode" feature. This will enable the user to enter their company name before getting the seed value. This value will not change unless they change their company name. If you intend to be talking to the customer when they enter the Passcode, then this is not an issue.

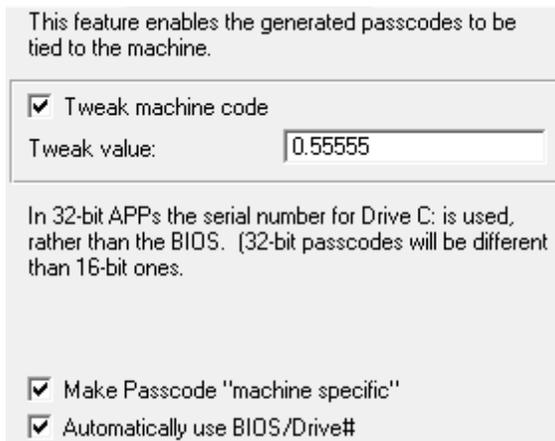
**"Expiry" Tab**



This setting controls when the temporary Passcode expires. If you didn't specify an "Expiry Field" in the "Config File" settings, then this setting will not be available.

The two options are "End of current period" and "One full period from now". Let's pretend that it's August 1st, 1995. If this setting is "End of current period", then the yearly Passcode would last until the end of 1995. If this setting is "One full period from now", then the yearly Passcode would last until July 31, 1996.

**"Machine Specific" Tab**



This setting controls whether the Passcode system includes the machine's BIOS information (for 16-bit programs) or hard drive serial number (for 32-bit programs) in the seed value. There are a number of thoughts to consider here.

1. Only the BIOS is used, not the CMOS. This means that a 16-bit user can change hard drives without needing a new Passcode, but a motherboard change would cause the Passcode to expire. In contrast, a 32-bit user needs a new passcode if the hard drive changes.
2. Memory managers and operating system sometimes change the BIOS at boot time. On a given machine, this should be consistent. However, if they are always changing operating systems and/or memory managers, there is a chance that you would have to provide new passcodes when they did this. If they are usually alternating between a couple of setups, then you could give them a Passcode for each setup, and reduce your hassle factor.

It's best not to mention that changing their system setup could cause a need for a new Passcode. If they know this, then they might try to trick you into giving them a Passcode for another machine, under the ruse of needing two for a single machine.

3. If your user purchased a number of identical machines at the same time, they will all have the same BIOS information. Even hard drives can sometimes come with the same serial number. It is hoped that a large customer with multiple machines would be honest enough to purchase a separate copy of your software for each machine, but it may not be the case. Unfortunately, no copy protection scheme is foolproof.

Along our upgrade path, we decided to make the passcode calculation more complex. However, to support existing users, we made this tweak optional. In most cases, you should leave the "Tweak" settings as is. Go ahead and change the tweak value if you want, though. It will randomize your passcodes even further.

If you decide to determine the "machine code" yourself, then turn off the "Automatic" mode, and specify a "GetMachineCode" function. It must take no parameters, and return a LONG.

### **"Hooks/Tasks" Tab**

"FetchTask" Procedure:

15-digit passcodes with embedded task digits

This optional function is only used when using the built-in ShowPass (i.e. Alt+F9). It will be called repeatedly for a single bank of codes.

Procedure Name:

Prototype:  
(STRING Period, \*BYTE Task, \*LONG Value)

---

"PerformTask" Procedure

Procedure Name:

Prototype:  
(STRING Period, BYTE Task, LONG Value), BYTE

It must return "True" if all is OK.

---

Prevent normal passcode messages

Prevent normal passcode updates for Prm/Yr/Mth/Day

If you want to perform various operations when a passcode is entered, then you can use a 15-digit passcode instead of a 6 digit one. The prompts here will depend on whether you're create a ShowPass utility, or a Normal/Server application. In the case of the FetchTask procedure, it is needed for any

application that can request a passcode (e.g. ShowPass, Normal or Server).

**NEW in 6.02:** You can specify the **InitShowPass** and **UpdateCodes** settings when creating a ShowPass utility for 6-digit passcodes too. These can be useful for Automating ShowPass.

### ***\*\*\* Prompts for ShowPass Utility \*\*\****

**"FetchTask" Procedure** - When the passcode is calculated, you must provide ShowPass with the necessary task information for it to insert in the 15-digit passcode. This function takes three parameters:

**STRING Period** - This is [P]ermanent, [Y]early, [M]onthly or [D]aily. The function will be called once for each of these periods. You may want to vary the task type and data depending on the period, or you may want to have all of them be the same. This really depends on your needs, and you'll often have all codes do the same thing.

**\*BYTE Task** - This is a pointer to a BYTE, and you'll assign the Task type (0 to 9) to this parameter.

**\*LONG Value** - This is a pointer to a LONG, and you'll assign any desired data (0000000 to 9999999) to this parameter.

**"InitShowPass" Procedure** - This optional procedure is called immediately before the ACCEPT loop on the ShowPass window. Usually you'll be running the ShowPass utility manually. However, you can automate it using this function along with the UpdateCodes procedure.

**SHORT MachineCtl** - This is the "Machine Code" control. You will enter the requesting user's machine code here.

**SHORT DateCtl** - This is the "Machine Date" control. You will enter the system date of the requesting user's machine here.

**SHORT OkButton** - This is the OK button on the window. It's called "OK" in your procedure.

**E.g.:**

```
0{PROP:Hide} = True
CHANGE(MachineCtl, |
    GETINI('Request', 'Machine',, '.\LICENSE.INI'))
CHANGE(DateCtl, |
    GETINI('Request', 'Date',, '.\LICENSE.INI'))
POST(EVENT:Accepted, OKButton)
```

**"UpdateCodes" Procedure** - This optional procedure is called immediately before the calculated passcodes are displayed. There are four parameters, for the Permanent, Yearly, Monthly and Daily codes (P, Y, M and D).

**E.g.:**

```
PUTINI('Response', 'P', P, '.\LICENSE.INI')
PUTINI('Response', 'Y', Y, '.\LICENSE.INI')
PUTINI('Response', 'M', M, '.\LICENSE.INI')
PUTINI('Response', 'D', D, '.\LICENSE.INI')
HALT
```

### \*\*\* Prompts for Normal/Server Application \*\*\*

**"PerformTask" Procedure** - After the user enters the 15-digit passcode, the passcode template will automatically call this function to perform the specified task. You must return True if the everything's OK, or False if it isn't. This function takes three parameters, as specified by your ShowPass utility. You may respond to these parameters in any way that suits your application.

