The MiddleMan Module of Frontman

Client/Server Front-end Development

Version: 2.3

MiniSoft, Inc.
1024 First Street
Snohomish, WA 98290
U.S.A.

1-800-682-0200
360-568-6602
Fax: 360-568-2923

MiniSoft Marketing AG
Papiermühle 1
CH-6048 Horw
Switzerland

Phone: 41.41.340.23 20
Fax: 41.41.340.38 66
minisoftag@centralnet.ch

Internet access: sales@minisoft.com
support@minisoft.com
http://www.minisoft.com

Disclaimer

The information contained in this documentation is subject to change without notice.

MiniSoft, Inc. makes no warranty of any kind with regard to this material, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. MiniSoft, Inc. or its agents shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishings, performance, or use of this material.

Printing History:

Vers. 2.1, September 1996
Vers. 2.3, May, 1998

MiddleMan features

MiddleMan is a powerful software development tool designed for building client/server applications in a networked Windows environment. MiddleMan combines the freedom, flexibility, and distributed processing power of the PC with the centralized control and maintenance of host systems.

New for version 2.5:

- Native support of 32-bit Windows.
- ActiveX Objects.

MiddleMan features:

- Direct Image and TurboImage database access. Developers can design applications that read, write, delete, and update databases, KSAM, or MPE files.
- No terminal emulation software is required to use applications developed with MiddleMan, as no session is needed on the HP 3000.
- Includes a server program that can be customized to meet the needs of specific applications. MiddleMan supports the concurrent use of multiple server programs.
- Powerful trace and debugging facility for application testing and error correction.
- A network file transfer facility is included and can be used simultaneously with multiple server programs.
- MiddleMan is accessible from a variety of Windows-based applications, including Visual Basic, Visual C, Excel, WordPerfect, Lotus, and many other applications that support DDE, OLE or ActiveX.
- With MiddleMan you can leverage the power of both the server and the client to easily integrate your applications with the desktop tools already in use.

MiddleMan requires:

- Hewlett Packard's ThinLAN Link on the HP 3000 (product #36923A ThinLAN 3000/iX Network Link) and MPE/iX 4.0, *or*

- MPE/iX 5.0 (which includes ThinLAN).

License agreement

In return for payment of a one-time fee for this software product, the Customer receives from MiniSoft, Inc. a license to use the product subject to these terms and conditions:

- The product may be used on one computer system at a time: i.e., its use is not limited to a particular machine or user but to one machine at a time.
- The software may be copied for archive purposes, program error verification, or to replace defective media. All copies must bear copyright notices contained in the original copy.
- The software may not be installed on a network server for access by more than one personal computer without written permission from MiniSoft, Inc.

Purchase of this license does not transfer any right, title, or interest in the software product to the Customer except as specifically set forth in the License Agreement, and the Customer is on notice that the software product is protected under the copyright laws.

90-Day limited warranty

MiniSoft, Inc. warrants that this product will execute its programming instructions when properly installed on a properly configured personal computer for which it is intended. MiniSoft, Inc. does not warrant that the operation of the software will be uninterrupted or error-free. In the event that this software product fails to execute its programming instructions, Customer's exclusive remedy shall be to return the diskette to MiniSoft, Inc. to obtain replacement. Should MiniSoft, Inc. be unable to replace the diskette within a reasonable amount of time, Customer shall be entitled to a refund of the purchase price upon the return of the product and all copies. MiniSoft, Inc. warrants the diskette upon which this product is recorded to be free from defects in materials and workmanship under normal use for a period of 90 days from the date of purchase. During the warranty period MiniSoft, Inc. will replace diskettes which prove to be defective. Customer's exclusive remedy for any diskette which proves to be defective shall be to return the diskette to MiniSoft, Inc. for replacement.

ANY IMPLIED WARRANTY OF MERCHANTABILITY OR FITNESS IS LIMITED TO THE 90-DAY DURATION OF THIS WRITTEN WARRANTY. Some states or provinces do not allow limitations on how long an implied warranty lasts, so the above limitation or exclusion may not apply to you. This warranty gives you specific rights, and you may also have other rights which vary from state to state or province to province.

LIMITATION OF WARRANTY: MiniSoft, Inc. makes no other warranty expressed or implied with respect to this product. MiniSoft, Inc. specifically disclaims the implied warranty of merchantability and fitness for a particular purpose.

EXCLUSIVE REMEDIES: The remedies herein are Customer's sole and exclusive remedies. In no event shall MiniSoft, Inc. be liable for any direct, indirect, special, incidental, or consequential damages, whether based on contract, tort, or any other legal theory.

Table of Contents

Chapter 1
Introduction

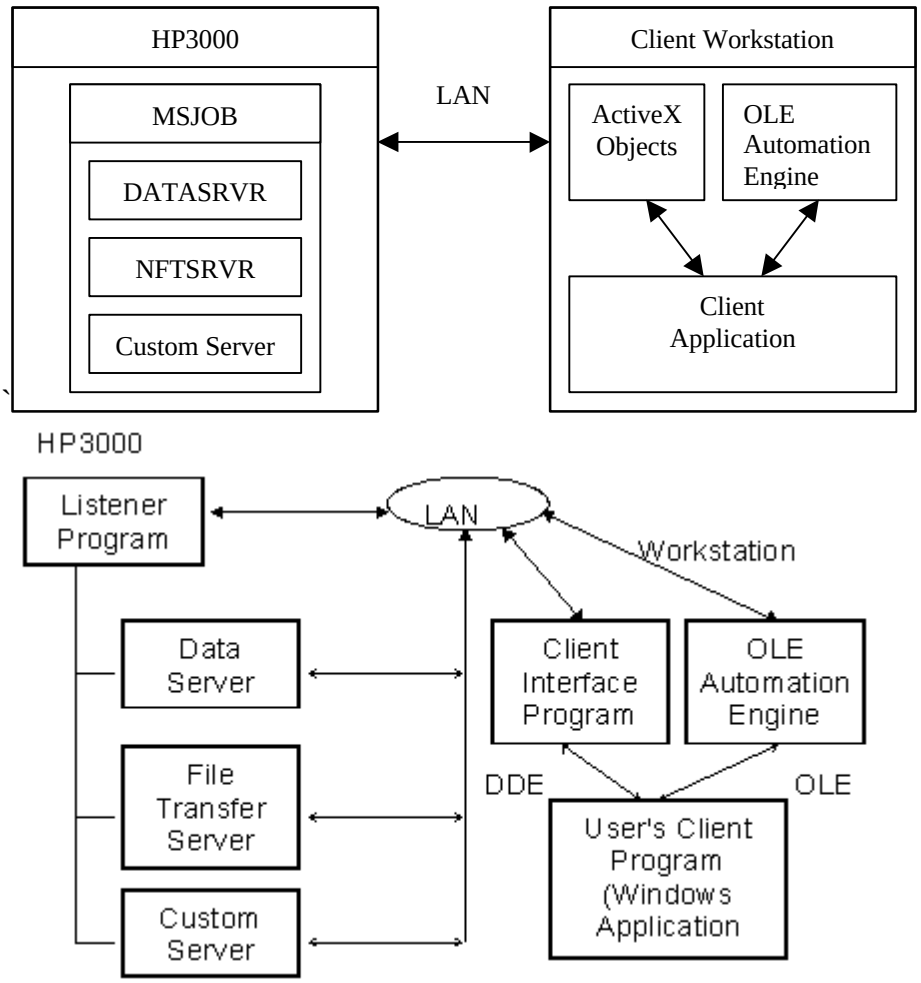
MiddleMan

MiddleMan lets you build client/server applications in a networked Windows environment.

MiddleMan consists of *client* and *server* software:

- The client software resides on the workstation (PC), where it acts as an interface between the server software and a Windows application running on the workstation. This Windows application is referred to in this manual as the *client program.* Chapter 2 discusses client software.
- The server software resides on the HP 3000, where it handles network communication and MPE functions. Chapter 3 discusses server software.

Here is a diagram of the MiddleMan software components and how they interact with programs on the HP 3000 and the PC workstation:



Terminology

Client

The *client* is an application on the PC that the user interacts with. The client accesses the server through the engine.

