Mirroring with 4D 2004.4, PART I

By Kent Wilbur

Technical Note 06-36

PartnerWare

Introduction

Two new commands for creating a mirroring system were introduced in 4D 2004.3. But as with most new things the original implementations of the commands pointed out the need for some additional improvements.

Also, my original Technical note received many requests for a more flexible complete solution. So the code in this technical note is far more robust with more features and error handling built in.

Therefore, the purpose of this technical note is to explain the revised mirroring commands for 4D 2004.4 and also to provide a more complete solution to mirroring. The code in this technical note is NOT compatible with the code from the first technical note. So if you used any of the methods from the first technical note. They should be removed and replaced with the ones from this technical note.

There is also a component available that can be added to your database to provide a mirroring solution. Updates to this technical note and component will be provided when necessary.

Since the code is different between the two technical notes. This technical note will repeat the important information so that you don't need to refer to the original technical note.

What is mirroring?

Mirroring is the concept of having a second (or more) identical database running in parallel to the database in active use. Its purpose is to provide the quickest possible solution in the event that the main database has become inoperative for any reason. Because mirroring keeps the database synchronized if the main database fails, the mirror database will be up to date as of the last synchronization. If the log file that contains everything since the last synchronization on the main server is recoverable, all that needs to happen is to synchronize those last few entries and you can be up and running again using the mirrored server as the primary server.

Since the mirrored server is already a server and is already running the whole process can literally be complete in a few minutes and the database will be fully functional again. No restoring from backups. Mirroring is the fastest way to get a backup server acting as the primary server possible.

Does the mirrored server database have any special requirements?

The rigid requirement of the mirror machine is that it absolutely cannot modify data in the data file on its own. So you will need to be sure that your On Startup doesn't modify any data automatically. And also be very sure that nobody accidentally logs into the database and does something via a client or web connection. This can be trapped in the On Web Authentication and On Server Open connection database methods.

In addition you must have a way for the database in its code to know that it is a mirror and NOT the live operational database. I can think of several ways to do this. One is to rename of the structure file and check the name on startup. A second is the presence of a particular file in the 4D Extensions folder. In my example database, I have an XML preferences file stored in the preferences folder that contains whether it is the mirror or the main live operational database. This will allow you to exclude 'in code' actions from happening to the mirrored server that would change the data in the data file and break the mirror.

Log files

At the heart of both the old mirroring system and the 2004 mirroring system are log files. To set up a mirror you simply backup the database and begin a log file. Shut down the server and then transfer all the files to a mirror machine. To create a mirror all the files must be identical. Then both the mirrored server and the original server are restarted.

To maintain the mirror, periodically on the original server a new log files is created and the one just closed is sent to the mirror server where it is integrated. This process can be continued indefinitely maintaining both the original and complete duplicate copy on the mirrored server.

Log file numbering scheme

Each backup and log from a backup gets a unique number. If you are familiar with using Backup on 4D Server you should easily recognize them. Ex: MyDatabase[0739].4BK and MyDatabase[0738].4BL.

When a mirroring system is involved each transfer of the log to the mirrored server creates an additional segmented log file. In order to identify these logs, they are given an additional segment number in the name of the file. MyDatabase[0739-0001].4BL; MyDatabase[0739-0002].4BL; MyDatabase[0739-0003].4BL; etc. It is these closed segmented log files that are sent to the mirrored machine for integration.

New log file function

The **New log file** function closes the current log file, renames it, and creates a new log file at the same location as the original log file. This is the same action that takes for a log file during a normal BACKUP command. The difference is that a backup does not take place. Only the new log file.

The new log file function returns the full Path Name of the log file that has just been closed and renamed. This name contains the last backup number and segment number as described above.

The function only executes on a 4D Server. If the command is not executed on 4D Server an error code of 1412 is generated and nothing else happens. The function also requires that a log file be active. If there is no log file active and error of 1403 is returned.

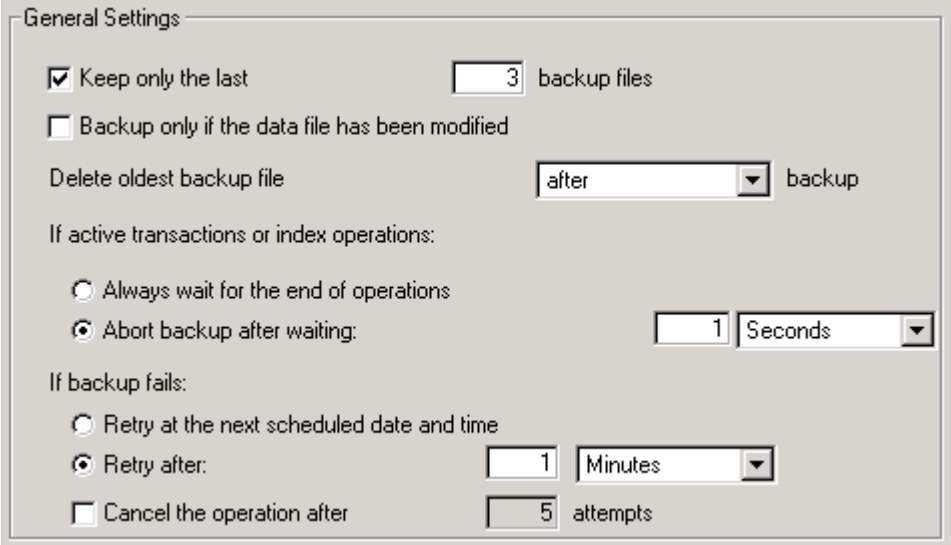
New log file function and critical operations

The **New log file** function and certain critical operations such as transactions or indexing are mutually exclusive. For example you cannot create a **New log file** while a client machine or stored procedure has an open transaction, nor can a new transaction be started while the **New log file** function is taking place.

Just like Backup in all the previous versions, you cannot do a backup with an open transaction. However, 4D 2004.4 offers a significant improvement to this dilemma. The **New log file** function stops new transactions from being started. Therefore, it was possible to cause 4D Client connections attempting

to start a new transaction to freeze until the **New log file** function timed out while it waited for an already existing transaction to complete. In 4D 2004.4 you can now set the timeout to a single second, thereby minimizing this risk.

If you plan to implement a mirroring solution you will most likely want to change the Backup preferences for the database. In the Backup General Settings be sure that the Abort backup after waiting radio button is selected. Also change the default time from 3 minutes to 1 Seconds.



Now that the Client machines will no longer lock up, (at least for more than 1 second.) we can code for the success or failure of the **New log file** function by looking at the error code. If the error code is 1411, the **New log file** failed because of a critical operation. Usually a transaction was in progress. Your scheduler could try again until successful, or the beginning of the next business day. We will examine this code later in the database.

However, when using a mirroring solution if your database uses transactions, (whose doesn't) keep them short. Rather than simply use a transaction on an input screen, you should consider doing all your subforms as arrays and using listboxes instead. This will eliminate the biggest single problem to a mirroring system which is open transactions that are waiting for user input. You definitely do NOT want to use the Automatic Transactions during data entry preference available on older converted databases.

Transfer the log file segment

Once a log file segment is created and closed it is transferred to the mirrored server(s). How the transfer is accomplished is up to the developer. It could be transferred manually/or programmatically using shared network volumes. Using the 4D Internet commands you could create an ftp file transfer mechanism, then use 4D Open to tell the mirrored database to integrate the

ftp'd file. In the example database provided, I have chosen to use a 4D SOAP call to transfer and integrate log files. I chose 4D SOAP as it was the simplest solution and provided an easy way to communicate success or failure of the log integration back to the main database.

INTEGRATE LOG FILE

The **INTEGRATE LOG FILE** command starts the integration of the log file given to the command. The command reads the entries from the log file and merges them into the data file. Mirroring is now complete and the two databases are synchronized.

However, several problems might occur along the way. First, the command will only work on a 4D Server. If it is not running on a 4D Server the file is not integrated and an error of 1412 is generated. Like its counterpart there must be a current activated log file. If there is no current log file a 1403 error is generated.

These two problems are easily overcome. But the bigger problem is if the files get out of sync. (A log file is not successfully sent and integrated, or data is changed accidentally on the mirrored server.) Internal to both the backup and log files are a set of numbers that for lack of a better term I will call sequence verification numbers. For successful integration to take place both the backup and log file segment internal sequence verification numbers must be correct. You can't change these internal numbers simply by renaming the file. These numbers are embedded into the files themselves. These numbers insure that the log file being integrated is the correct 'next' log file to go with the appropriate 'backup'. If anything happens to disrupt the correct sequence of database events in the log file, the integration of the log file will not occur and an error code of 1410 is generated which in sort says: "Wrong log file. The log file exists, but what it contains is not the correct next value expected."

ERROR HANDLING

The process of mirroring without problems is essential to those attempting to set up a mirror. Problems can sometimes occur. Some harmless, and mirroring should continue at the next available scheduled time. Some errors are fatal. No matter what the error, it is essential that proper error handling be observed. If you are not using an **ON ERROR CALL** routine any time you use one of the mirroring commands and an error occurs, the process will abort without completing the task and also without doing anything or even notifying you about the potential problem. In addition if the process is a self delaying process, since it aborted, it will not run the next time it should. It is absolutely essential to create and use an **ON ERROR CALL** routine when using any of the mirroring commands.

Backups

Backing up the primary active database will break the mirroring system. It is necessary to do this when setting up the system initially, or when the mirror gets out of sync. I have coded an **ALERT** into the example database that will warn you when you are about to break the mirror.

4D 2004.4, however, has added a new capability to a mirroring system. With 4D 2004.4 it is now possible to back up the mirrored machine without breaking the mirror. This new feature will finally allow you create up to date backups, for archival or data security reasons. You can either use the mirrored server scheduler to create backups like you would on a normal 4D Server. Or, as we will see in the example database, you can control the backups on the mirror from the main server.

The example database

The example database is designed to create a mirroring system that ranges from simple all the way to providing a rather complex system with multiple mirror machines. It requires two (or more) 4D servers running 4D 2004.4 or higher.

Creating the example database mirror

In order to do mirroring, you must first have a backup with log files established. Then you transfer the database to a mirroring machine. Establishment of a mirror requires that you follow several exact steps.

