Alpha Five

Runtime and Runtime+ (for Regular and Enterprise editions)

For Windows 2000/XP/Vista

Alpha Software Inc.
70 Blanchard Rd.
Burlington, MA 01803
781.229.4500

www.alphasoftware.com

Table of Contents

Alpha Five Runtime

The Alpha Five Runtime is an inexpensive way for you to distribute your Alpha Five applications to other users, without requiring these uses to purchase a full copy of Alpha Five.

The Alpha Five Runtime is ideal for commercial developers who would like to sell applications that they have developed in Alpha Five. It is also ideal for corporate developers who wish to distribute applications to many users within their organization.

You can design your application so that the fact that the application is written in Alpha Five is largely hidden from your customers or users. For example you can:

- Specify your own splash screen image to display when the Runtime loads
- Turn off the splash screen completely when the Runtime loads
- Specify your own icon to display on the Runtime Title bar, and on the Task bar
- Specify your own title for the Title bar and Task bar, rather than the standard "Alpha Five" title.
- Customize all menus and toolbars.

NOTE You (the developer) will need a full copy of Alpha Five in order to create the applications that you wish to distribute with the Alpha Five Runtime.

A key benefit of the Runtime is that allows developers to protect their intellectual property by preventing users from seeing the source code to scripts and functions. Alpha Five also gives an extra level of protection by allowing developers to compile scripts and functions. For more information on compiling scripts and functions, open the Alpha Five help file and search on "Compiling Scripts and Functions".

Four Runtime Editions

There are four versions of the Alpha Five Runtime as summarized below:

Standard Edition	Enterprise Edition
Runtime	Runtime
Runtime+	Runtime+

The 'Standard' Editions are for use with Alpha Five's .dbf files. You cannot use ADO or AlphaDAO to access data stored in SQL databases.

The 'Enterprise' Editions are for use with both Alpha Five's .dbf files and with data in SQL databases. Both ADO and AlphaDAO are enabled in the 'Enterprise' editions.

Both Runtime and Runtime+ allow other users to run your Alpha Five applications. With the Runtime edition, users cannot modify any aspects of your application, and cannot design their own reports, letters and labels. With the Runtime+ edition, users can create new reports, labels and letters on selected tables and sets, and can edit certain existing reports, letters and labels (as long as you specifically enable this in your application design).

The differences between the Runtime and Runtime+ are summarized in the table below:

	Runtime	Runtime+
Create or Edit Forms and Browse Layouts	No	No
Create or Edit Reports (reports, labels and letters)	No	Yes (The application developer controls which reports can be edited, and for which tables/sets users can build new reports). Reports can only

		be created against native Alpha Five tables (i.e. .dbf files)
Create and edit tables using the Table Builder	No	No
Edit Field Rules	No	No
Create and edit indexes using the Index Builder	No	No
Create and edit Scripts and Functions	No	No
Create and edit Operations using any of the Operation Builders	No	No

Distributing Runtime Applications - Licensing Issues

You can distribute as many copies of the Alpha Five Runtime as you wish. The number of users who can use your application concurrently on any network is based on the type of Runtime that you purchased. **The people to whom you distribute the Alpha Five Runtime may not, in turn, distribute any copies of the Alpha Five Runtime**.

For example, if you purchased an "Unlimited 10-User Runtime", then a maximum of 10 users can use your application concurrently on any particular network.

If you purchased an Unlimited User Runtime, then there are no limitations on the number of concurrent users of your application.

How to Distribute a Runtime Application

In order to distribute an application to another user, you must distribute a copy of your own application files, and a copy of the Alpha Five Runtime.

You retain the copyright to your application files. Alpha Software retains the copyright on the Alpha Five Runtime files.

You are free to make copies of the Alpha Five Runtime Installation Program (setup.exe) to distribute to your users. You can burn your own CDs with these files.

You will have to create another installation program to install your application files. These files are the files with the following extensions:

Extension	Description
ADB	Alpha Five Database
ALB	Data Dictionary for Database
ALM	Data Dictionary for Database (memo file)
ALX	Data Dictionary for Database (index file)
CDX	Index file for a table
DBF	Table
DDD	Data Dictionary for a table
DDM	Data Dictionary for a table (memo file)
DDX	Data Dictionary for a table (index file)
FPT	Memo file for a table
SEM	Set (memo files)
SET	Set

SEX	Set (index file)

It is recommended that when you develop your application, you place all of the files that belong to that application in a single folder. Then you can be sure that as long as you install all of the files from this folder on your users' machines, you will have included all of the necessary application files.

To create an installation program for your application you can use the Alpha Five Installation Maker, or you can use a 3rd party program (such as InstallShield or Wise Install Maker).

The Alpha Five Installation Maker

The Alpha Five Installation Maker is only enabled if you have both the full version of Alpha Five and the Runtime version of Alpha Five installed on the same computer. To access the Installation Maker, start the full version of Alpha Five (not the Runtime version), and select the Tools, Utilities menu from the Control Panel. Click Help on the dialog for more information.

