Super Import-Export - Documentation

Table of Contents

1 Getting Started

1.1 Introduction

The BoxSoft Super Import-Export templates provide ad hoc import and export support for your applications. It presents a simple interface to your users for importing new data to your system from another file format, or exporting existing data for use in other software packages. The real benefit is that your users can control which fields are import and exported, the order of the fields, and their format pictures. The current file format is the standard comma-delimited ASCII (*.CSV), made popular by the BASIC programming language. It is equivalent to Clarion's BASIC file driver. Here is a sample export file:

```
"Inv:Date","Inv:CusNo","Inv:Address1","Inv:Address2","Inv:Address3"
" 1/01/95",1,"123 Main St.","Unit 10","Toronto"
" 1/02/95",1,"15 Park Street","","North York"
" 1/03/95",2,"","",""
```

There is also an extension template that you can use in conjunction with Clarion's Process procedure template to quickly and easily setup a copy operation from one file to another. This can be used to export, archive, etc.

With the 7.0 version, we've added major new functionality:

- Save and restore your import and export configurations - When we first created Super Import-Export, we anticipated that it would be used for occasional ad hoc imports and exports, not for regular operations like monthly imports from a consistent outside source. For those types of things, I expected that developers would create a fixed import procedure to meet the requirement. However, our users have been urging us to support this extra functionality. Not only can you save field mapping, but we've also added significant support for automation of those saved jobs.
- Provide alternate names for your export fields, making it easier for your users to understand the data that the colum contain.
- Include local variables in exports - Use these to provide pre-defined calculations for export, which aren't saved directly in your database. For example: TodaysDate, ExtendedTotal, etc.
- User-defined formulas lets savvy users create simple calculations at runtime. It uses the EVALUATE function behind the scenes.

ABC and Legacy Template Chains

Now don't panic! With the new version we've decided to focus on the ABC template chain. We've retained the legacy chain as it was, but it's not been enhanced (beyond basic bug fixes, etc.). We felt this was an acceptable approach, as Super Import-Export procedures are functionally distinct from the rest of a system. Therefore, if you're main system uses the legacy templates, then you can create an ABC EXE application with the Import-Export functionality, and RUN this from your main system. For those who believe strongly in the legacy religion, we apologize for offending your sensibilities.

Consequently, this document pertains to our ABC template chain. The pre-existing functionality of our legacy template chain is basically a subset of this. If for some reason you choose to leave your Import-Export operations in the legacy chain, then this document will still be useful, although not entirely accurate.

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

Super Import-Export 6.x to 7.0

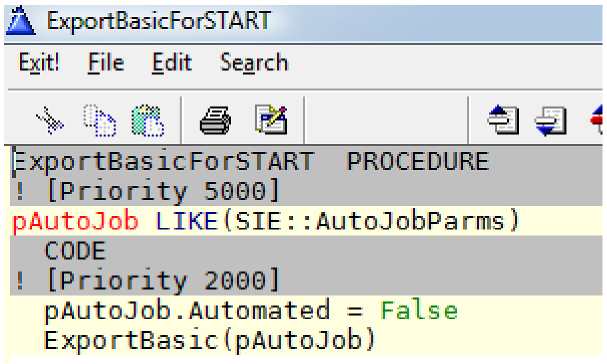
- All new features are only in the ABC template chain. (We've retained the legacy template chain to permit you to move your existing procedures over, but support is limited to bug fixes.) If your system uses the legacy templates, then create a separate ABC application for Import/ Export operations, and RUN that from your main system.

- You must import ImpExp.txd into your dictionary, and ImpExp.txa into your application. Refer to the Installation Notes to determine the location of these files on your system.

- The ImportBasic and ExportBasic procedures have required parameters now, so you must retrofit any of your existing procedures with the prototype `(SIE::AutoJobParms pAutoJob)`.

- Due to the new parameters, you won't be able to `START` an ExportBasic or ImportBasic procedure directly from your frame. You have two alternatives:

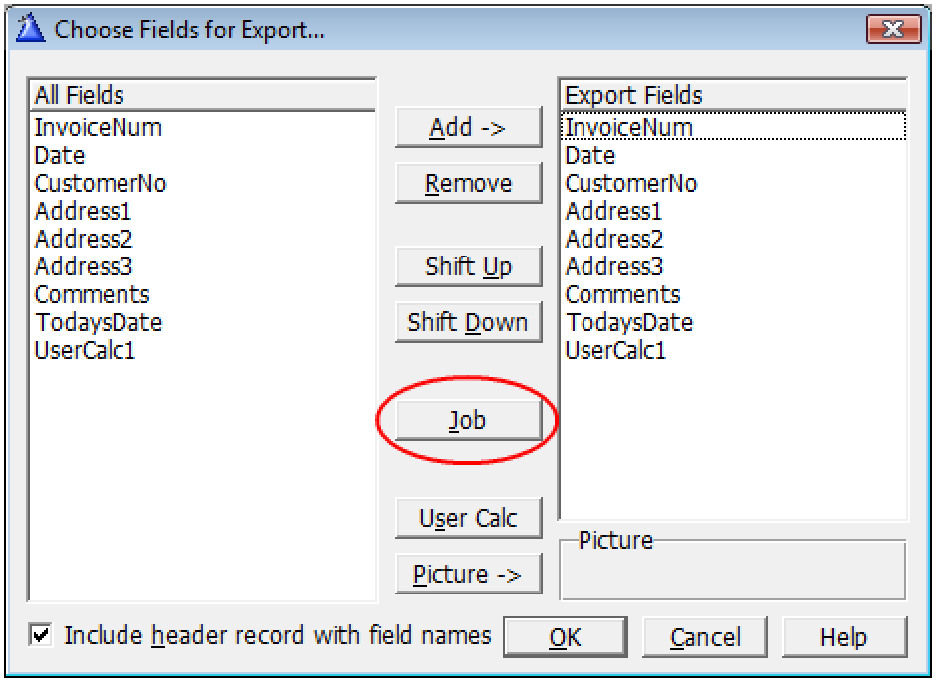
  1. You can create a regular Source procedure, looking something like this (see `ExportBasicForSTART` and `ImportBasicForSTART` in the Examples):



```
ExportBasicForSTART  PROCEDURE
! [Priority 5000]
pAutoJob LIKE(SIE::AutoJobParms)
  CODE
! [Priority 2000]
  pAutoJob.Automated = False
  ExportBasic(pAutoJob)
```

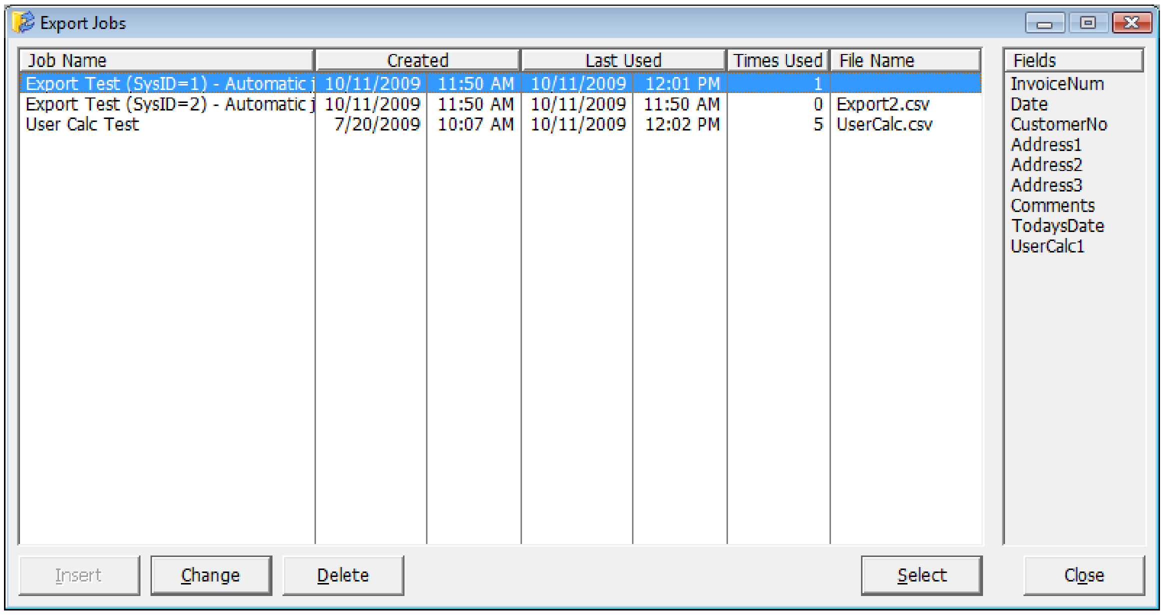
  2. You can use the new Job Caller Procedure Template. It prompts you for your desired behavior, and calls the specified ImportBasic or ExportBasic procedure appropriately.

- To add the new Job support, use the Job Support Control Template to any existing or new ImportBasic or ExportBasic window. It adds the following button to the window:



All Fields	Export Fields
InvoiceNum	InvoiceNum
Date	Date
CustomerNo	CustomerNo
Address1	Address1
Address2	Address2
Address3	Address3
Comments	Comments
TodaysDate	TodaysDate
UserCalc1	UserCalc1