**STRING Period** - This is [P]ermanent, [Y]early, [M]onthly or [D]aily, depending on which passcode was entered by the user.

**BYTE Task** - This is the task type (0 to 9).

**LONG Value** - This is the data value (0000000 to 9999999).

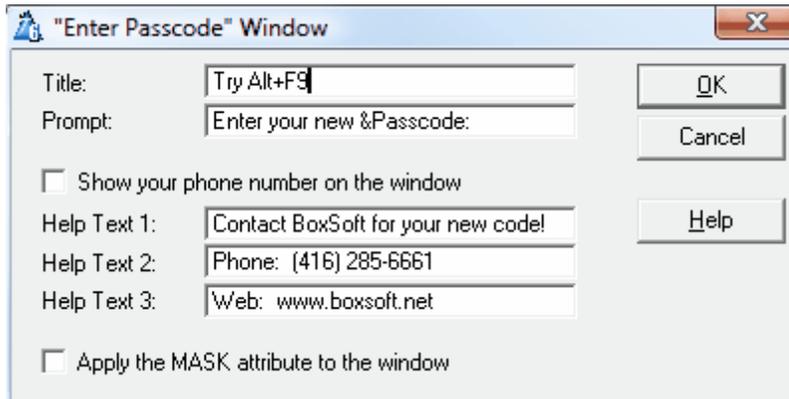
### "Splash Window" Tab

Splash Procedure:	About
When to Call:	Before Passcode

These settings let you call a splash procedure before or after the Passcode function is called. You enter the name of the procedure to be run, as well as when to run it.

### "Window Text" Tab

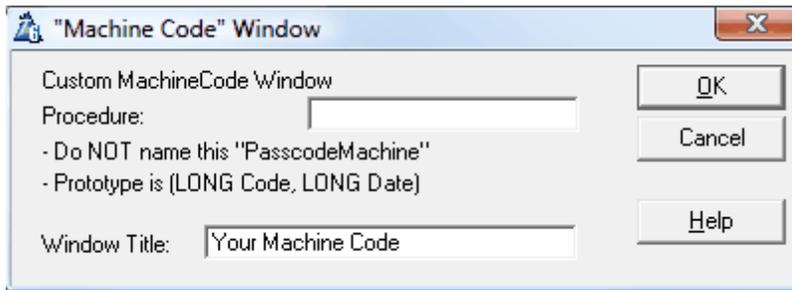
All of these except the bottom button refer to individual windows or message screens in the Passcode system. The last button relates to common buttons that are shared by more than one window.



Title:	Try Alt+F9	OK
Prompt:	Enter your new &Passcode:	Cancel
<input type="checkbox"/> Show your phone number on the window		Help
Help Text 1:	Contact BoxSoft for your new code!	
Help Text 2:	Phone: (416) 285-6661	
Help Text 3:	Web: www.boxsoft.net	
<input type="checkbox"/> Apply the MASK attribute to the window		

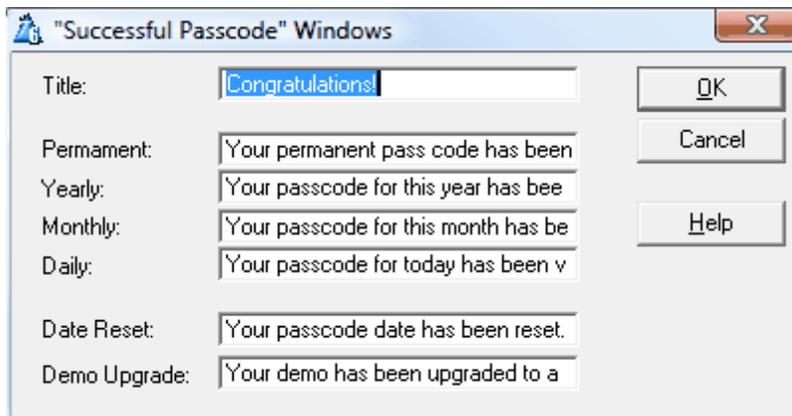
The **"Enter Passcode" Window** is where the user enters their Passcode. They would see this when the application first starts (optionally after the splash window described earlier). You may want to include your phone number, because your user may need to call you when they encounter this screen.

The MASK attribute is useful for a 15-digit Passcodes, as it automatically shows the dashes during entry (e.g. 123-456-789-012-345).



The **"Machine Code" Window** displays the user's unique machine code. You need this number (normally four or five digits) to assign passcodes over the phone. The user's Passcode date is also displayed on this screen. The date should be today; if it isn't, they will have already received a "Need Date Reset" warning.

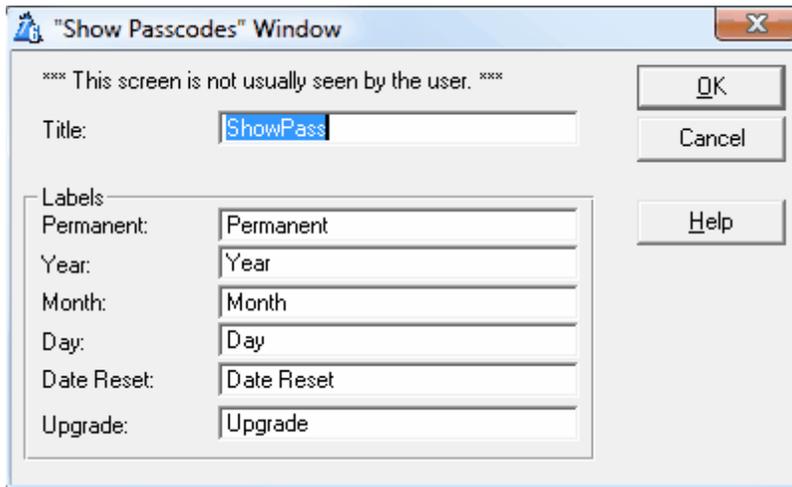
**New in Version 6.03** - We've added the option of calling an alternate procedure to display the machine code and date. This gives you the opportunity not only to display this basic information, but also to show contact details, or anything else that suits your fancy. Eventually we will be enhancing this to support improved registration automation.



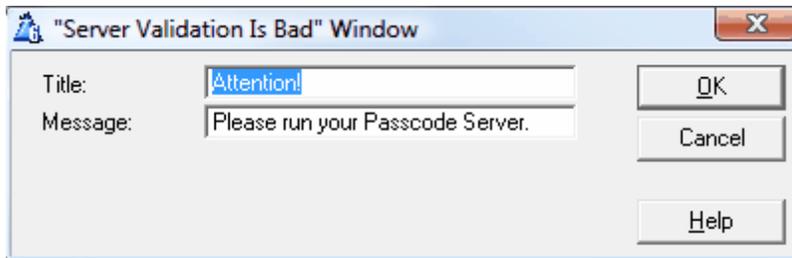
The **"Successful Passcode"** message windows are displayed for successfully entering the various Passcodes. For all of these the title is the same, and the message changes.

