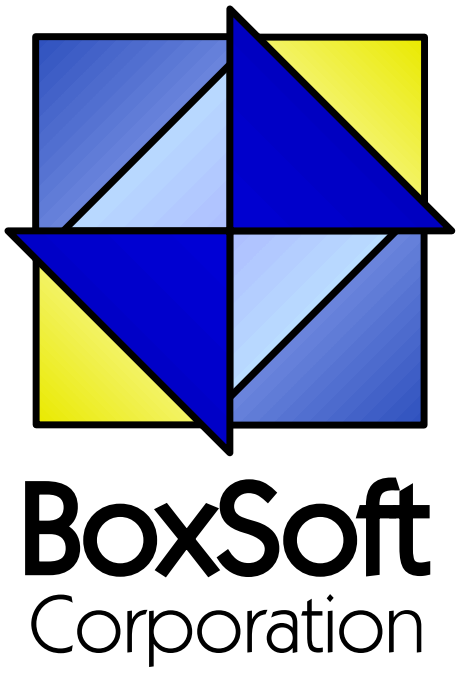


Super Limiter - Documentation

Table of Contents

1 Getting Started

1.1 Introduction

The BoxSoft Super Limiter templates allow you to limit the number of concurrent users of your package working on a network. This is very useful if you are selling "site licenses" of your software. You can specify the number of users, location of the check-in file, check-frequency, inactivity time-out for bumping, etc.

Version 4.0 incorporates an improved Inactivity monitoring method, and the API has been converted to an OOP class.

There are a number of templates included in this package to provide you with the following functionality:

LimiterGlobal - This global Extension template adds the global data and map segments to your application. It also generates the Limiter support module. This template also inserts the calls to Limiter.TakeEvent() and Limiter.UpdateActivity in all Windows and Reports throughout your APP. You control various general options which have an impact on the entire system.

LimiterUpdate - This Code template gives you a simple interface into the Limiter API to update the "active" status of the user.

LimiterLock and **LimiterUnlock** - If you anticipate that your user will not be able to update their active status for a long duration, then you can use these Code templates to "lock" their placeholder in the limiter file. This forces the Limiter system to wait for a longer time before bumping the user.

LimiterReset - This Code template gives you a simple interface into the Limiter.Reset() procedure to force a "restart" on the Super Limiter placeholder system. All users will automatically reestablish their position in the ST::Limiter file the next time they update their "active" status.

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:

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) 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.

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

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.

1.4 Upgrading From Earlier Versions

Upgrading from Super Limiter pre-1.50

If you are upgrading from a Super Limiter version prior to 1.50 (released March 1998), then there are a number of changes that you note:

- If you had not already imported the support files into your dictionary, you must do so now. The import file LIMITER.TXD. If you have already imported them, then the file labels and prefixes have been changed, and you must make a similar change in your existing dictionary. The physical file names on disk remain the same (LIMITER_). Look at LIMITER.DCT for an example.

  OLD: Limiter_,PRE(Lim_)
  NEW: ST::Limiter,PRE(SLim_)

- There are a number of changes in the Global Extension. Most notably, the Messages tab has disappeared. These settings have been moved to SUPER\LIBSRC\ST_LIM.TRN. The rest of the settings will only be available if the "Generate template globals and ABC's as EXTERNAL" is OFF.

- The Frame extension is now extinct. It should be removed from any windows where it exists. (If you forget to remove them all, the template will remind you when the code is generated.

- The API has been converted to OOP. Primarily, this means that instead of using Limiter_Method(), you use Limiter.Method(). We've also changed the CheckIn method to UpdateActivity. For more information on this, see API Reference.

2 Template Usage

2.1 Adding SuperLimiter to your Applications

The Super Limiter is a combination of templates and and OOP classes. The templates are designed to make the calling of the class methods very simple. If you wish, you can also call the methods directly. (For more information on this, see API Reference.)

To add the Super Limiter templates to your application, you must begin by adding the Support Files and the Global Support extension template. Once you've done this, you can proceed to add one or more of the other templates.