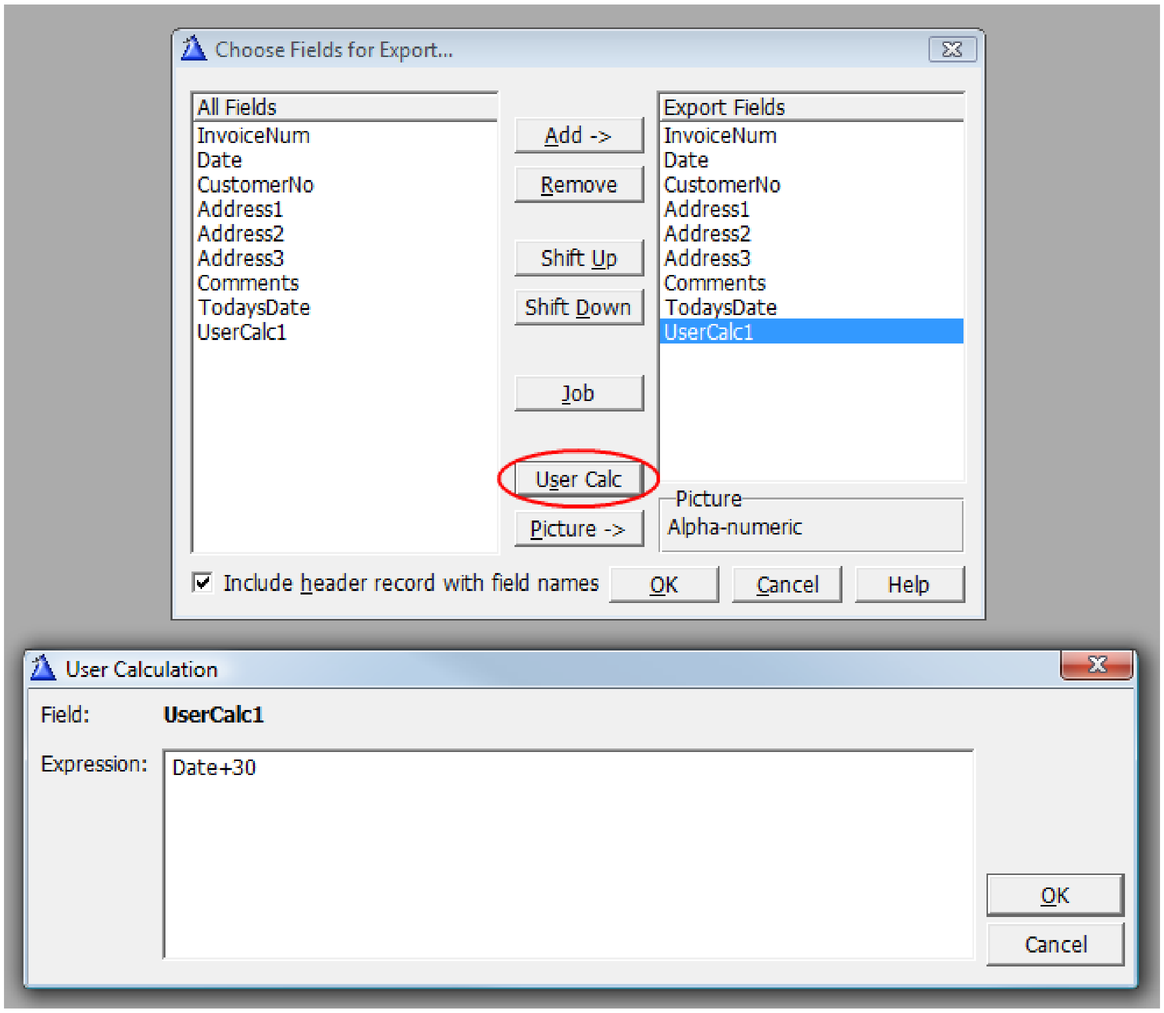
☑ Include header record with field names



Job Name	Created		Last Used		Times Used	File Name
Export Test (SysID=1) - Automatic j	10/11/2009	11:50 AM	10/11/2009	12:01 PM	1	
Export Test (SysID=2) - Automatic j	10/11/2009	11:50 AM	10/11/2009	11:50 AM	0	Export2.csv
User Calc Test	7/20/2009	10:07 AM	10/11/2009	12:02 PM	5	UserCalc.csv

Fields
InvoiceNum
Date
CustomerNo
Address1
Address2
Address3
Comments
TodaysDate
UserCalc1

- To add support for user-specified calculations, you have to add the User Calculations Control Template. It will give you this button, which leads to the following screen:



Field: **UserCalc1**

Expression: Date+30

2 Template Usage

2.1 Adding Import and Export to your Applications

The Super Import-Export templates are implemented via an combination of elements: of imported ImpExp.txd, ImpExp.txa, templates, classes, routines, etc. The templates should handle all interfacing with the API, so you should almost never have to call it directly. If you are adventurous, though, you can certainly give it a try. (For more information on this, see API Reference.)

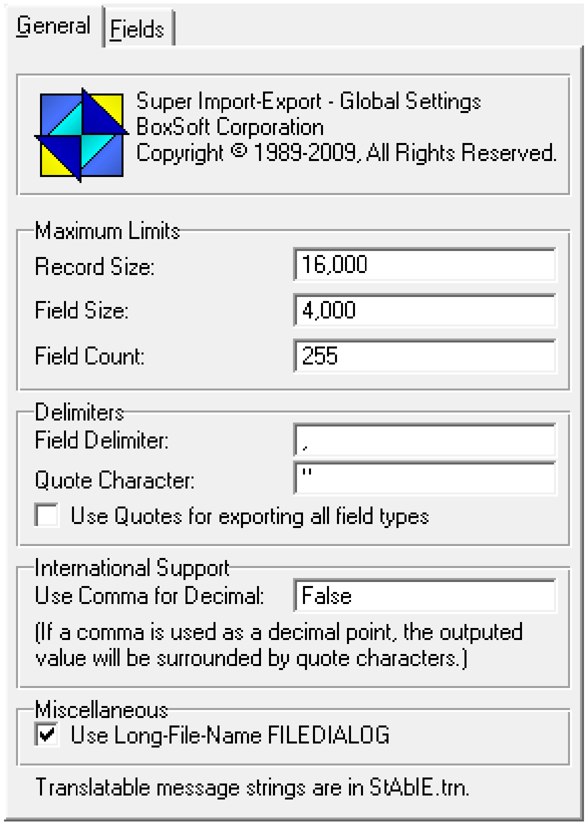
To add the Import-Export to your application, you must begin by importing ImpExp.txd into your dictionary, and ImpExp.txa into your APP. (For the location of these files, see the Installation Notes.) Then you add the Global Support extension template.

Once you've done this, you can proceed to add one or more of the ImportBasic, ExportBasic, FileCopy, and supporting templates.

2.2 Global Extension Template

This template generates a module called "App$IE.CLW", where "App" is the first five characters of your application name. There are various options you can use to control the behavior of the system. If you are creating a multi-APP system, then most of these options will be available only in the base APP.

General Tab



Super Import-Export - Global Settings
BoxSoft Corporation
Copyright © 1989-2009, All Rights Reserved.

Maximum Limits

Record Size:	16,000
Field Size:	4,000
Field Count:	255

Delimiters

Field Delimiter:	,
Quote Character:	"

☐ Use Quotes for exporting all field types

International Support

Use Comma for Decimal: False

(If a comma is used as a decimal point, the outputed value will be surrounded by quote characters.)

Miscellaneous

☑ Use Long-File-Name FILEDIALOG

Translatable message strings are in StAbIE.trn.

Maximum Limits - These limits control internal variables used by the functions. The smaller you can make them, the smaller your program will be.

Record Size - This is the largest allowable import/export record. The maximum is 64,000. The default is 16,000. Remember that this is a variable length record, so information takes less space that it would in a corresponding fixed length record.

Field Size - This is the largest allowable field. The maximum is 16,000. The default is 4,000.

Field Count - This is the largest number of export fields. The default is 255.

Delimiters - These entries control how the BASIC record is formatted. You can specify a constant string, or a variable preceded by an exclamation point. The record delimiter is a CR+LF, and cannot be changed.

Field Delimiter - This separates the individual fields. The default is a comma (,). There is no field delimiter before the first field or after the last field.

Quote Character - This is used to surround alpha-numeric fields and formatted picture fields (like dates). The default is a double quote (").

Use Quotes for exporting all field types - Do you want all fields (including numbers) surrounded by quotes?

International Support

Use Comma for Decimal - This is a conditional expression. If you want comma used for decimal points, this expression should be True. The default is "False". If this is true, all numbers will be exported enclosed in quotes, to prevent confusion with virtually all programs that read CSV files.

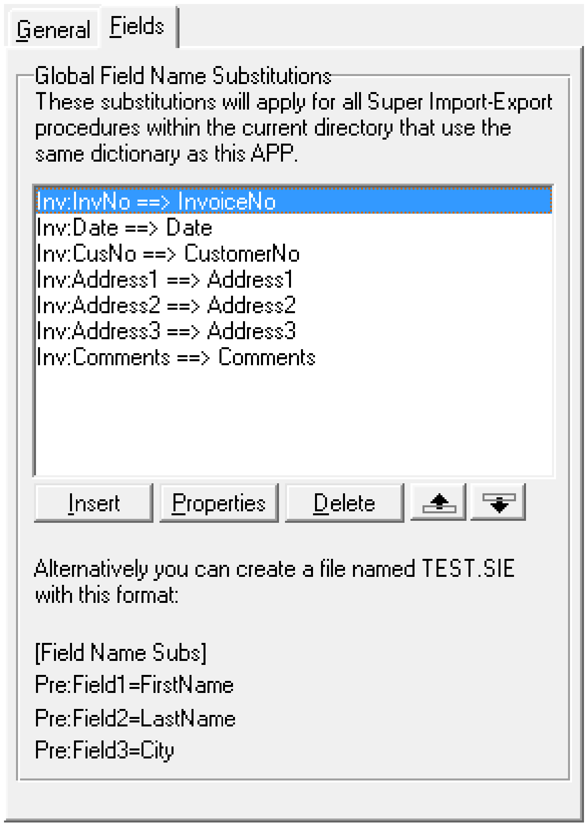
Miscellaneous

Use Long-File-Name DIALOG - It's recommended that you leave this on.

Note that various strings for messages and dialogs are defined in ...\3rdParty\LibSrc\StAbIE.trn.

Fields Tab

(New in Version 7)



Global Field Name Substitutions
These substitutions will apply for all Super Import-Export procedures within the current directory that use the same dictionary as this APP.

Inv:InvNo ==> InvoiceNo
Inv:Date ==> Date
Inv:CusNo ==> CustomerNo
Inv:Address1 ==> Address1
Inv:Address2 ==> Address2
Inv:Address3 ==> Address3
Inv:Comments ==> Comments

Alternatively you can create a file named TEST.SIE with this format:

```
[Field Name Subs]
Pre:Field1=FirstName
Pre:Field2=LastName
Pre:Field3=City
```

You can override the field names presented to your users. These are specified globally, so multiple procedures will automatically share these names.

The names that you specify must following the standard rules for labels (starts with a letter; is comprised of letters, numerals, underscores or colons; includes no spaces; etc.).

If entering all of your field name substitutions into this screen seems tedious, you can alternatively create a text file with the name pairs. The template actually saves a copy of these names into <YourDictionaryName>.SIE. You can edit this file and add any entries that you desire. As you can see, the structure follow that of a standard INI file. Note that this file is used only at design time, not when your application is running.