1) Launch the main operational database. (This database uses some XML preference settings, that need to be selected when the database is first launched. For this database choose the 'Main DB' setting.)
2) Perform a backup of the data. (Note: It would be a good idea to include the plugins folder in the backup)
3) Go to the backup preferences and create a new log file.
4) Shut down the database.
5) Transfer everything, structure, data files, backups, log files, etc. to a second server machine.
6) In the Preferences folder locate the Mirror folder and trash the folder.
7) Launch the mirror database on a 4D Server and this time choose the 'Mirror' setting. The mirror is now ready to go.
8) Launch the 'Main DB' on a 4D Server.

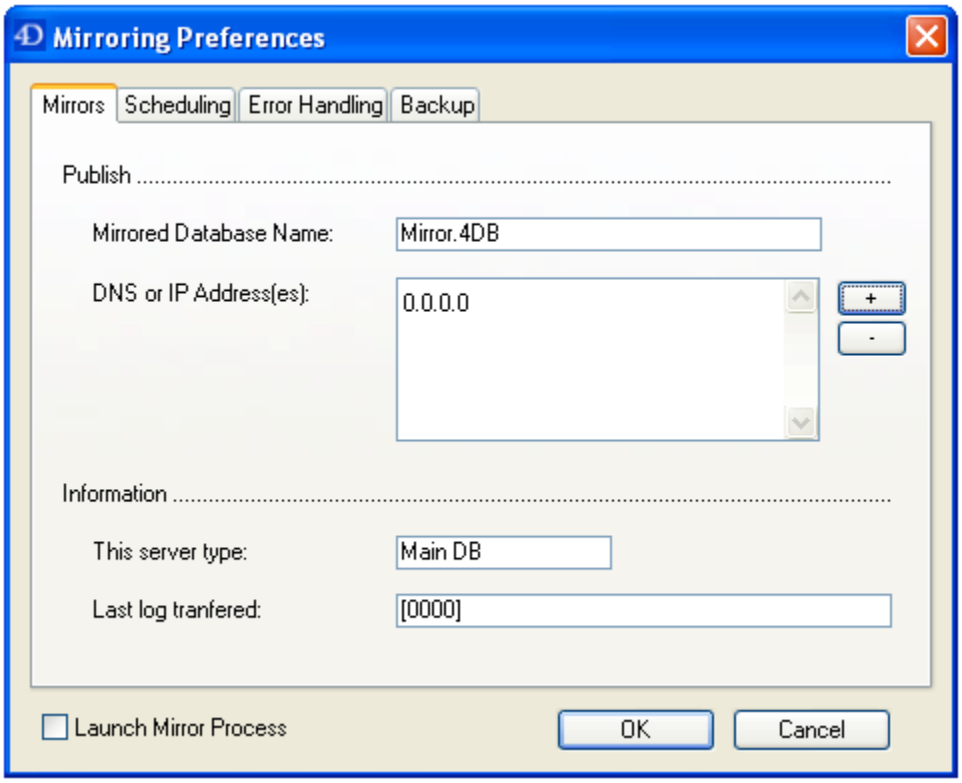
Any time the mirror is broken, you must go back and recreate the mirror from scratch, beginning with step two, the backup of the data.

The mirror preference settings

There is a single mirroring preference settings dialog in the database that is used for scheduling mirroring, transaction delays, error handling, and remote backup settings.

It is imparative to distinguish one database from the other. Since you can NOT use data to tell the databases apart, that would break the mirror, I chose an external preference file. It is a simple XML file that stores the settings. For the mirrored server the only setting used is the fact that it is a Mirrored server. Each time the mirrored server is recreated simply trash the preferences and choose the 'Mirror' setting and that is all the mirror database needs.

On the other hand, the Main server does all the work. It needs far more information.



For security purposes the example database requires the entry of the mirrored database name. I use this internal name to make sure that the mirrored database is running, and that the one you are trying to send the log

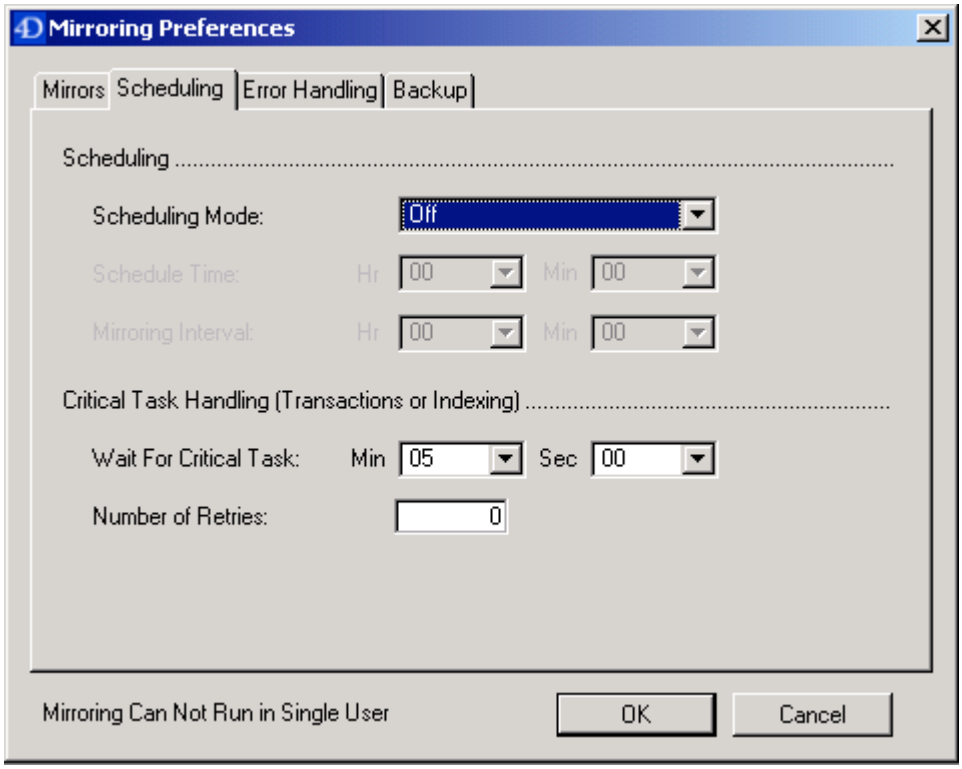
to is the correct database, before I even call the **New log file** function. I don't want to create a new log file that has nowhere to go.

The second setting necessary is the IP address or DNS name of the mirrored server(s). The code in this database will handle multiple mirror machines.

The type of the server and Last Log Transferred are for informational purposes only. It cannot be changed. It indicates the last log successfully integrated into the mirror server. This will aid in disaster recovery if necessary.

At the bottom of the dialog is a status indicator for the mirroring process on the server. If it is not running you can select the checkbox to launch the mirroring process on the server. It will launch, provided that you have entered a mirroring interval. If the process is already running it will be updated with your revised values.

Next is Scheduling. There are four options to choose from. 1) Off. 2) Time Only. This to set a single specific time to do the mirroring once every 24 hours. 3) Time & Interval. This will begin mirroring at the specified time and then repeat at the selected interval. 4) Interval only. This will repeat mirroring every specified interval in hours and/or minutes.

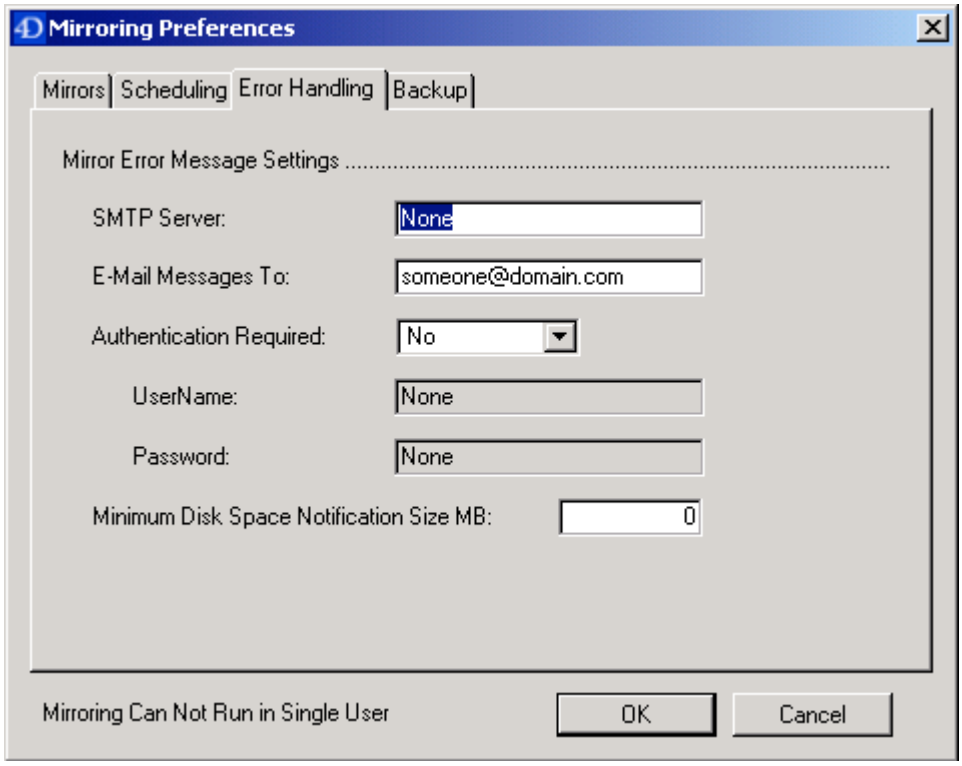


The lower half of this tab is for handling critical operations. Remember we changed the preference setting for the database earlier to time out after one second. But one second is too short to wait for a transaction to complete.

