OVERVIEW

Applications which make use of EAI databases will find that databases are unavailable at times[1]. In other cases, the application's needs can be better met without using a database connection at all. Cache files are a common example of a secondary data source. They provide a way to store and retrieve program information that sidesteps the complexities inherent in communicating with a database. Another advantage is that cache files can be used to speed up application processes for the end-user, since access to the data can happen anytime instead of only when the user makes a request.

The cache to support EAI applications consists of a single 'root' folder that contains subfolders, one for each application that will make use of the cache. Each subfolder contains the cached data files that are needed by the application. In the root folder are several additional files that are used to keep things organized (more information below).

USES OF THE CACHE

ECM

Use of the cache is divided into two categories: 1) Use by an application requesting data, 2) Use by a utility to manage the cache directory structure and its data files. To help facilitate both cases, an interface is provided by a central managing class called the ExternalCacheManager (**ECM**)[2]. The ECM is a mainly passive unit which prefers to provide information for requests coming from other applications; in other words, it is intended to be more responsive than active… the active part comes from outside.

SOM

The Serialization Object Model (SOM) files are the other half of the ECM. SOM files are reference-only DLL files containing information necessary for the ECM to do its job. Each application has its own associated SOM (SOM.Aire, SOM.Silo, etc.) and must disclose this module to the ECM in order to work with cache files. If you read the section on Registering New Data Files, you will see that a data file cannot be accessed if it has not been included in an SOM file[3].

APPLICATION USE CASE

When an application needs information from the cache, it requests the file (by name) from the ECM. To access the data in the file, the application also uses one of the deserializing services provided by the ECM. To successfully complete the data request, the application only needs to know the filename and

---

[1] Although this is generally true when many users are potentially making database requests, it is specifically true for the SPECTRUM database used by EAI, since several of SPECTRUM's modules have a pessimistic concurrency policy which locks out all users while those modules are being accessed. This policy makes even simple read-only queries impossible, so data redundancy is a de facto requirement for any application intended to provide a reasonable user experience.

[2] The ECM is a DLL file assigned to the 'Jumble' domain and its full project name is: Jumble.ExternalCacheManager.

[3] There is currently no way for the grabber to view a list of files contained in an SOM file. This should be added to future revisions under the "info" command.

the type of data contained in the file. Note that applications can only retrieve and read cache data from the ECM.

UTILITY USE CASE

Since applications cannot write or modify cache data, those tasks must fall to the purview of the utility application. There is also an additional use case required when, because of design considerations, the need for a new datafile is discovered and the file must be integrated into the cache structure.

The '**grabber'** is a command-line utility for managing the cache directory structure, the files it contains, and the additional use case of registering new files as they are added to the overall design. Although the grabber has several functions, this document will cover only the most important.

TASKS

CACHE-ING THE DATA

The most important of the primary tasks is to back up the data in the data files. Use the "cache /r" command to perform this task. Every file that is registered with the grabber will be included in the backup routine. Read more about registering data files below.

Every cache file is associated with a database data request. The job of the grabber is to execute this data request, process the results, then write the data to the cache file. Backup should be performed at night or whenever company databases are expected to be unused. In the off-chance that a data connection is not available when the backup is being run, the grabber will simply display a command line error message, then continue to the next file on its list and attempt the connection again.

The backup process will overwrite the current file contents, which is how the information is kept fairly up to date; if backups are run nightly, then users know that the information is never older than 15-17 hours (assuming a normal work day of 9-5)[4].

ADDING A NEW CACHE DATA FILE

The second of the two primary tasks that will need to be accomplished is the adding of cache data files as designs and application needs change. What does "adding a cache data file" really mean? It means that the new data file:

a. …is added to the cache folder for the application that will need to use the file.
b. …is registered with the ECM's internal file map so that it can be located by any application.
c. …has an associated class file, implementing the "ICacheableObject" interface, included in the specific SerialiazationObjectModel (**SOM**) for the application that will use it.
d. …is included in the data backup process controlled by the 'grabber' utility, which keeps the file contents up to date.

[4] Remember that, for time-sensitive information, a direct database request by an application is the preferred method of getting data. It's when this method fails that the application will be redirected to the cache as a data source. For non time-sensitive information, the cache is very likely to be the primary data source for the application.