For more information, see:

- Support Files
- Global Extension Template
- UpdateActivity Code Template
- Lock and Unlock Code Templates
- Reset Code Template

2.2 Support Files

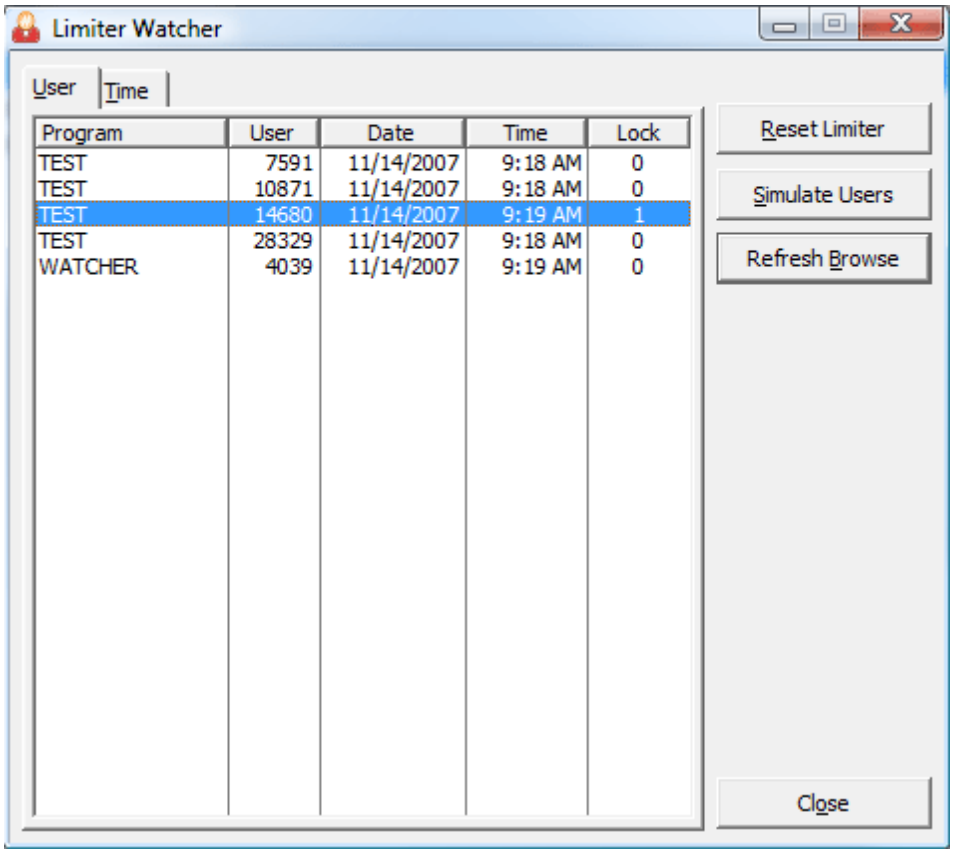
LIMITER.TXD

We've decided to always place the support files in the dictionary. It gives us many benefits, and clears up much confusion regarding the settings for the global extension template.

The first step is to load your dictionary, and import `LIMITER.TXD` from `SUPER\LIBSRC\LIMITER`. You'll end up with the following file in your dictionary:

ST::Limiter This file contains the "slots" for each active user. Each record contains the program name, the "user number" (normally a random number assigned during initialization), the last check-in date and time, and whether the slot is "locked".