IMPORTANT If you write your own install program, you must register a5controls.dll and a5contexteval.dll during installation. If you have a Runtime+ edition, then you must also register all files matching 'codejock*.ocx'.

Installing a Runtime Application on a User's Computer

Single User

If you are installing a single user application, installation is simple - your install program can install the Alpha Five Runtime files and your application files on the user's computer. You install program can create a shortcut to start your application.

Multi User

If you are installing a multi user application, the installation involves multiple steps – installing the application files on a shared server, installing the Runtime files on each user's machine and (optionally) installing a "bootstrap" application on each user's machine (if your application uses Network Optimization).

IMPORTANT You should install the Alpha Five Runtime on each user's local machine. DO NOT INSTALL A SINGLE COPY OF THE RUNTIME ON A SERVER FOR USERS TO SHARE FROM THEIR DESKTOPS. Doing so could result in serious performance degradation.

IMPORTANT Each user must have read/write access to the folder on the server where the application files are installed.

For multi user applications, you should consider taking advantage of Alpha Five's Network Optimization feature. Applications that run on a network will run **much** more quickly if you use Network Optimization.

Using Network Optimization

If you want your application to use Network Optimization (recommended), then follow these steps:

1. Install the application files on a shared folder on the server.
2. Install the Runtime files on each user's machine.
3. Install a special "bootstrap" application on each user's machine.

For more information on creating a "bootstrap" application, see "Creating a Bootstrap Application" in the Appendix.

Your install program for the "bootstrap" application should include a shortcut that launches the Alpha Five Runtime and loads the "bootstrap" application. The first time the user clicks the shortcut, the "bootstrap" application will be launched. This application will then prompt the user for the path to the master copy of the application (which is installed on the shared server) and it will then create a "shadow" copy of the application.

The next time the user clicks the shortcut to start the application, the real application is loaded (and not the "bootstrap" application).

Not using Network Optimization

If you do not want to use Network Optimization (not recommended), then follow these steps:

1. Install the application files on a shared folder on the server.
2. Install the Runtime files on each user's machine.

To start the Application, a user will start the Alpha Five Runtime on their machine and then use the File/Open command to open the application on the server.

After the user opens the application for the first time, they can create a shortcut that will start the Runtime and also open the master copy of the application.

How to Make a Shortcut

Start Alpha Five and from the Control Panel, select the Tools, Create Shortcut… command.

Designing an Application for the Runtime

When you design an application that you intend to run with the Alpha Five Runtime, you should make sure that you keep the following in mind:

1. **Be sure to specify a "start up" form, or create an Autoexec script that loads the startup form.**

The "start up" form is loaded automatically whenever Alpha Five loads a database file (i.e., an .adb file). The start up form should be the main menu of your application. For example, the AlphaSports application that ships with Alpha Five has a form called "Main Menu" that loads automatically whenever the alphasports.adb file is opened.

AlphaSports Start Up Form

The buttons on the start up form should open the various forms and reports in your application.

To specify a start up form when you are designing your application, select the File, Database Properties command, select the form from the list of available forms, and then check the Run on load checkbox.

As an alternative to specifying a startup form in File, Database Properties, you can create an Autoexec script that loads the startup form. For example, if you want your application to display the "Main Menu" form on startup, your Autoexec script would contain this command:

```
DIM pForm as p
pForm = form.view("Main Menu")
```

2. **Design a custom menu and toolbar for all forms displayed by your Application.**

You probably do not want your application to display the default Alpha Five menus and toolbars when your application runs. To specify a custom menu and toolbar (or no menu or toolbar at all), edit the form, select Form, Properties, click on the Menus/Toolbars tab and enter the name of the custom menu and toolbar.

3. **Ensure that when the user closes the start up form you also close the Alpha Five Runtime.**

If you allow the user to close your start up form, without also closing the Alpha Five Runtime itself, then the user will be left staring at the Alpha Five Control Panel. This is probably undesirable.

You should ensure that when the user closes the start up form, that the Alpha Five Runtime is also closed. You can do this by using the `A5.close()` method.

For example, assume that you put a button on the start up form to close the application. You could attach this code to the button's OnPush event:

```
response = ui_msg_box("","Are you sure you want to ↵
 close the application?", UI_YES_NO)
if response= UI_YES_SELECTED  then
  a5.close()
else
  choice="NO"
end if
```

You might also want to put a "Close" or "Exit" command on the custom menu that you design for the start up form. You would attach the same code above to this menu option.