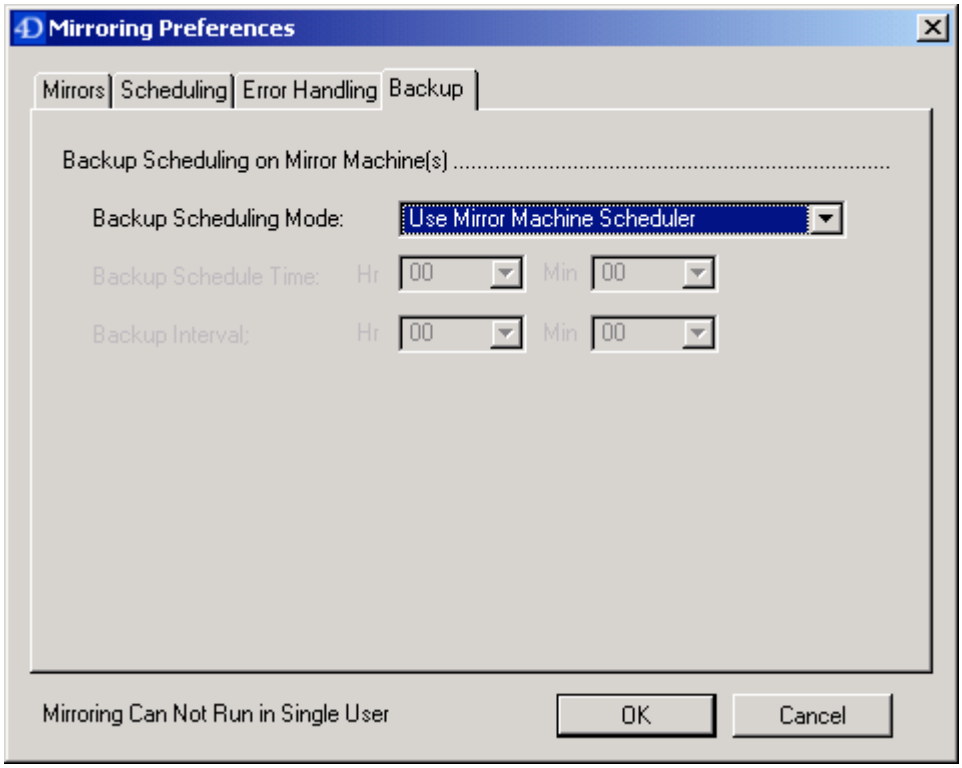
Therefore, the first setting here is the length of time to keep retrying the one second wait. It retries every 1/4 of a second. While this loop is going on Clients will not be able to start a transaction. However, you can test on the client machine to see if this is in progress and display a message that you are waiting for the backup to complete. The total amount of time you should set this for is equal to the maximum amount of time you consider it acceptable for a client machine to wait for the backup. This setting also provides a solution for the backup that is waiting for indexing to complete so it can continue quickly

The second setting is the number of times the code will attempt to create a new log file before it abandons the attempt and skips this scheduled time for mirroring. There is a one minute delay between attempts. This second setting is to allow other processes that might have been waiting on the new log process to now continue, without the penalty of having to abort this mirroring time completely. To get the approximate total time the mirroring will attempt to take place, multiply the Wait for Critical Task time times 1.25 seconds. Then add one minute times the Number of Retries. The result is how long the system will attempt to create a new log file. Testing has shown for that this total should be shorter than your Mirroring Scheduling interval. If you only back up once a day, you can wait a long time for things to complete. On the other hand if you are mirroring every 10 minutes. Perhaps you should skip this time if things are busy.

The third tab is for the Mirror Error Message Settings. In includes all of the settings needed for an SMTP server and e-mail address where you would like to send error messages. If you leave this to the default settings of ‘None’ no messages will be sent.



The last setting on this tab is a hard disk free space amount for the main operational server. As mirroring progresses new log file segments are created. These segments are separate file and are not deleted because they are NOT merged into the original log file. If you needed to restore your data from the last backup, you would need all these small log files to bring the main server back up to date. For a server that runs a long period of time this might begin to fill up the hard drive. Entering a number here will, if the SMTP section is activated, send a warning message when the free space gets below the indicated amount. This number is in MB. It does NOT check the mirror machines. Since normal backups can be performed on the mirror machines the backup routine code in the database will take care of deleting log files that have already been integrated and backed up to a normal backup.



The last tab controls backups on the mirror server(s). Again there are four options. 1) Use Mirror Machine Scheduler. This is an option. Simply, schedule backups to occur using the Backup preferences settings on the mirrored server. However, here is something to consider. What if the main server attempts to send a log file while the mirror machine is performing a backup? This will cause a conflict. Therefore, I have included three options to schedule backing up the mirror machines by controlling them from the main server. 2) via Main Server Time. Again this is a specific time. Since backing up the mirror server only takes place after sending a log file, you should schedule the time to occur at the same time a mirroring takes place. If you don't the backup will occur at the next time a log file is sent after the backup time has passed. The last two 3) via Main Server Time & Interval and 4) via Main Server Interval work the same way as described above.

Mirror preference settings – Server side

The mirror preference settings are saved and used on the server. Since 4D already creates a preferences folder for Backup and other settings the database will use the same architecture to create its own preferences and store them in an XML file.

Determining the location of the file: This is done in a single method. If you want to change the location the string found here is simply modified to the location where you would like the file saved. Don't worry that the values

appears to be a windows format. The code will convert Windows to Mac if needed.

```
If (False)
        ` Method: Mirror_tMirrorPath
        ` 4D Technote on Mirroring a 2004.4 database
        ` Created by: Kent Wilbur
        ` Date: 9/27/2005
        ` Modified Date: 5/16/2006

        ` Purpose: Contains the location of the Mirror path

    <>Mirror_f_Version2004x4:=True
    <>Mirror_fK_Wilbur:=True

End if

C_TEXT($0)
C_TEXT ($1;$tFile)

$tFile:=$1
Case of
    : ($tFile="Backup")
        $0:="Preferences\\Backup\\Backup.xml"
    : ($tFile="Multiples")
        $0:="Preferences\\Mirror\\"
    Else
        $0:="Preferences\\Mirror\\Settings.xml"
End case
```

Note: If you are comparing this tech note to the previous one. Right away you should notice this method is different. This method, along with many others is why the code should be 100% replaced.

Loading/Creating the preferences file: When the 4D Server launches the Mirroring process which attempts to load the preferences file using the *Mirror_HandleMirrorPreferences* method. All of the manipulation of the preference file and all of time setting calculations occur in the single recursive method *Mirror_HandleMirrorPreferences*, so it might be a little confusing to read at first. Try first following the code in the method below assuming that the preference file and folders do not even exist using the portion of the case statement where $tAction= "Load" located near the middle of page 15.

Following through the code with using 'Load' will soon call the method again for 'create', simply jump back to the top of the method and follow it through the second time with $tAction="Create". Page 12 and Page 22.

```
If (False)
    ` Method: Mirror_HandleMirrorPreferences
    ` 4D Technote on Mirroring a 2004.4 database
    ` Created by: Kent Wilbur
    ` Date: 5/4/2006

    ` Purpose: Creates a preferences file for the mirror
```

```
      ` Parameters
      `    $1    -    Action

   <>Mirror_f_Version2004x4:=True
   <>Mirror_fK_Wilbur:=True

End if

   ` Declare parameters
C_TEXT($1;$tAction)

   ` Declare local variables
C_BOOLEAN($fAbort)
C_BOOLEAN($fIntervalValid)
C_BOOLEAN($fTimeValid)
C_DATE($dDate)
C_DATE($dCurrentDate)
C_STRING(16;$sXML_Reference)
C_STRING(16;$sXML_ElementReference)
C_LONGINT($LProcessID)
C_TEXT($tMirror_Time;$tMirror_TimeInterval;$tMirror_BackupTime;$tMirror_BackupTimeInterval;$tMirror_
TransactionDelay;$tMirror_TransactionRetries)
C_TEXT($tCurrentTime)
C_TEXT($tMinimumDiskFreeSpace)
C_TEXT($tPreferencesPath)
C_TEXT($tSettingsFolderPath)
C_TEXT($tXML_ElementValue)
C_TIME($hMirroringInterval)
C_TIME($hMirroringTime)
C_TIME($hTime)

   ` Reassign for readability
$tAction:=$1

   ` Declare default values
$fAbort:=False
$tPreferencesPath:=Mirror_tFormatPathname (Mirror_tMirrorPath ("Mirror"))
```

After getting the path name from *Mirror_tMirrorPath*, the *Mirror_tFormatPathname* method simply modifies the path name for the appropriate platform. (Macintosh or Windows.)

```
Case of
   : ($tAction="Create")
      CONFIRM("Choose the type for this server.";"Main DB";"Mirror")
      If (OK=1)
         <>Mirror_tServerType:="Main DB"
      Else
         <>Mirror_tServerType:="Mirror"
      End if
      <>Mirror_tDatabaseName:="None"
      <>Mirror_tServerIPAddress:="0.0.0.0"
      <>Mirror_tScheduleMode:="Off"
      $tMirror_Time:="00:00:00"
      $tMirror_TimeInterval:="00:00:00"
      <>Mirror_tBackupScheduleMode:="Use Mirror Machine Scheduler"
      $tMirror_BackupTime:="00:00:00"
      $tMirror_BackupTimeInterval:="00:00:00"
      $tMirror_TransactionDelay:="00:05:00"
      <>Mirror_tNextTimeIntervalMode:="00000000000000"
      <>Mirror_tNextBackupIntervalMode:="00000000000000"
```

```
      <>Mirror_LTransactionRetries:=0
      <>Mirror_tLastLogNumber:="[0000]"
      <>Mirror_tSMTPServer:=<>Mirror_tDatabaseName
      <>Mirror_tErrorEMailAccount:="someone@domain.com"
      <>Mirror_tAuthenticationRequired:="No"
      <>Mirror_tAuthenticationUserName:=<>Mirror_tDatabaseName
      <>Mirror_tAuthenticationPassword:=<>Mirror_tDatabaseName
      <>Mirror_LCurrentDiskFreeSpace:=0
      <>Mirror_LMinimumDiskFreeSpace:=0

   : (Count parameters#1)
      $fAbort:=True    ` Not enough parameters
   Else
      $tMirror_BackupTime:=String(<>Mirror_hBackupTime;HH MM SS )
      $tMirror_BackupTimeInterval:=String(<>Mirror_hBackupTimeInterval;HH MM SS )
      $tMirror_Time:=String(<>Mirror_hTime;HH MM SS )
      $tMirror_TimeInterval:=String(<>Mirror_hTimeInterval;HH MM SS )
      $tMirror_TransactionDelay:=String(<>Mirror_hTransactionDelay;HH MM SS )
End case

Case of
   : ($fAbort)
```

The next section of the code deals with setting the next time interval for a mirroring to occur. It's not very exciting to read.

```
   : ($tAction="SetNextTimeInterval")
      Case of
         : (<>Mirror_tScheduleMode="Time Only")
            If (Current time(*)><>Mirror_hTime)
               <>Mirror_tNextTimeIntervalMode:=Mirror_tDateTimeStamp (Add to date(
                        Current date(*);0;0;1);<>Mirror_hTime)
```