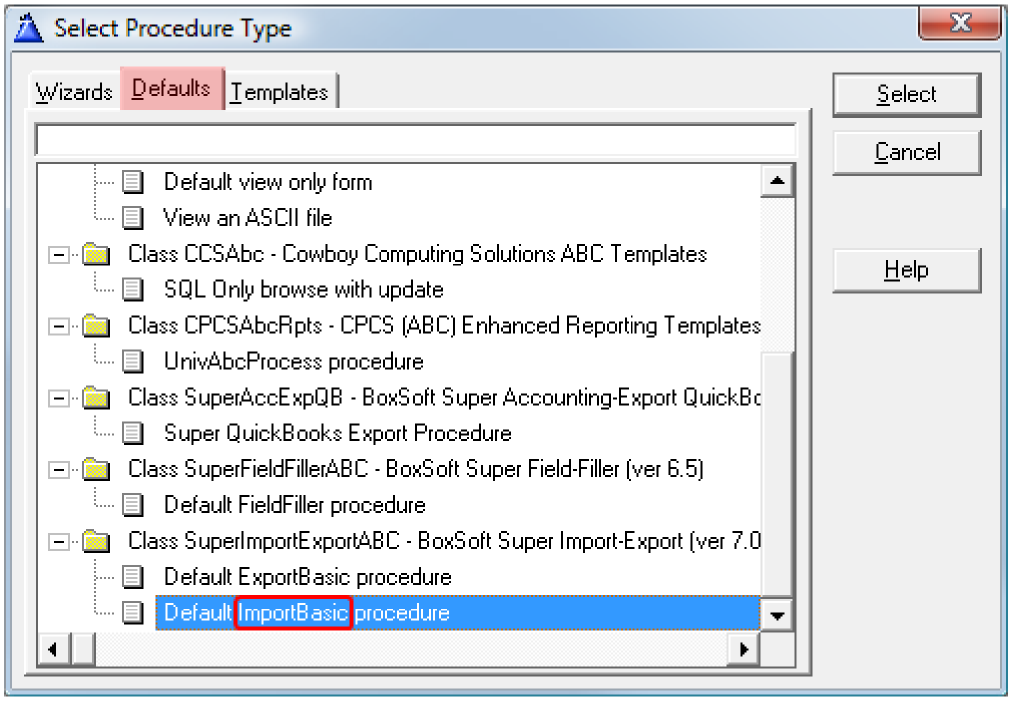
2.3 ImportBasic Procedure Template

This procedure template allows your users to import data from a comma-delimited ASCII file into one of your data files. The field assignments are decided by the user at run-time. They can also specify the deformatting picture.

The user can preview import records to determine which import fields are to be matched to the target fields. They can also strip the first record, in case it contains field names. If the output came from the same program and the field names match, then you can also "auto assign" the import fields to the target fields using the values from the first import record.

New in Version 7 - The ImportBasic procedure has required parameters now, so you must retrofit any of your existing procedure with the prototype `(SIE::AutoJobParms pAutoJob)`.

NOTE: When adding the new procedure, be sure to select the procedure template from the "Default" tab:


Select Procedure Type
Wizards
Defaults
Templates
Select
Cancel
Help
Default view only form
View an ASCII file
Class CCSAbc - Cowboy Computing Solutions ABC Templates
SQL Only browse with update
Class CPCSAbcRpts - CPCS (ABC) Enhanced Reporting Templates
UnivAbcProcess procedure
Class SuperAccExpQB - BoxSoft Super Accounting-Export QuickBc
Super QuickBooks Export Procedure
Class SuperFieldFillerABC - BoxSoft Super Field-Filler (ver 6.5)
Default FieldFiller procedure
Class SuperImportExportABC - BoxSoft Super Import-Export (ver 7.0
Default ExportBasic procedure
Default ImportBasic procedure

Once you have created your ImportBasic procedure, you must specify the target file. You do this using the [Files] button on the procedure properties window. Related secondary files are not supported by this template.

The default window contains the following controls:

Import Field List Box - This list box contains a separate entry for each field found in the import file. Beside each field is the picture. (For @S fields they will see "Alpha-num", for @N fields they will see "Numeric", and for all others they will see the actual "@..." picture.) Beside the picture is the value of the field in the current import record. If the first record in the import file contains field names, then the first values that the user sees will be these names. This can make it very handy for assigning the import fields to the target fields. They can press the [Next] button to view other records in the import file. As you move the highlighter bar over Import Fields that have been assigned to Target Fields, the Target Field list box will move its highlighter bar to the corresponding entry. You should not remove this list box.

Target Field List Box - This list box shows which import fields will be assign to which target fields. (The fields that are available here are controlled by the Include and Exclude options behind the [ImportBasic Properties] button on the procedure properties window). As you move the highlighter bar over Target Fields with assigned Import Fields, the Import Field list box will move its highlighter bar to the corresponding entry. You should not remove this list box.

[Next] Button - This button causes the next record to be read from the import file and displayed in the "Import Fields" list box. If the user reaches the end of the file, then the [Next] button is disabled. If desired, you may remove this button.

"Rec #" and "RecordNumber" Strings - These display the current import record number. This is for information purposes only, and it has no effect on the import process. If desired, you may remove these strings.

[Picture] Button - This button calls the picture entry window for the currently highlighted Import Field. This picture will be used to deformat the import field. This is especially useful if the import field is a formatted date. If desired, you may remove this button.

[Assign] Button - This assigns the currently highlighted Import Field to the currently highlighted Target Field. You should not remove this button.

[Auto] Button - The full name for this button is "Auto Assign". It is only available when the first import record is displayed. If the field names from the import header record happen to match your target fields, then you can automatically assign anything that matches. If desired, you may remove this button.

[Unassign] Button - This button clears the assignment for the currently highlighted Target Field. You should not remove this button.

"Strip header record (field names)" Check Box - This controls whether the first record of the import file is ignored. If the user presses the [Auto] button, then this defaults to true. You should not remove this control.

[OK] Button - This begins the import process. You should not remove this control.

[Cancel] Button - This cancels the import process. You should not remove this control.

[Help] Button - This calls the help window. You must set-up the help topics yourself. If desired, you may remove this control.

The rest of the settings are behind the [ImportBasic Properties] button on the main Procedure Properties window. They are as follows:

Import File Tab

Import Filename - You can allow the user to choose the filename at run-time, or you can specify the filename yourself. If you specify the filename here, you can provide a constant string, or a variable preceded by an exclamation point.

Quick-Scan Records - Turning this on will speed up the import. The file is opened ReadOnly+DenyWrite, so there is almost no benefit to turning it off.

Record Filter - This filter expression is tested after the import fields have been assigned, but before the record has been added. If the expression is false, then the record is skipped.

"No Import Fields Selected" Message - This message is displayed if the user presses the [OK] button before assigning any of the import fields to the available target fields.

Target File Tab

Always empty target before starting - If you turn this on, then the target file will be emptied every time you run the import. You can specify whether the file should be closed, created then re-opened, or locked, emptied then unlocked.

Prime fields using Dictionary settings - Do you want to enforce the dictionary priming settings for the file?

Handle Insert (or Update) code manually - If you turn this ON, you'll be expected to issue the necessary Insert/ADD statement yourself (or update existing records if applicable). It will be left completely up to you. The template will continue to process the import records and assign the field values, but the rest is left up to you.

Handle errors manually with TryInsert - This is similar to the previous setting, except that the template will attempt to add the record with TryInsert, and then you're responsible for handling any errors.

Determine new auto-numbers for target - If you turn this on, then the template will use the dictionary information to auto-number each new record added to the target file.

Depending on your settings above, a bit of sample code will appear. You can use this to help determine which combination of settings best suits your needs.

Include Tab

Here you can specify individual fields from the target file that will be presented to the user. This overrides your settings for "dictionary auto-populate" and "exclude groups" on the "Exclude" Tab. The template automatically includes all fields, so this is necessary only as an override against automatic exclusions.

If you want to specify a dimensioned field, it is treated like a group in the pop-up window. You must expand its tree node before selected the desired array element.

Exclude Tab

Use "Auto-populate" options from dictionary - This tells the template to check your dictionary to determine which fields that the user shouldn't see. Individual fields can be overridden on the "Include" tab.

Automatically exclude GROUPs - This tells the template to exclude any GROUP fields that it finds in your file. Individual groups can be overridden on the "Include" tab.

Individual Fields - This list box contains any additional fields that you wish to exclude. If you want to specify a dimensioned field, it is treated like a group in the pop-up window. You must expand its tree before selected the desired array element.

2.4 ExportBasic Procedure Template

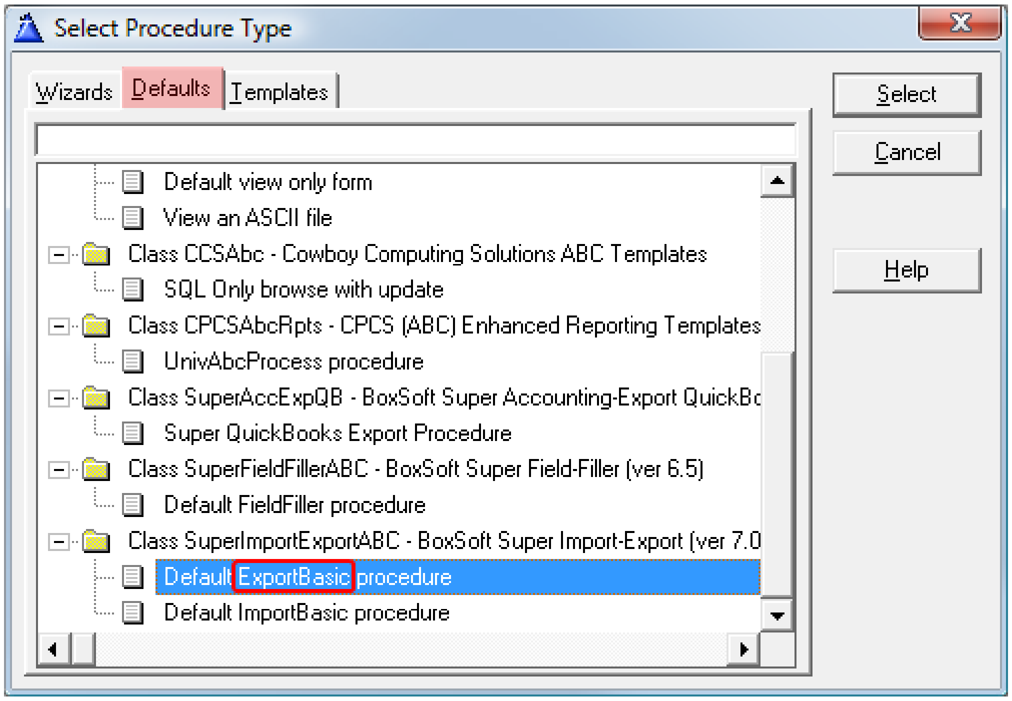
New required parameters means that it cannot be called directly with START. Instead, call it from a Source with the new template.

This procedure template allows your users to export data from your system to a comma-delimited ASCII file. The user can specify the fields to be exported, the field order within the record, and the formatting picture. They can also specify whether to include a header record with the field names.