The user can also close the start up form by clicking on the X button in the start up form's title bar. You should therefore attach the following script to the form's CanExit event:

```
response = ui_msg_box("","Are you sure you want to ↵
 close the application?", UI_YES_NO)
if response= UI_YES_SELECTED  then
  a5.close()
else
  cancel()
end if
```

4. **Put buttons, or menu items, to perform system maintenance options on your start up form and menu.**

System maintenance functions, such as rebuilding corrupt indexes, packing tables, refreshing network optimized databases etc. should all be made available to users of your applications. You should have a button on your start up form, or a command on the start up form's menu that allows the user to perform certain system maintenance tasks.

While in normal operation, indexes should never become corrupt, if an application is improperly terminated, or a power glitch occurs, indexes can become corrupt. Therefore, your application should include a button to update indexes.

NOTE Creating a script to update indexes in all tables is easy. There is an option in Action Scripting to create a script that updates indexes and packs tables for multiple tables.

If your application is designed to run on a network, and you are using Alpha Five's Network Optimization feature, you might need to include a command that allows the Alpha Five Runtime user to refresh the application. This is

necessary if the master copy of the application is updated. The Refresh Shadow command must be run from each workstation to ensure that the latest copy of the master application is copied to the local workstation.

You can create a command button or menu choice on the start up form with the following code to refresh the application:

```
'Update local copy of Network Optimized database
'-------------------------------------------------------------
refresh_shadow()
'Tell user that Alpha Five must be restarted
'-------------------------------------------------------------
ui_msg_box("Warning", "The application must now be restarted")
a5.close()
```

NOTE An alternative to making the Refresh Shadow command available to users of your application is to set an option when you create the Network Optimized database to automatically update the shadow databases whenever the master database is updated.

5. Create a global script called Autoexec

You may want to create a global script (click on the Code tab of the Control Panel and click the New button) called Autoexec to perform "start up" tasks for your application. The autoexec script is automatically executed when Alpha Five starts up.

For example, you might want to open several forms when the application is started, or prompt for a user name and password, or set the value in some variable etc.

6. Creating Demo Versions of your Application

You may want to create demo versions of your application that your customers can use in some limited fashion until they have purchased the full application. For example, you want to supply the whole application to them, but restrict them to entering a certain number of records. To do this, you would attach a script to each table's CanSaveRecord event, which is defined in Field Rules. The following script shows how to restrict the user to just 50 records in a table:

```
T=table.current()
Records = t.records_get()
If Records > 50 then
   Cancel()
End If
```

Runtime+ Edition

Features of the Runtime+ Edition

- Allow users to edit selected reports, labels and letters
- Allow users to create new reports, labels and letters
- Protect "system" reports, labels and letters so that users cannot overwrite them.
- Designate which tables and sets users can create reports on.
- Provide friendly names for tables and set when users select a data source for a new report.

Using the Report, Label or Letter Editor in the Runtime+ Edition

The Runtime+ includes the Alpha Five report editor, label editor and letter error (collectively referred to as the report editors).

NOTE: Runtime+ users cannot invoke the report editors from the Control Panel. As part of your application design, you must create menu entries that execute the appropriate Xbasic (described below) to launch the report editor. Runtime+ users can only edit and create reports that are based on native Alpha Five tables – i.e. .dbf files.

Editing an Existing Report

The Xbasic command to launch the report editor is:

```
<ObjectP> = a5_layout_design as c (object_name as c, type as c )
```

Where *object_name* is the name of the Layout, and *type* is "report", "letter", or "label". The *object_name* can be optionally qualified with a data dictionary to distinguish between objects with the same name from different dictionaries, or to resolve *object_names* for tables and sets that are not part of the current database.

For example:

```
a5_layout_design("invoice","report")
a5_layout_design("invoice@invoice.set","report")
a5_layout_design("ship_labels","label")
a5_layout_design("ship_labels@c:\data\customer.ddd","label")
a5_layout_design("thankYouLetter@customer.ddd","letter")
```

Creating a New Report

You can also launch the Report, Label and Letter editors to create new layouts. The syntax is:

```
a5_new_report_layout(C report_type ,C table_set_name , C method )
```

ObjectP	The function returns a pointer to the report editor window. This is useful if you want to change a property in the report. E.g. `ObjectP.window_title = "MyCustomReport"`
report_type	Options are: report, label or letter
table_set_name	This is the table or set on which the report is based. In the case of a table you do not include the extension. If the table is part of the current database, you do not need to include the drive/path.

	In the case of a set you must include the .set extension. If the set is part of the current database, you do not need to include the drive/path.
method	Options are: "prompt" – prompt the user whether to go directly to the editor, or first display the genie, "genie", or "editor". Note: These options are only honored for reports. For labels and letters, Alpha Five always prompts as to whether you use the genie.

