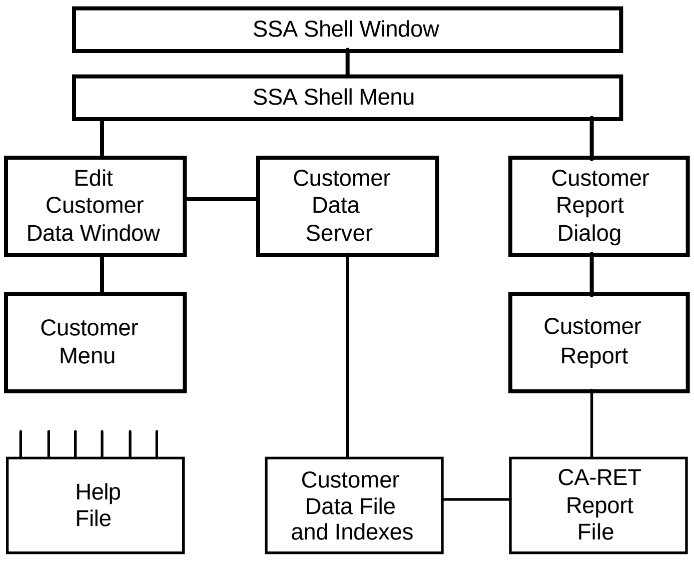
# Appendix A
# Creating a Path-Independent Application

## Overview

This appendix describes what you must do to make sure that an application can run successfully when it is installed on any drive or directory.   Creating such path-independent applications is an important design objective.

The following diagram of the primary South Seas Adventures application building blocks indicates that there are three types of external files—help files, data/index files, and CA-RET report definition files:



**Note:** Rectangles with thick borders are the primary building blocks, while those with thin borders are external files. Design linkages are shown as thick lines, while external linkages are shown with thin lines.

An application must be able to locate these files at runtime, so there are several key steps that must be taken to remove possible path dependency while you are creating the building blocks of your application. CA-RET report files themselves contain information that directs the report runtime engine to the location of the related data files, so additional steps to remove path information from each report's query statement are required.

## Establishing Drive and Directory Independence

The "Exploring the CA-Visual Objects Integrated Development Environment" chapter describes how the SSATUTOR directory structure was chosen to facilitate the development process. In addition to carefully planning the directory structure for the developer's development environment, it is important to think ahead to the drive and directory possibilities when the application is installed by an end user.

There are two possible approaches when planning your directory structure, depending on where you want your data to reside.

### Fixed Paths

If you are sure that the data files will always reside on a given drive and directory, then you may want to use fixed paths for the data file and index files specified in the DB Server Editor. This may be a good choice if the data files are maintained on a single LAN drive and directory. During the development process, you can keep your test data files in these designated locations.

## User-Defined Paths

If the location of the data files depends upon the user's choice of installed directory, you should plan to keep your application help files, data files, and report files in the application directory (that is, where the .EXE file resides). In this case, you must not use any path information when you define each DATA server and report. During the development process, these files should be kept in the application's .EXE file directory (for example, D:\CAVO\SAMPLES\SSATUTOR) specified in the Application Properties dialog box.

This appendix addresses the second approach—since this is the more likely scenario—which places some restrictions on what you can do. The fact that the application starts up from the directory that holds the .EXE file allows you to specify path-independent linkages for your external files.

There are several runtime considerations relating to finding these external files:

- When running an executable file, the current directory is the directory that holds the .EXE file. External files that have no explicit path specified in the related application building blocks will be successfully located if they are placed in this directory.
- When you use Save or Save As to store a report definition, the path is stored in the report's class entity. This path is required during the development process. However, you can create a special access entity to change this path to NULL_STRING at runtime, allowing report files stored in the application directory to be successfully opened.
- Each report file executes a query at runtime to locate the data files and index files. One or more data servers must be specified when you define a new report and any path information stored in the data server at that time becomes part of the query statement.

Achieving drive and directory independence for your reports requires that you not use any path information in defining data servers. Otherwise, the runtime query executed for each report will fail if the data files are not in the fully specified location. The South Seas Adventures application was designed to achieve drive and directory independence. Therefore, there is no path information associated with any data server.

## Help Files

You can specify a help file name for a shell window or a data window by using the Help File Name property that is specified in the Window Editor. The Window Editor generates source code that creates the linkage. For example, if the help file is SSA.HLP, the code is

```
SELF:HelpDisplay := HelpDisplay{"ssa.hlp"}
```

where SELF is a shell window or data window. Here, a help display object is created using the SSA.HLP file, and then using the HelpDisplay assign of the Window class. This help linkage is path-independent.

If we had included path information in the Help File Name property, then it would be included in the source code. This would require that the help file always be in the designated location, or that the path is modified at runtime. Generally, you can simply remove all path information from the Help File Name property and locate the help file in the directory in which the application is installed.