Mirror_tDateTimeStamp returns a text string of the passed values in YYYYMMDDHHMMSS format.

```
            Else
               <>Mirror_tNextTimeIntervalMode:=Mirror_tDateTimeStamp (Current date(*);<>Mirror_hTime)
            End if

         : (<>Mirror_tScheduleMode="Time & Interval")
            $dCurrentDate:=Current date(*)
            $hMirroringTime:=Current time(*)
            If (<>Mirror_tNextTimeIntervalMode<Mirror_tDateTimeStamp ($dCurrentDate;$hMirroringTime))
                        ` Something happened and the last scheduled mirroring did not take place
               If (<>Mirror_tNextTimeIntervalMode="00000000000000")
                  $dDate:=$dCurrentDate
                  $hTime:=$hMirroringTime
               Else
                  $dDate:=Date(Substring(<>Mirror_tNextTimeIntervalMode;5;2)+"/"+
                        Substring(<>Mirror_tNextTimeIntervalMode;7;2)+"/"+
                        Substring(<>Mirror_tNextTimeIntervalMode;1;4))
                  $hTime:=Time(Substring(<>Mirror_tNextTimeIntervalMode;9;2)+":"+
                        Substring(<>Mirror_tNextTimeIntervalMode;11;2)+":"+
                        Substring(<>Mirror_tNextTimeIntervalMode;13;2))
               End if

               While (Mirror_tDateTimeStamp ($dDate;$hTime)<Mirror_tDateTimeStamp
                        ($dCurrentDate;$hMirroringTime))
                  $hTime:=$hTime+<>Mirror_hTimeInterval
                  If ($hTime>=?24:00:00?)    ` If the next interval is tomorrow
```

```
            $hTime:=$hTime-?24:00:00?
            $dDate:=Add to date($dDate;0;0;1)
         End if

      End while
      <>Mirror_hTime:=$hTime
   Else

      <>Mirror_hTime:=<>Mirror_hTime+<>Mirror_hTimeInterval    ` Set the next time interval

   End if

   If (<>Mirror_hTime>=?24:00:00?)    ` If the next interval is tomorrow
      <>Mirror_hTime:=<>Mirror_hTime-?24:00:00?
   End if

   If ($hMirroringTime<<>Mirror_hTime)
      <>Mirror_tNextTimeIntervalMode:=Mirror_tDateTimeStamp (Current date(*);<>Mirror_hTime)
   Else
      <>Mirror_tNextTimeIntervalMode:=Mirror_tDateTimeStamp (Add to date(Current
date(*);0;0;1);<>Mirror_hTime)
   End if

Else
   <>Mirror_tNextTimeIntervalMode:="00000000000000"
End case
```

The next section of the code deals with setting the next time interval for a backup to occur when it is being controlled by the main server.

```
: ($tAction="SetNextBackupInterval")
   Case of
      : (<>Mirror_tBackupScheduleMode="via Main Server Time")
         If (Current time(*)><>Mirror_hBackupTime)
            <>Mirror_tNextBackupIntervalMode:=Mirror_tDateTimeStamp (Add to date(
                  Current date(*);0;0;1);<>Mirror_hBackupTime)
         Else
            <>Mirror_tNextBackupIntervalMode:=Mirror_tDateTimeStamp (
                  Current date(*);<>Mirror_hBackupTime)
         End if

      : (<>Mirror_tBackupScheduleMode="via Main Server Time & Interval")
         $hMirroringTime:=Current time(*)
         $dCurrentDate:=Current date(*)
         If (<>Mirror_tNextBackupIntervalMode<Mirror_tDateTimeStamp
               ($dCurrentDate;$hMirroringTime))
                  ` Something happened and the last scheduled mirroring did not take place
            If (<>Mirror_tNextBackupIntervalMode="00000000000000")
               $dDate:=$dCurrentDate
               $hTime:=$hMirroringTime
            Else
               $dDate:=Date(Substring(<>Mirror_tNextBackupIntervalMode;5;2)+"/"+
                     Substring(<>Mirror_tNextBackupIntervalMode;7;2)+"/"+
                     Substring(<>Mirror_tNextBackupIntervalMode;1;4))
               $hTime:=Time(Substring(<>Mirror_tNextBackupIntervalMode;9;2)+":"+
                     Substring(<>Mirror_tNextBackupIntervalMode;11;2)+":"+
                     Substring(<>Mirror_tNextBackupIntervalMode;13;2))
            End if

            While (Mirror_tDateTimeStamp ($dDate;$hTime)<Mirror_tDateTimeStamp
                  ($dCurrentDate;$hMirroringTime))
```

```
      $hTime:=$hTime+<>Mirror_hBackupTimeInterval
      If ($hTime>=?24:00:00?)    ` If the next interval is tomorrow
         $hTime:=$hTime-?24:00:00?
         $dDate:=Add to date($dDate;0;0;1)
      End if

   End while

   <>Mirror_hBackupTime:=$hTime

Else

   <>Mirror_hBackupTime:=<>Mirror_hBackupTime+<>Mirror_hBackupTimeInterval
                                       ` Set the next time interval

End if

If (<>Mirror_hBackupTime>=?24:00:00?)    ` If the next interval is tomorrow
   <>Mirror_hBackupTime:=<>Mirror_hBackupTime-?24:00:00?
End if

If ($hMirroringTime<<>Mirror_hBackupTime)
   <>Mirror_tNextBackupIntervalMode:=Mirror_tDateTimeStamp
                  (Current date(*);<>Mirror_hBackupTime)
Else
   <>Mirror_tNextBackupIntervalMode:=Mirror_tDateTimeStamp
                  (Add to date(Current date(*);0;0;1);<>Mirror_hBackupTime)
End if

Else
   $fAbort:=False
   $dDate:=Current date(*)
   $hTime:=Current time(*)
   $hMirroringTime:=$hTime+<>Mirror_hBackupTimeInterval
   If ($hMirroringTime>?24:00:00?)    ` Check to see if midnight has passed
      $hMirroringTime:=$hMirroringTime-?24:00:00?
      $tCurrentTime:=Mirror_tDateTimeStamp (Add to date($dDate;0;0;1);$hMirroringTime)
   Else
      $tCurrentTime:=Mirror_tDateTimeStamp ($dDate;$hMirroringTime)
   End if

   Case of
      : (<>Mirror_tNextBackupIntervalMode="00000000000000")
         $dDate:=Current date(*)
         $hTime:=Current time(*)

      : ($tCurrentTime<<>Mirror_tNextBackupIntervalMode)
            ` We are changing values and we should keep the existing time already scheduled
         $fAbort:=True

      Else
         $dDate:=Date(Substring(<>Mirror_tNextBackupIntervalMode;5;2)+"/"+
                  Substring(<>Mirror_tNextBackupIntervalMode;7;2)+"/"+
                  Substring(<>Mirror_tNextBackupIntervalMode;1;4))
         $hTime:=Time(Substring(<>Mirror_tNextBackupIntervalMode;9;2)+":"+
                  Substring(<>Mirror_tNextBackupIntervalMode;11;2)+":"+
                  Substring(<>Mirror_tNextBackupIntervalMode;13;2))
   End case

   If (Not($fAbort))
      $hTime:=$hTime+<>Mirror_hBackupTimeInterval
      If ($hTime>?24:00:00?)    ` Check to see if midnight has passed
```

```
            $hTime:=$hTime-?24:00:00?
            $dDate:=Add to date($dDate;0;0;1)
         End if

         <>Mirror_tNextBackupIntervalMode:=Mirror_tDateTimeStamp ($dDate;$hTime)

         If ($tCurrentTime><>Mirror_tNextBackupIntervalMode)
               ` The next scheduled time has already passed, reset to the current scheduled time
            <>Mirror_tNextBackupIntervalMode:=$tCurrentTime
         End if

      End if

   End case


: ($tAction="Load")
   $tSettingsFolderPath:=Mirror_tGetFolderPathnames ($tPreferencesPath)
```

Mirror_tFolderPathnames method separates the name of the path folders from the name of the file itself.

```
   If (Test path name($tSettingsFolderPath)#0)    ` See if the directory exists
      Mirror_PreparePath ($tSettingsFolderPath)
   End if
```

The **Test path name** function will return a 0 if the value being checked is a folder path. Since we are testing for a folder, if any other value is returned, one or more of the folders do not exist. *Mirror_PreparePath* is a method that will create any missing folders in the path.

Below we again check for the file using the **Test path name** function It will return a 1 if the value being checked is a file. By checking the full path, file name included, if anything other than a 1 is returned the file doesn't exist.

```
   If (Test path name($tPreferencesPath)#1)    ` Check to see if a valid file exists
```

So we call the *Mirror_HandleMirrorPreferences* method again to create the preference file.

```
      Mirror_HandleMirrorPreferences ("Create")    ` If not create a preferences file

   Else
```