WATCHER.TXA



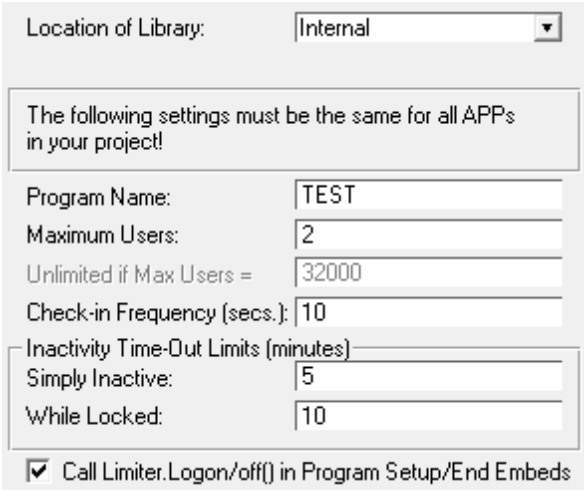
This is a utility application that can display the current users in ST::Limiter check-in file. It has the ability to clear all seats, after which all current users will automatically create a new seat when they next check-in. You can also simulate the presence of additional users, to test what happens when too many users are on the system.

2.3 Global Extension Template

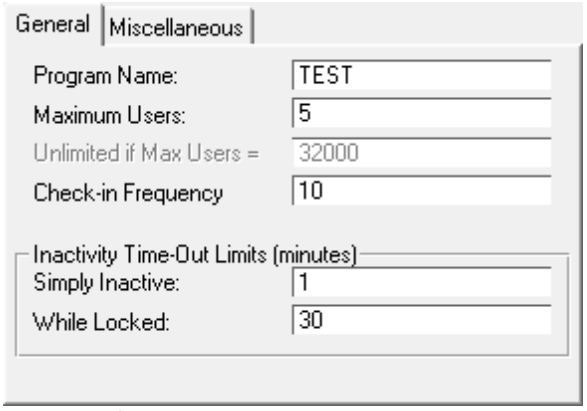
To use Super Limiter, you must populate this template into all APPs within your project. It automatically updates the user's "active" status from all Windows and Reports.

If *"Generate template globals and ABC's as EXTERNAL"* is OFF (i.e.: in a single-APP project, or in the base support APP of a multi-APP project), then it will also call `Limiter.Init(), Limiter.Logon()`, `Limiter.Logoff()`, and `Limiter.Kill()` in the "Program Setup" and "Program End" embeds. There are various settings that you can use to control the behavior of the system.

NOTE: The Global Extension should be populated into all the APPs of your project. However, the settings are only available only if *"Generate template globals and ABC's as EXTERNAL"* is OFF.



Settings for Clarion/Legacy template chain



General tab for ABC template chain

Location of Library - This setting is only in the Clarion/Legacy chain. It specifies whether this is the base (datafiles) APP, in which case you set it to *Internal*. Otherwise, this should be set to *External*.

Program Name - This is the common program name for all APPs that will be sharing the same "slots". It defaults to your APP's name, but should be changed if you have a multi-APP project. If you want a particular APP to be independant of the others, then use a different name here.

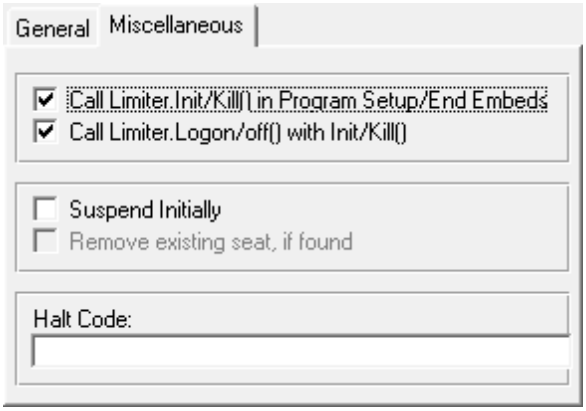
Maximum Users - This is the maximum number of users that can access the system concurrently. If you wish to control this from a configuration file, then you can place the name of a field here, preceded by an exclamation point. If you do this, you must use the "Program Setup" embed to read the configuration record before the `Limiter.LogOn()` procedure is called.

Using a configuration file can be very handy if you want to increase the size of site licenses without recompiling the program. You can simply send them a disk with a new configuration file on it.

Unlimited if Max Users = - If your "Maximum Users" is a variable (e.g. !Glo:MaxUsers), then you can specify which value will be interpreted as permitting unlimited users. In other words, if the MaxUsers variable equals this value, then the Limiter will be disabled.