For example:

```
a5_new_report_layout("report","customer","prompt")
a5_new_report_layout("report","invioice.set","editor")
a5_new_report_layout("report","customer","genie")
a5_new_report_layout("letter","customer","prompt")
a5_new_report_layout("label","customer","prompt")
a5_new_report_layout("label","customer","editor") 'not supported
a5_new_report_layout("label","customer","genie")  'not supported
```

Controlling the Tables and Sets on which Users can Create Reports

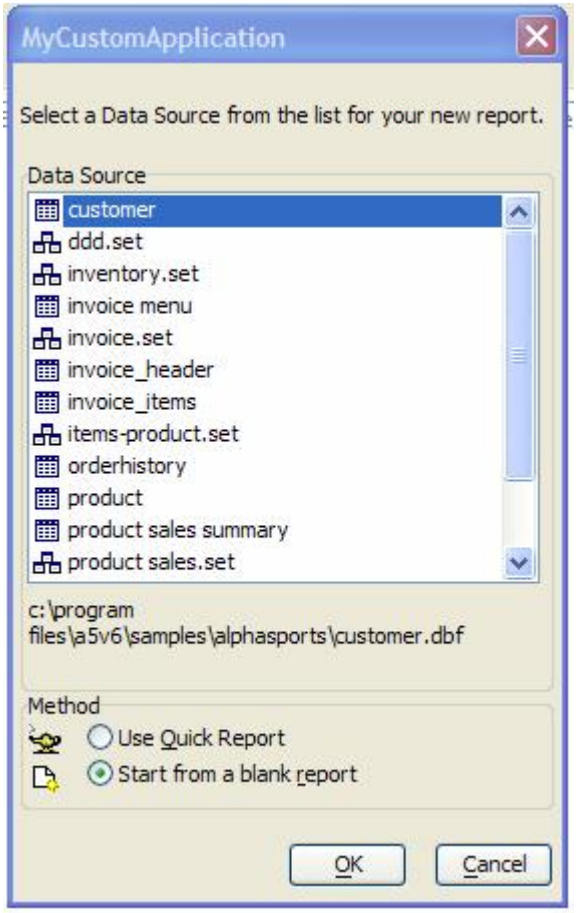
As part of your application design, you may only want users of your application to be able to create new reports on selected tables and sets. For example, you may have certain tables in your application that contain configuration information, and you may not want users to be able to report on these tables. You may also want to provide friendly names for the tables and sets on which users are allowed to report.

You can create your own Xdialog which allows users to select the table or set on which to base their new report, or you can use the **a5_new_report_dialog()** function, which allows you to display a custom dialog from which the user can create a new report, label, or letter.

The a5_new_report_dialog() allows you to control the list of tables and sets in two ways. You can either provide a list of tables and sets to show in the dialog, or you can provide a list of tables and sets to exclude from the dialog. If you have a lot of tables and sets in your application, it may be easier to supply a list of the "hidden" tables and sets. The user will then be able to create a report on any table or set that is not in this list.

Here is an example of a dialog created by this function:

```
a5_new_report_dialog("Report","Select a Data Source from the list for your new
report.","MyCustomApplication")
```



The syntax of the command is:

<*ObjectP*> = a5_new_report_dialog(C *type* [,C *prompt* [,C *title* [,*C table_set_list* [,C *alias_names* [,L *exclude_tables*]]]]])"

ObjectP	The function returns a pointer to the report editor window. This is useful if you want to change a property in the report. E.g. `ObjectP.window_title = "fred"`
type	Type can be "report", "label" or "letter"
prompt	This is the text of a user defined prompt that appears just above the list of data sources.
title	This is the title of the dialog box.
table_set_list	This parameter is optional. If blank, then Alpha Five shows all of the tables/sets in the database in the "Data Source" list box. *Table_set_list* is a CR-LF delimited list of table and/or set names. If specified, the meaning of this parameter is dependent on the *excude_tables* flag. If *exclude_tables* is .f. then the dialog shows only the tables and sets specified by this parameter. If *exclude_tables* is .t. then the dialog shows all of the tables and sets in the database, minus those specified by this parameter.

alias_names	This parameter is optional. It is a CR-LF delimited list of aliases to present to the user instead of actual table and set names. See below for the format of the *alias_names* parameter. NOTE: Once the report editor is launched, the title bar on the report will show the alias name. However, once the report is saved, the title will revert to the Alpha Five generated title (which will show the real name). You must add code to the Fiile/Save and File/Save As actions that you define in your custom report editor menus to reset the title.
exclude_tables	A logical flag that controls the meaning of the *table_set_list* parameter.

How to Specify Alias Names

Alias names are specified in a CR-LF delimited string using the format:

```
Table/setname|Alias
```

For example:

```
Customers|Master Customer List
Vendor|Master Vendor List
Invoice_header|Invoice Headings
Invoice_items|Invoice Line Items
Invoice.set|Invoices
```

This alias list can be stored in a file on disk, or you can define a UDF that returns the alias list.