The **"Invalid Code" Window** is displayed if your user doesn't enter an acceptable Passcode. It contains your business name, address, phone numbers, etc., so that your user can contact you. There is also an optional ICON to be displayed with your company logo or program icon.

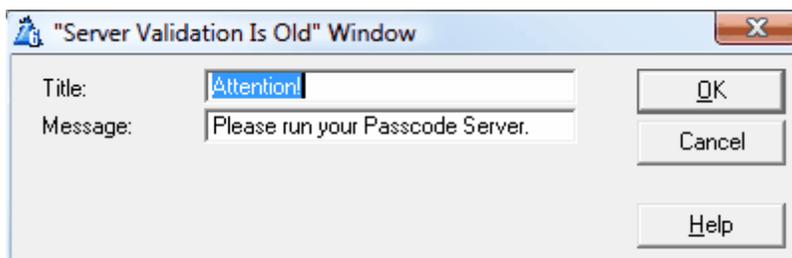
The **"Need Date Reset" Window** is displayed when a new Passcode is about to be entered, but the Passcode date field doesn't match the current machine date. In this situation, you should give the user the "date reset" code first, then their regular Passcode date.



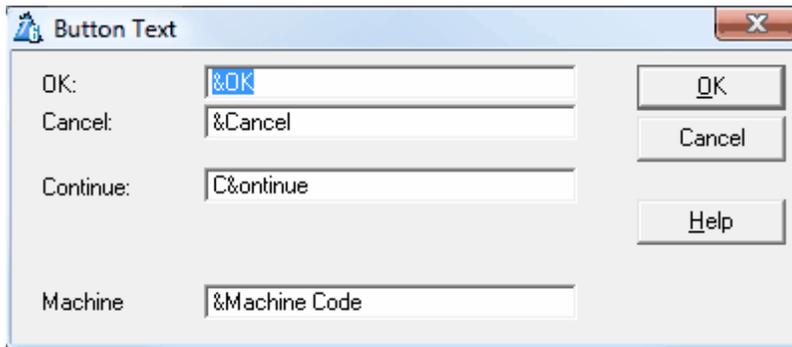
The **"Show Passcodes" Window** is used to determine the various passcodes. If the "Normal Application or ShowPass Utility" switch is OFF, then you will be allowed to specify only the window title. This is so that the user cannot easily spot and memorize the permanent Passcode when you intend to assign something less. You can still see the labels for your off-site ShowPass utility, though.



The settings for **"Server Validation Is Bad"** are available only when the Program Type is set to "Client".

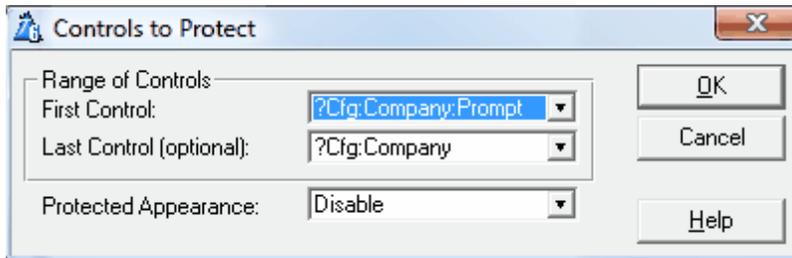


The settings for **"Server Validation Is Old"** are available only when the Program Type is set to "Client".



The "**Button Text**" is used on the various buttons found throughout the various Passcode windows.

## 2.5 DemoProtect Extension Template



This template is normally used in a Form procedure. It protects your company field (and optionally any other fields) when the system is in demo mode. The program is in demo mode when the company field in the configuration file contains the demo company name. Don't confuse this with the Trial mode.

To implement this, go to your configuration update screen. Press the [Extensions] button, then the [Add] button. Find "Class SuperPasscode", then "DemoProtect" below that. Highlight this entry and press [Select].

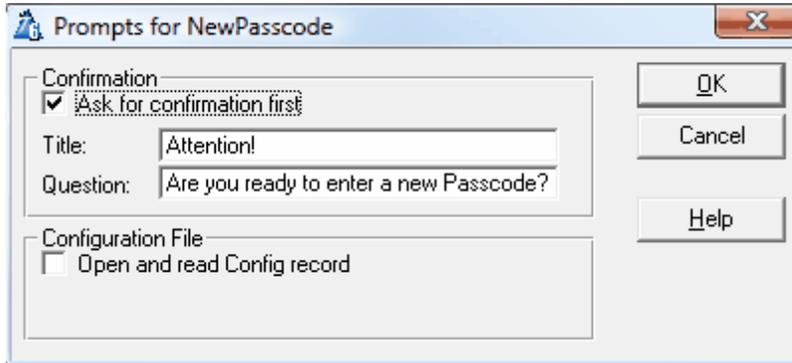
Now press the [Controls to Protect] button, then [Insert] to add a range of controls. You must specify the first control, and can optionally specify the last control to protect a range. You should protect the prompts along with the entry controls.

Once you've specified the control range, then you must declare how the protected controls should appear. You can choose "Disable", "Hide" or "Read-Only".

Here is a sample of the code generated:

```
IF IsDemoCompany( )
  DISABLE(?Cfg:Company:Prompt , ?Cfg:Company)
END! IF
IF IsDemoCompany( )
  HIDE(?Cfg:Company)
END! IF
IF IsDemoCompany( )
  LOOP X# = ?Cfg:Company:Prompt TO ?Cfg:Company
    X#{PROP:ReadOnly} = True
  END! LOOP
END! IF
```

## 2.6 NewPasscode Code Template



This code template is used to force entry of a new Passcode. This is handy if you have given them a temporary Passcode while awaiting for payment, then you've received the payment. Rather than waiting for the Passcode to expire and have your user panic not knowing what to do, you can have them force re-entry at a convenient, controlled time.

This code template can be called as part of any embed. You may want to place it on your pulldown menu, or as a button in an update form, or as part of a registration process. You may also want to call it automatically if they change their company name, rather than waiting for them to restart their application. (Of course, if they change their company name and you are including it in the Passcode calculation, then they will need to give you their machine code for you to calculate the proper Passcode.)