Check-in Frequency (secs.) - This number controls the time between each file access of the `Limiter.UpdateActivity()` procedure. For example, if this is set to 10, then you can call it repeatedly, but it will access the file only after at least 10 seconds have elapsed.

Inactivity Time-Out Limits (minutes) - These two settings control how long a user is allowed to be inactive before they can be bumped by another user. The first is for normal usage. The second applies if the `Limiter.Lock()` procedure has been called. Both of these can be either constants (e.g.: 10) or variables preceded by an exclamation point (e.g. `!Cfg:TimeOut`).



Miscellaneous tab for ABC template chain

Call Limter.Init/Kill() in Program Setup/End Embeds - If this is checked, then `Limiter.Init` will be called in the "Program Setup" embed, and `Limiter.Kill` will be called in the "Program End" embed. If you have a complex start-up process, you may have to turn this off and call the methods manually.

Call Limiter.Logon/off() with Init/Kill() - Do you also want to Logon and Logoff when the Init and Kill are called? You may not want to do this, if you need to initialize your user variables further into the programs. If you turn this off, you must call the methods manually.

Suspend Initially - When the program first starts up, do you want the Limiter functionality to be turned off (i.e. Suspended)?

Remove existing seat, if found - This setting is available if "Suspend Initially" is turned *On*. It automatically clears any seat that was previously occupied by the user. (This should only happen if the program was exited without a proper Limiter.Logoff being processed.)

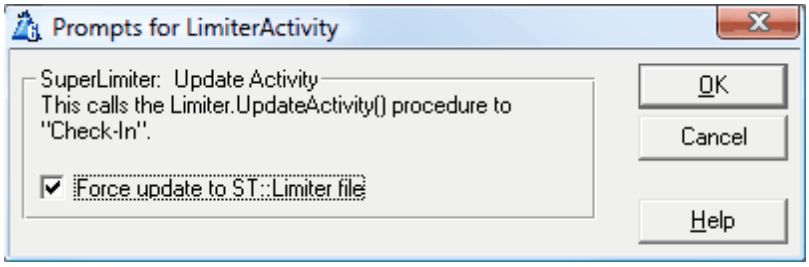
Halt Code - This code will be called whenever the Limiter forces the program to exit (whether

that's caused by an inactivity timeout, or some error condition). Use this to call any clean-up procedure that your program requires.

2.4 UpdateActivity Code Template

(Code Template)

This code template provides a simple interface into the `Limiter.UpdateActivity()` procedure.



Force Update to ST::Limiter File - Normally the ST::Limiter record will be updated only as often as is specified in the global extension template. If, for some reason, you wish to force it to update the disk file, then turn this ON.

2.5 Lock and Unlock Code Templates

(Code Templates)

These two code templates provide a simple interface into the `Limiter.Lock()` and `Limiter.Unlock()` procedures. There are no prompts to enter, as the procedures do not take any parameters.

In normal operation, the Super Limiter Frame Extension should automatically check-in for the user, unless some operation has taken control of the Windows event structure so that the `EVENT:Timer` is not being generated for the frame.

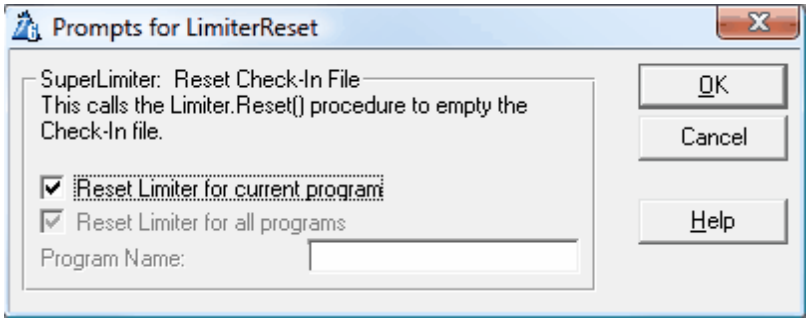
However, if you wished to disable the timer in the Frame you may want to Lock then Unlock the user's `ST::Limiter` record while this is being done.