Engine

The *engine* is an OLE Automation server or ActiveX object on the PC that provides communication between the client and server applications.

Server

The *server* is an application running on the host that communicates with the engine on a PC.

Client software

Client access options

The client software can use either OLE (Object Linking and Embedding) automation or an ActiveX object as its main interface to MiddleMan. Since ActiveX is newer, faster, and smaller than OLE, using an ActiveX object to MiddleMan is the preferred option. OLE should be used for 16-bit applications and where ActiveX is not supported. DDE should be used only if the client software being used does not support OLE automation or ActiveX.

ActiveX objects

The two ActiveX objects included in MiddleMan are Data Access, and Streams Access.

- The Data Access Engine provides read/write data access to TurboImage databases, KSAM files, and MPE files.

- The Streams Access Engine allows programmers to define their own interface where an existing interface does not provide sufficient capability.

You can use any OCX-compatible language, application, or utility with these engines. Examples include Active Server Pages, Visual Basic, Excel, Visual C++, PowerBuilder, and Access.

OLE automation engines

The four OLE automation engines included in MiddleMan are Data Access, File Transfer, Streams Access, and File Transfer Protocol (FTP).

- The Data Access Engine provides read/write data access to TurboImage databases, KSAM files, and MPE files.
- The File Transfer Engine provides for file transfers between the HP 3000 and your PC application.
- The Streams Access Engine allows programmers to define their own interface where an existing interface does not provide sufficient capability.
- The File Transfer Protocol (FTP) Engine provides for file transfers between any FTP compliant host and your PC application.

You can use any OLE2-compatible language, application, or utility with these engines. Examples include Visual Basic, Excel, Visual C++, PowerBuilder, MicroFocus Cobol, and Access.

Client Interface Program (CIP)

If your client software cannot use the OLE automation interface to MiddleMan, then you must use the DDE interface. This is done through the Client Interface Program (CIP). This program acts as a bridge between the client program (Windows application) and the MiddleMan server programs. Requests from the client are translated if necessary, then transmitted to the server. Responses from the server are received by the

MiddleMan CIP and then organized for receipt by the client program. The MiddleMan CIP also handles file transfer requests and validates data for input to the data server.

The interface between the MiddleMan CIP and the client program is through the Windows *Dynamic Data Exchange* (DDE) interprocess communication standard. The client program sends commands and data to the MiddleMan CIP with DDE-EXECUTE and DDE-POKE commands, and receives data from the MiddleMan CIP with DDE-REQUEST commands.

Using this interface, any Windows application that can communicate using the DDE standard can be a client to the HP 3000 server.

The MiddleMan CIP also includes a trace facility, interactive network file transfer, and configuration options.

Server software

MiddleMan server software consists of server programs and a network interface library. A client can connect to one server, multiple copies of the same server, or multiple copies of multiple servers. This makes all the functionality of all the servers available to your client application. The primary server programs are:

- The Master Server (MSSERVER)
- The Listener Server (SERVER)

The secondary server programs are used by each client connection to the host system. The Listener Server creates these servers.

- The Data Access Server (DATASRVR)
- The File Transfer Server (NFTSRVR)
- Your own custom server program, created with the network interface library (NIFOBJ).

Master Server

The Master Server is the application that starts all other server programs. Based upon your licensing information and the parameters supplied in its startup job, it creates listener servers to listen for client connection requests. It also interacts with the Server Console client program and the MSJOB Command Utility.

Listener Server

The Listener Server waits for connection requests from client programs. When a request is made that matches what it is listening for, it spawns a copy of the appropriate server program to respond to the client. In this way, each client program has its own copy of the server program with which to converse. Each type of server program will have a listener to listen for connection requests for it.

For more information on the Listener Server, see *Listener server*, on page 3-5.

Data Access Server

The Data Access Server handles input/output for TurboImage databases, KSAM files, and MPE files. Client programs can connect to this server to access data in these data sources directly.

For more information on the Data Access Server program, see *Data access* server, on page 3-6.

File Transfer Server

The File Transfer Server handles file transfer between the HP 3000 and the PC. This is the preferred way to transmit large amounts of data. This server can also execute MPE commands, programs, and command files.

For more information on the File Transfer Server program, see *File transfer server*, on page 3-6.

Stream Server

The Stream Server is a user-written program that is called by the listener and uses the network interface library. The network interface library is a set of functions employed by a custom server program to enable it to communicate with a client program. The library is accessible from any program language on the HP 3000, as well as from many 4GL-type development systems.

For detailed information on the network interface library, see *Stream server*, on page 3-6.

File Transfer Protocol

The FTP client will communicate with any ftp server. The FTP listener, daemon, or JOB must be active prior to making any connections. Host types are available for connection to most common systems.

Requirements

Hardware

MiddleMan requires this host and workstation hardware:

Host

- HP 3000 with LAN interface

PC workstation

- 386 processor, 4MB RAM, and a LAN interface card

Software

MiddleMan requires this host and workstation software:

Host

- MPE/iX 4.0 and ThinLAN Link *or*
- MPE/iX 5.0 (which includes ThinLAN Link)
- Any TCP/IP-based FTP server (for the FTP client only)

PC workstation

- Winsock-compliant TCP/IP stack
- Windows, v. 3.1/3.11 *or*
  Windows for Workgroups, v.3.11 *or*
  Windows 95, *or*
  Windows NT
- Windows application that supports OLE, *or*
  Windows programming language that supports OLE, *or*
  Windows application or utility that supports DDE or OLE.

Installation

Installing the server software on the HP 3000

Follow these steps to install the MiddleMan server software on the HP 3000:

1. Insert the tape into a tape drive.

2. Log on to the HP 3000 as MANAGER.SYS and restore the installation job:

   ```
   :HELLO MANAGER.SYS

   :FILE T;DEV=TAPE

   :RESTORE *T;@.PUB.SYS;SHOW
   ```

3. Remount the tape and run the installation job.

   ```
   :INSTMS
   ```

4. When the message *Installation Complete* appears, start the MiddleMan server:

   ```
   :STREAM MSJOB.MM.MINISOFT
   ```

See chapter 3, *Server Software,* for more information on the server software.

MiddleMan is now installed and running on the HP 3000.

Installing the client software on the PC

The server software must be installed on the HP 3000 before the client software can be installed on the PC. All MiniSoft client software is installed with an Installer program. If the Installer program has not been installed on your PC start at step 1, else start at step 3.

1. Insert the MiddleMan diskette in a drive. In Windows 3.1 Program Manager, select Run from the File menu, or in Windows95, select Run from the Start button. In the Command Line text box, type:

   ***drive*:\setup.exe**

   where *drive* is the drive letter where you inserted the diskette (usually *a* or *b*).

2. Click OK and follow the installation instructions.

   A client installer Start Menu item or icon will be created.

3. Start the Installer icon or Start Menu item.

4. Enter valid logon identification for your HP 3000.

5. Select MiddleMan from the list by clicking on it.

6. Click Install and follow the installation instructions.

7. Steps 3 through 5 may be repeated for additional MiniSoft products.

MiddleMan is now installed on the PC.

Notes:

Chapter 2
Client Software

MiddleMan provides access to the host computer's server programs by presenting an API (Application Program Interface) to client programs using OLE Automation. Any client may create an object of the needed type and then use the object's properties and methods to communicate with the server program.

Data access engine

The client program uses the data access engine objects to communicate with the data server program on the host. This gives the client read/write data access to TurboImage databases, KSAM files, and MPE files.

Object creation

In order to access the data access engine from a client program, the client program must create the *session object* of the data access engine as follows:

Examples

Visual Basic:

```
Dim DASession As Object
  ...
Set DASession = CreateObject("MdmDA.Session")
```

Visual C++:

```
CreateDispatch("MdmDA.Session");
```

Object destruction

When the client program wishes to stop using the data access engine, it must delete the *session object* as follows:

Examples

Visual Basic:

```
Set DASession = Nothing
```

Visual C++:

```
ReleaseDispatch( );
```

Note: A more complete example of using the data access engine from Visual Basic is shown by the CUSTINFO.MAK project in the VB subdirectory of the MiddleMan installation directory.