## DB Server Data Files

If you create a data server by importing a .DBF file, path information will automatically be placed in the File Name edit control.  If you do not remove the path information from the server before you save it, the path will be stored in the CLASS definition, as follows:

```
CLASS Customer INHERIT DBServer

INSTANCE Customer_DBF_Path := ;
      "c:\cavo\samples\ssatutor\" AS STRING
```

Therefore, you must remove any path information for the data file or any index file, leaving only the name of the file.  If you do this, NULL_STRING is stored in the instance variable, as shown below:

```
INSTANCE Customer_DBF_Path := "" AS STRING
```

If, for some reason, you do want path information in the CLASS definition, you can still change it at runtime by creating a special access method for the instance variable.  This access entity removes path dependence at runtime:

```
ACCESS Customer_DBF_Path CLASS Customer
RETURN ""
```

This next code allows you to specify the DBFS subdirectory of the current directory,

```
ACCESS Customer_DBF_Path CLASS Customer
RETURN ".\dbfs\"
```

However, neither of these access methods address the problem of path independence for reports.  Nor does use of the SetDefault() function, which allows you to specify a path that will be searched when performing direct database actions.

***Important!*** *To achieve path independence in reports, you must* ***not*** *have any path information stored in any data server building blocks.*

## Report Files

### CA-RET File Path

When you create a report you will use either the Save or Save As command in the Report Editor to store the CA-RET file. If you are creating a new report, the default location is your application .EXE file directory; therefore, you must use Save As to locate it elsewhere. The actual path location of the file is stored in the CLASS entity as follows:

```
CLASS CAdvRpt INHERIT ReportQueue

INSTANCE CAdvRpt_File := ;

"c:\cavo\samples\ssatutor\cadvrpt.ret" AS STRING
```

This path is required so that the file can be opened when you want to edit the report. Thus you must use a special access method to achieve path independence in the end-user runtime environment. The South Seas Adventures application includes an access that removes the path dependency for each report. For example:

```
   ACCESS CAdvRpt_File CLASS CAdvRpt

   RETURN "cadvrpt.ret"
```

While this access method achieves path independence at runtime, you must also address the location of the report file during the development process. The CA-RET file path in the CLASS statement must be changed, if you have imported the South Seas Adventures .AEF file into a directory other than C:\CAVO and you wish to edit the report definition.

You can change the CLASS statement by entering the Report Editor and using Save As to save it to the desired location. When you double-click on the report entity for the first time, a dialog box asks you to type in the full path of the .RET file. When you do this and close the dialog box, the Report Editor opens. The path in the CLASS definition is updated to the new path.

Query Path

When you create a new report, the Report Editor dialog box requires that you choose one or more servers to define what fields are on the report. If any of these servers contain path information for a data or index file, it will be included in the query statement.

In order to achieve path independence, you must manually remove the paths by using the Database Edit Query command. After doing this, save the CA-RET file to the proper location. If you do not manually remove path information from the query, you will get a CA-RET runtime error if the data files cannot be found by using the designated path.

***Important!*** *The best approach is to remove the path information from the data servers, so that it is not passed to the Report Editor.*

## Icon, Cursor, and Bitmap Files

Icons, cursors, and bitmaps can be treated differently than help files, data files, and report files, since they are directly incorporated in any .AEF export file or a generated .EXE file. When the South Seas Adventures application was initially created, the RESOURCE definitions of the icons, cursors, and bitmaps contained the path for the .ICO file on the developer's disk drive. For example:

```
RESOURCE ITEM_ICON Icon ;
      c:\cavo\samples\ssatutor\files
```

If you have installed CA-Visual Objects to C:\CAVO, the file is stored in this directory during the .AEF file import process. This path is not useful on your machine, however, if you installed to some other drive and directory. This is not a problem because CA-Visual Objects automatically modifies the path during the import process. The file is placed into the directory from which the import is taking place and the RESOURCE statements will be automatically changed. For example, if you installed to drive D:, the icon, cursor, and bitmap files will be placed in the D:\CAVO\SAMPLES\SSATUTOR\FILES subdirectory.

Because this is automatically handled during the import process, no further steps are necessary unless you wish to edit an icon or cursor itself. If you try to edit such an icon entity and it is not in the same location as the original .AEF file, a dialog box is displayed, indicating that the file cannot be found. You must close this dialog box in order to invoke the Icon Editor.

When in the Icon Editor, use the File Open command to open the file from its new location. Then, use the Save command, type in the same entity name, and choose OK to save the icon file and the icon entity. Select Yes when asked about overwriting the existing file. If you are working with a cursor entity, you must first change to the Cursor Mode (from the Options menu) when you invoke the Icon Editor.

## Summary

As you have just seen, there are essentially no runtime issues associated with the paths for icons, cursors, and bitmaps, since they are bundled into the .EXE file. The only issue is one of moving an .AEF file between two developer's machines with different directory structures.

In addition, you have learned what you can do to make your help, data, index, and report files path-independent.