This procedure template is actually a superset of Clarion's Process procedure template. All of the same settings and embeds are available. This means that you can specify a series of related files under your primary file to create a complex view of your system (e.g.: Customer-Invoice-InvItem-Product-Supplier). The main difference is that there is one extra object: the WindowManager is regular one (like a Form), while the ExportManager plays the part of the ProcessManager. The ExportManager is not called until the WindowManager tells it to proceed.

Default Procedure

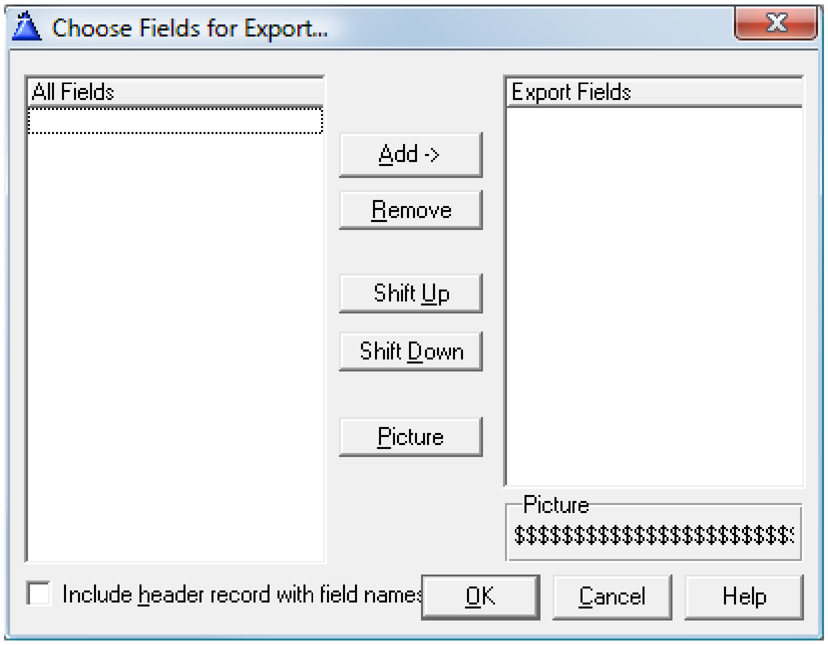
When adding the new procedure, be sure to select the procedure template from the "Default" tab:


Select Procedure Type

Wizards | Defaults | Templates

- Default view only form
- View an ASCII file
- Class CCSAbc - Cowboy Computing Solutions ABC Templates
  - SQL Only browse with update
- Class CPCSAbcRpts - CPCS (ABC) Enhanced Reporting Templates
  - UnivAbcProcess procedure
- Class SuperAccExpQB - BoxSoft Super Accounting-Export QuickBo
  - Super QuickBooks Export Procedure
- Class SuperFieldFillerABC - BoxSoft Super Field-Filler (ver 6.5)
  - Default FieldFiller procedure
- Class SuperImportExportABC - BoxSoft Super Import-Export (ver 7.0
  - Default ExportBasic procedure
  - Default ImportBasic procedure

The default window contains the following controls:



All Fields List Box - This is a list of all fields available for export. The fields that are available here are controlled by the Include and Exclude options behind the [ImportBasic Properties] button on the procedure properties window. This list box supports the complete windows style marking for highlighting fields. You should not remove this control.

Export Fields List Box - This is a list of the fields that will be exported, in order of placement within the export record. As you move from entry to entry, the "Picture" box will display the output format. This list box supports the complete windows style marking for highlighting fields. You should not remove this control.

[Add ->] Button - This button copies the currently highlighted fields from the "All" list box to the "Export" list box. If some of the fields are already in the Export list box, then they will be left in their pre-exiting positions, although they will be highlighted after the operation. You should not remove this control.

[Remove] Button - This button removes the currently highlighted fields from the "Export" list box. You should not remove this control.

[Shift Up] Button - This button shifts the currently selected Export field up one position. If desired, you may remove this button.

[Shift Down] Button - This button shifts the currently selected Export field down one position. If desired, you may remove this button.

[Picture] Button - This button calls the picture entry window for the currently highlighted Export field. This picture will be used to format the field in the export file. This is especially useful if the import field is a formatted date. If desired, you may remove this button.

Picture Box and String - This displays the format picture for the currently highlighted Export field. If desired, you may remove these two controls.

"Include header record with field names" Check Box - This controls whether the first record of the export file will be populated with the field names. You should not remove this control.

[OK] Button - This begins the export process. You should not remove this control.

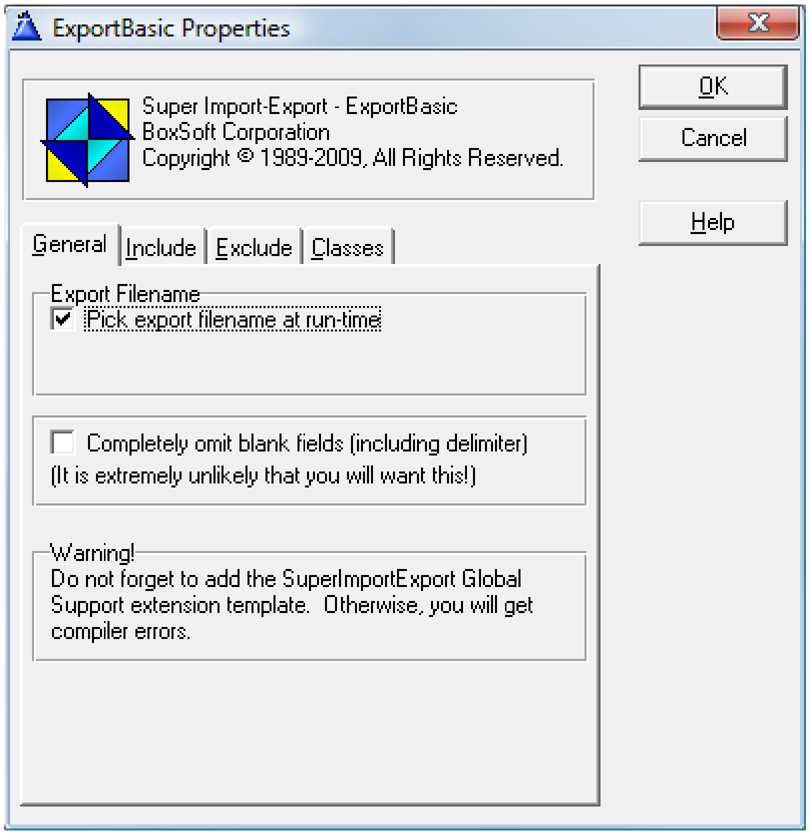
[Cancel] Button - This cancels the export process. You should not remove this control.

[Help] Button - This calls the help window. You must set-up the help topics yourself. If desired, you may remove this control.

Template Settings

There are a number of settings behind the **[Process Properties]** button. These are identical to a regular "Process".

The rest of the settings are behind the **[ExportBasic Properties]** button on the main Procedure Properties window.


ExportBasic Properties

Super Import-Export - ExportBasic
BoxSoft Corporation
Copyright © 1989-2009, All Rights Reserved.

Export Filename
☑ Pick export filename at run-time

☐ Completely omit blank fields (including delimiter)
(It is extremely unlikely that you will want this!)

Warning!
Do not forget to add the SuperImportExport Global Support extension template. Otherwise, you will get compiler errors.

General Tab

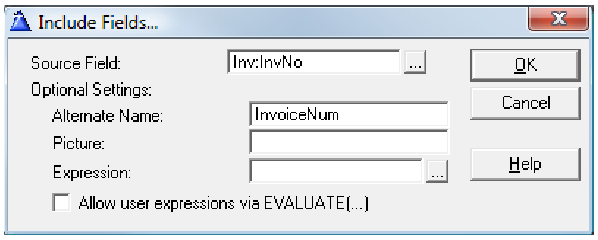
Export Filename - You can allow the user to choose the filename at run-time, or you can specify the filename yourself. If you specify the filename here, you can provide a constant string, or a variable preceded by an exclamation point.

Completely omit blank fields - This particular feature may be helpful if you're exporting the data for inclusion in a regular text document. It will prevent fields like Address2, HomePhone, etc.from being included in the export record, conditionally determined on a per-record basis.

Include Tab

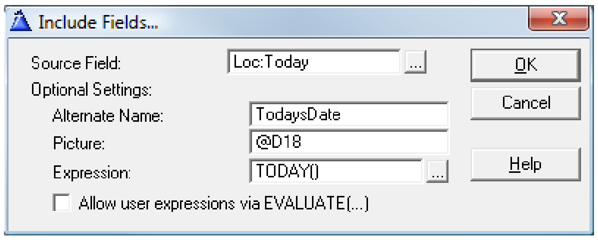
Here you can specify which fields from the source view will be presented to the user. This overrides your settings for "dictionary auto-populate" and "exclude groups" on the "Exclude" Tab. The template automatically includes all fields, so this is necessary only as an override against automatic exclusions.

You can also specify an alternate description for the field. This will be shown at run-time instead of the field name itself. Here are a few examples of usage:


Include Fields...

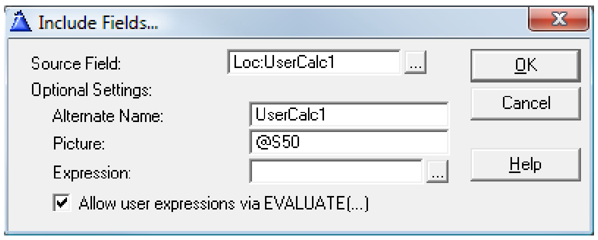
Source Field: Inv:InvNo
Optional Settings:
Alternate Name: InvoiceNum
Picture:
Expression:
☐ Allow user expressions via EVALUATE(...)

This is a regular field with an alternate name. This is similar to providing the alternate names at the global template level, except that you can tweak it on a per-procedure basis.


Include Fields...

Source Field: Loc:Today
Optional Settings:
Alternate Name: TodaysDate
Picture: @D18
Expression: TODAY()
☐ Allow user expressions via EVALUATE(...)