If the file exists we simply load the preference settings.

```
      While (Semaphore("$ModMirrorPrefs"))    ` Limit acces
         Mirror_MyDelay (Current process;1)
      End while
      $sXML_Reference:=DOM Parse XML source($tPreferencesPath)
      $sXML_ElementReference:=DOM Find XML element
                  ($sXML_Reference;"/Mirror/PreferenceSetting/Mirror_ServerType")
      DOM GET XML ELEMENT VALUE ($sXML_ElementReference;<>Mirror_tServerType)
      $sXML_ElementReference:= DOM Find XML element
                   ($sXML_Reference;"/Mirror/PreferenceSetting/Mirror_ServerName")
      DOM GET XML ELEMENT VALUE ($sXML_ElementReference;<>Mirror_tDatabaseName)
      $sXML_ElementReference:= DOM Find XML element
                   ($sXML_Reference;"/Mirror/PreferenceSetting/Mirror_ServerIPAddress")
      DOM GET XML ELEMENT VALUE ($sXML_ElementReference;<>Mirror_tServerIPAddress)
```

```
$sXML_ElementReference:= DOM Find XML element
            ($sXML_Reference;"/Mirror/PreferenceSetting/Mirror_ScheduleMode")
DOM GET XML ELEMENT VALUE ($sXML_ElementReference;<>Mirror_tScheduleMode)
$sXML_ElementReference:= DOM Find XML element
            ($sXML_Reference;"/Mirror/PreferenceSetting/Mirror_Time")
DOM GET XML ELEMENT VALUE ($sXML_ElementReference;$tMirror_Time)
$sXML_ElementReference:= DOM Find XML element
            ($sXML_Reference;"/Mirror/PreferenceSetting/Mirror_TimeInterval")
DOM GET XML ELEMENT VALUE ($sXML_ElementReference;$tMirror_TimeInterval)
$sXML_ElementReference:= DOM Find XML element
            ($sXML_Reference;"/Mirror/PreferenceSetting/Mirror_NextTimeInterval")
DOM GET XML ELEMENT VALUE ($sXML_ElementReference;<>Mirror_tNextTimeIntervalMode)
$sXML_ElementReference:= DOM Find XML element
            ($sXML_Reference;"/Mirror/PreferenceSetting/Mirror_BackupScheduleMode")
DOM GET XML ELEMENT VALUE ($sXML_ElementReference;<>Mirror_tBackupScheduleMode)
$sXML_ElementReference:= DOM Find XML element
            ($sXML_Reference;"/Mirror/PreferenceSetting/Mirror_BackupTime")
DOM GET XML ELEMENT VALUE ($sXML_ElementReference;$tMirror_BackupTime)
$sXML_ElementReference:= DOM Find XML element
            ($sXML_Reference;"/Mirror/PreferenceSetting/Mirror_BackupTimeInterval")
DOM GET XML ELEMENT VALUE ($sXML_ElementReference;$tMirror_BackupTimeInterval)
$sXML_ElementReference:= DOM Find XML element
            ($sXML_Reference;"/Mirror/PreferenceSetting/Mirror_NextBackupIntervalMode")
DOM GET XML ELEMENT VALUE ($sXML_ElementReference;
           <>Mirror_tNextBackupIntervalMode)
$sXML_ElementReference:= DOM Find XML element
            ($sXML_Reference;"/Mirror/PreferenceSetting/Mirror_TransactionDelay")
DOM GET XML ELEMENT VALUE ($sXML_ElementReference;$tMirror_TransactionDelay)
$sXML_ElementReference:= DOM Find XML element
            ($sXML_Reference;"/Mirror/PreferenceSetting/Mirror_TransactionRetries")
DOM GET XML ELEMENT VALUE ($sXML_ElementReference;$tMirror_TransactionRetries)
$sXML_ElementReference:= DOM Find XML element
            ($sXML_Reference;"/Mirror/PreferenceSetting/Mirror_LastSegmentNumber")
DOM GET XML ELEMENT VALUE ($sXML_ElementReference;<>Mirror_tLastLogNumber)
$sXML_ElementReference:= DOM Find XML element
           ($sXML_Reference;"/Mirror/PreferenceSetting/Mirror_ErrorSMTPServer")
DOM GET XML ELEMENT VALUE ($sXML_ElementReference;<>Mirror_tSMTPServer)
$sXML_ElementReference:= DOM Find XML element
            ($sXML_Reference;"/Mirror/PreferenceSetting/Mirror_ErrorEMail")
DOM GET XML ELEMENT VALUE ($sXML_ElementReference;<>Mirror_tErrorEMailAccount)
$sXML_ElementReference:= DOM Find XML element
            ($sXML_Reference;"/Mirror/PreferenceSetting/Mirror_ErrorEMailAuthentication")
DOM GET XML ELEMENT VALUE ($sXML_ElementReference;<>Mirror_tAuthenticationRequired)
$sXML_ElementReference:= DOM Find XML element
            ($sXML_Reference;"/Mirror/PreferenceSetting/Mirror_ErrorEMailUsername")
DOM GET XML ELEMENT VALUE ($sXML_ElementReference;
           <>Mirror_tAuthenticationUserName)
$sXML_ElementReference:= DOM Find XML element
           ($sXML_Reference;"/Mirror/PreferenceSetting/Mirror_ErrorEMailPassword")
DOM GET XML ELEMENT VALUE($sXML_ElementReference;<>Mirror_tAuthenticationPassword)
$sXML_ElementReference:= DOM Find XML element
            ($sXML_Reference;"/Mirror/PreferenceSetting/Mirror_MinimumDiskFreeSpace")
DOM GET XML ELEMENT VALUE($sXML_ElementReference;$tMinimumDiskFreeSpace)
DOM CLOSE XML($sXML_Reference)
CLEAR SEMAPHORE("$ModMirrorPrefs")
<>Mirror_hTime:=Time($tMirror_Time)
<>Mirror_hTimeInterval:=Time($tMirror_TimeInterval)
<>Mirror_hBackupTime:=Time($tMirror_BackupTime)
<>Mirror_hBackupTimeInterval:=Time($tMirror_BackupTimeInterval)
<>Mirror_hTransactionDelay:=Time($tMirror_TransactionDelay)
<>Mirror_LTransactionRetries:=Num($tMirror_TransactionRetries)
<>Mirror_LMinimumDiskFreeSpace:=Num($tMinimumDiskFreeSpace)
```

```
   End if
```

These next few sections handle using the mirroring preference setting dialog popups.

```
: ($tAction="HandleModePopup")
   Case of
      : (Mirror_atMode{Mirror_atMode}="Off")    ` Keep the last settings
            `Mirror_atTimeHr:=1    ` Set to Zero
            `Mirror_atTimeMin:=1
            `Mirror_atHr:=1    ` Set to Zero
            `Mirror_atMin:=1

      : (Mirror_atMode{Mirror_atMode}="Time Only")
         Mirror_atHr:=1    ` Set to Zero
         Mirror_atMin:=1
         Mirror_HandleMirrorPreferences ("HandleBackupModePopup")

      : (Mirror_atMode{Mirror_atMode}="Interval Only")
         Mirror_atTimeHr:=1    ` Set to Zero
         Mirror_atTimeMin:=1

      Else
         Mirror_HandleMirrorPreferences ("HandleBackupModePopup")

   End case
   <>Mirror_hTime:=Time(Mirror_atTimeHr{Mirror_atTimeHr}+":"+
                  Mirror_atTimeMin{Mirror_atTimeMin}+":00")
   <>Mirror_hTimeInterval:=Time(Mirror_atHr{Mirror_atHr}+":"+Mirror_atMin{Mirror_atMin}+":00")
   Mirror_HandleMirrorPreferences ("SetModePopup")

: ($tAction="SetModePopup")
   Case of
      : (Mirror_atMode{Mirror_atMode}="Off")
         DISABLE BUTTON (*;"Mirror_Time@")
         DISABLE BUTTON (*;"Mirror_Interval@")
         SET RGB COLORS(*;"Mirror_Time@";0x00BBBBBB;0x00FFFFFF)
         SET RGB COLORS (*;"Mirror_Interval@";0x00BBBBBB;0x00FFFFFF)

      : (Mirror_atMode{Mirror_atMode}="Time Only")
         ENABLE BUTTON (*;"Mirror_Time@")
         DISABLE BUTTON (*;"Mirror_Interval@")
         SET RGB COLORS (*;"Mirror_Time@";0x0000;0x00FFFFFF)
         SET RGB COLORS (*;"Mirror_Interval@";0x00BBBBBB;0x00FFFFFF)

      : (Mirror_atMode{Mirror_atMode}="Interval Only")
         DISABLE BUTTON (*;"Mirror_Time@")
         ENABLE BUTTON (*;"Mirror_Interval@")
         SET RGB COLORS (*;"Mirror_Time@";0x00BBBBBB;0x00FFFFFF)
         SET RGB COLORS (*;"Mirror_Interval@";0x0000;0x00FFFFFF)

      Else
         ENABLE BUTTON (*;"Mirror_Time@")
         ENABLE BUTTON (*;"Mirror_Interval@")
         SET RGB COLORS (*;"Mirror_Time@";0x0000;0x00FFFFFF)
         SET RGB COLORS (*;"Mirror_Interval@";0x0000;0x00FFFFFF)

   End case

: ($tAction="HandleBackupModePopup")
   Case of
```

```
      : (Mirror_atBackupMode{Mirror_atBackupMode}="Use Mirror Machine Scheduler")
         Mirror_atBackupTimeHr:=1    ` Set to Zero
         Mirror_atBackupTimeMin:=1
         Mirror_atBackupHr:=1    ` Set to Zero
         Mirror_atBackupMin:=1

      : (Mirror_atMode{Mirror_atMode}="Time Only") &
                  (Mirror_atBackupMode{Mirror_atBackupMode}#"via Main Server Time")
         ALERT("Backups on the Mirrored Server may only occur at the secheduled mirroring time."+
                  "\rTo use a different time you must use the preference settings on the Mirrored
                  mac"+"hine itself.")
         Mirror_atBackupMode:=2
         Mirror_atBackupTimeHr:=Mirror_atTimeHr
         Mirror_atBackupTimeMin:=Mirror_atTimeMin
         Mirror_atBackupHr:=1    ` Set to Zero
         Mirror_atBackupMin:=1

      : (Mirror_atMode{Mirror_atMode}#"Interval Only") &
                  (Mirror_atBackupMode{Mirror_atBackupMode}="via Main Server Interval")
         ALERT("Backups on the Mirrored Server may only be set to Interval mode when the
                        Scheduli"+"ng Mode is also Interval Mode")
         Mirror_atBackupMode:=Mirror_atMode
         Mirror_atBackupTimeHr:=Mirror_atTimeHr
         Mirror_atBackupTimeMin:=Mirror_atTimeMin
         Mirror_atBackupHr:=Mirror_atHr
         Mirror_atBackupMin:=Mirror_atMin

      : (Mirror_atBackupMode{Mirror_atBackupMode}="via Main Server Time")
         Mirror_atBackupHr:=1    ` Set to Zero
         Mirror_atBackupMin:=1

      : (Mirror_atBackupMode{Mirror_atBackupMode}="via Main Server Interval")
         Mirror_atBackupTimeHr:=1    ` Set to Zero
         Mirror_atBackupTimeMin:=1

   End case
   <>Mirror_hBackupTime:=Time(Mirror_atBackupTimeHr{Mirror_atBackupTimeHr}+":"+
                  Mirror_atBackupTimeMin{Mirror_atBackupTimeMin}+":00")
   <>Mirror_hBackupTimeInterval:=Time(Mirror_atBackupHr{Mirror_atBackupHr}+":"+
                  Mirror_atBackupMin{Mirror_atBackupMin}+":00")
   Mirror_HandleMirrorPreferences ("SetBackupModePopup")

: ($tAction="SetBackupModePopup")
   Case of
      : (Mirror_atBackupMode{Mirror_atBackupMode}="Use Mirror Machine Scheduler")
         DISABLE BUTTON(*;"Mirror_BackupTime@")
         DISABLE BUTTON (*;"Mirror_BackupInterval@")
         SET RGB COLORS (*;"Mirror_BackupTime@";0x00BBBBBB;0x00FFFFFF)
         SET RGB COLORS (*;"Mirror_BackupInterval@";0x00BBBBBB;0x00FFFFFF)

      : (Mirror_atBackupMode{Mirror_atBackupMode}="via Main Server Time")
         ENABLE BUTTON (*;"Mirror_BackupTime@")
         DISABLE BUTTON (*;"Mirror_BackupInterval@")
         SET RGB COLORS (*;"Mirror_BackupTime@";0x0000;0x00FFFFFF)
         SET RGB COLORS (*;"Mirror_BackupInterval@";0x00BBBBBB;0x00FFFFFF)

      : (Mirror_atBackupMode{Mirror_atBackupMode}="via Main Server Interval")
         DISABLE BUTTON (*;"Mirror_BackupTime@")
         ENABLE BUTTON (*;"Mirror_BackupInterval@")
         SET RGB COLORS (*;"Mirror_BackupTime@";0x00BBBBBB;0x00FFFFFF)
         SET RGB COLORS (*;"Mirror_BackupInterval@";0x0000;0x00FFFFFF)
```

```
   Else
      ENABLE BUTTON (*;"Mirror_BackupTime@")
      ENABLE BUTTON (*;"Mirror_BackupInterval@")
      SET RGB COLORS (*;"Mirror_BackupTime@";0x0000;0x00FFFFFF)
      SET RGB COLORS (*;"Mirror_BackupInterval@";0x0000;0x00FFFFFF)

End case
```