Session object

Once the *session object* has been created, the client can use its properties and methods to access the host's data server program. Typically the

properties specifying the host's IP address, TCP/IP port number, and login information are set, then the Connect method is called to connect the host's data server program.

Properties

AccountPassword, type String, write access
The string used for the account password when connecting to the host's data server program.

ConnectStatus, type Boolean, read access
Contains True if the last connection attempt was successful, else False.

ErrorMessage, type String, read access
Contains the last error message generated.

ErrorNumber, type integer, read access
Contains the last error number generated.

GroupPassword, type String, write access
The string used for the group password when connecting to the host's data server program.

HostAddress, type String, write access
The IP address used when attempting to connect to the host.

JobName, type String, write access
The string used for the job name when connecting to the host's data server program.

LoginAccount, type String, write access
The string used for the account name when connecting to the host's data server program.

LoginGroup, type String, write access
The string used for the group name when connecting to the host's data server program.

LoginStatus, type Boolean, read access
Contains True if the login information supplied when connecting to the host's data server program was valid, else False.

LoginUser, type String, write access
The string inserted for the user name when connecting to the host's data server program.

Port, type integer, write access
A number that uniquely identifies the server program on the host that the client wishes to connect to. This number may be any number between 1 and 32767, though a number above 30000 is recommended. The number for each type of server is specified in the host software's startup job.

TraceLevel, type integer, write access
A number representing the tracing options activated. It is the sum of the individual values of each option. The options are: 1 - trace errors, 2 - trace network traffic, 4 - trace server transactions, and 8 - trace client transactions. For example, a value of 5 would trace errors and server transactions.

UserPassword, type String, write access
The string used for the user password when connecting to the host's data server program.

VendorId, type String, write access
A six-digit code to verify access to the host server program. MiddleMan's host data server requires the code 000008, which is the default. Third-party software developers who wish to have their own code to verify access for clients they develop may obtain a unique one from MiniSoft.

Methods

database object AddImageDBRef (String DatabaseName)

This method creates an object to reference a TurboImage database the client program wishes to access. The DatabaseName parameter specifies the database to access. Once a database object is created, the client may open and access the database using the properties and methods of the database object.

ksam file object AddKsamFileRef (String FileName)

This method creates an object to reference a KSAM file the client program wishes to access. The FileName parameter specifies the file to access. Once a ksam file object is created, the client may open and access the file using the properties and methods of the ksam file object.

MPE file object AddMpeFileRef (String FileName)

This method creates an object to reference a MPE file the client program wishes to access. The FileName parameter specifies the file to access. Once a MPE file object is created, the client may open and access the file using the properties and methods of the MPE file object.

Boolean Connect ()

This method attempts to connect to the host's data server program at the IP address specified by the HostAddress property. The Port property is used to identify the requested host program. Connect sets the ConnectStatus property and returns True if the connection request succeeded, otherwise it returns False.

void Disconnect ()

This method disconnects from the host's data server program.

integer HostCommand (String CommandString)

This method executes a MPE command or MPE command file. The CommandString parameter specifies the command or command file to be executed. A return of 0 indicates the command was executed successfully, while a return of 1 indicates the command failed.

Boolean SaveConfig (String FileName)

This method saves the writeable session object properties to the named file. Any database object, ksam file object, or MPE file object that was created will also be saved. This file can later be used to initialize a session object when created.

Boolean LoadConfig (String FileName)

This method loads the session object properties from the named file. Any database object, ksam file object, or MPE file object that was saved in the file will also be loaded. A return of True indicates the properties were loaded successfully.

Boolean SetSessionConfig ()

This method displays the Session Configuration dialog of the Data Access Engine. This dialog allows the user to set these properties: HostAddress, Port, LoginUser, UserPassword, LoginGroup, GroupPassword, LoginAccount, AccountPassword, and JobName. A return of True indicates the user clicked OK to exit the dialog; a return of False indicates the user clicked Cancel to exit the dialog.

Boolean SetTraceConfig ()

This method displays the Trace Options dialog of the Data Access Engine. This dialog allows the user to set tracing options used for debugging. A return of True indicates the user clicked OK to exit the dialog; a return of False indicates the user clicked Cancel to exit the dialog.

Obsolete methods

These methods are from pre-2.1.a versions of MiddleMan. They will continue to work, but should not be used for new code.

DBObject Define (String DBName, 0)

Replace with the AddImageDBRef method described earlier in this chapter.

FileObject Define (String FileName, 1)

Replace with the AddKsamFileRef or AddMpeFileRef method described earlier in this chapter.

Examples

Visual Basic:

```
Dim DASession As Object
Dim Demo1 As Object
Dim Customer As Object
  ...
Sub Form_Load ( )
 Dim Password As String

 ' Create session object
 Set DASession = CreateObject("MdmDA.Session")
 ' Set all tracing options on
 DASession.TraceLevel = 15
 ' Set the Host IP address and the Data
 ' Server TCP-IP port number. These values are
 ' dependent on your host configuration and
 ' the JOB used to start the Data Server program.
 DASession.HostAddress = "192.6.1.2"
 DASession.Port = 30002
 ' Set the login information before connecting.
 ' This must be done if the Data Server program was
 ' started in "secure" mode.
 DASession.JobName = "CUSTINQ"
 DASession.LoginUser = "MGR"
 DASession.LoginAccount = "MINISOFT"
 Do
  'Prompt for and set user password.
  Password = InputBox$("Enter Login Password", "Order
             Inquiry")
  If Len (Password) = 0 Then Exit Do
  DASession.UserPassword = Password
  ' Connect to the Data Server.
  DASession.Connect
  ' Check to see that the login was successful.
```

```
Loop While DASession.LoginStatus = False
' Check to see if the connection was established.
If DASession.ConnectStatus = True Then
 ' Define the databases and datasets that will be
 ' used.
 Set Demo1 = DASession.AddImageDBRef ("DEMO1")
 Set Customer = Demo1.AddDatasetRef ("CUSTOMER")
 ' Open the database
 If Demo1.Open ("DEMO1", "EXE", "MINISOFT", 5,
                "DEMO1") Then
  ' Initialization complete.
  Exit Sub
 Else
  ' DEMO1 Database Open Error.
  MsgBox Demo1.ErrorMessage, MB_ICONSTOP,
         "Order Inquiry"
 End If

 ' Delete the database object previously created.
 Set Demo1 = Nothing
 Set Customer = Nothing

 ' Disconnect from the Data Server.
 DASession.Disconnect
End If


' Delete the session object.
Set DASession = Nothing

' Exit the client program.
End
End Sub
```

Database object

Database objects are created by the AddImageDBRef method of the session object. A database object must be created for each database the

client wishes to access. Once a database object has been created, operations on the database can be performed using the object's methods.

Properties

DatasetList, type String, read access

Contains a comma-delimited list of the dataset names and types of all the datasets within this database. The format of the list is "<dataset name>,<dataset type>,...". The dataset types are M for manual masters, A for automatic masters, and D for details. This list will not be valid unless the database is opened using the Open method or the database object was loaded from a file.

DBStatus (integer Index), type integer, read access

Contains the value of a specific word of the Status array returned from the last intrinsic call to the database. The specific word of the Status array it contains is specified by the Index parameter. Valid values for the Index parameter are 1 through 10.

DBStatusDW (integer Index), type long integer, read access

Contains the value of a specific double word of the Status array returned from the last intrinsic call to the database. The specific double word of the Status array it contains is specified by the Index parameter. Valid values for the Index parameter are 1 through 5.

DictionaryMode, type integer, read/write access

This property specifies how to obtain the information about the database.

- The default value of 1 indicates that the information about the database should be obtained dynamically from the database when it is opened.
- A value of 0 indicates that the client program will supply the information about the database.

See *Using manual dictionary mode* on page 2-73.

ErrorMessage, type String, read access
Contains the last error message generated.

ErrorNumber, type integer, read access
Contains the last error number generated.