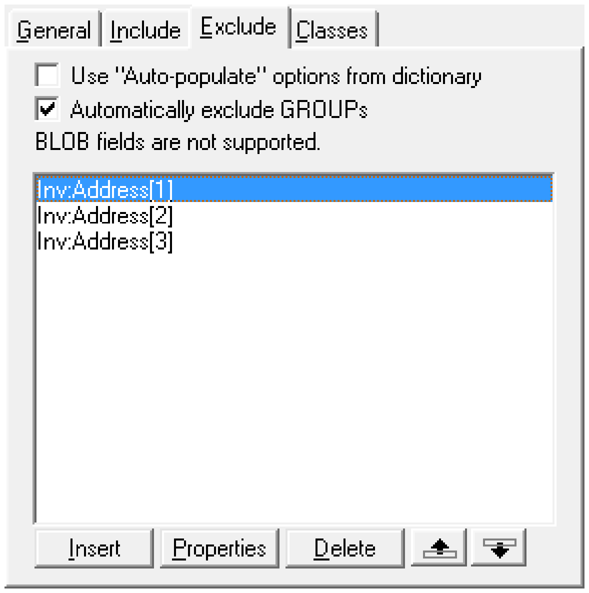
This is a run-time calculated field that your users can include in the export. Note that it names a local variable, an alternate name, a picture for export, and the runtime expression. Note that this expression is generated directly into the code, so it's not dependent on Clarion's EVALUATE function.


Include Fields...

Source Field: Loc:UserCalc1
Optional Settings:
Alternate Name: UserCalc1
Picture: @S50
Expression:
☑ Allow user expressions via EVALUATE(...)

This is how you support a user-editable expression, to be calculated at run-time with EVALUATE. You start by specifying a local variable to house the result of the expression. This will usually be a STRING(50) or something similar, with corresponding picture. (Recall that EVALUATE returns a STRING.) The expression is left blank here, and you turn ON "Allow user expressions via EVALUATE(...)". The user will see UserCalc1 in the list of available fields. They can add it to the list of export fields, then edit the expression. (The expression is entered with SIE:: AskUserExpression.)

Exclude Tab



☐ Use "Auto-populate" options from dictionary
☑ Automatically exclude GROUPs
BLOB fields are not supported.

Inv:Address[1]
Inv:Address[2]
Inv:Address[3]

Use "Auto-populate" options from dictionary - This tells the template to check your dictionary to determine which fields that the user shouldn't see. Individual fields can be overridden on the "Include" tab.

Automatically exclude GROUPs - This tells the template to exclude any GROUP fields that it finds in your file. Individual groups can be overridden on the "Include" tab.

Individual Fields - This list box contains any additional fields that you wish to exclude.

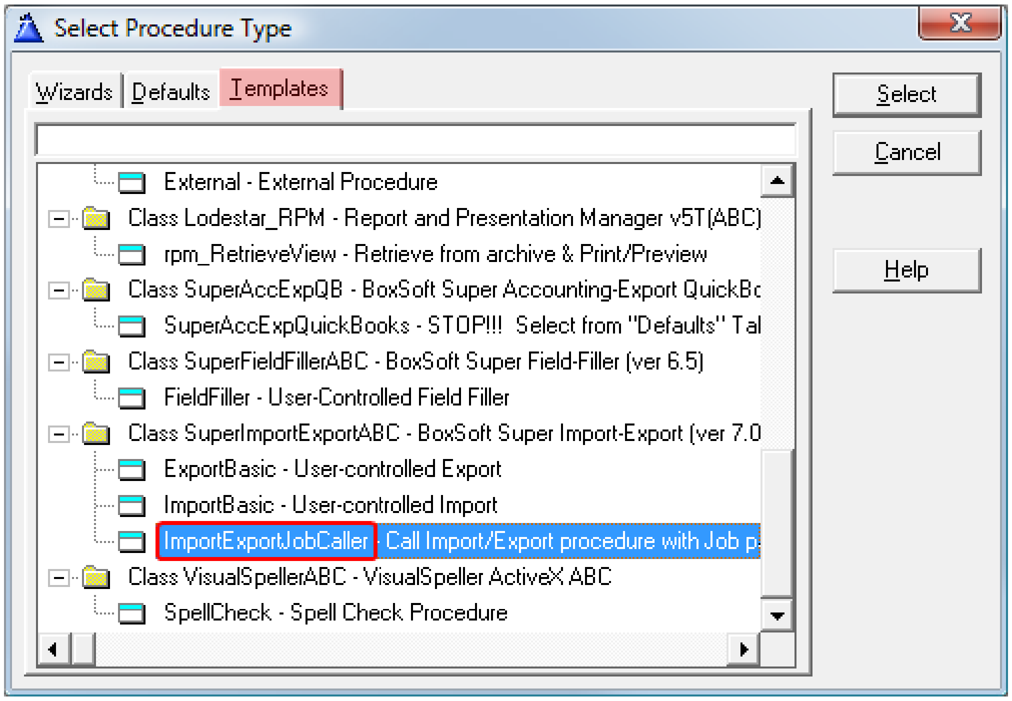
Classes Tab

This contains the standard ABC class settings. As a default, the ExportManager is called "Export".

2.5 Job Caller Procedure Template

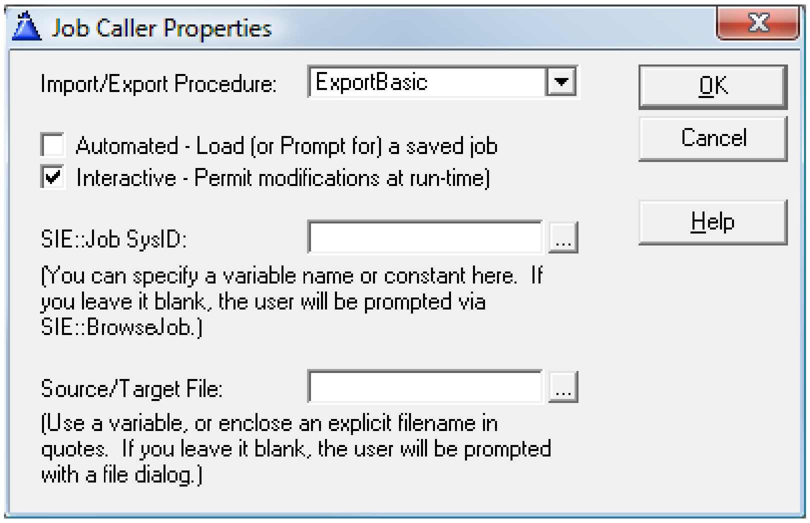
The new job parameters required for ImportBasic and ExportBasic procedures make them more complicated that before. In particular, you can no longer START those procedures directly. This new procedure template makes this a non-issue. It prompts you for the desired behavior, and then sets the parameters accordingly and calls the procedure.

You start by creating a new procedure with the template (from the Templates tab):



External - External Procedure
Class Lodestar_RPM - Report and Presentation Manager v5T(ABC)
rpm_RetrieveView - Retrieve from archive & Print/Preview
Class SuperAccExpQB - BoxSoft Super Accounting-Export QuickBc
SuperAccExpQuickBooks - STOP!!! Select from "Defaults" Tal
Class SuperFieldFillerABC - BoxSoft Super Field-Filler (ver 6.5)
FieldFiller - User-Controlled Field Filler
Class SuperImportExportABC - BoxSoft Super Import-Export (ver 7.0
ExportBasic - User-controlled Export
ImportBasic - User-controlled Import
ImportExportJobCaller - Call Import/Export procedure with Job p
Class VisualSpellerABC - VisualSpeller ActiveX ABC
SpellCheck - Spell Check Procedure

Once you create the procedure, you'll be presented with the following options. You'll likely want to experiment with various combinations of settings until it flows as you wish.



Import/Export Procedure: ExportBasic

☐ Automated - Load (or Prompt for) a saved job

☑ Interactive - Permit modifications at run-time)

SIE::Job SysID:

(You can specify a variable name or constant here. If you leave it blank, the user will be prompted via SIE::BrowseJob.)

Source/Target File:

(Use a variable, or enclose an explicit filename in quotes. If you leave it blank, the user will be prompted with a file dialog.)

Import/Export Procedure - This is the ImportBasic or ExportBasic procedure to be called.

Automated - If you want the procedure to start out completely *blank*, then leave this OFF. If, however, you expect your users to chose an existing job, or this caller is specifying the name of a saved job, then you should turn this ON.

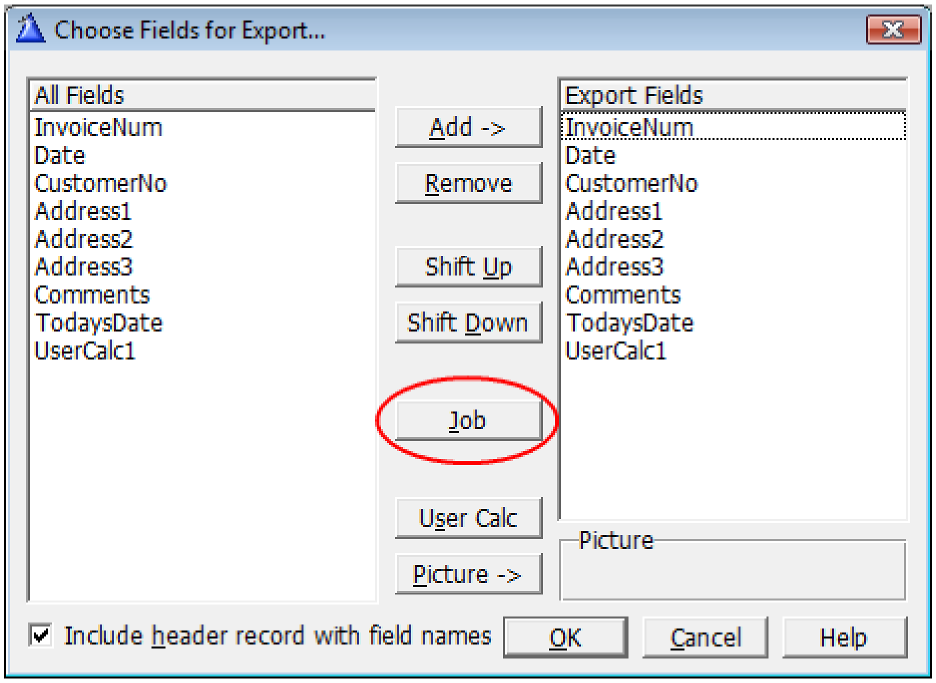
Interactive - This is related to Automated, but is distinct. If you want the procedure to process with as little user intervention as possible, then turn this ON. Otherwise, the user will be given the opportunity to override settings from the saved job, etc. Also, if Interactive is OFF, then the procedure window is hidden from the window.