This sections checks to see if the time chose for backups will work with the time settings for mirroring. If they are not compatible it warns the user that the backup will not happen exactly at the time that they have currently chosen.

```
: ($tAction="ValidateBackupScheduling")
   If (Mirror_atMode{Mirror_atMode}="Time Only")
      If ((Mirror_atBackupTimeHr#Mirror_atTimeHr) | (Mirror_atBackupTimeMin#Mirror_atTimeMin))
         ALERT("Backups on the Mirrored Server may only occur at the secheduled mirroring time."+
                  "\rTo use a different time you must use the preference settings on the Mirrored
                  mac"+"hine itself.")
         Mirror_atBackupMode:=2
         Mirror_atBackupTimeHr:=Mirror_atTimeHr
         Mirror_atBackupTimeMin:=Mirror_atTimeMin
      End if

   Else
      $fTimeValid:=False
      $fIntervalValid:=False

      If ((<>Mirror_hTime#?00:00:00?) | (<>Mirror_hBackupTime#?00:00:00?))
         If (<>Mirror_hTime<=<>Mirror_hBackupTime)
            $hMirroringTime:=<>Mirror_hTime
            Repeat
               If ($hMirroringTime=<>Mirror_hBackupTime)
                  $fTimeValid:=True
               End if
               $hMirroringTime:=$hMirroringTime+<>Mirror_hTimeInterval
            Until (($fTimeValid) | ($hMirroringTime>?24:00:00?) | (<>Mirror_hTimeInterval=?00:00:00?))
         Else
            $hMirroringTime:=<>Mirror_hTime
            If ($hMirroringTime=<>Mirror_hBackupTimeInterval)
               $fIntervalValid:=True
            Else
               Repeat
                  $hMirroringTime:=$hMirroringTime-<>Mirror_hTimeInterval
                  If ($hMirroringTime=<>Mirror_hBackupTime)
                     $fTimeValid:=True
                  End if
               Until (($fTimeValid) | ($hMirroringTime<<>Mirror_hTimeInterval) |
                           (<>Mirror_hTimeInterval=?00:00:00?))
            End if
         End if
      End if

      If ((<>Mirror_hTimeInterval#?00:00:00?) & (<>Mirror_hBackupTimeInterval#?00:00:00?))
         $hMirroringTime:=<>Mirror_hTimeInterval
         Repeat
            If ($hMirroringTime=<>Mirror_hBackupTimeInterval)
               $fIntervalValid:=True
            End if
            $hMirroringTime:=$hMirroringTime+<>Mirror_hTimeInterval
```

```
      Until (($fIntervalValid) | ($hMirroringTime>?24:00:00?) | (<>Mirror_hTimeInterval=?00:00:00?))
    End if
  End if

  Case of
    : ($fIntervalValid) & (Mirror_atBackupMode{Mirror_atBackupMode}="via Main Server Interval")
        ` OK

    : (((Not($fTimeValid)) & (<>Mirror_hTimeInterval#?00:00:00?)) | ((Not($fIntervalValid)) &
        (<>Mirror_hBackupTimeInterval#?00:00:00?)))
      ALERT("Backup not synchronized with scheduled Mirroring. Best available backup time will"+
        " be used.")
  End case



: ($tAction="Save")
  While (Semaphore("$ModMirrorPrefs"))
    Mirror_MyDelay (Current process;1)
  End while
  $sXML_Reference:=DOM Parse XML source($tPreferencesPath)
  $sXML_ElementReference:=DOM Find XML element
      ($sXML_Reference;"/Mirror/PreferenceSetting/Mirror_ServerType")
  DOM SET XML ELEMENT VALUE($sXML_ElementReference;<>Mirror_tServerType)
  $sXML_ElementReference:=DOM Find XML element
      ($sXML_Reference;"/Mirror/PreferenceSetting/Mirror_ServerName")
  DOM SET XML ELEMENT VALUE($sXML_ElementReference;<>Mirror_tDatabaseName)
  $sXML_ElementReference:=DOM Find XML element
      ($sXML_Reference;"/Mirror/PreferenceSetting/Mirror_ServerIPAddress")
  DOM SET XML ELEMENT VALUE($sXML_ElementReference;<>Mirror_tServerIPAddress)
  $sXML_ElementReference:=DOM Find XML element
      ($sXML_Reference;"/Mirror/PreferenceSetting/Mirror_ScheduleMode")
  DOM SET XML ELEMENT VALUE($sXML_ElementReference;<>Mirror_tScheduleMode)
  $sXML_ElementReference:=DOM Find XML element
      ($sXML_Reference;"/Mirror/PreferenceSetting/Mirror_Time")
  DOM SET XML ELEMENT VALUE($sXML_ElementReference;$tMirror_Time)
  $sXML_ElementReference:=DOM Find XML element
      ($sXML_Reference;"/Mirror/PreferenceSetting/Mirror_TimeInterval")
  DOM SET XML ELEMENT VALUE($sXML_ElementReference;$tMirror_TimeInterval)
  $sXML_ElementReference:=DOM Find XML element
      ($sXML_Reference;"/Mirror/PreferenceSetting/Mirror_NextTimeInterval")
  DOM SET XML ELEMENT VALUE($sXML_ElementReference;<>Mirror_tNextTimeIntervalMode)
  $sXML_ElementReference:=DOM Find XML element
      ($sXML_Reference;"/Mirror/PreferenceSetting/Mirror_BackupScheduleMode")
  DOM SET XML ELEMENT VALUE($sXML_ElementReference;<>Mirror_tBackupScheduleMode)
  $sXML_ElementReference:=DOM Find XML element
      ($sXML_Reference;"/Mirror/PreferenceSetting/Mirror_BackupTime")
  DOM SET XML ELEMENT VALUE($sXML_ElementReference;$tMirror_BackupTime)
  $sXML_ElementReference:=DOM Find XML element
      ($sXML_Reference;"/Mirror/PreferenceSetting/Mirror_BackupTimeInterval")
  DOM SET XML ELEMENT VALUE($sXML_ElementReference;$tMirror_BackupTimeInterval)
  $sXML_ElementReference:=DOM Find XML element
      ($sXML_Reference;"/Mirror/PreferenceSetting/Mirror_NextBackupIntervalMode")
  DOM SET XML ELEMENT VALUE($sXML_ElementReference;<>Mirror_tNextBackupIntervalMode)
  $sXML_ElementReference:=DOM Find XML element
      ($sXML_Reference;"/Mirror/PreferenceSetting/Mirror_TransactionDelay")
  DOM SET XML ELEMENT VALUE($sXML_ElementReference;$tMirror_TransactionDelay)
  $sXML_ElementReference:=DOM Find XML element
      ($sXML_Reference;"/Mirror/PreferenceSetting/Mirror_TransactionRetries")
  DOM SET XML ELEMENT VALUE($sXML_ElementReference;String(<>Mirror_LTransactionRetries))
  $sXML_ElementReference:=DOM Find XML element
```

```
                ($sXML_Reference;"/Mirror/PreferenceSetting/Mirror_LastSegmentNumber")
   DOM SET XML ELEMENT VALUE($sXML_ElementReference;<>Mirror_tLastLogNumber)
   $sXML_ElementReference:=DOM Find XML element
                ($sXML_Reference;"/Mirror/PreferenceSetting/Mirror_ErrorSMTPServer")
   DOM SET XML ELEMENT VALUE($sXML_ElementReference;<>Mirror_tSMTPServer)
   $sXML_ElementReference:=DOM Find XML element
                ($sXML_Reference;"/Mirror/PreferenceSetting/Mirror_ErrorEMail")
   DOM SET XML ELEMENT VALUE($sXML_ElementReference;<>Mirror_tErrorEMailAccount)
   $sXML_ElementReference:=DOM Find XML element
                ($sXML_Reference;"/Mirror/PreferenceSetting/Mirror_ErrorEMailAuthentication")
   DOM SET XML ELEMENT VALUE($sXML_ElementReference;<>Mirror_tAuthenticationRequired)
   $sXML_ElementReference:=DOM Find XML element
                ($sXML_Reference;"/Mirror/PreferenceSetting/Mirror_ErrorEMailUsername")
   DOM SET XML ELEMENT VALUE($sXML_ElementReference;<>Mirror_tAuthenticationUserName)
   $sXML_ElementReference:=DOM Find XML element