TpiEnabled, type Boolean, read/write access
After the database is opened, this indicates whether the database supports TPI (Third Party Indexing) or not. If the database supports TPI, additional modes for retrieving data are enabled. We recommend that the value of this property not be changed.

Methods

void AddDataset (String DatasetName, String DatasetType)
This method should only be used when the Dictionary property is set to 0. This allows the client program to provide the information about the database, instead of getting it from the data server.

dataset object AddDatasetRef (String DatasetName)
This method creates an object to reference a dataset the client program wishes to access. Once a dataset object is created, the client may access the dataset using the properties and methods of the dataset object.

Boolean DBBegin (DatabaseObject Database, Integer Mode, String Text)
This method directly calls TurboImage's DBBegin intrinsic. It designates the beginning of a sequence of TurboImage/XL procedure calls regarded as a static or multiple database transaction (based on the Mode) for the purposes of logging and recovery.

- When the Mode parameter is equal to 1, the transaction is a static, single database transaction. In this case the Database parameter should be a single database object.

- When the Mode parameter is equal to 3 or 4, the transaction is a multiple database transaction. In this case the Database parameter should be an array of database objects.
- The Text parameter can be used to log user information to the log file.

The DBBegin method is used in conjunction with the DBEnd method to begin and end a static or multiple database transaction. For more information on the DBBegin intrinsic, refer to the TurboImage/XL manual from Hewlett-Packard.

Boolean DBClose (String DatasetName, Integer Mode)

This method directly calls TurboImage's DBClose intrinsic. It terminates access to a database or a data set, or rewinds a data set.

- When the Mode parameter is equal to 1, all access to the database is terminated. In this case the DatasetName parameter is ignored.
- When the Mode parameter is equal to 2, the dataset referenced by the DatasetName parameter is closed.
- When the Mode parameter is equal to 3, the dataset referenced by the DatasetName parameter is reinitialized.

For more information on the DBClose intrinsic, refer to the TurboImage/XL manual from Hewlett-Packard.

Boolean DBControl (Integer Mode)

This method directly calls TurboImage's DBControl intrinsic. It sets or resets various options for accessing the database. For more information on the DBControl intrinsic, refer to the TurboImage/XL manual from Hewlett-Packard.

Boolean DBEnd (DatabaseObject Database, Integer Mode, String Text)

This method directly calls TurboImage's DBEnd intrinsic. It designates the end of a sequence of TurboImage/XL procedure calls regarded as a

static or multiple database transaction (based on the Mode) for the purposes of logging and recovery.

- When the Mode parameter is equal to 1 or 2, the transaction is a static, single database transaction. In this case the Database parameter should be a single database object.
- When the Mode parameter is equal to 3 or 4, the transaction is a multiple database transaction. In this case the Database parameter should be an array of database objects.
- The Text parameter can be used to log user information to the log file.

The DBEnd method is used in conjunction with the DBBegin method to begin and end a static or multiple database transaction. For more information on the DBEnd intrinsic, refer to the TurboImage/XL manual from Hewlett-Packard.

String DBError ()

This method directly calls TurboImage's DBError intrinsic. It returns an error message for the last TurboImage intrinsic called. For more information on the DBError intrinsic, refer to the TurboImage/XL manual from Hewlett-Packard.

String DBInfo (String Qualifier, Integer Mode)

This method directly calls TurboImage's DBInfo intrinsic. It provides information about the database being accessed. The contents of the Qualifier parameter depend on the value of the Mode parameter:

Mode	Qualifier	Output
101	Data item name or number	Data item number; positive indicates read access, negative indicates read/write access
102	Data item name or number	Data item name,type,sub-item length,sub-item count

103	ignored	Number of data items, data item number *1*,...,data item number *n*
104	Dataset name or number	Number of data items, data item number *1*,...,data item number *n*
201	Dataset name or number	Dataset number; positive indicates read access, negative indicates read/write access
202	Dataset name or number	Dataset name,type,entry length,blocking factor,entries in set,capacity of set
203	ignored	Number of datasets, dataset number *1*,...,dataset number *n*
204	Data item name or number	Number of datasets, dataset number *1*,...,dataset number *n*
205	Dataset name or number	Dataset name,type,entry length,blocking factor,entries in set,capacity of set,high water mark,maximum capacity,initial capacity,incremental number of entries,incremental percent,dynamic expansion flag
301	Dataset name or number	Path count,dataset number of path *1*,search item number of path *1*,sort item number of path *1*,...,dataset number of path *n*,search item number of path *n*,sort item number of path *n*
302	Master dataset name or number	Key item number
302	Detail dataset name or number	Search item number,master dataset number
401	ignored	Log identifier name,database log flag,user log flag, transaction flag,user transaction number
402	ignored	ILR log flag,date,time
403	ignored	Log identifier name,database log flag,user log flag, transaction flag,user transaction number,XM log set size,XM log set type,database attach flag,dynamic transaction flag,XM log set name

404	ignored	Database log flag,user log flag,roll-back log flag,ILR log flag, MUSTRECOVER flag,database remote flag,logical transaction flag,log identifier name,log index,multiple database transaction ID,number of databases,database IDs,...
501	ignored	Subsystem access
502	ignored	Critical item update flag,current setting for accessor
801	ignored	TPI product name,version,date installed,time installed
802	ignored	number of external files,number of internal files
803	ignored	TPI enabled flag
811	Data item name or number,dataset name or number	Data item number; positive indicates read access, negative indicates read/write access
812	Data item name or number,dataset name or number	Data item name,type,sub-item length,sub-item count
813	ignored	Number of data items,data item number *1*,...,data item number *n*
814	Dataset name or number	Number of data items,data item number *1*,...,data item number *n*
821	Third-party key name or number	Number of datasets, dataset number *1*,...,dataset number *n*
831	Dataset name or number	Number of keys,item number of key *1*,...,item number of key *n*
832	Dataset name or number	Number of keys,item number of key *1*,...,item number of key *n*

833	Dataset name or number,data item number of key	Third-party key number,type,key length,set number,item number,length,use,number of components,item number of component *1*,offset of component *1*,length of component *1*,type of component *1*,sub-item length of component *1*,sub-item count of component *1*,..., item number of component *n*,offset of component *n*,length of component *n*,type of component *n*,sub-item length of component *n*,sub-item count of component *n*
901	ignored	Language ID

For more information on the DBInfo intrinsic, refer to the TurboImage/XL manual from Hewlett-Packard.

Boolean DBLock (String Qualifier, Integer Mode)

This method directly calls TurboImage's DBLock intrinsic. The contents of the Qualifier parameter depend on the value of the Mode parameter.

Mode	**Qualifier**
1 or 2	ignored
3 or 4	Dataset name or number
5 or 6	Number of lock descriptors,dataset name or number of descriptor *1*,data item name or number of descriptor *1*,relational operator of descriptor *1*,item value of descriptor *1*,..., dataset name or number of descriptor *n*,data item name or number of desciptor *n*,relational operator of descriptor *n*,item value of descriptor *n*

For more information on the DBLock intrinsic, refer to the TurboImage/XL manual from Hewlett-Packard.

Boolean DBMemo (String Text)

This method directly calls TurboImage's DBMemo intrinsic. It writes the data in the Text parameter to the log file. For more information on

the DBMemo intrinsic, refer to the TurboImage/XL manual from Hewlett-Packard.

Boolean DBOpen (String DatabaseName, String Password, Integer Mode)

This method directly calls TurboImage's DBOpen intrinsic. It opens a database.

- The DatabaseName parameter specifies the database to open. It may be fully qualified.
- The Password parameter specifies the database password to use when opening the database.
- The OpenMode parameter specifies the mode to open the database with. Valid values are 1 through 8.

A return of True indicates the database was opened successfully. For more information on the DBOpen intrinsic, refer to the TurboImage/XL manual from Hewlett-Packard.

Boolean DBUnlock ()

This method directly calls TurboImage's DBUnlock intrinsic. It removes the lock applied by a previous DBLock method. For more information on the DBUnlock intrinsic, refer to the TurboImage/XL manual from Hewlett-Packard.