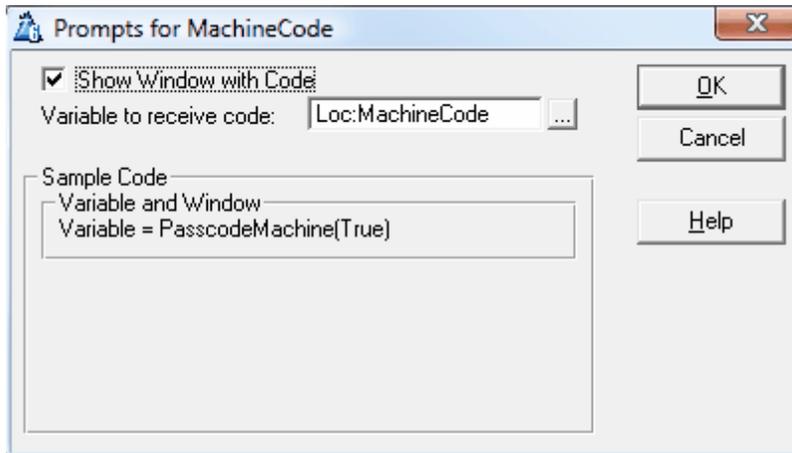
To add the code template, get to any embed list, and press [Add], find "Class SuperPasscode" and highlight "NewPasscode" below that, then press [Select].

Your first option is to "Ask for confirmation first". It is strongly suggested that you use this, so that the person doesn't accidentally cancel their old Passcode in the process. If this is turned on, then you will be allowed to specify the title and message for the confirmation screen.

Here is a sample of the generated code:

```
IF MESSAGE('Question', 'Title', ICON:Question, |
    BUTTON:Yes+BUTTON:No, BUTTON:No) = BUTTON:Yes
    NewPasscode
END!IF
```

## 2.7 MachineCode Code Template



The MachineCode code template optionally displays the user's unique machine code and Passcode date. It also returns this number from the function so that you can print it on a report to be used for registration.

Remember, the machine code is used to determine the user's passcodes off-site. This means you could call this procedure after the user enters their personal information. This code would be printed on a registration form and mailed or faxed to you. While you are responding with the Passcode, the user can be running the program in trial mode. Let's pretend you want to use this technique:

1. Set the Passcode system to allow a trial period. It is strongly suggested that you don't use the demo mode at the same time (especially in this situation).
2. Create an update form called `UpdateConfig` to enter the configuration information.
3. In the "Initialize the Procedure" embed of `UpdateConfig`, place the following code:
 

```
GlobalRequest = ChangeRecord
```
4. In the "End of Procedure, Before Closing Files" embed of `UpdateConfig`, place the following code:
 

```
IF LocalResponse = RequestCompleted
  RegistrationReport
END!IF
```
5. Create a new "Report" procedure called `RegistrationReport`. Specify your configuration file as the primary file.
6. Press the `[Data]` button and add a variable called `Loc:Machine`.
7. In the "Procedure Setup" embed, add the "MachineCode" code template from the "SuperPasscode" class. Specify `Loc:Machine` as the "Variable to receive code", and turn the "Show window with code" option OFF.
8. Place `Loc:Machine` somewhere on your report.

## 2.8 Trial versus Demo

In earlier DOS versions of the Passcode system, there was no trial mode. Only the demo mode support was available. Here's how the demo mode works:

1. When the user runs the APP for the first time (without a configuration file), it adds a record to the file and places the demo company name into the company field.
2. The user must contact the software developer to get a temporary Passcode to try the software.
3. This company field is placed on numerous screens and reports to embarrass the user into purchasing the software.
4. The user is prevented from changing the company name until the upgrade Passcode is entered.
5. Once they have entered the upgrade Passcode, then must enter their proper company name.
6. Then they have to enter a new Passcode based upon their new company name.

As you can see, this is a major hassle. To make this procedure simpler, an optional trial mode was added to the system.

1. When the user runs the APP for the first time (without a configuration file), it adds a record to the file and starts the trial mode. If you're concerned about the user deleting the configuration file to start a new trial, then you can specify a function that will be called by the Passcode template to determine the starting date from one of your date-oriented files.
2. At any time the user can change their company information.
3. The user can use the software until it expires, or contact the developer for a Passcode to get it out of trial mode.
4. At that time, the developer gives them the proper Passcode and they are finished.
5. If the trial period expires, you have the option to allow the user to continue with a restricted version of the program. You control what these restrictions will be.
6. If you have specified the company field in the Passcode settings, and the user changes his company name (maybe giving a copy to his friend), then they will need a new Passcode. You can also use the machine specific BIOS support to protect it from copying.

## 2.9 Performing Tasks with 15-digit Passcodes

Usually passcodes are 6-digits, and they are responsible only for the protection of the system. However, assigning passcodes involves communicating with your customer and interfacing with their machines with a "magic" number. This scenario gives us the opportunity to do more than prevent copy protection. You might want to change the number of concurrent users controlled by our companion product, SuperLimiter. Or you could run a special procedure, like a data archiver that requires personal programmer involvement.

By associating these tasks with the passcode, you have the opportunity to pass data through as part of the passcode, and the user is none the wiser.

To accomplish this, you must do a number of things:

1. Enable the "Support Tasks with 15-digit passcode" setting on the Tasks tab of the global Passcode extension template. This must be done both in your program and in your ShowPass APP.
2. In ShowPass, you must create a procedure (usually called FetchTask) with the prototype (`STRING Period, *BYTE Task, *LONG Value`). When you are calculating passcodes with ShowPass, this procedure will be called for each passcode type. Your procedure must set the value of the Task and Value external parameters. In many cases this will involve calling a window that will ask you (the developer) what values you wish to send. You can also choose whether to respond to the Period parameter (Permanent, Yearly, Monthly, etc.) If your task is to occur regardless of the period, then just ignore it.
3. Your end-user application also needs a FetchTask procedure. If you plan to assign passcodes while personally visiting the client's site, and you want the passcode system to perform tasks just as it would for remotely assigned codes, then your program also needs to call the FetchTask procedure specified in step 2 above. If, however, you don't plan to assign passcodes on-site, or you don't want it to perform tasks when you are on-site, then you can just create a dummy procedure that does nothing.
4. Your end-user application also needs a PerformTask procedure. The prototype is (`STRING Period, BYTE Task, LONG Value`), `BYTE`. This procedure will receive the values that you provided with your FetchTask procedure. It will be called only once (for the appropriate passcode that was entered). If your procedure executed the task properly and you wish for the program to continue, then return True. Otherwise, return False and the program will abort.