There are no prompts for these templates.

2.6 Reset Code Template

(Code Template)

This code template provides a simple interface into the `Limiter.Reset()` procedure.



Reset Limiter for current program - If this is checked, then the `ST::Limiter` records for the current program are deleted. To access the other prompts, you must uncheck this option.

Reset Limiter for all programs - If this is checked, then all of the `ST::Limiter` records will be deleted. To access the "Program Name" prompt, you must uncheck this option.

Program Name - This is the name of the program for which the `ST::Limiter` records are to be deleted. It can be either a 1-8 character string (uppercase), or a variable name preceded by an exclamation point.

2.7 Suppress Check-In

(Extension Template)

Use this procedure extension template to prevent the automatic activity updater from checking in for the current procedure. There are not many situations where this is necessary, but it does come in handy occasionally. For example, if the file system is being disconnected and reconnected while waiting in a timer loop, you don't want the Limiter attempting to access its file.

To the rest of the users, this will look like an inactive seat that hasn't logged out. If it takes too long, the Limiter seat for this user may be taken by another. In that case, the next time the current user checks in, the Limiter will look for another free seat. If unavailable, the current user will be kicked out of the program.

There are no prompts for this template.

3 Appendices

3.1 Interface Modification and Translation

We've moved all displayable strings to STABLIM.TRN. (See the Installation section for location of the file.) This file contains equates for elements throughout the Super Limiter system.

Feel free to change any or all of these. In most cases, you should copy the file into your application directory. That way new versions of the templates don't overwrite your preferences. For each new APP you create, you can copy your version of the file into that directory. Or you can have your own version of the file somewhere in the RED path before %ROOT%\LIBSRC.

3.2 Multi-APP Development using LIBs/DLLs

If you are using the Super Limiter templates and you want to split your application into multiple LIBs/DLLs and a single EXE, here's how you do it.

LIB/DLL Including the Support Code

1. If you haven't already done so, import the Limiter support file definition from SUPER\LIBSRC\LIMITER\LIMITER.TXD.
2. Create or open the APP destined to be a DLL/LIB. This must be your base APP in which your data files are marked as "Internal" and "Exported"
3. Choose "Application / Properties ..." from the pulldown menu.
4. Blank the "First Procedure" field.
5. Ensure that your "Destination Type" is set to "Dynamic Link Library (.DLL)" or "Library (.LIB)".
6. Click [OK].
7. Press the [Global] button.
8. Press the [Extensions] button.
9. Press the [Insert] button.
10. Select "Super LimiterGlobal" under "Class Super Limiter".
11. Press [OK] to exit the "Extensions and Control Templates" window.
12. Turn OFF the "Generate template globals and ABC's as EXTERNAL" check box.
13. Press [OK] to exit the "Global Properties" window.
14. Make the DLL/LIB. (Press the "Lightning" on the button bar, select "Project / Make" from the pulldown, or press Ctrl-M.) You'll find a LIB file in your application directory.
15. If you specified your destination type to "DLL" back in step #5, then you will also have a DLL in your current directory. Remember to include this file when you distribute your APP.

EXE/LIB/DLL Excluding the Support Code

1. Create or open the APP that will call the procedures in the APP described above.
2. Press the [Global] button.
3. Press the [Extensions] button.
4. Press the [Insert] button.
5. Select "Super LimiterGlobal" under "Class Super Limiter".

6. Press [OK] to exit the "Extensions and Control Templates" window.

7. Turn ON the "Generate template globals and ABC's as EXTERNAL" check box.

8. Press [OK] to exit the "Global Properties" window.

9. Select "Application / Insert Module", then choose "ExternalDLL".

10. Specify your base application name with a .LIB extension as the Name (e.g.: BASE_APP.LIB), then press [OK] to exit the Module Properties window. (Even if your base APP is a DLL, you still use a .LIB extension here.)