Boolean DBXBegin (Integer Mode, String Text)

This method directly calls TurboImage's DBXBegin intrinsic. It designates the beginning of a sequence of TurboImage/XL procedure calls that are to be regarded as a dynamic transaction for the purposes of logging and dynamic roll-back recovery.

- The Text parameter can be used to log user information to the log file.
- The Mode parameter must be set to 1.

The DBXBegin method is used in conjunction with the DBXEnd method to begin and end a dynamic transaction. For more information on the DBXBegin intrinsic, refer to the TurboImage/XL manual from Hewlett-Packard.

Boolean DBXEnd (Integer Mode, String Text)

This method directly calls TurboImage's DBXEnd intrinsic. It designates the end of a sequence of TurboImage/XL procedure calls that are to be regarded as a dynamic transaction for the purposes of logging and dynamic roll-back recovery.

- The Text parameter can be used to log user information to the log file.
- The Mode parameter must be set to 1 or 2.

The DBXEnd method is used in conjunction with the DBXBegin method to begin and end a dynamic transaction. For more information on the DBXEnd intrinsic, refer to the TurboImage/XL manual from Hewlett-Packard.

Boolean DBXUndo (Integer Mode, String Text)

This method directly calls TurboImage's DBXUndo intrinsic. It rolls back the active sequence of TurboImage/XL procedure calls which are considered a dynamic transaction.

- The Text parameter can be used to log user information to the log file.
- The Mode parameter must be set to 1.

For more information on the DBXUndo intrinsic, refer to the TurboImage/XL manual from Hewlett-Packard.

Boolean Close ()

This method closes an open database. A return of True indicates the database was successfully closed.

Boolean Open (String DatabaseName, String OpenGroup, String OpenAccount, integer OpenMode, String Password)
This method opens a database.

- The DatabaseName parameter specifies the database to open.
- The OpenGroup parameter specifies the group that the database is located in.
- The OpenAccount parameter specifies the account that the database is located in.
- The OpenMode parameter specifies the mode to open the database with. Valid values are 1 through 8.
- The Password parameter specifies the database password to use when opening the database.

A return of True indicates the database was opened successfully.

Obsolete methods

These methods are from pre-2.1.a versions of MiddleMan. They will continue to work, but should not be used for new code.

Boolean CloseDB ()
This method operates exactly like the Close method. It is only needed for use in Visual Basic 3.0, where the word Close is a reserved word.

dataset object Define (String DatasetName)
Replace with the **AddDatasetRef** method as shown above.

Dataset object

Dataset objects are created by the AddDatasetRef method of a database object. A dataset object must be created for each dataset the client wishes

to access. Once a dataset object has been created, operations on the dataset can be performed using the object's methods.

Properties

Delimiter, type String, read/write access
Sets the delimiter to use between data item values when data is stored and retrieved with the ItemBuffer property. If the Delimiter property is empty, data is formatted to a fixed length. The default for the Delimiter is a comma.

ErrorMessage, type String, read access
Contains the last error message generated.

ErrorNumber, type integer, read access
Contains the last error number generated.

IsKey (String ItemName), type Boolean, read access
Indicates if a data item is a key or not. The ItemName parameter specifies the name of the data item.

Item (String ItemName), type String, read/write access
This property is used to read or write the value of a data item within the dataset. The ItemName parameter specifies the name of the data item to read or write. If the Delimiter property is empty, the data is formatted to a fixed length. The Item property is the dataset object's default property, so it is not necessary to use the property's name in Visual Basic. If your programming language does not allow this property to be assigned a value, an alternate method is WriteItem.

ItemBuffer (String RWItemList), type String, read/write access
This property is used to store and retrieve the values of the data items listed in the RWItemList parameter. If the Delimiter property is not empty, data item values are delimited by the character in the Delimiter

property. If the Delimiter property is empty, data is formatted to a fixed length.

ItemFormatLength (String ItemName), type integer, read access
Contains the length in characters used to format a data item value. The data item is specified by the ItemName parameter. Data item values are formatted to this length when the Delimiter property is empty.

ItemLength (String ItemName), type integer, read access
Contains the data item length as specified in the database schema or by the AddItem method. The data item is specified in the ItemName parameter. The value of the ItemLength property depends on the ItemType property. Both properties are used to determine the internal storage format for the item. See the AddItem method for more information.

ItemList, type String, read access
Contains a comma-delimited list of all the items of the dataset.

ItemSigned (String ItemName), type Boolean, read/write access
Indicates if data item of type P or Z is signed or not. The data item is specified by the ItemName parameter. This property is set automatically when data is read from a dataset.

If you are not reading data before writing data to a dataset, you should set this property to make sure that data of type P or Z is written correctly. Also, when finding or retrieving data using the DBFind, Find, mode 7 DBGet, ReadByKey, ReadFirst, or ReadLast methods, the ItemSigned property of the key item (if it is type P or Z) should be set to reflect the way data is stored in the key.

The default value for this property is false.

ItemType (String ItemName), type String, read access
Contains the TurboImage data item type as specified in the database schema or by the AddItem method. The data item is specified in the

ItemName parameter. The value of the ItemType property determines the meaning of the ItemLength property. Both properties are used to determine the internal storage format for the item. See the AddItem method for more information.

KeyType (String ItemName), type String, read access
Contains the type of key for data item that are keys. The data item is specified in the ItemName parameter. Types are I for a TurboImage key, G for a third-party generic key, M for a third-party multiple key, and B for a third-party generic and multiple key.

LinkDataset, type String, read access
For a *master* dataset, the LinkDataset property contains a list of the detail datasets this master dataset links to. For a *detail* dataset, the LinkDataset property contains a list of the master datasets this detail dataset links to.

RecordNumber, type long, read access
Contains the record number of the record last read, updated, or written.

Type, type String, read access
Contains the type of the dataset. Values are M for a manual master dataset, A for an automatic master dataset, and D for a detail dataset.

SubItems (String ItemName), type integer, read access
Contains the data item sub-item count as specified in the database schema or by the AddItem method. The data item is specified in the ItemName parameter.

Methods

AddItem (String ItemName, String Type, integer Length, integer Offset, integer SubItems, Boolean Signed, Boolean Key, String KeyType, Boolean WriteAccess)

This method is used to add a data item. This allows you to redefine the record layout of the dataset.

- The ItemName parameter specifies the name of the new data item.
- The Type parameter specifies the TurboImage type of the new data item. Valid types are I, J, K, E, R, U, X, Z, and P.
- The Length parameter specifies the TurboImage length of the new data item.
- The SubItems parameters specify the number of sub-items contained in the new data item.
- The Signed parameter specifies if the new data item will contain a sign (type P or Z only).
- The Key parameter specifies if the new data item is a key.
- The KeyType parameter specifies the key type of the new data item.
- The WriteAccess parameter specifies if there is write access to the new data item.

The following table describes how the Type and Length parameters determine the internal storage format of the item:

Length Unit	**Type**	**Type description**
16 bits	E	A real (IEEE floating point) number.
16 bits	I and J	A signed binary integer in 2's complement form.
16 bits	K	An absolute binary quantity (no negative values).
16 bits	R	A real (HP 3000 floating point) number.
4 bits	P	A packed decimal number.
8 bits	X	An unrestricted ASCII character string.
8 bits	U	An ASCII character string containing no lowercase alphabetic characters.

8 bits	Z	A zoned decimal format number.

For more information on TurboImage data types and lengths, refer to the TurboImage/XL manual from Hewlett-Packard.

Boolean DBDelete (integer Mode)

This method directly calls TurboImage's DBDelete intrinsic. It deletes the current entry from a manual master or detail data set. The Mode parameter should be set to 1. A return of True indicates the record was successfully deleted. For more information on the DBDelete intrinsic, refer to the TurboImage/XL manual from Hewlett-Packard.

Boolean DBFind (integer Mode, String KeyName, String KeyValue)

This method directly calls TurboImage's DBFind intrinsic. It finds the chain head of a TurboImage detail dataset chain. It can be used only on an Image detail dataset.