For a file to be a usable member of the data cache, these three conditions must be met. Conditions “a”, “b”, and “d” are accomplished at once by using the grabber’s “register” command. Condition “c” requires that the code class be added to the associated SOM DLL file for the application.

EXAMPLE

Here is a specific example, in which we want to add a new cache file (named “*overstock.dat*”) for use by the Aire application:

1. UPGRADE THE SOM DLL FILE
    a. After opening the SOM.Aire source code, we create a new public class and name it with the “C” prefix, which stands for “cache”. For sake of the example, we will name it “COverstock”.
    b. For the new class, implement the “ICacheableObject” interface, then assign the required information to the class.
    c. Recompile the SOM.Aire project. Use the recompiled DLL file “SOM.Aire.dll” to replace the existing DLL the application folders for ‘grabber’[5] and Aire. In the future, we will probably add the DLL to the GAC during installation; but for now, upgrading the file means overwriting the existing file wherever it is found.
2. REGISTER THE FILE WITH THE SYSTEM
    a. Now that the DLL has been upgraded, the file can be registered to the system. Open the grabber and run the “register” command. The command will perform the following tasks (compare these against conditions “a”, “b”, and “d” to verify for yourself that they will be met):
        i. Creates the file “overstock.dat” in the folder “(Root Cache Directory) - - > Aire”
        ii. Adds an entry for the file in the “filemap.xml” file contained in the root cache directory.
        iii. Includes the file in the backup process; run “grabber cache /r”, then observe that the file, which has a 0 kb size if freshly created, now contains data.

Though this process will hopefully be streamlined in the future, for now it should be that only a qualified person with access to development tools should be performing this task. If a future application feature somewhere requires the use of custom cache data files, AND those data files should not be located on the local user drive for some reason, then this issue will need to be revisited in greater detail.

RENAMING FILES

Ideally, no files should be renamed. However, to accommodate those situations where renaming is necessary, the grabber provides a rename function for data files (but not for cache folders).

When a file is renamed, no additional actions are necessary. All applications search for data files using the default original name of the file and it is the job of the ECM to resolve the filename to locate the data. How does the ECM accomplish this? By searching through a list of rename events when mismatches occur. Since all rename events are recorded, there is always a linear trail from the original name of the file to its current name. However, let it be said again that renaming of files is a highly

[5] ..or whichever location is chosen for SOM files; all that matters is that the DLL is visible to the ‘grabber’ utility.

discouraged practice – just know that if it really needs to done, the system is designed to handle it without breaking.

RELOCATING THE CACHE ROOT DIRECTORY

While renaming files will not break the system, renaming of folders in the cache structure WILL break the system. The only folder that can be renamed is the cache root directory (the one containing all the application folders and everything else), and it can only be renamed indirectly, by changing the entire root directory path.

This is accomplished by using the grabber's "path" command, in which the new directory path must be provided as an argument. If the provided directory does not currently exist, the grabber will create it, thus *indirectly* allowing the user to rename the root directory. ***Directly renaming the cache root directory will break the system.***

To relocate the root directory, "path" will copy all files and folders in the current directory to the new location, update the configuration files to reflect the name change, then delete the old root directory and all of its contents.

THE "CACHEROOTDIRECTORY.DAT" FILE

The "*cacherootdirectory.dat*" file is critical for correctly loading the ECM. If the file cannot be found, the ECM will not be accessible by any application. Stated differently, **the grabber must have "*cacherootdirectory.dat*" installed in its application root folder in order to make use of cache data.**

If there are problems, first check that the file is located in the same directory as "grabber.exe". If the file is missing from this directory AND you are unable to verify if the file contents are up to date, then do not attempt to copy-paste from another known instance of the file – you must regenerate the file using the grabber's "restore /d" command. This will create a new file whose contents are known to be current.

RESTORE

The "restore" command is the same as the "init" command used to initialize the cache root directory structure and generate the application files. If "restore" is called on an existing cache root directory configuration, all data will be erased and all data files not listed in the original default settings will be destroyed. If that's what you want to do, then refer to the source code of the "RestoreCommand" object contained in grabber's source files.

When running this command, grabber will ask you to confirm and warn you that the process is not reversible[6].

[6] Later, this feature can be expanded to take a snapshot of the current directory structure and save it to a configuration file. Then, when "restore" is called, we have several choices of which configuration can be recreated. This is a good way to also keep track of all the cache files added after the original set.