11. Make the APP. (Press the "Lightning" on the button bar, select "Project / Make" from the pulldown, or press [Ctrl-M].)

12. Test your APP.

Remember, if you are using DLLs, then all references must be down. For example, you are three APPs: AAA, BBB and CCC. BBB and CCC are DLLs, while AAA is an EXE. AAA calls routines in BBB and CCC. BBB calls routines in CCC. Therefore, CCC must be made first, then BBB, then AAA. If BBB did not call anything in CCC, then you could make BBB before CCC, if desired.

Libraries are different. You can have as many cross references as you wish. LIBs can even call routines in the parent EXE. This is because the linker has all modules (EXE and LIB) together at once, so it can rectify all references, none of which are resolved in a LIB. This contrasts a DLL, which is a standalone module with all of its external references resolved.

3.3 Example Programs

There are two example programs provided with the Super Limiter templates. Both can be found in the SUPER\EXAMPLES\LIMITER directory, and both user TEST.DCT as their dictionary.

TEST	This simple program doesn't do anything but test the Super Limiter. It uses all of the templates (Global Extension, Frame Extension, Lock Code, Unlock Code and Reset Code), plus it can call WATCHER. (Make sure that WATCHER is compiled before you try calling it.)
WATCHER	This program is a basic Browse showing the records in the ST::Limiter file. It can be used to help test the Super Limiter once it has been added to your application. You can reset the Limiter file, which will cause all current users to create a new seat during their next check-in. You can also simulate multiple new users to see what happens when an existing user gets bumped.

3.4 Reference Section

Limiter.Init (STRING ProgName, SHORT MaxUsers, SHORT Timeout, SHORT LockTimeout, BYTE CheckInFreq)

This procedure initializes the Super Limiter class.

Limiter.Kill ( )

This procedure shuts down the Super Limiter class.

Limiter.Logon (<BYTE Bumped>), BYTE

This procedure attempts to find a spare slot within the Limiter file for the new user. If a full compliment of users is already active, then it checks to see if one of the existing users has been inactive and can be bumped. If so, then the new user takes the old user's place. Otherwise, a "Try again later" message is displayed.

If the optional "Bumped" parameter is included, then the "You have been bumped" message is displayed if a slot cannot be found. This is usually used only when the `Limiter.CheckIn()` procedure has determined that the user's record is gone, and has to attempt another Logon.

Returns TRUE if logon was successful.

Limiter.Logoff ( )

This procedure removes the user's record from the `ST::Limiter` file.

Limiter.TakeEvent ( )

This procedure is called from the Windows ACCEPT loop. It calls Limiter.UpdateActivity for all events except EVENT:Timer.

Limiter.UpdateActivity (<BYTE Force>)

This procedure updates the user's record in the `ST::Limiter` file with the current date and time. The file itself is accessed only as often as you specify in the "Check-In Frequency" in the global settings, or if the Force parameter is used.

If the user's record has disappeared, then they may have been bumped. In this situation, the `Limiter.Logon()` procedure is called to attempt to find another open slot.

Limiter.Lock ( )

This procedure sets the "Lock" field in the user's `ST::Limiter` record. This causes other users that would like to bump the current user to wait for the "While Locked" time limit, rather than the "Simply Inactive" time limit. These time limits are controlled by the global extension template.

Limiter.Unlock ( )

This procedure resets the "Lock" field in the user's `ST::Limiter` record to zero. This means that the current user can be bumped by another user if they don't check-in for the "Simply Inactive" time limit. This time limit is controlled by the global extension template.

Limiter.Reset (<STRING Program>)

This procedure clears all records from the `ST::Limiter` file. This will NOT cause everyone to be

bumped, as the `Limiter.CheckIn()` procedure attempts to Logon again if it discovers that its record is missing.