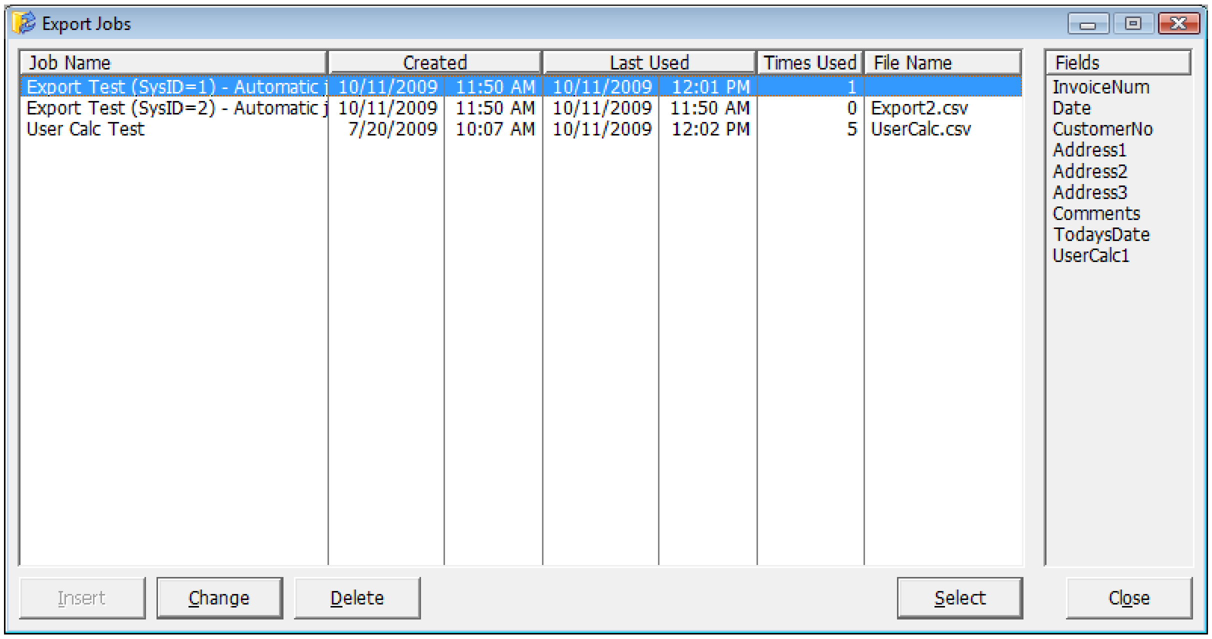
SIE::JobSysID - If you want to force it to load a particular saved job, then you can enter that job's SysID here. This is an expression prompt, so you can specify a literal number or variable name. If you leave it blank, then the user will be presented with a blank slate (with variations in behavior based upon your settings for *Automated* and *Interactive*).

Source/Target File - This is the name of the file to be imported or exported. It's an expression, so you can specify a quoted 'Filename', or a variable name. If no filename is specified here, then the user will be prompted for one at runtime.

2.6 Job Support Control Template

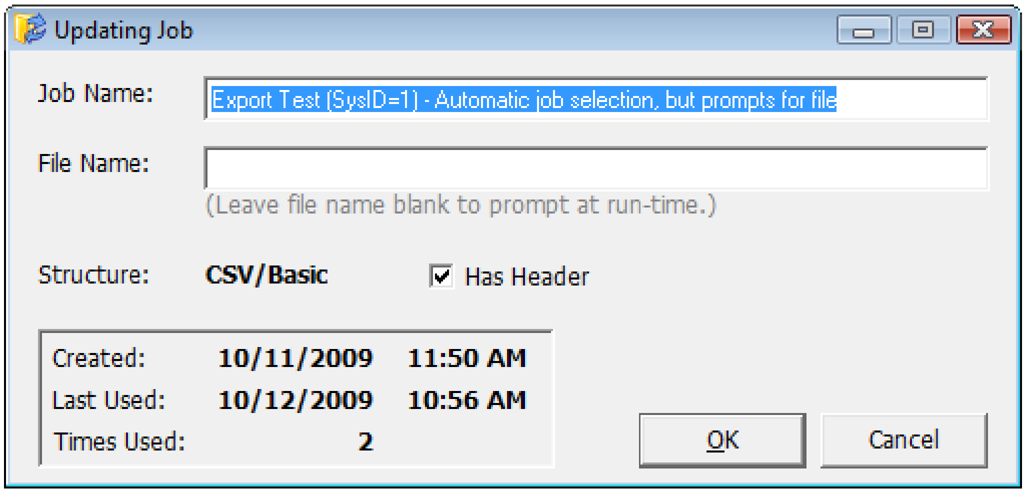
This control template enables your users to save and restore their field mappings. It works on both ImportBasic and ExportBasic procedures. When the user presses the button, they are presented with a browse of saved field mappings that are associated with that procedure:




Export Jobs

Job Name	Created		Last Used		Times Used	File Name
Export Test (SysID=1) - Automatic j	10/11/2009	11:50 AM	10/11/2009	12:01 PM	1	
Export Test (SysID=2) - Automatic j	10/11/2009	11:50 AM	10/11/2009	11:50 AM	0	Export2.csv
User Calc Test	7/20/2009	10:07 AM	10/11/2009	12:02 PM	5	UserCalc.csv

The user can specify the name of the job, a fixed file to be used for import/export, and whether the file has a header with field names. It automatically records when the job was created, when it was last used, and how many times it's been used. (A high count could indicate that you need to create a regular, code-based version of this import/export job.)


Updating Job

Job Name: Export Test (SysID=1) - Automatic job selection, but prompts for file

File Name:
(Leave file name blank to prompt at run-time.)

Structure: **CSV/Basic** ☑ Has Header

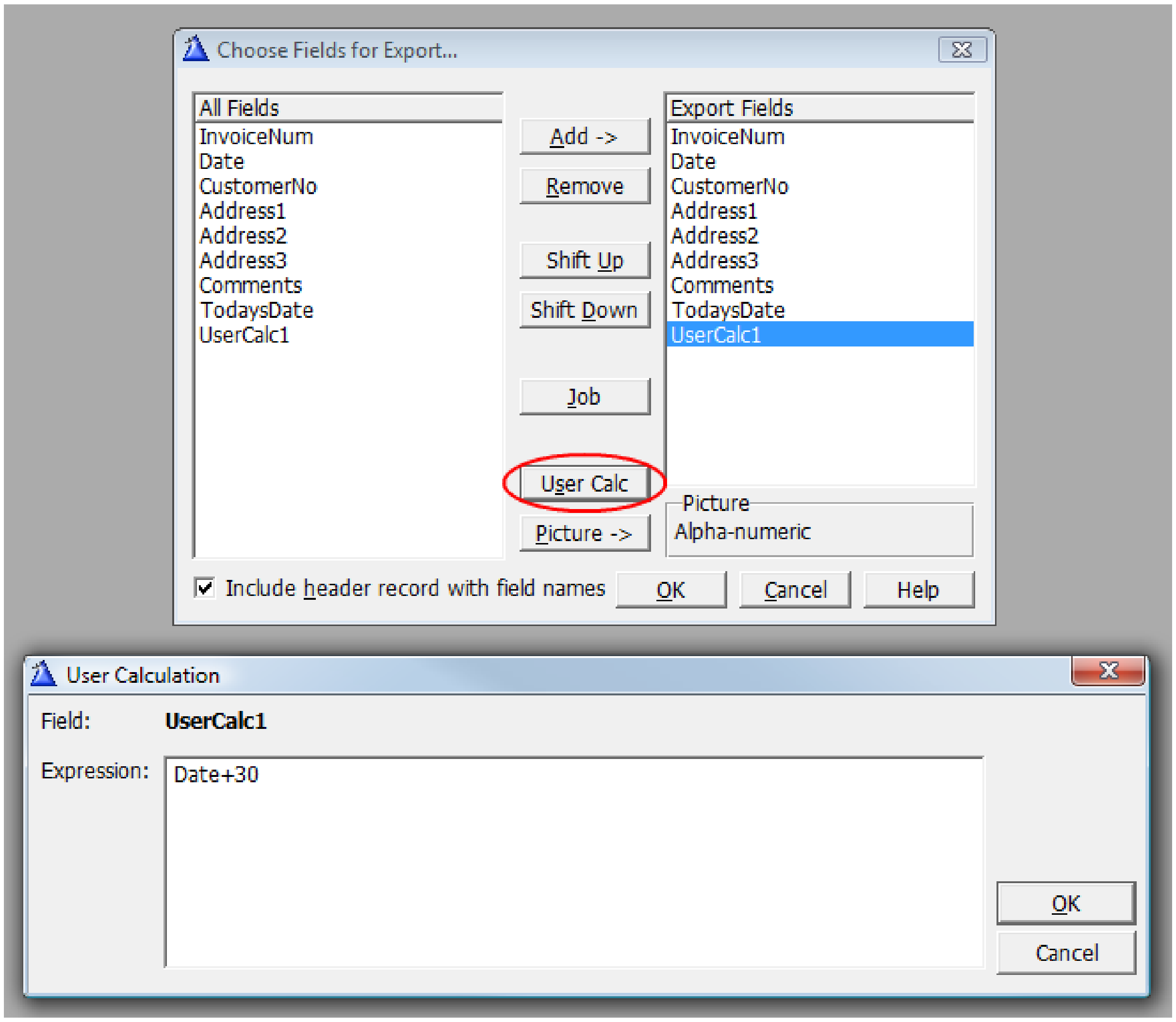
Created: 10/11/2009 11:50 AM
Last Used: 10/12/2009 10:56 AM
Times Used: 2

The jobs' configuration details are stored in two tables defined in your dictionary:

```
SIE::Job        FILE,DRIVER('TOPSPEED'),RECLAIM,NAME('SIE_Job'),PRE(SIE_J),BINDABLE,CREATE,
THREAD
PrimaryKey        KEY(SIE_J:SysID),PRIMARY
ProcNameKey       KEY(SIE_J:ProcedureName,SIE_J:JobName),NOCASE,OPT
ProcUsedKey       KEY(SIE_J:ProcedureName,-SIE_J:LastUsedDate,-SIE_J:LastUsedTime),DUP,
NOCASE,OPT
Record            RECORD,PRE()
SysID               LONG       !Auto-numbered primary key
Task                STRING(1)  !I/E for Import or Export
ProcedureName       STRING(50) !Associated Procedure name (uppercase)
JobName             STRING(100)        !User-specified Job Name
FileName            STRING(255)        !Fixed filename, to prevent prompting at runtime.
Format              BYTE       !1=CSV (Future use)
HasHeader           BYTE       !Includes header record with field names?
OverwriteTarget     BYTE       !Currently ignored in lieu of FileName
CreatedDate         LONG       !When was this created?
CreatedTime         LONG       !When was this created?
LastUsedDate        LONG       !When was this last loaded?
LastUsedTime        LONG       !When was this last loaded?
UsedCount           LONG       !How many times has this been loaded (not just run)
                  END
                END
SIE::JobField   FILE,DRIVER('TOPSPEED'),RECLAIM,NAME('SIE_JobF'),PRE(SIE_JF),BINDABLE,
CREATE,THREAD
PrimaryKey        KEY(SIE_JF:SysID),PRIMARY
JobKey            KEY(SIE_JF:JobSysID),DUP,OPT
UserExpression    MEMO(1000)   !User-defined expression
Record            RECORD,PRE()
SysID               LONG       !Auto-numbered primary key
JobSysID            LONG       !Parent Job's SysID
FieldName           STRING(50) !Field name, as shown in program
UserFieldName       STRING(50) !Future use
SourceFieldName     STRING(50) !Import source field (e.g. Field_1)
Picture             STRING(30) !Formatting picture
                  END
                END
```

2.7 User Calculations Control Template

This control template enables your users to enter their own calculations in ExportBasic procedures. (We plan to support ImportBasic in the near future.)