- The KeyName parameter specifies the name of the key item used to search on.
- The KeyValue parameter specifies the value of the key.
- The Mode parameter must be set to 1 unless Third Party Indexing is enabled for the database. If Third Party Indexing is enabled for the database, refer to the vendor's documentation for additional modes.

A return of True indicates a chain for the supplied key value was found. For more information on the DBFind intrinsic, refer to the TurboImage/XL manual from Hewlett-Packard.

Boolean DBGet (integer Mode, String RWItemList, String Argument)

This method directly calls TurboImage's DBGet intrinsic. It reads a record from a dataset.

- Valid values for the Mode parameter are 1 through 8.

- The RWItemList parameter is a comma-delimited list of the items to read.
- The Argument parameter is used only for mode 7.

A return of True indicates that a record was read successfully. For more information on the DBGet intrinsic, refer to the TurboImage/XL manual from Hewlett-Packard.

Boolean DBPut (integer Mode, String RWItemList)

This method directly calls TurboImage's DBPut intrinsic. It adds a new record to a manual master or detail dataset.

- The Mode parameter must be set to 1.
- The RWItemList parameter is a comma-delimited list of the items to write.

A return of True indicates the record was written successfully. For more information on the DBPut intrinsic, refer to the TurboImage/XL manual from Hewlett-Packard.

Boolean DBUpdate (integer Mode, String RWItemList)

This method directly calls TurboImage's DBUpdate intrinsic. It modifies the values of data items in the current record of the dataset.

- The Mode parameter must be set to 1.
- The RWItemList parameter is a comma-delimited list of the items to update.

A return of True indicates the record was updated successfully. For more information on the DBUpdate intrinsic, refer to the TurboImage/XL manual from Hewlett-Packard.

Boolean Delete ()

This method deletes the current record from the dataset. A return of True indicates the record was successfully deleted.

Boolean Find (String KeyName, String KeyValue)

This method finds the chain head of an Image detail dataset chain. It can be used only on an Image detail dataset.

- The KeyName parameter specifies the name of the key item used to search on.
- The KeyValue parameter specifies the value of the key.

A return of True indicates a chain for the supplied key value was found.

Boolean ReadByKey (String KeyName, String KeyValue, String RWItemList)

This method reads a record from an Image master dataset by key. It can be used only on an Image master dataset.

- The KeyName parameter specifies the name of the key item used to search on.
- The KeyValue parameter specifies the value of the key.
- The RWItemList parameter is a comma-delimited list of the items to read.

A return of True indicates a record was found and read.

Boolean ReadDirect (long RecordNumber, String RWItemList)

This method reads a record from a dataset by record number.

- The RecordNumber parameter specifies the record number of the record to read.
- The RWItemList parameter is a comma-delimited list of the items to read.

A return of True indicates a record was found and read.

Boolean ReadFirst (String KeyName, String KeyValue, String RWItemList)

This method reads the first record of an Image detail dataset chain. It can be used on an Image detail dataset only.

- The KeyName parameter specifies the name of the key item used to search on.
- The KeyValue parameter specifies the value of the key.
- The RWItemList parameter is a comma-delimited list of the items to read.

A return of True indicates a record was found and read.

Boolean ReadLast (String KeyName, String KeyValue, String RWItemList)

This method reads the last record of an Image detail dataset chain. It can be used on an Image detail dataset only.

- The KeyName parameter specifies the name of the key item used to search on.
- The KeyValue parameter specifies the value of the key.
- The RWItemList parameter is a comma-delimited list of the items to read.

A return of True indicates a record was found and read.

Boolean ReadNext (String RWItemList)

This method reads the next record in an Image detail dataset chain. It can be used on an Image detail dataset only. The RWItemList parameter is a comma-delimited list of the items to read. A return of True indicates a record was found and read.

Boolean ReadPrevious (String RWItemList)

This method reads the previous record in an Image detail dataset chain. It can be used on an Image detail dataset only. The RWItemList parameter is a comma-delimited list of the items to read. A return of True indicates a record was found and read.

Boolean ReadSerialNext (String RWItemList)

This method reads the next record in a dataset. The RWItemList

parameter is a comma-delimited list of the items to read. A return of True indicates a record was found and read.

Boolean ReadSerialPrevious (String RWItemList)
This method reads the previous record in a dataset. The RWItemList parameter is a comma-delimited list of the items to read. A return of True indicates a record was found and read.

RemoveItem (String ItemName)
This method is used to remove a data item. This allows you to redefine the record layout of the dataset. The ItemName parameter specifies the name of the data item to remove.

Boolean Rewind ()
This method sets the dataset to the beginning of the file. A return of True indicates the dataset was rewound successfully.

Boolean Update (String RWItemList)
This method updates the current record of the dataset. The RWItemList parameter is a comma-delimited list of the items to update. A return of True indicates the record was updated successfully.

Boolean Write (String RWItemList)
This method writes a new record to the dataset. The RWItemList parameter is a comma-delimited list of the items to write. A return of True indicates the record was written successfully.

Boolean WriteItem (String ItemName, String ItemValue)
This method is used to write the value of a data item within the dataset. The ItemName parameter specifies the name of the data item to read or write. If the Delimiter property is empty, the data is formatted to a fixed length. This method is used in languages that do not allow the Item property to be assigned values.

The RWItemList parameter

Special symbols can be used in the RWItemList parameter in the above properties and methods:

- Specifying a RWItemList of @ indicates all items.
- Specifying a RWItemList of * indicates the RWItemList last used should be used again.

Obsolete methods

These methods are from pre-2.1.a versions of MiddleMan. They will continue to work, but should not be used for new code.

Boolean DeleteRec ()
This method operates exactly like the Delete method. It is only needed for use in Visual Basic 3.0, where the word Delete is a reserved word.

Boolean UpdateRec (String RWItemList)
This method operates exactly like the Update method. It is only needed for use in Visual Basic 3.0, where the word Update is a reserved word.

Boolean WriteRec (String RWItemList)
This method operates exactly like the Write method. It is only needed for use in Visual Basic 3.0, where the word Write is a reserved word.

Example

Visual Basic:

```
' Read from the Customer dataset by ACCT-NO
If Customer.ReadByKey("ACCT-NO", "000001",
            "NAME,ADDRESS-1;") Then
    ' Read item values
    SoldToName = Customer.Item ("NAME")
    SoldToAddress = Customer ("ADDRESS-1")
Else
```

```
    MsgBox Customer.ErrorMessage
    Exit Sub
End If
            .
            .
            .
' Update item values
Customer.Item ("NAME") = SoldToName
Customer ("ADDRESS-1") = SoldToAddress
If Not Customer.Update("NAME,ADDRESS-1;") Then
    MsgBox Customer.ErrorMessage
End If
```

KSAM file object

Ksam file objects are created by the AddKsamFileRef method of the session object. A ksam file object must be created for each KSAM file the client wishes to access. Once a ksam file object has been created, operations on the file can be performed using the object's methods.

Properties

Delimiter, type String, read/write access
Sets the delimiter to use between data item values when data is stored and retrieved with the ItemBuffer property. If the Delimiter property is empty, data is formatted to a fixed length. The default for the Delimiter is a comma.

ErrorMessage, type String, read access
Contains the last error message generated.

ErrorNumber, type integer, read access
Contains the last error number generated.

IsKey (String ItemName), type Boolean, read access
Indicates if a data item is a key or not. The ItemName parameter specifies the name of the data item.

Item (String ItemName), type String, read/write access
This property is used to read or write the value of a data item within the file. The ItemName parameter specifies the name of the data item to read or write. If the Delimiter property is empty, the data is formatted to a fixed length. The Item property is the ksam file object's default property, so it is not necessary to use the property's name in Visual Basic.

ItemBuffer (String RWItemList), type String, read/write access
This property is used to store and retrieve the values of the data items listed in the RWItemList parameter. If the Delimiter property is not empty, data item values are delimited by the character in the Delimiter property. If the Delimiter property is empty, data is formatted to a fixed length.

ItemFormatLength (String ItemName), type integer, read access
Contains the length in characters used to format a data item value. The data item is specified by the ItemName parameter. Data item values are formatted to this length when the Delimiter property is empty.

