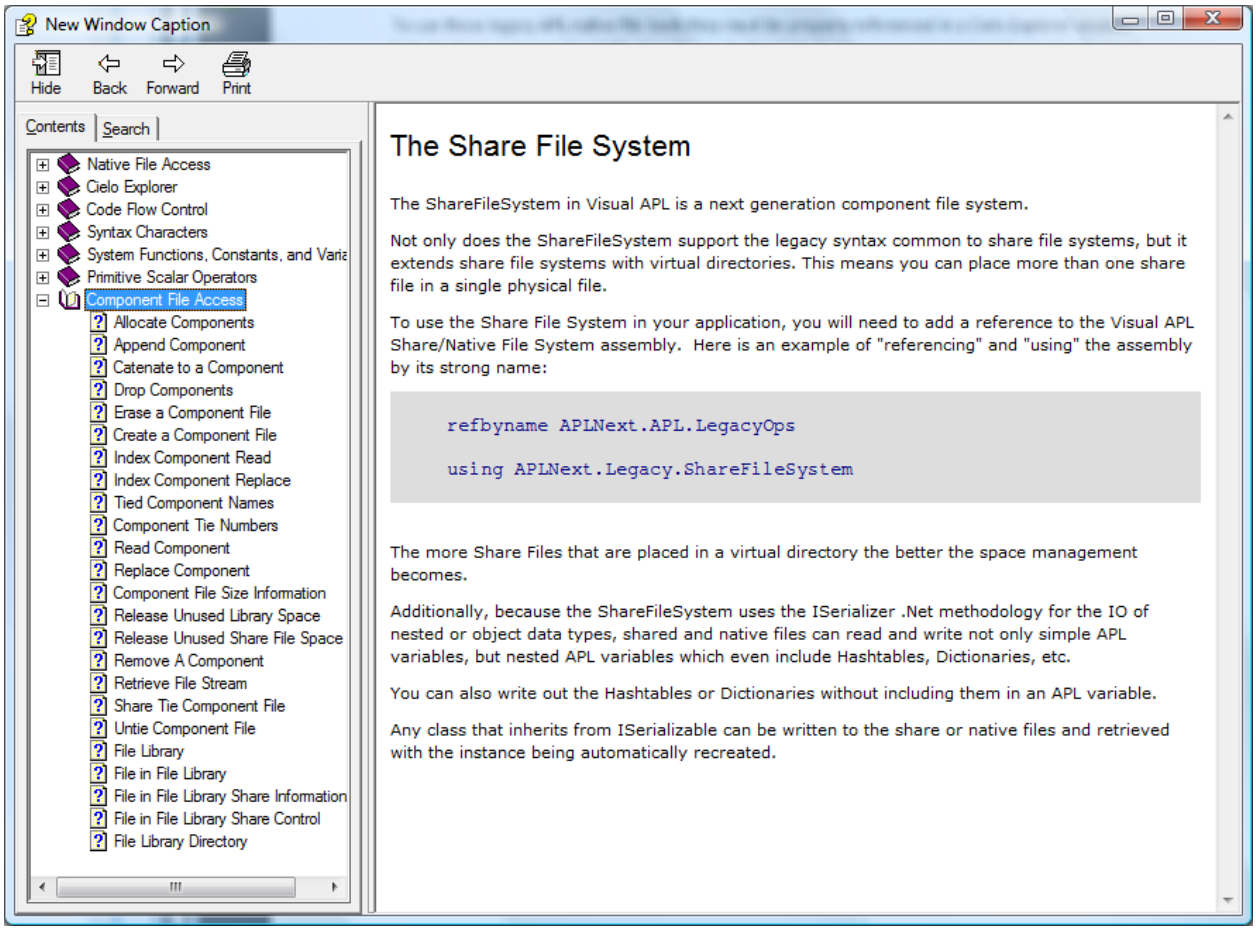
VisualAPL Component File System

The VisualAPL component file system represents a significant improvement over legacy APL implementations of 'component' files. New features of the VisualAPL component file system include:

- Allocate components
- Catenate t o a component
- Improved remove/dropping of components
- Erase a component
- Index read/replace component
- Share file associated with multiple component files within a virtual directory
- Retrieve file stream
- Read/ Write ISerializable objects

The documentation file "…\AplNext\VisualAPLProfessional\Documentation**Language_Interaction.chm**" installed when VisualAPL is installed to the programmer's machine incorporates a "Component File Access" section:



A careful study of this documentation will explain the features of the VisualAPL component file system.

Of particular importance is the fact that since a VisualAPL .Net assembly (class library) is compiled by the Microsoft JIT compiler and a .Net manifest is automatically created, the VisualAPL component file system tools are not included in the manifest by default. Instead, the VisualAPL programmer must indicate that the VisualAPL component file system tools are necessary for the project being programmed by adding the appropriate reference and using statements to the source code:

r**efbyname APLNext.APL.LegacyOps**

u**sing APLNext.Legacy.ShareFileSystem;**

Note that since the VisualAPL installer puts the "AplNext.Apl.LegacyOps.dll" in the GAC (.Net global assembly cache), the "APLNext.APL.LegacyOps" namespace can be referenced by name in the above statements.

The VisualAPL component file is deemed part of the legacy APL features implemented in VisualAPL. To understand this philosophy, some facts about any current or prior implementation of an APL component file system need to be stated:

- APL component files are inherently non-hierarchial, non-indexed files in which each component's structure and contents are independent of all other components.
- Essentially APL component files support 'blob' data, i.e. each component can accept any type of data and no inherent or consistent record structure is required when utilizing APL component files in an application system.
- APL component files have a proprietary underlying format, so they cannot be conveniently accessed by mainstream file access tools.
- Any index or record structure imposed upon an APL component file is purely a construct of the application software programmer.

Generally, in .Net applications, application data is stored in a format which permits wider use and future re-purposing of that data by multiple application systems and users, without the limitation of a format which is proprietary to a specific programming language. This generally means that a commercial database, such as Microsoft SQL Server, IBM DB2 or Oracle, or a mainstream data format, such as XML, Microsoft Excel, etc., are selected for data storage.

In cases where the data to be stored is entirely heterogeneous and limited in use to the current application system, the APL component file may still be appropriate.

Since every APL component file system uses a proprietary format, the APL+Win colossal component file system and its predecessor APL+Win component file system are not compatible with the VisualAPL component file system. To exchange data in APL component files between the APL+Win and VisualAPL systems, the data should be converted by the source system to a format which is compatible with the target system such as Microsoft Excel, Microsoft SQL Server, XML serialization or native (Unicode) text files, etc.