Choose Fields for Export...

All Fields		Export Fields
InvoiceNum	Add ->	InvoiceNum
Date	Remove	Date
CustomerNo		CustomerNo
Address1	Shift Up	Address1
Address2	Shift Down	Address2
Address3		Address3
Comments	Job	Comments
TodaysDate	User Calc	TodaysDate
UserCalc1	Picture ->	UserCalc1

Picture: Alpha-numeric

☑ Include header record with field names

OK Cancel Help

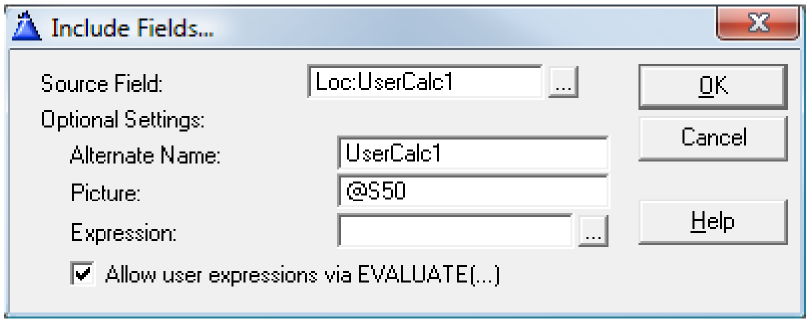
User Calculation

Field: **UserCalc1**

Expression: Date+30

OK Cancel

Calculations results need to be stored someplace, usually a local variable. You must add this to the list of fields in the ExportBasic's "Include" list, and turn ON the setting *"Allow user expressions via EVALUATE(...)"*:



Include Fields...

Source Field: Loc:UserCalc1

Optional Settings:

Alternate Name: UserCalc1

Picture: @S50

Expression:

☑ Allow user expressions via EVALUATE(...)

OK Cancel Help

These expressions are processed with Clarion's EVALUATE function, with its inherent limitations (labels must be bound, data types must be simple, etc.).

2.8 FileCopy Extension Template

This template is used to extend Clarion's Process template to copy records from one file to another. The source is the view (primary and secondary files) specified for the process. The target is a single file from your dictionary.

This template has two major benefits which make it much faster than hand-coding:

1. All matching field names are automatically assigned. This is especially helpful if you add new fields to the source and target files, but forget to edit your source code. The template doesn't forget to add the new fields.
2. Auto-numbering of the output is fully supported. This can be very complicated to write by hand.

Remember that you are working within a process template. This means that you can use this to archive records by telling the Process to DELETE the source records. Or you could set an "Archive" field in your source file and tell the process to PUT the source records.

Once you add the FileCopy extension to your Process, you must specify the target file behind the [Files] button. After that, you can go onto the FileCopy extension's various settings:

General Tab

Always empty target before starting - If you turn this on, then the target file will be emptied every time you run the process. You can specify whether the file should be closed, created then re-opened, or locked, emptied then unlocked.

Prime fields using Dictionary settings - Do you want to enforce the dictionary priming settings for the file?

Handle Insert (or Update) code manually - If you turn this ON, you'll be expected to issue the necessary Insert/ADD statement yourself (or update existing records if applicable). It will be left completely up to you. The template will continue to process the import records and assign the field values, but the rest is left up to you.

Handle errors manually with TryInsert - This is similar to the previous setting, except that the template will attempt to add the record with TryInsert, and then you're responsible for handling any errors.

Determine new auto-numbers for target - If you turn this on, then the template will use the dictionary information to auto-number each new record added to the target file.

Include Tab

Automatically match field names - This tells the system to look for matching field names between the source and target fields. For any fields that don't match, you can manually assign them using the Individual Fields list box, or the "After Assign Target Fields" embed point.

Individual Fields - This list box contains any additional fields that you wish to include. This overrides the "exclude groups" setting if the field is a group.

For each field you specify both the source and target, so you can use this to manually override a field assignment, when two fields have the same name but do not contain the same information.

If you want to specify a dimensioned field, it is treated like a group in the pop-up window. You must expand its tree before selected the desired array element.

Exclude Tab

Automatically exclude GROUPs - This tells the template to exclude any GROUP fields that it finds in your file. Individual groups can be overridden on the "Include" tab.

Individual Fields - This list box contains any additional fields that you wish to exclude. This overrides any fields automatically matched by name.

If you want to specify a dimensioned field, it is treated like a group in the pop-up window. You must expand its tree before selected the desired array element.

3 Appendices

3.1 Interface Modification and Translation

We've moved all displayable strings to ...\3rdParty\LibSrc\StAbIE.trn. This file contains equates for messages, titles, buttons and prompts throughout the Super Import-Export 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 LibSrc.

NOTE: If you have copied StAbIE.trn into your APP directory, then upgraded to a new version of Super Import-Export, don't forget to check the new TRN file for changes. If you don't keep your own copy of this file up to date, then you will get compiler errors.

3.2 Examples

Please refer to the Installation Notes for the location of the example APP and DCT. It demonstrates all of the Import-Export templates using two data files: Invoice and ArchiveInvoice. The ArchiveInvoice file has one extra field, ArchiveDate. Other than that, the two files are identical.

First take a look at the contents of the Invoice and ArchiveInvoice file. Then use the **FileCopy** procedure uses the FileCopy Extension Template to copy records from Invoice to ArchiveInvoice. This example does not delete the records from the Invoice file.

Then take a look at ad hoc exports. The central procedure for this is **ExportBasic**., which uses the ExportBasic Procedure Template, as well as as a variety of extension templates. It features local variables with developer calculations, as well as user-specified calculations, editable at runtime. It's called by **ExportBasicForSTART** (so it can be STARTed from the frame), as well as **AutoExport1-4**, which apply various levels of automation, loading pre-saved jobs, etc. Note that you can restore the standard test jobs for the example (#1 and #2) if you mess them up.

After that take a look at ad hoc imports. The central procedure for this is **ImportBasic**, which uses the ImportBasic Procedure Template. As with the exports, it's called by **ImportBasicForSTART** (for STARTing from the frame), and **AutoImport1-4** to demonstrate the automation options. You can restore the standard test jobs for the example (#3 and #4) if you mess them up.

ExportBasicForSTART and **ImportBasicForStart** are standard Source procedures, which manually handle the parameters for calling **ExportBasic** and **ImportBasic**. Alternatively, **AutoExport1-4** and **AutoImport1-4** use the Job Caller Procedure Template procedure template.

3.3 API Reference

Job Parameters for ImportBasic and ExportBasic

New in Version 7 - The ImportBasic procedure has required parameters now, so you must retrofit any of your existing procedure with the prototype `(SIE::AutoJobParms pAutoJob)`. For a full description of these attributes, refer to the Job Caller procedure template.

```
SIE::AutoJobParms    GROUP,TYPE
Automated              BYTE
SysID                  LONG
Filename               STRING(255)
Interactive            BYTE
                     END
```

Imported Procedures

`SIE::AskPicture (STRING Field, STRING Picture), STRING`

This window function is imported into your application from StAbIE.txa. It prompts the user for an alternate picture for a field. The Field parameter is the name of the field, while the Picture parameter is the current picture. It applies basic validation to the entered value, and then returns the new picture.

`SIE::AskUserExpression (STRING Field, STRING Expression), STRING`

This window function is imported into your application from StAbIE.txa. It prompts the user for a run-time expression, to be passed into the EVALUATE function as records are processed. The Field parameter is the name of the field, while the Picture parameter is the current expression. It applies basic validation to the entered value, and then returns the new expression.

`SIE::AskUserFieldName (STRING Field, STRING Expression), STRING`

This window function is imported into your application from StAbIE.txa. It prompts the user for an alternate name to be used as the column name in the exported header record.

Internal Support Procedures

`ImportBasic_OpenFile (<STRING Filename>), LONG`

This function opens the import file and issues the `SET` command. If the optional filename is omitted, then the user will be presented with the `FILEDIALOG()` to choose a file. The function returns the file size, in bytes. If the file cannot be found, the function returns zero (0).

`ImportBasic_CloseFile`

This procedure closes the currently open import file.

`ImportBasic_ResetFile (BYTE QuickScan)`

This procedure issues the `SET` command on the currently open import file to reset processing to the beginning of the file. If the `QuickScan` parameter is `True`, then quickscan will be turned on.

`ImportBasic_NextRecord, LONG, PROC`

`ImportBasic_PreviousRecord, LONG, PROC`

These functions reads the next/previous record from the import file into the import record buffer. It is not yet parsed. The return value is the number of bytes read. If the record is not available (i.e. end of file or error), the functions return zero (0). They can also be called as a procedure (if you already know that the record is there).

`ImportBasic_ResetRecord`

This procedure resets the field parser to the beginning of the record (i.e. field #1).

`ImportBasic_NextField, BYTE`

This function parses the next field from the import record buffer. The current state of the field is unformatted (i.e. the picture has not been applied). The function returns one (1) if it's a string field, two (2) if it's a number, or zero (0) if there are no more fields. The field's value is stored in `G::Import-Export_Field`.

`ImportBasic_DeformatField (STRING Picture)`

This procedure deformats the field buffer using the picture parameter. The resulting value will be stored in `G::Import-Export_Field`. Only pictures other than `@S` and `@N` are applied. Repeatedly calling `ImportBasic_DeformatField()` for a single field has no additional effect.

`ImportBasic_DeformatComma, STRING`

This procedure converts from the European use of commas as decimal points. It searches for the comma, and does a direct replacement for a decimal point. It returns the value after replacement. If there is no comma, then the original string is returned (clipped).