ItemLength (String ItemName), type integer, read access
Contains the data item length as specified by the AddItem method. The data item is specified in the ItemName parameter. The value of the ItemLength property depends on the ItemType property. Both properties are used to determine the internal storage format for the item. See the AddItem method for more information.

ItemList, type String, read access
Contains a comma-delimited list of all the items of the dataset.

ItemSigned (String ItemName), type Boolean, read/write access
Indicates if data item of type P or Z is signed or not. The data item is

specified by the ItemName parameter. This property is set automatically when data is read from a dataset.

If you are not reading data before writing data to a dataset, you should set this property to make sure that data of type P or Z is written correctly. Also, when finding or retrieving data using the FFindByKey, FFindN, Find, or ReadByKey methods, the ItemSigned property of the key item (if it is type P or Z) should be set to reflect the way data is stored in the key. The default value for this property is false.

ItemType (String ItemName), type String, read access
Contains the data item type as specified by the AddItem method. The data item is specified in the ItemName parameter. The value of the ItemType property determines the meaning of the ItemLength property. Both properties are used to determine the internal storage format for the item. See the AddItem method for more information.

RecordNumber, type long, read access
Contains the record number of the record read by the last ReadByKey, ReadDirect, ReadFirst, ReadLast, ReadNext, ReadPrevious, ReadSerialNext, or ReadSerialPrevious method.

SubItems (String ItemName), type integer, read access
Contains the data item sub-item count as specified in the database schema or by the AddItem method. The data item is specified in the ItemName parameter.

Methods

AddItem (String ItemName, String Type, integer Length, integer Offset, integer SubItems, Boolean Signed, Boolean Key, Boolean WriteAccess)
This method is used to add a data item. This allows you to define or redefine the record layout of the file.

- The ItemName parameter specifies the name of the new data item.
- The Type parameter specifies the TurboImage type of the new data item. Valid types are I, J, K, E, R, U, X, Z, and P.
- The Length parameter specifies the TurboImage length of the new data item.
- The SubItems parameters specify the number of sub-items contained in the new data item.
- The Signed parameter specifies if the new data item will contain a sign (type P or Z only).
- The Key parameter specifies if the new data item is a key.
- The WriteAccess parameter specifies if there is write access to the new data item.

The following table describes how the Type and Length parameters determine the internal storage format of the item:

Length Unit	Type	Type description
16 bits	E	A real (IEEE floating point) number.
16 bits	I and J	A signed binary integer in 2's complement form.
16 bits	K	An absolute binary quantity (no negative values).
16 bits	R	A real (HP 3000 floating point) number.
4 bits	P	A packed decimal number.
8 bits	X	An unrestricted ASCII character string.
8 bits	U	An ASCII character string containing no lowercase alphabetic characters.
8 bits	Z	A zoned decimal format number.

For more information on TurboImage data types and lengths, refer to the TurboImage/XL manual from Hewlett-Packard.

Boolean Close (String CloseDomain, SpaceDisposition)

This method closes an open KSAM file. The CloseDomain parameter determines the domain of the file after it is closed. Valid values are default, permanent, temporary, and delete. The SpaceDisposition parameter determines what to do with unused space after the EOF. Valid values are noreturn, change-limit, and return. A return of True indicates the file was successfully closed.

Boolean Delete ()

This method deletes the current record from a KSAM file. A return of True indicates the record was successfully deleted.

Boolean FClose (integer Disposition)

This method directly calls MPE/iX's FClose intrinsic. The Disposition parameter corresponds directly to the disposition parameter of the FClose intrinsic. For more information on the FClose intrinsic, refer to the KSAM/3000 reference manual from Hewlett-Packard.

Boolean FControl (integer ControlCode)

This method directly calls MPE/iX's FControl intrinsic. The ControlCode parameter corresponds directly to the controlcode parameter of the FControl intrinsic. For more information on the FControl intrinsic, refer to the KSAM/3000 reference manual from Hewlett-Packard.

Boolean FFindByKey (String KeyName, String KeyValue, integer Relop)

This method directly calls MPE/iX's FFindByKey intrinsic.

- The KeyName parameter provides the value for the keylocation parameter of the FFindByKey intrinsic.
- The KeyValue and Relop parameters correspond directly to the keyvalue and relop parameters of the FFindByKey intrinsic.

For more information on the FFindByKey intrinsic, refer to the KSAM/3000 reference manual from Hewlett-Packard.

Boolean FFindN (String KeyName, long RecordNumber)

This method directly calls MPE/iX's FFindN intrinsic.

- The KeyName parameter provides the value for the keylocation parameter of the FFindN intrinsic.
- The RecordNumber parameter corresponds directly to the number parameter of the FFindN intrinsic.

For more information on the FFindN intrinsic, refer to the KSAM/3000 reference manual from Hewlett-Packard.

Boolean Find (String KeyName, String KeyValue, String Relation)

This method sets the record pointer for a KSAM file based upon a key value.

- The KeyName parameter specifies the name of the key item used to search on.
- The KeyValue parameter specifies the value of the key.
- The Relation parameter specifies how the search is done. Valid values for the Relation parameter are =, >, >=, and <=.

A return of True indicates the record pointer was set successfully.

Boolean FLock (Boolean WaitFlag)

This method directly calls MPE/iX's FLock intrinsic. The WaitFlag parameter corresponds directly to the lockcond parameter of the FLock intrinsic. For more information on the FLock intrinsic, refer to the KSAM/3000 reference manual from Hewlett-Packard.

Boolean FOpen (String FormalDesignator, integer FOptions, integer AOptions)

This method directly calls MPE/iX's FOpen intrinsic. The FormalDesignator, FOptions, and AOptions parameters correspond directly to the formaldesignator, foptions, and aoptions parameters of the FOpen intrinsic. For more information on the FOpen intrinsic, refer to the KSAM/3000 reference manual from Hewlett-Packard.

Boolean FPoint (long RecordNumber)

This method directly calls MPE/iX's FPoint intrinsic. The RecordNumber parameter corresponds directly to the recnum parameter of the FPoint intrinsic. For more information on the FPoint intrinsic, refer to the KSAM/3000 reference manual from Hewlett-Packard.

Boolean FRead (integer Length)

This method directly calls MPE/iX's FRead intrinsic. The Length parameter specifies a byte count used as the tcount parameter of the FRead intrinsic. For more information on the FRead intrinsic, refer to the KSAM/3000 reference manual from Hewlett-Packard.

Boolean FReadByKey (integer Length, String KeyName, String KeyValue)

This method directly calls MPE/iX's FReadByKey intrinsic.

- The Length parameter specifies a byte count used as the tcount parameter of the FReadByKey intrinsic.
- The KeyName parameter provides the value for the keylocation parameter of the FReadByKey intrinsic.
- The KeyValue parameter corresponds directly to the keyvalue parameter of the FReadByKey intrinsic.

For more information on the FReadByKey intrinsic, refer to the KSAM/3000 reference manual from Hewlett-Packard.

Boolean FReadC (integer Length)

This method directly calls MPE/iX's FReadC intrinsic. The Length parameter specifies a byte count used as the tcount parameter of the FReadC intrinsic. For more information on the FReadC intrinsic, refer to the KSAM/3000 reference manual from Hewlett-Packard.

Boolean FReadDir (integer Length, long RecordNumber)

This method directly calls MPE/iX's FReadDir intrinsic.

- The Length parameter specifies a byte count used as the tcount parameter of the FReadDir intrinsic.
- The RecordNumber parameter corresponds directly to the recnum parameter of the FReadDir intrinsic.

For more information on the FReadDir intrinsic, refer to the KSAM/3000 reference manual from Hewlett-Packard.

Boolean FRemove ()

This method directly calls MPE/iX's FRemove intrinsic. For more information on the FRemove intrinsic, refer to the KSAM/3000 reference manual from Hewlett-Packard.

Boolean FSetMode (integer ModeFlags)

This method directly calls MPE/iX's FSetMode intrinsic. The ModeFlags parameter corresponds directly to the modeflags parameter of the FSetMode intrinsic. For more information on the FSetMode intrinsic, refer to the KSAM/3000 reference manual from Hewlett-Packard.