                ($sXML_Reference;"/Mirror/PreferenceSetting/Mirror_ErrorEMailPassword")
   DOM SET XML ELEMENT VALUE($sXML_ElementReference;<>Mirror_tAuthenticationPassword)
   $sXML_ElementReference:=DOM Find XML element
                ($sXML_Reference;"/Mirror/PreferenceSetting/Mirror_MinimumDiskFreeSpace")
   DOM SET XML ELEMENT VALUE($sXML_ElementReference;
                String(<>Mirror_LMinimumDiskFreeSpace))
   DOM EXPORT TO FILE($sXML_Reference;$tPreferencesPath)
   DOM CLOSE XML($sXML_Reference)
   CLEAR SEMAPHORE("$ModMirrorPrefs")

: ($tAction="SaveFromClient") | ($tAction="Save&LaunchFromClient")
   Mirror_HandleMirrorPreferences ("Save")
   Case of
      : ($tAction="Save&LaunchFromClient")
         $LProcessID:=New process("Mirror_P_MirrorProcess";32000;"MirroringProcess";*)
      Else
         $LProcessID:=Process number("MirroringProcess";*)
         If ($LProcessID>0)
            Mirror_MyDelay ($LProcessID;0)    ` Wake up the process to update the parameters
         End if
   End case

: ($tAction="Create")
   While (Semaphore("$ModMirrorPrefs"))
      Mirror_MyDelay (Current process;1)
   End while

   $sXML_Reference:=DOM Create XML Ref("Mirror")
   $tXML_ElementValue:="/Mirror/PreferenceSetting/Mirror_ServerType"
   $sXML_ElementReference:=DOM Create XML element($sXML_Reference;$tXML_ElementValue)
   DOM SET XML ELEMENT VALUE($sXML_ElementReference;<>Mirror_tServerType)
   $tXML_ElementValue:="/Mirror/PreferenceSetting/Mirror_ServerName"
   $sXML_ElementReference:= DOM Create XML element ($sXML_Reference;$tXML_ElementValue)
   DOM SET XML ELEMENT VALUE ($sXML_ElementReference;<>Mirror_tDatabaseName)
   $tXML_ElementValue:="/Mirror/PreferenceSetting/Mirror_ServerIPAddress"
   $sXML_ElementReference:= DOM Create XML element ($sXML_Reference;$tXML_ElementValue)
   DOM SET XML ELEMENT VALUE ($sXML_ElementReference;<>Mirror_tServerIPAddress)
   $tXML_ElementValue:="/Mirror/PreferenceSetting/Mirror_ScheduleMode"
   $sXML_ElementReference:= DOM Create XML element ($sXML_Reference;$tXML_ElementValue)
   DOM SET XML ELEMENT VALUE ($sXML_ElementReference;<>Mirror_tScheduleMode)
   $tXML_ElementValue:="/Mirror/PreferenceSetting/Mirror_Time"
   $sXML_ElementReference:= DOM Create XML element ($sXML_Reference;$tXML_ElementValue)
   DOM SET XML ELEMENT VALUE ($sXML_ElementReference;$tMirror_Time)
   $tXML_ElementValue:="/Mirror/PreferenceSetting/Mirror_TimeInterval"
   $sXML_ElementReference:= DOM Create XML element ($sXML_Reference;$tXML_ElementValue)
   DOM SET XML ELEMENT VALUE ($sXML_ElementReference;$tMirror_TimeInterval)
   $tXML_ElementValue:="/Mirror/PreferenceSetting/Mirror_NextTimeInterval"
```

```
      $sXML_ElementReference:= DOM Create XML element ($sXML_Reference;$tXML_ElementValue)
      DOM SET XML ELEMENT VALUE ($sXML_ElementReference;<>Mirror_tNextTimeIntervalMode)
      $tXML_ElementValue:="/Mirror/PreferenceSetting/Mirror_BackupScheduleMode"
      $sXML_ElementReference:= DOM Create XML element ($sXML_Reference;$tXML_ElementValue)
      DOM SET XML ELEMENT VALUE ($sXML_ElementReference;<>Mirror_tBackupScheduleMode)
      $tXML_ElementValue:="/Mirror/PreferenceSetting/Mirror_BackupTime"
      $sXML_ElementReference:= DOM Create XML element ($sXML_Reference;$tXML_ElementValue)
      DOM SET XML ELEMENT VALUE ($sXML_ElementReference;$tMirror_BackupTime)
      $tXML_ElementValue:="/Mirror/PreferenceSetting/Mirror_BackupTimeInterval"
      $sXML_ElementReference:= DOM Create XML element ($sXML_Reference;$tXML_ElementValue)
      DOM SET XML ELEMENT VALUE ($sXML_ElementReference;$tMirror_BackupTimeInterval)
      $tXML_ElementValue:="/Mirror/PreferenceSetting/Mirror_NextBackupIntervalMode"
      $sXML_ElementReference:= DOM Create XML element ($sXML_Reference;$tXML_ElementValue)
      DOM SET XML ELEMENT VALUE ($sXML_ElementReference;<>Mirror_tNextBackupIntervalMode)
      $tXML_ElementValue:="/Mirror/PreferenceSetting/Mirror_TransactionDelay"
      $sXML_ElementReference:= DOM Create XML element ($sXML_Reference;$tXML_ElementValue)
      DOM SET XML ELEMENT VALUE ($sXML_ElementReference;$tMirror_TransactionDelay)
      $tXML_ElementValue:="/Mirror/PreferenceSetting/Mirror_TransactionRetries"
      $sXML_ElementReference:= DOM Create XML element ($sXML_Reference;$tXML_ElementValue)
      DOM SET XML ELEMENT VALUE ($sXML_ElementReference;String(<>Mirror_LTransactionRetries))
      $tXML_ElementValue:="/Mirror/PreferenceSetting/Mirror_LastSegmentNumber"
      $sXML_ElementReference:= DOM Create XML element ($sXML_Reference;$tXML_ElementValue)
      DOM SET XML ELEMENT VALUE ($sXML_ElementReference;<>Mirror_tLastLogNumber)
      $tXML_ElementValue:="/Mirror/PreferenceSetting/Mirror_ErrorSMTPServer"
      $sXML_ElementReference:= DOM Create XML element ($sXML_Reference;$tXML_ElementValue)
      DOM SET XML ELEMENT VALUE ($sXML_ElementReference;<>Mirror_tSMTPServer)
      $tXML_ElementValue:="/Mirror/PreferenceSetting/Mirror_ErrorEMail"
      $sXML_ElementReference:= DOM Create XML element ($sXML_Reference;$tXML_ElementValue)
      DOM SET XML ELEMENT VALUE ($sXML_ElementReference;<>Mirror_tErrorEMailAccount)
      $tXML_ElementValue:="/Mirror/PreferenceSetting/Mirror_ErrorEMailAuthentication"
      $sXML_ElementReference:= DOM Create XML element ($sXML_Reference;$tXML_ElementValue)
      DOM SET XML ELEMENT VALUE ($sXML_ElementReference;<>Mirror_tAuthenticationRequired)
      $tXML_ElementValue:="/Mirror/PreferenceSetting/Mirror_ErrorEMailUsername"
      $sXML_ElementReference:= DOM Create XML element ($sXML_Reference;$tXML_ElementValue)
      DOM SET XML ELEMENT VALUE($sXML_ElementReference;<>Mirror_tAuthenticationUserName)
      $tXML_ElementValue:="/Mirror/PreferenceSetting/Mirror_ErrorEMailPassword"
      $sXML_ElementReference:= DOM Create XML element ($sXML_Reference;$tXML_ElementValue)
      DOM SET XML ELEMENT VALUE($sXML_ElementReference;<>Mirror_tAuthenticationPassword)
      $tXML_ElementValue:="/Mirror/PreferenceSetting/Mirror_MinimumDiskFreeSpace"
      $sXML_ElementReference:= DOM Create XML element ($sXML_Reference;$tXML_ElementValue)
      DOM SET XML ELEMENT VALUE($sXML_ElementReference;
                  String(<>Mirror_LMinimumDiskFreeSpace))
      DOM EXPORT TO FILE($sXML_Reference;$tPreferencesPath)
      DOM CLOSE XML($sXML_Reference)
      CLEAR SEMAPHORE("$ModMirrorPrefs")
      <>Mirror_hTime:=Time($tMirror_Time)
      <>Mirror_hTimeInterval:=Time($tMirror_TimeInterval)
      <>Mirror_hBackupTime:=Time($tMirror_BackupTime)
      <>Mirror_hBackupTimeInterval:=Time($tMirror_BackupTimeInterval)
      <>Mirror_hTransactionDelay:=Time($tMirror_TransactionDelay)
      <>Mirror_LMinimumDiskFreeSpace:=Num($tMinimumDiskFreeSpace)
End case
   `End of method
```