The `Period` parameter contains the passcode type. You should use the equates specified in the reference section of the documentation.

The `Task` parameter is a 1-digit number (0-9). You supply this value from your FetchTask procedure, and respond to it in PerformTask.

The `Value` parameter is a 7-digit number (0-9999999). You supply this value from your FetchTask procedure, and respond to it in PerformTask.

**TIP:** If you happen to need eight data digits instead of seven, and you don't need a task digit, then you can use the task digit for an extra data digit. You'll have to split it apart in FetchTask, and reassemble it in PerformTask.

For more information on Tasks and 15-digit passcodes, take a look at the global template settings, and the example in `SUPER\EXAMPLES\PASSCODE2`.



## 2.10 Client/Server Passcodes

The primary purpose of SuperPasscode is to generate a machine-specific, so that your program cannot be copied to unauthorized computers. However, this machine-specific aspect can be a problem in network settings. If each machine is passcode protected, it means you must assign individual passcodes to each and every machine. This is obviously unacceptable.

In the past we suggested that you turn off the machine-specific setting, and have the passcode determined using only company name. Of course, this doesn't offer nearly as much protection against piracy as you would like. Our solution was to implement a Client/Server paradigm.

Basically, one machine (the "server") is responsible for entering and authenticating the passcode. It must be run occasionally (e.g. once per day), to verify that the passcode is still acceptable.

Meanwhile, the client doesn't care what the passcode is. All he wants to know is that it's been verified "recently". (The default is sometime in the last 3 days, although you can change this in the global settings.)

The server and client communicate via a common configuration file that you define in your dictionary. If there will be only one server, then you can also store the server's passcode settings in the same file.

If, however, you wish to have multiple servers (for a large network where one server might not be able to run), then each server must have it's own personal passcode configuration file (or record). All the servers and clients share the "recent verification" information. The only field that **MUST** be in the Server's personal configuration file is the Passcode field itself. The rest can be in the shared file.

For more information on implementing a client/server passcode system, take a look at the global template settings, as well as the examples in SUPER\EXAMPLES\PASSCODE3 and 4.

## 2.11 Assigning Passcodes

### On-Site: [Alt-F9]

To assign a Passcode on-site, perform the following steps:

1. Get to the "Enter New Passcode" window. It may be coming up automatically, or you may have to get to the place where the "NewPasscode" template is called.
2. Press [Alt-F9]. You will be presented with the ShowPass window.
3. Type the 15-digit backdoor password into the field then press [Enter]. If you are using 15-digit passcodes, then your FetchTask procedure will be called for each passcode type before displaying the codes.
4. You will now see the six valid codes. Because the user may be looking over your shoulder, the labels for the Passcode types are not displayed. You must remember that they are in the following order: Permanent, Yearly, Monthly, Daily, Date Reset, and Upgrade Demo. (For those of you who used earlier versions of Passcode for DOS, take note that these last two codes have switched.)
5. Remember the desired Passcode, then press [Continue].
6. Type the Passcode into the field and press [OK].
7. Read the message to verify that the user has been granted the proper access.

### Off-Site: The ShowPass Utility

The ShowPass utility is used when the user calls you requesting a new Passcode. Before you get started, you should compile the SHOWPASS.APP in the SUPER\EXAMPLES\PASSCODE directory. Although both the CPD 2.1, CDD 3.0, and CW 1.x versions should produce the same codes, we suggest that you use the same version of Clarion with ShowPass that you used for the user's application (just in case of possible differences in rounding).

**IMPORTANT:** You need a separate ShowPass utility for each backdoor value that you use.

#### ***By Phone:***

1. The user calls.
2. You run the ShowPass utility.
3. You verify that the user has the Passcode window up on the screen.
4. You ask the user to press the "Machine Code" button. This will display another window with a seed value (usually four or five digits) and the Passcode date (the latest date on which the Passcode was checked, regardless of the actual system date).
5. The user recites the machine code.
6. You type the machine code into the ShowPass window and press [OK].

7. You verify that the date displayed on the user's system is today's date. If not, you may want to give them the "Reset Date" code before giving them the normal code. You can also change the date in the ShowPass utility to match the user's displayed date. Press [OK] to see the various passcodes will appear on the ShowPass screen. If you are using 15-digit passcodes, then your FetchTask procedure will be called for each passcode type before displaying the codes.
8. You ask the user to press [OK] to return to the Passcode screen.
9. You give the new Passcode to the user to type into the system, followed by [OK].
10. Verify that the user sees the message "Your Passcode for this <period> has been verified". (The message for the permanent Passcode is slightly different.)
11. You instruct them to press any key, and their system is back to normal. If you gave them the "Date Reset" Passcode, then you will need to return to step #4 (or #6, if you remember their machine code) and continue from there.

***By Mail:***

1. You receive the user's registration with their machine code in the mail.
2. You run the ShowPass utility.
3. You type the machine code into the ShowPass window and press [OK].
4. The various passcodes will appear on the ShowPass screen.
5. You record the desired Passcode and give it to the user.
6. Instruct the user on how to get to the Passcode (automatically, from the trial message, or through the NewPasscode template).

## 2.12 NEW: Automating ShowPass

You may have a custom program for tracking your users. You might even be using a special `GetMachineCode` procedure to assign your own machine codes to your customers. In this situation, being required to re-enter the information back-and-forth from a separate ShowPass utility can be a bit of a bother.

In the global extension template in your ShowPass APP., you'll find a tab entitled "Hooks/Tasks". Within it, you can specify two procedures: **InitShowPass** and **UpdateCodes**. `InitShowPass` is called when the screen is first opened, before the ACCEPT loop. `UpdateCodes` is called after new passcodes have been calculated. These give you the opportunity to tweak ShowPass so that you never need to see it. Here's what you do:

1. From your customer administration program, you need to store the machine code and entry date in an INI file (or the system registry, or a data file, or any other method that permits you to share information between two programs). This file is only on your local machine, so don't be concerned with security. To execute ShowPass from your program, do something like this:

```
SP" = '.\SHOWPASS.INI'
PUTINI('Req', 'M', Cus:MachineCode , SP")
PUTINI('Req', 'D', Glo:PasscodeDate, SP")
!---
SETCURSOR(CURSOR:Wait)
RUN('SHOWPASS',1)
SETCURSOR
!---
Cus:PasscodeDate = Glo:PasscodeDate
Cus:PasscodePerm = GETINI('Rsp', 'P', , SP")
Cus:PasscodeYear = GETINI('Rsp', 'Y', , SP")
Cus:PasscodeMth = GETINI('Rsp', 'M', , SP")
Cus:PasscodeDay = GETINI('Rsp', 'D', , SP")
RetVal = Access:AllMachinesChild.Update()
```

In this case, it's assumed that you already have the customer's machine code recorded in `Cus:MachineCode`. This may be your own custom code, as assigned by `GetMachineCode` at the customer site. Or it may be the machine code that your customer has given to you over the phone or via e-mail.