`ExportBasic_OpenFile (*BYTE OverwriteTarget), BYTE`

This function creates the export file, opens it, then issues the `SET` command. It uses the filename stored in G::ImportExport_Filename. If this is blank then it prompts the user for the filename (using the strings defined in StAbIE.trn). You must pass it a BYTE variable, will will indicate whether an existing file is being overwritten. The function returns `True` (1) if the operation is completed, or zero (0) if the file cannot be opened or the user aborted. Note that no header record is written when appending.

`ExportBasic_CloseFile`

This procedure closes the currently open export file.

`ExportBasic_ClearRecord`

This procedure clears the export record to begin formatting individual fields.

`ExportBasic_AppendField(*? Field, STRING Picture, <BYTE IsFieldName>)`

This procedure appends a formatted field to the existing export record. If the picture is "`@S...`", then no formatting is performed (other than bracketing with the quote characters). If the picture is "`@N...`", then no formatting is performed. (The quote character are not added to a number field.) Otherwise, the picture is used with the FORMAT command, and the result is surrounded with the quote characters. If you are formatting a header record with field names, you can include "True" as the third parameter to convert array punctuation `[,]` to a more standard format.

```
ExportBasic_AddRecord, BYTE
```

This function writes the current export record buffer to the file. If successful, it returns `True` (1). Otherwise, it returns `False` (0).

```
SIE::IsValidExpression (STRING Expression), BYTE
```

This function checks to see if `EVALUATE` can handle the passed string. All elements of the expression must be bound! If it's invalid, it displays a message and returns `False` (0). Otherwise, it returns `True` (1). The message is defined in StAbIE.trn.

```
SIE::IsValidLabel (STRING Label), BYTE
```

This function checks to see if the passed string is a acceptable Clarion label (i.e. variable name). If it's invalid, it displays a message and returns `False` (0). Otherwise, it returns `True` (1). The message is defined in StAbIE.trn.

```
SIE::IsValidPicture (STRING Picture), BYTE
```

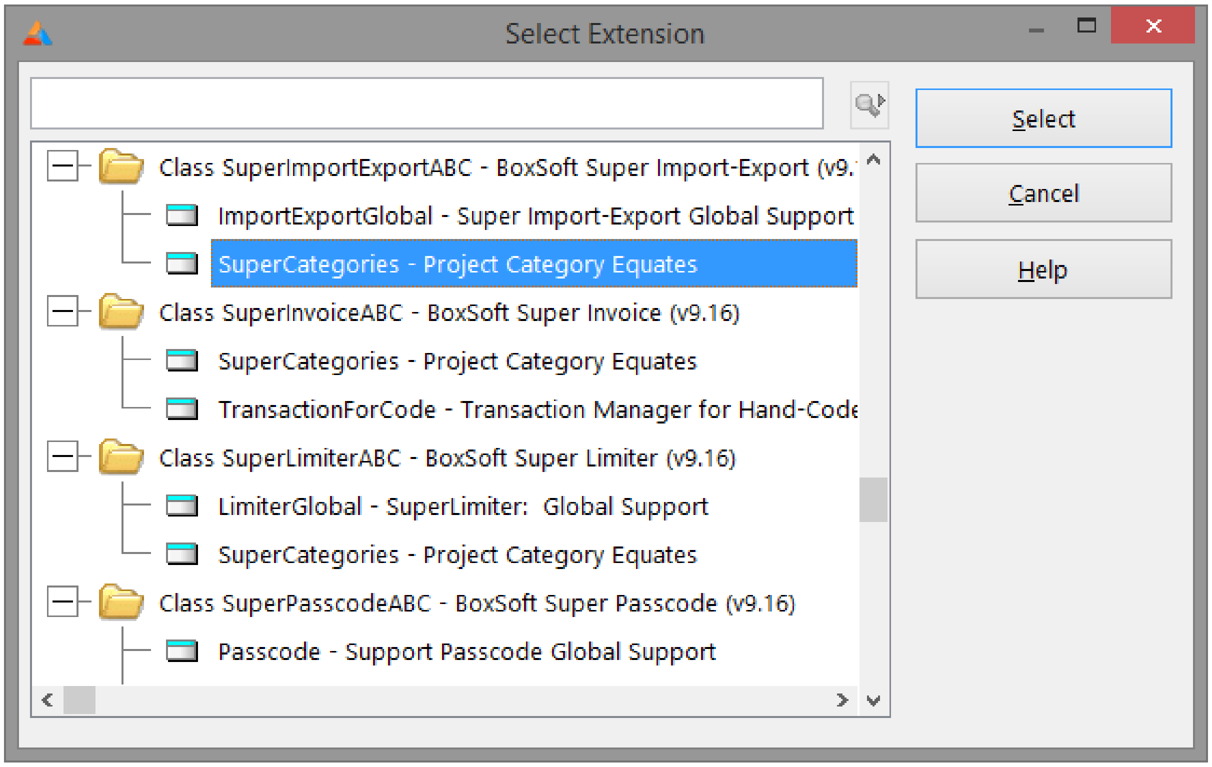
This function checks to see if the passed string is a valid Clarion picture. At this time verification is minimal: it ensures that it's not blank, and that its starts with "`@`". If it's invalid, it display a message and returns `False` (0). Otherwise, it returns `True` (1). The message is defined in StAbIE.trn.

3.4 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.

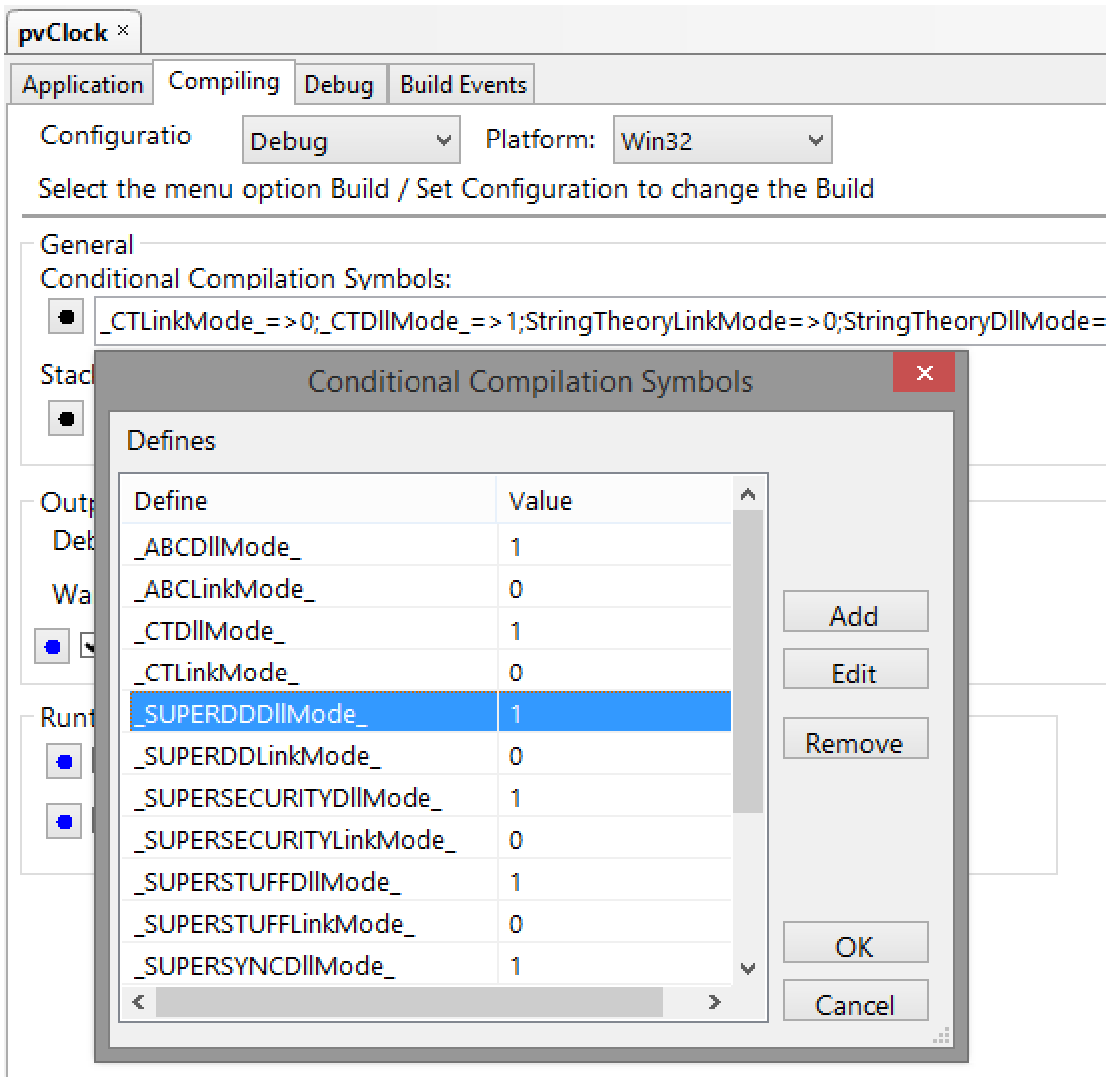
To make this inclusion more selective, as of our 7.0 versions we changed to use various other switches. Some of these are product related (e.g. Super QBE uses _SuperQBELinkMode_ and _SuperQBEDllMode_), while others are associated with one of our shared base classes (e.g. Drag & Drop uses _SuperDDLinkMode_ 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.



Class SuperImportExportABC - BoxSoft Super Import-Export (v9.
ImportExportGlobal - Super Import-Export Global Support
SuperCategories - Project Category Equates
Class SuperInvoiceABC - BoxSoft Super Invoice (v9.16)
SuperCategories - Project Category Equates
TransactionForCode - Transaction Manager for Hand-Code
Class SuperLimiterABC - BoxSoft Super Limiter (v9.16)
LimiterGlobal - SuperLimiter: Global Support
SuperCategories - Project Category Equates
Class SuperPasscodeABC - BoxSoft Super Passcode (v9.16)
Passcode - Support Passcode Global Support

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:



Select the menu option Build / Set Configuration to change the Build

Conditional Compilation Symbols:

CTLinkMode=>0;_CTDllMode_=>1;StringTheoryLinkMode=>0;StringTheoryDllMode=

Defines

Define	Value
ABCDllMode	1
ABCLinkMode	0
CTDllMode	1
CTLinkMode	0
SUPERDDDllMode	1
SUPERDDLinkMode	0
SUPERSECURITYDllMode	1
SUPERSECURITYLinkMode	0
SUPERSTUFFDllMode	1
SUPERSTUFFLinkMode	0
SUPERSYNCDllMode	1

3.5 Troubleshooting

Problem:

After creating a new ImportBasic or ExportBasic procedure, there is no default Window structure.

Solution:

When adding the new procedure, be sure to select the procedure template from the "Default" tab.

3.6 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

3.7 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