Example

Assume that the aliases are stored in a file called "MyApplication.TableAliases" in the Alpha Five program folder:

```
Dim alias_file as c
Dim alias_list as c
Alias_file = a5.get_exe_path() + chr(92) + "MyApplication.TableAliases"
If file.exists(alias_file) then
    Alias_list = get_from_file(alias_file)
Else
    Alias_list = ""
End if

Dim table_set_list as c
Table_set_list = ""

Dim Prompt as c
Prompt = "Select the data source for your new report…"
Dim dlg_title as c
Dlg_title = "New MyApplication Report"

a5_new_report_dialog("report",prompt,dlg_title,table_set_list,alias_list,.f.)
```

Protecting System Reports

The "System" Reports are the Reports, Letters and Labels that you include as part of your application. You may want to prevent users of your application from overwriting these reports. For example, you may allow users to create new reports, but when saving a new report, you don't want them to be able to overwrite a system report. You may also want to allow users to edit a "system" report, but require them to save the edited report with a new name (so that the "system" report remains unchanged).

In order to protect the system reports, must create a list that defines which reports are "system" reports (see below), and you must customize the menus and toolbars for the Report editors to replace the built-in actions for the File, Save and File, Save As commands with your own, user-defined actions.

You must replace the built-in action for File, Save with a call the to the **a5_layout_save()** function, and you must replace the build-in action for File, Save As with a call to the **a5_layout_saveas()** function.

Both of these functions take, as one of their arguments, a list of the "system" reports. When a user saves a report, the dialog box that is displayed for the user to type in the report name will exclude all "system" reports and will prevent the user from manually entering the name of any "system" reports. Both of these functions are discussed below.

Note: The customized system menus are stored in the **a_menus_system.dbf** table in the Alpha Five program folder. The customized toolbars are stored in the **a_toolbars_customized.dbf** table in the Alpha Five program folder. You will therefore have to ship these tables (along with the associated .fpt files) with your application. Note that the Alpha Five patches which are released from time to time may overwrite these tables, so you will have to be sure to restore these tables after applying a patch, or you will have to write you own install program for the patches so that these tables are not overwritten.

Specifying the "System" Reports

Note: The list of reports, labels and letters that constitute the "system" reports can be stored in a text file on disk (this file can be encrypted), or you could create a function that returns this list. This function could be compiled into an .AEX file so that the user cannot change the list. You are free to use either approach.

Regardless of whether you use a function, or a text file, the format of the "system" reports list is:

```
Layout_name|type
```

Where "layout_name" is the qualified name of the layout (i.e. the format is *layout_name*@*dictionary_name*). Note that the *dictionary_name* is a relative name. I.e. If the dictionary is in the same folder as the current Database (.adb file), then you do not need to qualify the *dictionary_name* with a drive and path.

For example, assume your application has the following reports:

"Outstanding Invoices" and "Invoices" for the Invoice.set, and "Customer List" for the "Customer" table. The application might also have these labels: "Ship Labels" for the "Customer" table, and "Product Labels" for the "Product" table.

The list should have the following CR-LF delimited entries:

```
Outstanding Invoices@invoice.set|report
Invoices@invoice.set|report
Customer List@customer.ddd|report
Ship Labels@customer.ddd|label
Product Labels@product.ddd|label
```

The a5_Layout_SaveAs() and a5_Layout_Save() Functions

The **a5_layout_saveAs()** function displays a customized "Save As" dialog box. The **a5_layout_save()** function displays a customized "Save" dialog box if the report has not been previously saved, otherwise, it just saves the report using its existing name.

The syntax of the **a5_Layout_SaveAs()** function is:

```
V a5_layout_saveas(C dlg_title ,C type [,C exclude_list  [,C restricted_name_message ]])
```

The syntax of the **a5_Layout_Save ()** function is:

```
V a5_layout_save(C dlg_title ,C type [,C exclude_list  [,C restricted_name_message ]])
```

dlg_title	The title that appears on the dialog box
type	The type of layout: report, label, letter
exclude_list	The CR-LF delimited list of "System" reports that should be removed from the list of layouts that are displayed in the dialog box. Users cannot save a layout with one of these names.
restricted_name_message	The message that appears when the user tries to save a layout

	that is in the excluded list.

Customizing the Xbasic for the "Save" and "Save As" Menu Commands

Assume that you have created a file in the Alpha Five program folder called "SystemReports.Txt". The file contains this information:

```
Outstanding Invoices@invoice.set|report
Invoices@invoice.set|report
Customer List@customer.ddd|report
Ship Labels@customer.ddd|label
Product Labels@product.ddd|label
```

To customize the "Save" menu and toolbar button for a report, you would define the following Xbasic code to be executed when the user selected the "Save" command, or pressed the "Save" button:

```
Dim list as c
Dim fn as c
Fn  = a5.get_exe_path() + chr(92) + "SystemReports.Txt"
If file.exists(fn) = .t. then
    List = get_from_file(fn)
    A5_layout_save("Save Report As…","report",list,\
    "Please choose another name.")
End if
```

Note: In the case of a Label or Letter, the Xbasic would be the same, except that the "report" argument would be replaced with either "label", or "letter".