`Glo:PasscodeDate` contains the "applicable" date when you expect the customer to enter the passcode. If you're providing a daily Passcode, then this date is very important! However, if you're providing a permanent Passcode, then this date is ignored. You might just want to use `TODAY()`.

2. In ShowPass itself, create a Source procedure called "InitWindow". It's responsible for pre-filling the fields in the ShowPass window, and then pressing the OK button, like this:

```
InitWindow PROCEDURE(SHORT M,SHORT D,SHORT OK)
CODE
SP" = '.\SHOWPASS.INI'
0{PROP:Hide} = True
CHANGE(M, GETINI('Req', 'M', , SP"))
CHANGE(D, GETINI('Req', 'D', , SP"))
POST(EVENT:Accepted, OK)
```

3. Create another Source procedure in ShowPass called "UpdateCodes". It's responsible for putting the new codes back into the INI file, and then exiting ShowPass. It should look something like this:

```
UpdateCodes PROCEDURE(STRING P,STRING Y,STRING M,STRING D)
CODE
SP" = '.\SHOWPASS.INI'
PUTINI('Rsp', 'P', P, SP")
PUTINI('Rsp', 'Y', Y, SP")
PUTINI('Rsp', 'M', M, SP")
PUTINI('Rsp', 'D', D, SP")
HALT
```

4. If you are actually using 15-digit passcodes, then you would also need to create a FetchTask procedure. However, this is no different than any other 15-digit scenario.

## 3 Appendices

### 3.1 Example Programs

#### Example Directories

There are four example directories included with SuperPasscode, which can be found below the SUPER\EXAMPLES directory. By examining the dictionary files, extension settings, custom procedures, etc., you should get a good idea how to suite the templates to suit your needs. They demonstrate the following scenarios:

<b>PASSCODE</b>	This is the simplest implementation, and it also happens to be the original style of Passcode usage. There is a test program that requires a passcode, and a ShowPasscode utility for assigning passcode off-site. It uses the 6-digit passcodes.
<b>PASSCODE2</b>	This is quite similar to the previous example, except that 15-digit passcodes are used. Sample procedures are shown for FetchTask, PerformTask, etc.
<b>PASSCODE3</b>	This is the standard Client/Server implementation, when only one machine is designated to be a server. Therefore, all Passcode fields are contained in the one configuration file. The smaller 6-digit passcodes are used.
<b>PASSCODE4</b>	This is another example of Client/Server implementation. In this case, there will be multiple servers, which each server storing its passcode information in local files. In this case, the filename, C:\SCONFIG.DAT, is hard-coded in the dictionary; you can use any location, as long as the record loaded is specific to the current server. All servers and clients share "check-in" information in a common file called CONFIG.DAT. The smaller 6-digit passcodes are used.

#### Example Files

Throughout the example directories, you'll find the following files:

<b>TEST.DCT</b>	This file contains a sample CONFIG.DAT containing all of the necessary passcode fields. You may import this file into your data dictionary. It uses the CLARION file driver, and is set to be encrypted with !Glo:Lock as the owner name. This dictionary is used by TEST.APP.
<b>TEST.APP</b>	This sample application uses all of the passcode templates, and some sample source code as well. Take note of the following sections: <ol style="list-style-type: none"> <li>1. The Passcode global extension template is populated in the global area with all features enabled.</li> </ol>

	<p>2. The Main procedure calls the NewPasscode and PasscodeMachine code templates from buttons on its button bar.</p> <p>3. The UpdCfg procedure uses the DemoProtect extension template to prevent users from changing the demo company name. Also, the passcode fields are displayed on the form. Because Cfg:PassDate (last check date), Cfg:PassTrial (trial start date), and Cfg:PassExpiry (expiry date) are encrypted, the Passcode_Encrypt() function is used to decrypt these. You could use GetPasscode_PassDate(), GetPasscode_TrialDate() and GetPasscode_ExpiryDate(); however, some of those functions are not present (depending on the Passcode Global Extension settings) so we used the generic method.</p>
<b>SHOWPASS.APP</b>	This is a ShowPass utility. It is used to assign passcodes off-site (when your customer is talking to you on the phone, or has sent you their unique machine code seed value via mail or fax). The codes it generates are dependant upon your backdoor setting. Therefore, if you use different backdoors for different applications, then you must have a different ShowPass utility for each of those backdoors.
<b>SERVER.APP</b>	This is a sample server APP. It's responsible for the entry of passcodes, along with occsionally verifying that they are still valid.
<b>CLIENT.APP</b>	This client expects the server to athenticate the passcode regularly. (The default is at least one every three days.)

## 3.2 Reference Section

e_PasscodeType_None	EQUATE(' ')
e_PasscodeType_Permanent	EQUATE('P')
e_PasscodeType_Yearly	EQUATE('Y')
e_PasscodeType_Monthly	EQUATE('M')
e_PasscodeType_Daily	EQUATE('D')
e_PasscodeType_Trial	EQUATE('T')
e_PasscodeType_TrialExpired	EQUATE('X')
e_PasscodeType_DemoCompany	EQUATE('C')

These equates are used whenever the system passes a passcode type into a function.

### Passcode ( ), BYTE

This function returns True if the user has a valid passcode, or False if he does not. It performs any user input required to get the new passcode, displays "Invalid Passcode" errors messages, etc.

If you are creating a ShowPass program, this performs the operation. None of the other procedures and functions will be available.

### NewPasscode ( )

This procedure forces the user to enter a new passcode. It doesn't confirm first, so make sure that you verify that they want to do this before proceeding. You can call this using the NewPasscode code template.

### IsDemoCompany ( ), BYTE

This function returns True if the current value of the "Company Field" in the Configuration file is equal to the "Demo Company" value. This only applies if you are support the demo mode.

### PasscodeMachine ( <ShowCode> ), LONG, PROC

This function returns the unique machine code for the user's computer. If you include the optional <ShowCode> parameter, a window will appear with the machine code and passcode date.

Because this function has the PROC attribute, you can ignore the return value if you wish.

### SetPasscode\_PassDate ( Date )

This procedure encrypts the date parameter into the "Passcode Date" field in the Configuration file.