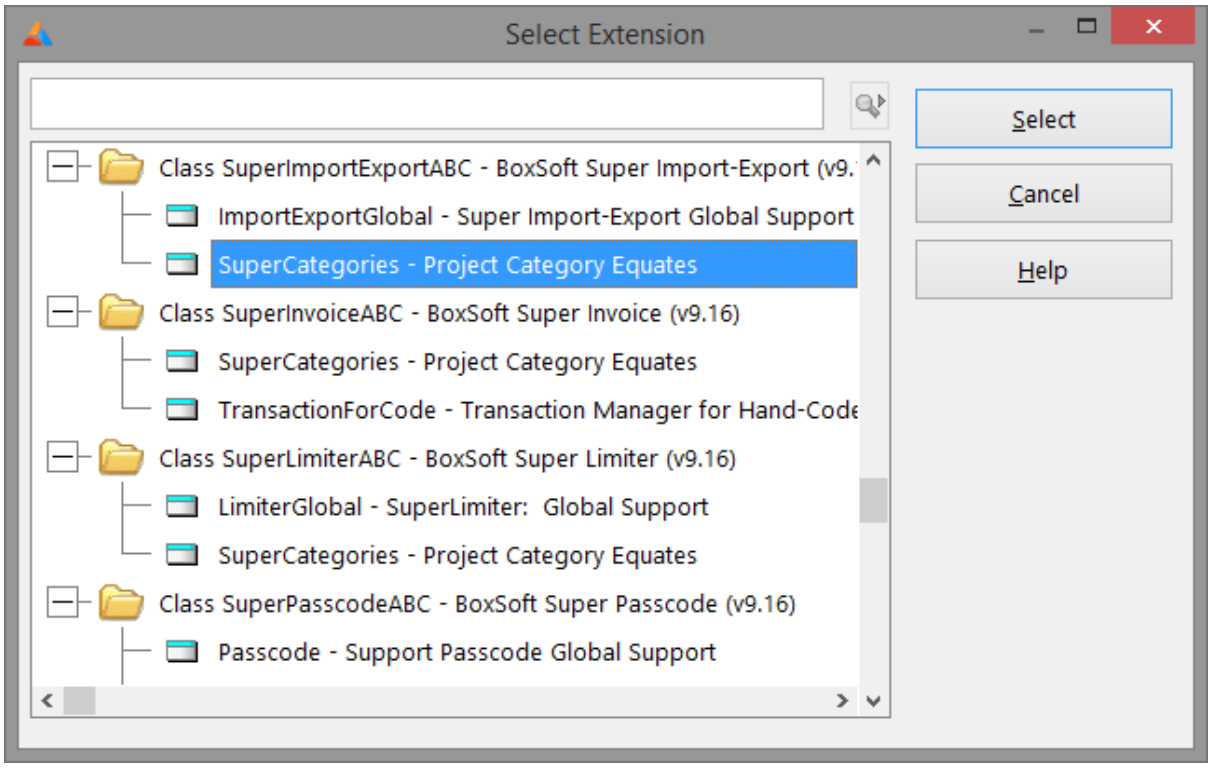
If you omit the optionally Program parameter, then the entries for the current program are erased. If you explicitly specify the parameter, then only that program's records are deleted. Finally, specifying an asterisk (`'*'`) will cause ALL program's entries to be erased.