To customize the "Save as" menu (the default toolbar in the report editors does not have a "Save as" button), the Xbasic would be identical to the above example, except that the **a5_layout_save()** function would be replaced with **a5_layout_saveas()**

If, instead of defining the list of "system" reports in a text file, you defined a function to return this list, your code would look like this:

Here is how you would define your function:

```
Function SystemReportsList as c ()
SystemReportsList = <<%txt%
Outstanding Invoices@invoice.set|report
Invoices@invoice.set|report
Customer List@customer.ddd|report
Ship Labels@customer.ddd|label
Product Labels@product.ddd|label
%txt%
End Function
```

Here is how you would define the Xbasic for the "Save" menu and toolbar button:

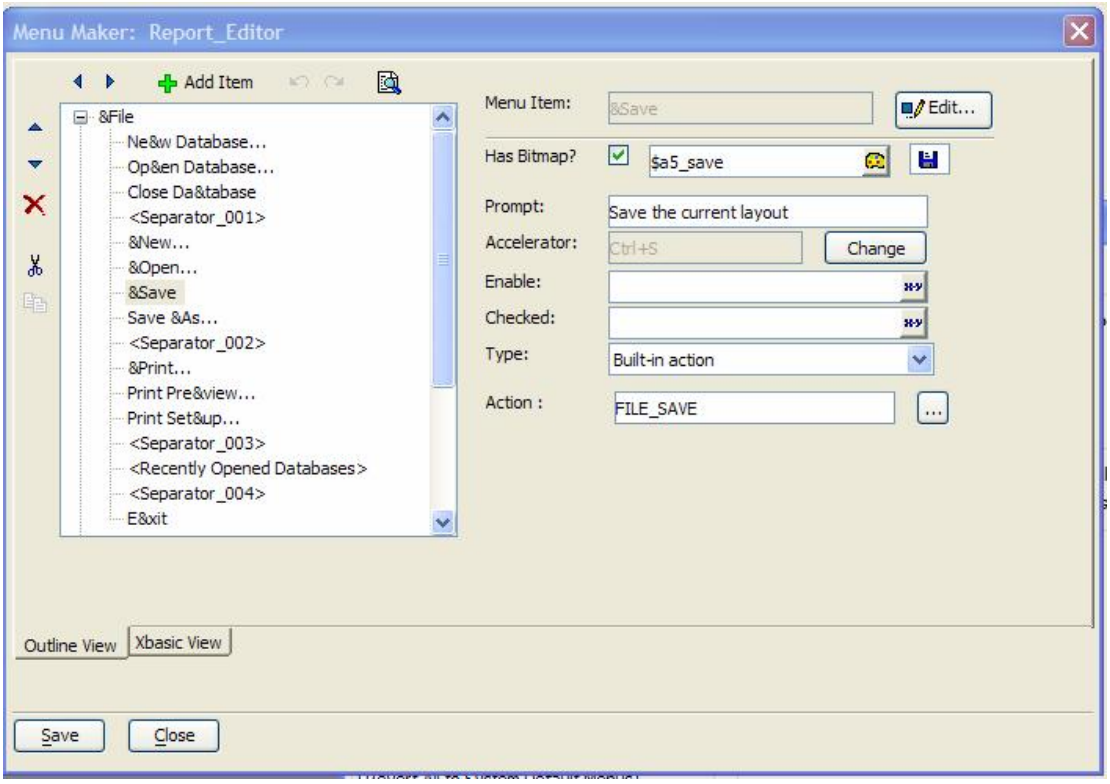
```
Dim list as c
List = SystemReportsList()
A5_layout_save("Save Report As…","report",list,\
    "Please choose another name.")
```

Customizing the System Menus in the Report, Letter and Label Editors

To customize the menu for the Report, Label and Letter editors so that your customized Xbasic (described above) is called when the user selects the File, Save, or File, Save as command, follow these steps:

1. Right click on the white background on the Code tab of the Control Panel.
2. Select "Customize System Menus".
3. Choose Report Editor, Label Editor, or Letter Editor.
4. Navigate in the menu tree to the File, Save command.

The default action for the File, Save command is shown below:

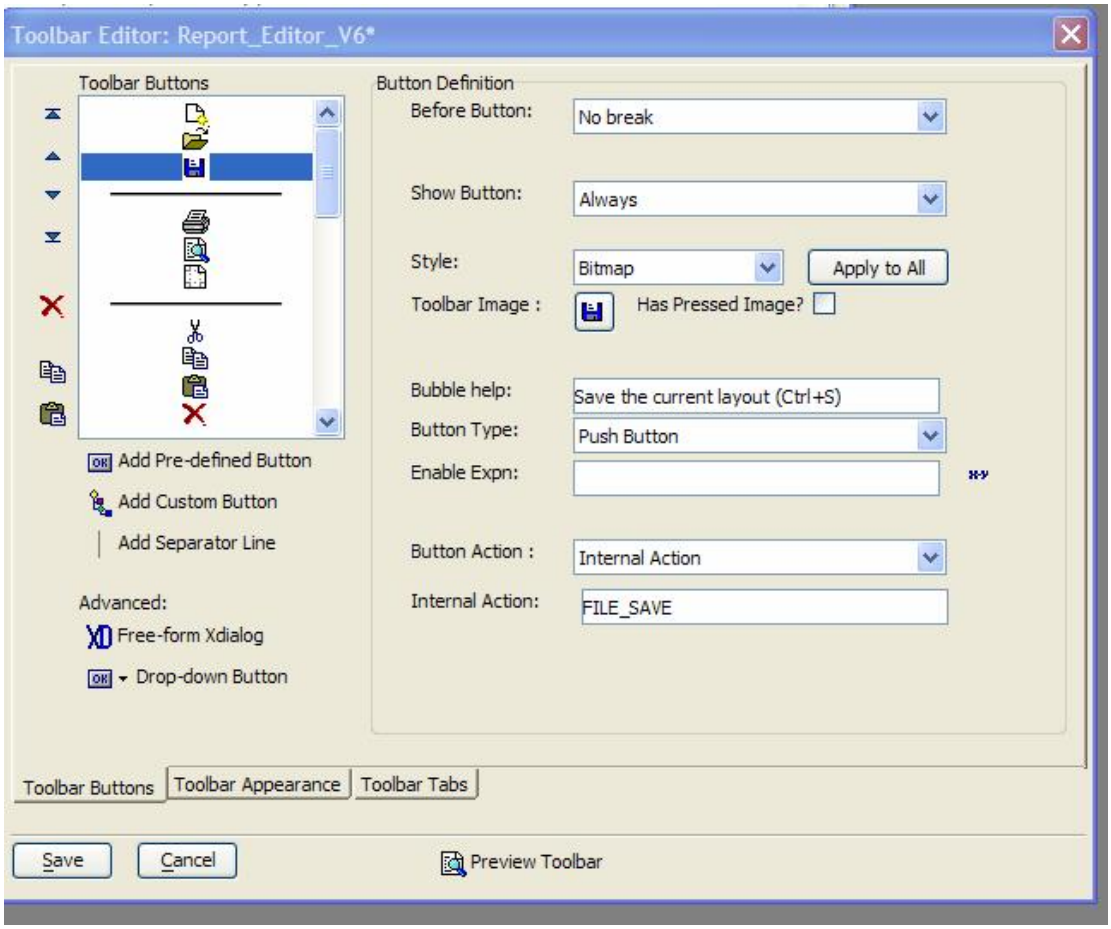


5. Change the "Type" from "Built-in action" to "Xbasic".
6. Enter the Xbasic for the action (see "Example of How to Customize the Xbasic for the "Save" and "Save As" Menu Commands" above).

Customizing the System Toolbars in the Report, Letter and Label Editors

To customize the toolbar for the Report, Label and Letter editors so that your customized Xbasic (described above) is called when the user clicks the Save button, follow these steps:

1. Right click on the white background on the Code tab of the Control Panel.
2. Select "Customize System Toolbars".
3. Choose Report Editor, Label Editor, or Letter Editor.
4. Select the Save button.



5. Change the Button Action from "Internal Action" to "Xbasic".
6. Enter the Xbasic for the action (see "Example of How to Customize the Xbasic for the "Save" and "Save As" Menu Commands" above).

IMPORTANT NOTE: The customized toolbars are stored in the "a5_system_toolbars.al*" files in your Alpha Five program folder. You will need to install these files with your runtime application.

Starting Alpha Five Runtime – Command Line Options

To start the Alpha Five Runtime, you can use the following command:

```
alpha5 <.adb file>  [-TITLE=<title>] [-ICON-<icon>] [-SPLASH=<splash>]
[-NOSPLASH] [-INCLUDE=<file>]
```

NOTE An easy way to generate the command line syntax for starting Alpha Five is to use the Shortcut Genie (See the Tools, Create Shortcut… command) to create a shortcut. The right click on the shortcut that Alpha Five created and examine the properties of the shortcut. You will see that Alpha Five has generated the correct syntax for starting your application.