### GetPasscode\_PassDate ( ), LONG

This function returns the decrypted date value from the "Passcode Date" field in the Configuration file.

### SetPasscode\_ExpiryDate ( Date )

If you are supporting the variable expiry feature (the passcode can expire on a date other than a period boundary), then this this procedure encrypts the date parameter into the "Expiry Date" field in the Configuration file.

**GetPasscode\_ExpiryDate ( ), LONG**

If you are supporting the variable expiry feature (the passcode can expire on a date other than a period boundary), then this function returns the decrypted date value from the "Expiry Date" field in the Configuration file.

**SetPasscode\_TrialDate ( Date )**

If you are supporting the trial feature, then this this procedure encrypts the date parameter into the "Trial Date" field in the Configuration file.

**GetPasscode\_TrialDate ( ), LONG**

If you are supporting the trial feature, then this function returns the decrypted date value from the "Trial Date" field in the Configuration file.

**Passcode\_Encrypt ( Value, Start ), LONG**

You shouldn't need to call this function yourself. It's used for encryption and decryption by the other functions. The first parameter is the value to be en/decrypted. The second parameter tells the encryption routine where to start in the encryption string (total of 15 characters). Make sure that you don't start too near the end of the encryption string, as the number of bytes in your value may overrun the string. It returns the en/decrypted value.

**Passcode\_RestrictAfterTrial ( ), BYTE**

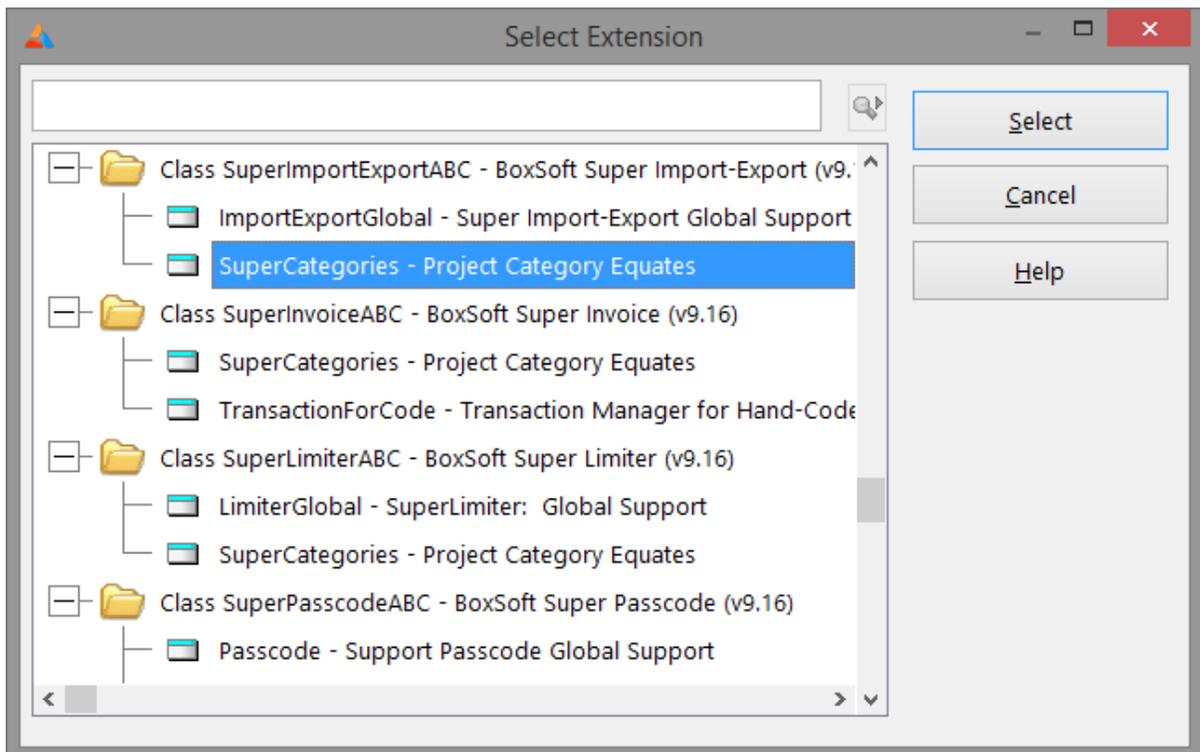
You are using the Trial feature, and you have chosen to allow your users to continue with a restricted version of the program after the trial has expired. When you wish to restrict a part of the program, call this function to determine if the system is in "restricted" mode. A return value of True means that it should be restricted. False indicates that the trial is still in effect, or that it's a full working program.