If you are having trouble with all the XML stuff you might want to study a little bit about XML by searching the knowledgebase for technical notes on XML. This is using the DOM (Document Object Model) technique which allows you to build the structure completely in memory.

Mirror preference settings – Client side

The XML preference file resides on the Server machine in its preferences folder. It would be challenging to try to get a client machine to directly manipulate the file and modify its content. It is also awkward to have to go to a Client machine and **Execute on server** a method that would bring up a dialog on the Server machine, then run to that machine to enter the data.

Instead we will use some interprocess communication to accomplish the task by using the **GET PROCESS VARIABLE** and **SET PROCESS VARIABLE** commands. For both these commands you specify a process ID, and the name of the variable in the other process, as well as the local process variable that is to receive or currently contains the value. If the process ID is negative, you are talking about a process ID on the server, not the Client. Finally we can **Execute on server** to have the server update the file.

Everything needed takes place in the form method for the dialog that changes the preference settings

```
If (False)
      ` Form Method: Mirror_Preferences in the [zDialogs] table
      ` 4D Technote on Mirroring a 2004.4 database
      ` Created by: Kent Wilbur
      ` Date: 5/4/2006

      ` Purpose: Handles user interface for setting Mirroring Preferences

   <>Mirror_f_Version2004x4:=True
   <>Mirror_fK_Wilbur:=True

End if

   ` Declare local variables
C_LONGINT($LApplicationType)
C_LONGINT($LFormEvent)
C_LONGINT($LProcessID)
C_TEXT($tTimeIncrement)
C_TEXT($$tSettingsFullPath)

   ` Declare default
$tSettingsFullPath:=Mirror_tMirrorPath ("Mirror")

$LFormEvent:=Form event
$LApplicationType:=Application type

Case of
   : ($LFormEvent=On Load )
      ARRAY TEXT(atMirror_Authentication;2)
      atMirror_Authentication{1}:="No"
      atMirror_Authentication{2}:="Yes"

      Case of
         : ($LApplicationType=4D Client )
            GET PROCESS VARIABLE(LMirrorProcessID;<>tMirror_ServerType;
               <>tMirror_ServerType)
            GET PROCESS VARIABLE(LMirrorProcessID;<>tMirror_DatabaseName;
               <>tMirror_DatabaseName)
            GET PROCESS VARIABLE(LMirrorProcessID;<>tMirror_ServerIPAddress;
```

```
            <>tMirror_ServerIPAddress)
        GET PROCESS VARIABLE (Mirror_LProcessID;<>Mirror_tScheduleMode;
            <>Mirror_tScheduleMode)
        GET PROCESS VARIABLE (Mirror_LProcessID;<>Mirror_hTime;<>Mirror_hTime)
        GET PROCESS VARIABLE (Mirror_LProcessID;<>Mirror_hTimeInterval;
            <>Mirror_hTimeInterval)
        GET PROCESS VARIABLE (Mirror_LProcessID;<>Mirror_tNextTimeIntervalMode;
            <>Mirror_tNextTimeIntervalMode)
        GET PROCESS VARIABLE (Mirror_LProcessID;<>Mirror_tBackupScheduleMode;
            <>Mirror_tBackupScheduleMode)
        GET PROCESS VARIABLE (Mirror_LProcessID;<>Mirror_hBackupTime;
            <>Mirror_hBackupTime)
        GET PROCESS VARIABLE (Mirror_LProcessID;<>Mirror_hBackupTimeInterval;
            <>Mirror_hBackupTimeInterval)
        GET PROCESS VARIABLE (Mirror_LProcessID;<>Mirror_tNextBackupIntervalMode;
            <>Mirror_tNextBackupIntervalMode)
        GET PROCESS VARIABLE (Mirror_LProcessID;<>Mirror_hTransactionDelay;
            <>Mirror_hTransactionDelay)
        GET PROCESS VARIABLE (Mirror_LProcessID;<>Mirror_LTransactionRetries;
            <>Mirror_LTransactionRetries)
        GET PROCESS VARIABLE (Mirror_LProcessID;<>Mirror_tLastLogNumber;
            <>Mirror_tLastLogNumber)
        GET PROCESS VARIABLE (Mirror_LProcessID;<>Mirror_tSMTPServer;
            <>Mirror_tSMTPServer)
        GET PROCESS VARIABLE (Mirror_LProcessID;<>Mirror_tErrorEMailAccount;
            <>Mirror_tErrorEMailAccount)
        GET PROCESS VARIABLE (Mirror_LProcessID;<>Mirror_tAuthenticationRequired;
            <>Mirror_tAuthenticationRequired)
        GET PROCESS VARIABLE (Mirror_LProcessID;<>Mirror_tAuthenticationUserName;
            <>Mirror_tAuthenticationUserName)
        GET PROCESS VARIABLE (Mirror_LProcessID;<>Mirror_tAuthenticationPassword;
            <>Mirror_tAuthenticationPassword)
        GET PROCESS VARIABLE (Mirror_LProcessID;<>Mirror_LMinimumDiskFreeSpace;
            <>Mirror_LMinimumDiskFreeSpace)

    : ($LApplicationType=4th Dimension )
        Mirror_HandleMirrorPreferences ("Load")

    Else
        CANCEL
End case

atMirror_Authentication:=Find in array(atMirror_Authentication;<>tMirror_AuthenticationRequired)
If (atMirror_Authentication=2)    ` authentication required
    SET ENTERABLE(<>tMirror_AuthenticationUserName;True)
    SET ENTERABLE(<>tMirror_AuthenticationPassword;True)
Else
    SET ENTERABLE(<>tMirror_AuthenticationUserName;False)
    SET ENTERABLE(<>tMirror_AuthenticationPassword;False)
End if

$tTimeIncrement:= Substring (String(<>Mirror_hTime;HH MM );1;2)
Mirror_atTimeHr:= Find in array (Mirror_atTimeHr;$tTimeIncrement)
$tTimeIncrement:= Substring (String (<>Mirror_hTime;HH MM );4;2)
Mirror_atTimeMin:= Find in array (Mirror_atTimeMin;$tTimeIncrement)
$tTimeIncrement:= Substring (String (<>Mirror_hTimeInterval;HH MM );1;2)
Mirror_atHr:= Find in array (Mirror_atHr;$tTimeIncrement)
$tTimeIncrement:= Substring (String (<>Mirror_hTimeInterval;HH MM );4;2)
Mirror_atMin:= Find in array (Mirror_atMin;$tTimeIncrement)
Mirror_atMode:= Find in array (Mirror_atMode;<>Mirror_tScheduleMode)

$tTimeIncrement:= Substring (String (<>Mirror_hBackupTime;HH MM );1;2)
```

```
      Mirror_atBackupTimeHr:= Find in array (Mirror_atBackupTimeHr;$tTimeIncrement)
      $tTimeIncrement:= Substring (String (<>Mirror_hBackupTime;HH MM );4;2)
      Mirror_atBackupTimeMin:= Find in array (Mirror_atBackupTimeMin;$tTimeIncrement)
      $tTimeIncrement:= Substring (String (<>Mirror_hBackupTimeInterval;HH MM );1;2)
      Mirror_atBackupHr:= Find in array (Mirror_atBackupHr;$tTimeIncrement)
      $tTimeIncrement:= Substring (String (<>Mirror_hBackupTimeInterval;HH MM );4;2)
      Mirror_atBackupMin:= Find in array (Mirror_atBackupMin;$tTimeIncrement)
      Mirror_atBackupMode:= Find in array (Mirror_atBackupMode;<>Mirror_tBackupScheduleMode)

      $tTimeIncrement:= Substring (String (<>Mirror_hTransactionDelay;HH MM );4;2)
      Mirror_atTransactionMin:= Find in array (Mirror_atTransactionMin;$tTimeIncrement)
      $tTimeIncrement:= Substring (String (<>Mirror_hTransactionDelay;HH MM SS );7;2)
      Mirror_atTransactionSec:= Find in array (Mirror_atTransactionSec;$tTimeIncrement)

      Mirror_Text2Array (<>Mirror_tServerIPAddress;->Mirror_atServers;",")
      Mirror_HandleMirrorPreferences ("SetModePopup")
      Mirror_HandleMirrorPreferences ("SetBackupModePopup")

` Set dialog defaults
      Mirror_ckLaunchProcess:=0
       SET VISIBLE(Mirror_ckLaunchProcess;False)
       SET VISIBLE(*;"MirrorStatus";True)
    Case of
          : (<>tMirror_ServerType="Mirror")
                tMessage:="This server is acting as the Mirrored Server"

          : (Application type#4D Client )
                tMessage:="Mirroring Can Not Run in Single User"

          : (LMirrorProcessID=-1)    ` There is no Mirroring process running on the Server
                SET VISIBLE(Mirror_ckLaunchProcess;True) )    ` Turn on the start process checkbox
                SET VISIBLE(*;"MirrorStatus";False)
                tMessage:=""
          Else
                tMessage:="Mirroring Process is Running on the Server"
      End case


: ($LFormEvent=On Unload )
      ARRAY TEXT(atMirror_Authentication;0)

      If (bOK=1)
            Mirror_HandleMirrorPreferences ("SetNextTimeInterval")
            Mirror_HandleMirrorPreferences ("SetNextBackupInterval")


            Case of
                  : ($LApplicationType=4D Client )
                     SET PROCESS VARIABLE(LMirrorProcessID;<>tMirror_ServerType;
                           <>tMirror_ServerType)
                     SET PROCESS VARIABLE(LMirrorProcessID;<>tMirror_DatabaseName;
                           <>tMirror_DatabaseName)
                     SET PROCESS VARIABLE(LMirrorProcessID;<>tMirror_ServerIPAddress;
                           <>tMirror_ServerIPAddress)
                     SET PROCESS VARIABLE (Mirror_LProcessID;<>Mirror_tScheduleMode;
                           <>Mirror_tScheduleMode)
                     SET PROCESS VARIABLE (Mirror_LProcessID;<>Mirror_hTime;<>Mirror_hTime)
                     SET PROCESS VARIABLE (Mirror_LProcessID;<>Mirror_hTimeInterval;
                           <>Mirror_hTimeInterval)
                     SET PROCESS VARIABLE (Mirror_LProcessID;<>Mirror_tNextTimeIntervalMode;
                           <>Mirror_tNextTimeIntervalMode)
                     SET PROCESS VARIABLE (Mirror_LProcessID;<>Mirror_tBackupScheduleMode;
```

```
                        <>Mirror_tBackupScheduleMode)
                    SET PROCESS VARIABLE (Mirror_LProcessID;<>Mirror_hBackupTime;
                        <>Mirror_hBackupTime)
                    SET PROCESS VARIABLE (Mirror_LProcessID;<>Mirror_hBackupTimeInterval;
                        <>Mirror_hBackupTimeInterval)
                    SET PROCESS VARIABLE (Mirror_LProcessID;<>Mirror_tNextBackupIntervalMode;
                        <>Mirror_tNextBackupIntervalMode)
                    SET PROCESS VARIABLE (Mirror_LProcessID;<>Mirror_hTransactionDelay;
                        <>Mirror_hTransactionDelay)
                    SET PROCESS VARIABLE (Mirror_LProcessID;<>Mirror_LTransactionRetries;
                        <>Mirror_LTransactionRetries)
                    SET PROCESS VARIABLE (Mirror_LProcessID;<>Mirror_tLastLogNumber;
                        <>Mirror_tLastLogNumber)
                    SET PROCESS VARIABLE (Mirror_LProcessID;<>Mirror_tSMTPServer;
                        <>Mirror_tSMTPServer)
                    SET PROCESS VARIABLE (Mirror_LProcessID;<>Mirror_tErrorEMailAccount;
                        <>Mirror_tErrorEMailAccount)
                    SET PROCESS VARIABLE (Mirror_LProcessID;<>Mirror_tAuthenticationRequired;
                        <>Mirror_tAuthenticationRequired)
                    SET PROCESS VARIABLE (Mirror_LProcessID;<>Mirror_tAuthenticationUserName;
                        <>Mirror_tAuthenticationUserName)
                    SET PROCESS VARIABLE (Mirror_LProcessID;<>Mirror_tAuthenticationPassword;
                        <>Mirror_tAuthenticationPassword)
                    SET PROCESS VARIABLE (Mirror_LProcessID;<>Mirror_LMinimumDiskFreeSpace;
                        <>Mirror_LMinimumDiskFreeSpace)
                      If (Mirror_ckLaunchProcess=1)
                          $LProcessID:=Execute on server("Mirror_HandleMirrorPreferences";16000;
                        "$UpdateMirrorPreferences";"Save&LaunchFromClient")
                      Else
                          $LProcessID:=Execute on server("Mirror_HandleMirrorPreferences";16000;
                        "$UpdateMirrorPreferences";"SaveFromClient")
                      End if

                  : ($LApplicationType=4th Dimension )
                      Mirror_HandleMirrorPreferences ("Save")
              End case
          End if
End case
   `End of form method
```