3.5 Project Defines

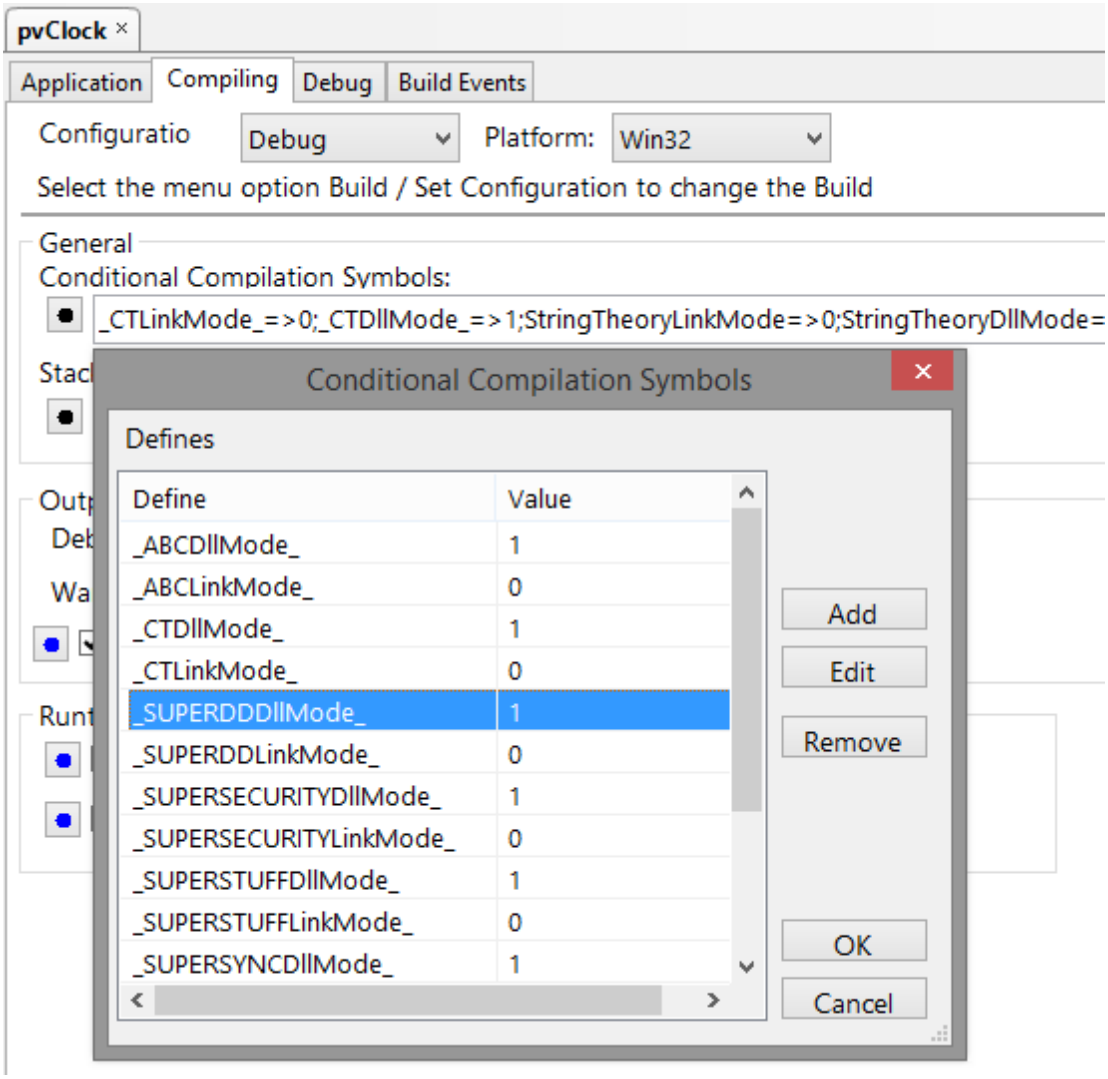
Prior to our 7.0 template versions, we were utilizing the same LINK and DLL Project Defines as the ABC libraries: _ABCLinkMode_ and _ABCDllMode_. This caused all of our libraries to be included in your base/dictionary DLL, even if you weren't using them in a particular development project.

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 _SuperQBEDllMode_), while others are associated with one of our shared base classes (e.g. Drag & Drop uses _SuperDDLinkMode_ and _SuperDDDllMode_). 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:



3.6 Troubleshooting

We have no common problems to describe at this time.

3.7 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
answers@mittensoftware.com
www.boxsoft.net
www.boxsoft.net/contact.htm

3.8 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

- A -

- C -

- D -

- E -

- F -

- G -

- I -

- L -

- M -

- P -

- R -

- S -

- T -

- U -

- W -