### 3.3 Project Defines

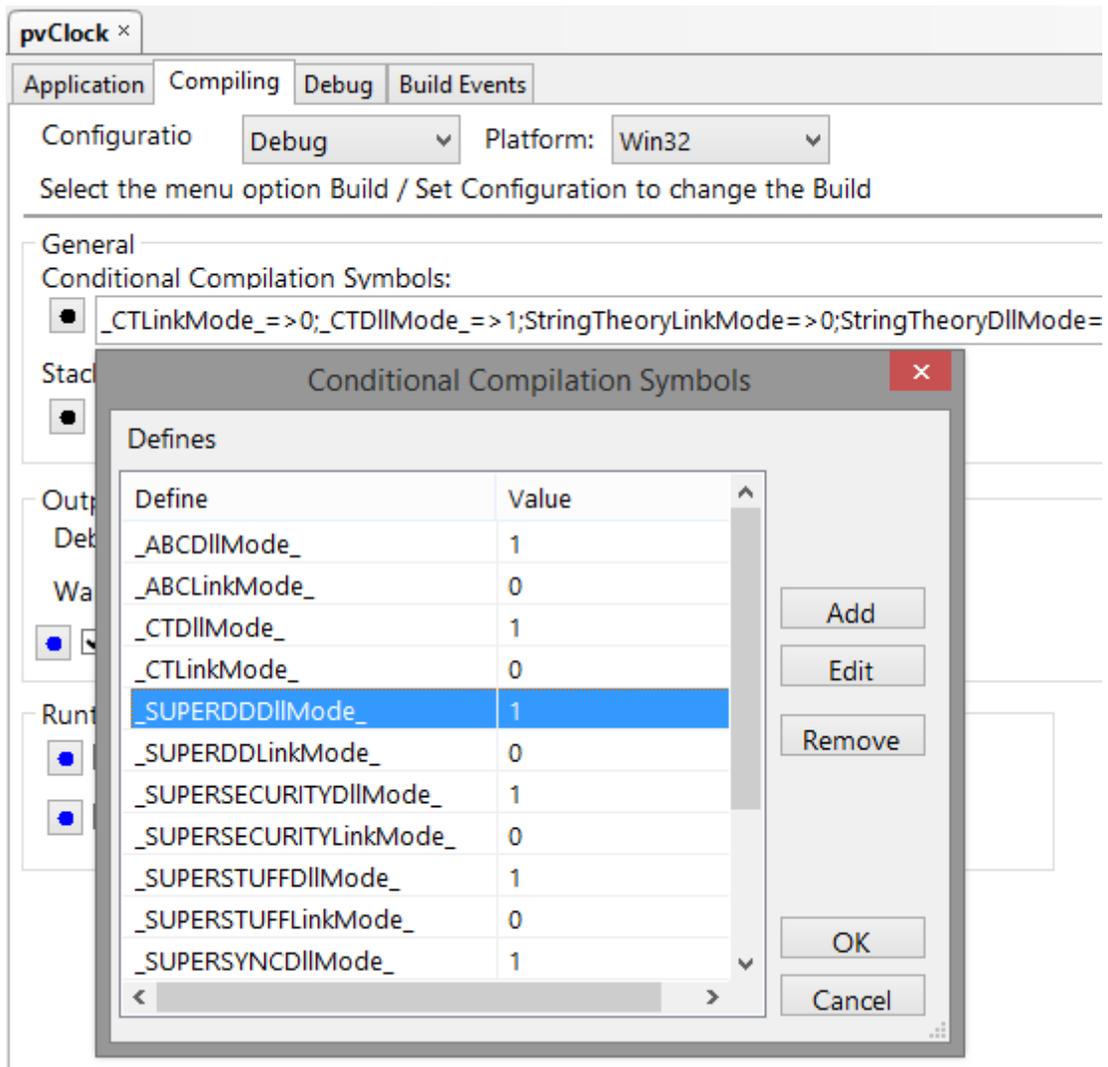
Prior to our 7.0 template versions, we were utilizing the same LINK and DLL Project Defines as the ABC libraries: `_ABCLinkMode_` and `_ABCDIIMode_`. This caused all of our libraries to be included in your base/dictionary DLL, even if you weren't using them in a particular development project.

To make this inclusion more selective, as of our 7.0 versions we changed to use various other switches. Some of these are product related (e.g. Super QBE uses `_SuperQBELinkMode_` and `_SuperQBEDIIMode_`), while others are associated with one of our shared base classes (e.g. Drag & Drop uses `_SuperDDLLinkMode_` and `_SuperDDDIIMode_`). Usually the templates (especially the global ones) automatically add the necessary entries to your Project Defines. If you happen to use the templates in your APPs in the wrong combination, these can be inadvertently omitted.

For APP-based systems, you can force the switches to be included by using the Super Categories global extension template. Every one of the Super Templates has this extension to apply its own switches, so if you're using multiple templates in a particular APP, you may have to add this extension for each of the products. (As was mentioned above, if there's already a global extension populated for a given Super Template, then you don't have to add this extension for that product.) Even if it's not needed, there's no problem with adding the SuperCategories global extension.



For hand-coded PRJ-based systems, you must add the switches manually. Take note of their names in the INC files, and then add them to the project settings like this:



## 3.4 Troubleshooting

There are no common troubleshooting problems to mention at this time.

---

## 3.5 Contacting Technical Support

If you have any troubles with this product, then please contact:

Mitten Software  
2354 West Wayzata Blvd  
Second Floor, Suite H  
Long Lake, MN 55356

Voice: (952) 745-4941  
Fax: (952) 745-4944

Internet: [www.mittensoftware.com](http://www.mittensoftware.com)  
[answers@mittensoftware.com](mailto:answers@mittensoftware.com)  
[www.boxsoft.net](http://www.boxsoft.net)  
[www.boxsoft.net/contact.htm](http://www.boxsoft.net/contact.htm)

## 3.6 License Agreement

### **One License per Developer**

This Super Template product is comprised of the templates, default applications, libraries, source code, documentation, and help files provided with the package. You must have a separate registered copy for each developer using it.

### **Redistribution**

You are allowed to use the product for any programs that you create, and you are permitted to distribute the generated source code. You may not, however, distribute any portion of the product in its original or modified form without the prior written consent from BoxSoft Development.

One exception to this is the example programs provided with this installation or separately from BoxSoft or its agents: these may be distributed without penalty, in either their original or a modified state.

### **Disclaimer**

BoxSoft Development does not warranty this software for any use. Any expenses or lost time due to errors in this product are not the responsibility of BoxSoft Development. We will attempt to fix any errors that are brought to our attention, but we are not legally liable for any lack of correctness of the product.

# Index

## - 1 -

15-digit CodeSplash15-digit Passcodes 16  
15-digit Passcodes 33

## - A -

API 42  
Assigning Passcodes 36, 38

## - B -

Backdoor 16

## - C -

Class Libraries 8, 44  
Client 3, 16, 40  
Company 13, 29  
Config File 16  
Configuration File 13, 15, 40  
Contacting Technical Support 47

## - D -

Daily 3  
Defines 44  
Demo 13, 16, 29, 32  
DemoProtect Extension Template 29  
Directories 8  
Disappear 44  
DLLMode 44

## - E -

Examples 40  
Expire 32  
Expiry 16

## - F -

FetchTask 33  
Freeze 44  
Function Reference 42

## - G -

Glo:Lock 15  
Global Data 15  
Global Extension 16  
GPF 44

## - I -

Include Files 8  
Installation 8  
Introduction 3

## - L -

License Agreement 48  
LinkMode 44

## - M -

Machine Code 3, 16, 31  
MachineCode Code Template 31  
Monthly 3

## - N -

NewPasscode Code Template 30

## - O -

Off-Site 36, 38  
On-Site 36, 38

## - P -

PassCliDate 13  
Passcode 13  
Passcode Fields 13

PassDate 13  
PassExpiry 13  
PassTrial 13  
PassType 13  
Performing Tasks with 15-digit Passcodes 33  
PerformTask 33  
Permanent 3  
Project Defines 44

## - R -

Redirection File 8  
Reference Section 42  
Registering the Template 8  
RTFM Warning 7

## - S -

Server 3, 16, 40  
ShowPass 3, 16, 36, 38, 40  
Support 47

## - T -

Task 33  
Tasks 16  
Tech Support 47  
Template Registry 8  
Trial 13, 16, 32  
Trial versus Demo 32  
Troubleshooting 46

## - U -

Upgrade 32

## - Y -

Yearly 3