t Error Handling *492*

A datafile primitive returns, as its leftmost output, an integer error value. This may be interpreted as follows:

o If the returned error value is zero, the primitive completed successfully.

o If the error is negative, the error is a Macintosh File Manager error. Some common error numbers:

-33
Directory full

-34
Disk full

-35
No such volume

-36
I/O error

-37
Bad name in the file system

-38
File not open

-39
End of file

-43
File not found

-44
Diskette is write protected

-45
File is locked

-46

Volume is locked

-47
File is busy (on file delete attempt)

-48
Duplicate filename (on file rename attempt)

-49
File already open with write permission

-54
Permissions error (on file open attempt)

-61
Write permissions error

-5000
Access denied

-5015
No more locks

-5020
Range not locked

-5021
Range overlap

o If the error value is positive, it is a Prograph datafile error. *493*

The following are the possible datafile error numbers returned by the datafile primitives. These may be used by your code to take further action, as appropriate.

1
Class names are not available for remapping. This only happens in the compiler. The fix to this error is to recompile with the Include Class/Method Names option specified in the Project Info dialog box.

2
File version doesn’t match software version. This error occurs if you are using an old datafile file with a

new version of the software. The fix to this is to run the appropriate conversion utility to convert the file version to the new version.

3
Name in class map can not be found. This means that there is an object (instance of a class) in the data file but its class is not defined in your interpreted or compiled program. The fix to this is to define the class in your program.

4
Primitive can not be performed in the current mode. This refers to the different modes (query, update, or shared) you can have for your datafile. For example, in query mode you can not perform any write or replace operations. The fix to this problem is to open the datafile in the correct mode.

5
Table already exists. This occurs when you try to create a new table, or rename an existing table, using a name already in use for a table. Table names must be unique within a datafile.

6
Table does not exist. This occurs if you try to do an open or delete operation on a table, or try to rename a table, that has not already been defined.

7
Key already exists. Each key must have its own unique name. If you get an error 7, give the key a new name.

8
Key does not exist. In this case, create a new key.

9
Cluster already exists. This error is returned if you try to cluster-undelete or to replace or write over a cluster that already exists.

10
Cluster does not exist. This is returned if you attempt to access a cluster that has been deleted or does not exist (for example, if you attempt to read through a table sequentially and come to the end of the table).

11
Verify database ID, table ID, or key ID.   This is returned if an invalid key is used.

12
Record Locked. This is returned if you attempt to access a locked record.

Most of the datafile primitives refer to path IDs (database ID, table ID, key ID or cluster ID). Path IDs can become invalid. For example, if you store a path ID somewhere, quit your application, and then restart it and try to reference that path ID, that path will no longer be valid.

Your code should provide some mechanism for storing path IDs, for disposing of path IDs when they’re no longer needed, and for properly initializing path IDs wherever they are stored when your program starts up. Path IDs can be stored in persistents or in class attributes (for more persistent storage), or in instance attributes.

Catching Errors *494*

The error code returned by a datafile primitive indicates whether it performed successfully (0) or not (non-0). The actual value of a non-0 error code gives information on which to base further action. Since dealing with error codes is such a common thing, a generalized error handler is a useful thing to write.