Command Line Option	Description	Example
-TITLE=<title>	Change the displayed title for the main Alpha Five window.	-TITLE="Customers"
-SPLASH=<bitmap>	Change the splash screen bitmap that displays when the Alpha Five Runtime loads.	-SPLASH=clock.bmp
-ICON=<icon>	Change the icon that displays on the Alpha Five Runtime title bar.	-ICON=logo.ico
-NOSPLASH	Do not display a splash screen when the Alpha Five Runtime loads.	-NOSPLASH
-COMMAND=<xbasic>	Runs the specified Xbasic commands after the Alpha Five Runtime starts.	
-INCLUDE=<file>	When you have more command line options than can fit on one line, you can create an ascii file with the command line options in the file. Put one option per line. Use this option to specify the name of the ascii file.	-INCLUDE=start.txt
-HELP	Get help on the command line parameters	

IMPORTANT: Do not leave any spaces between the option name and the '=' sign. E.g. -TITLE="Invoice", NOT -TITLE = "Invoice"

The following example shows how the Alpha Five Runtime Title bar can be changed:

	Runtime Title bar with a custom icon and title
	Runtime Title bar with standard icon and title

The following example shows how the Windows Task bar can be customized to show your own icon for the Alpha Five Runtime:

	Task bar showing Runtime with a custom icon and title
	Task bar showing Runtime with standard icon and title

Examples

- The following command starts the Alpha Five Runtime, loads a database (an .adb file) called "Orders" that is stored in the "c:\Orders" folder, changes the title of the Alpha Five Runtime title bar to "Orders", uses a bitmap called "NewAgeSolutions.bmp" for the splash screen, and uses an icon called "orders.ico" for the icon on the Title bar and the Task bar:

```
C:\a5runtime\alpha5.exe "c:\orders\orders.adb" -title="Orders"
       -splash="NewAgeSolutions.bmp" -icon="orders.ico"
```

- The following command starts the Alpha Five Runtime, loads a database called orders, and opens a file called "start.txt" which contains a list of command line options:

```
C:\a5runtime\alpha5.exe "c:\orders\orders.adb" -include=start.txt
```

Start.txt is an ascii file that might contain the following text:

```
-title="Orders"
-splash="NewAgeSolutions.bmp"
-icon="orders.ico"
```

- The following command starts the Alpha Five Runtime, loads a database called orders, and defines, then sets a global variables:

```
C:\a5runtime\alpha5.exe "c:\orders\orders.adb"
 -COMMAND="DIM GLOBAL VPath AS C; VPath = \"myfile.adb\""
```

NOTE: that the quotes in the command `Vpath = "myfile.adb"` need to be prefixed with a slash (\ ) to indicate that they are not the end of the command being passed to the Alpha Five Runtime.

Creating a Bootstrap Application

When you install a multi user application that uses Network Optimization, you need to install a master copy of the application on a shared server, and you need to have a "shadow" copy of the application on each user's machine. This "shadow" application points to the master copy of the application.

Your install program cannot install the "shadow" application on the user's machine because (at the time you create the install program) you don't have any way of knowing what the path is from the user's machine to the server (because you don't know where the user will install the master copy of the application). To get around this problem, you install a special "bootstrap" application on the user's machine. The first time the user starts your application, the "bootstrap" application is loaded and it asks the user for the path to the master application (which is on the shared server). The "bootstrap" application then automatically creates the "shadow" application on the user's machine.

Here is how to create a "Bootstrap" application.

1. Create a dummy application with no tables, forms etc. The name of the dummy application must be the same as the "Master" application. E.g. "AutoPartsStore.adb"
2. Add an Autoexec script to the application (See below).

Here is the code for the Autoexec Script in the "bootstrap" application:

```
dim master_db_name as C
dim shadow_db_name as C
master_db_name = ""
shadow_db_name = ""

ui_modeless_dlg_box("Install MyAppName",<<%dlg%
{can_exit=exit}
{lf};
Specify the filename of the master copy of the application:;
[%P=ui_get_file("Select file","(*.adb)",master_db_name)%.100master_db_name];
{lf};
<Click Here to Start Install!oK?.not.(master_db_name="")> <Cancel!cancel>;
```

```
%dlg%,<<%code%

if a_dlg_button = "cancel" then
    ui_modeless_dlg_close("Install MyAppName")
    a_dlg_button = ""
end if

if a_dlg_button = "oK" then
    if file.exists(master_db_name) = .f. then
        ui_msg_box("Error","Master Database not found.")
    else
        shadow_db_name = a5.Get_Name()
        a5.Load(master_db_name)
             If version() = 5 then
                a5_NOpt_CopyAddFiles(shadow_db_name,master_db_name)
                create_shadow(shadow_db_name,master_db_name,.t.)
             Else
                create_shadow2(shadow_db_name,master_db_name,.t.)
             End If
        ui_modeless_dlg_close("Install MyAppName")
    end if
end if
%code%)
```

Your installation program will install the "Master" copy of the application in a shared folder on the server and it will install the "Bootstrap" application on each runtime user's workstation.

When the user starts the "Bootstrap" application (which in this example is called "AutoParts.adb") it will immediately display a dialog asking for the path to where the Master copy of "Autoparts.adb" is installed. Once the user fills this in and clicks the OK button, the script will automatically open the master copy of the database. It will then do the initial Network Optimize, and then open the shadow copy of the application on the user's machine.

The next time that the user opens "AutoParts.adb" on his machine he automatically is working with a Network Optimized (i.e. “shadow”) version of the master database.