Boolean FSpace (integer Displacement)

This method directly calls MPE/iX's FSpace intrinsic. The Displacement parameter corresponds directly to the displacement parameter of the FSpace intrinsic. For more information on the FSpace intrinsic, refer to the KSAM/3000 reference manual from Hewlett-Packard.

Boolean FUnlock ()

This method directly calls MPE/iX's FUnlock intrinsic. For more information on the FUnlock intrinsic, refer to the KSAM/3000 reference manual from Hewlett-Packard.

Boolean FUpdate (integer Length)

This method directly calls MPE/iX's FUpdate intrinsic. The Length parameter specifies a byte count used as the tcount parameter of the

FUpdate intrinsic. For more information on the FUpdate intrinsic, refer to the KSAM/3000 reference manual from Hewlett-Packard.

Boolean FWrite (integer Length)

This method directly calls MPE/iX's FWrite intrinsic. The Length parameter specifies a byte count used as the tcount parameter of the FWrite intrinsic. For more information on the FWrite intrinsic, refer to the KSAM/3000 reference manual from Hewlett-Packard.

Boolean Open (String FileName, String OpenGroup, String OpenAccount, String FopDomain, String AopAccess)

This method opens a KSAM file.

- The FileName parameter specifies the file to be opened.
- The OpenGroup parameter specifies the group that the file is located in.
- The OpenAccount parameter specifies the account that the file is located in.
- The FopDomain parameter specifies the domain of the file. Valid values are New for a new file, Permanent for an old permanent file, Temporary for an old temporary file, and Old for an old permanent or temporary file.
- The AopAccess parameter specifies the access mode for the file. Valid values are Read, Write, Write-Save, Append, Read-Write, and Update.

A return of True indicates the file was opened successfully.

Boolean PointAt (long RecordNumber)

This method sets the record pointer for a KSAM file. The RecordNumber parameter specifies the record to set the pointer to. A return of True indicates the record pointer was set successfully.

Boolean ReadByKey (String KeyName, String KeyValue, String RWItemList)

This method reads a record from a KSAM file by key.

- The KeyName parameter specifies the name of the key item used to search on.
- The KeyValue parameter specifies the value of the key.
- The RWItemList parameter is a comma-delimited list of the items to read.

A return of True indicates a record was found and read.

Boolean ReadDirect (long RecordNumber, String RWItemList)

This method reads a record from a KSAM file by record number.

- The RecordNumber parameter specifies the record number of the record to read.
- The RWItemList parameter is a comma-delimited list of the items to read.

A return of True indicates a record was found and read.

Boolean ReadNext (String RWItemList)

This method reads the next record in a KSAM file. The RWItemList parameter is a comma-delimited list of the items to read. A return of True indicates a record was found and read.

RemoveItem (String ItemName)

This method is used to remove a data item. This allows you to redefine the record layout of the file. The ItemName parameter specifies the name of the data item to remove.

Boolean Rewind ()

This method sets the record pointer to the beginning of a KSAM file. A return of True indicates the file was rewound successfully.

Boolean Update (String RWItemList)

This method updates the current record of a KSAM file. The RWItemList parameter is a comma-delimited list of the items to update. A return of True indicates the record was updated successfully.

Boolean Write (String RWItemList)

This method writes a new record to a KSAM file. The RWItemList parameter is a comma-delimited list of the items to write. A return of True indicates the record was written successfully.

The RWItemList parameter

Special symbols can be used in the RWItemList parameter in the above properties and methods:

- Specifying a RWItemList of @ indicates all items.
- Specifying a RWItemList of * indicates the RWItemList last used should be used again.

For KSAM files, the items listed in the RWItemList only determine the length of the operation, since data must be read, updated, and written in continuous blocks. The length of the operation is determined by finding the item in the list with the largest value for the sum of its offset and storage length. All items in the record layout up to and including that item are included in the operation.

Obsolete methods

These methods are from pre-2.1.a versions of MiddleMan. They will continue to work, but should not be used for new code.

Boolean CloseFile (String CloseDomain, SpaceDisposition)

This method operates exactly like the Close method. It is only needed for use in Visual Basic 3.0, where the word *Close* is a reserved word.

Boolean DeleteRec ()

This method operates exactly like the Delete method. It is only needed for use in Visual Basic 3.0, where the word *Delete* is a reserved word.

Boolean UpdateRec (String RWItemList)

This method operates exactly like the Update method. It is only needed for use in Visual Basic 3.0, where the word *Update* is a reserved word.

Boolean WriteRec (String RWItemList)

This method operates exactly like the Write method. It is only needed for use in Visual Basic 3.0, where the word *Write* is a reserved word.

Example

Visual Basic:

```
Dim DASession As Object   'Session object
Dim FKsam As Object
    .
    .
    .
Private Sub Form_Load()
 Dim Password As String

 Set DASession = CreateObject("MdmDA.Session")
 DASession.HostAddress = "204.250.148.132"
 DASession.Port = 30002
 DASession.VendorId = "8"
 DASession.JobName = "CUSTINQ"
 DASession.LoginUser = "DEVMGR"
 DASession.LoginAccount = "MINISOFT"
 DASession.UserPassword = "XMSXDEV"
 DASession.Connect
 If DASession.ConnectStatus = True Then
  DASession.HostCommand "FILE F=FKSAM.EXE.MINISOFT"
  Set FKsam = DASession.AddKsamFileRef("FKSAM")
  If FKsam.FOpen("*F", 3, 5) Then
   FKsam.AddItem "ACCT-NO", "X", 6, 0, 1, False, True, True
   FKsam.AddItem "NAME", "X", 20, 6, 1, False, False, True
```

```
  FKsam.AddItem "ADDRESS-1", "X", 20, 26, 1, False, False, True
  FKsam.AddItem "ADDRESS-2", "X", 20, 46, 1, False, False, True
  FKsam.AddItem "STATE", "X", 2, 66, 1, False, False, True
  FKsam.AddItem "ZIP", "X", 6, 68, 1, False, True, True
  FKsam.AddItem "REGION", "X", 4, 74, 1, False, True, True
  FKsam.AddItem "RATING", "X", 2, 78, 1, False, True, True
  FKsam.AddItem "LIMIT", "P", 12, 80, 1, True, True, True
  FKsam.AddItem "BALANCE", "P", 8, 86, 1, True, False, True
  FKsam.AddItem "TAX-ID", "X", 2, 90, 1, False, False, True
  Exit Sub
 Else
  MsgBox FKsam.ErrorMessage, MB_ICONSTOP, "FOpen"
 End If
     Set FKsam = Nothing

 ' Disconnect from the Data Server.
 DASession.Disconnect
End If
Set DASession = Nothing
' Exit the client program.
End
End Sub
     .
     .
     .
Private Sub ReadRec_Click()
 If FKsam.FReadByKey(92, "ACCT-NO", AccountNumber) Then
  RecBuf$ = FKsam.ItemBuffer("@")
 Else
  MsgBox FKsam.ErrorMessage
 End If
End Sub
```

MPE file object

MPE file objects are created by the AddMpeFileRef method of the session object. A MPE file object must be created for each MPE file the client wishes to access. Once a MPE file object has been created, operations on the file can be performed using the object's methods.

Properties

Delimiter, type String, read/write access
Sets the delimiter to use between data item values when data is stored and retrieved with the ItemBuffer property. If the Delimiter property is empty, data is formatted to a fixed length. The default for the Delimiter is a comma.

ErrorMessage, type String, read access
Contains the last error message generated.

ErrorNumber, type integer, read access
Contains the last error number generated.

Item (String ItemName), type String, read/write access
This property is used to read or write the value of a data item within the file. The ItemName parameter specifies the name of the data item to read or write. If the Delimiter property is empty, the data is formatted to a fixed length. The Item property is the MPE file object's default property, so it is not necessary to use the property's name in Visual Basic.

ItemBuffer (String RWItemList), type String, read/write access
This property is used to store and retrieve the values of the data items listed in the RWItemList parameter. If the Delimiter property is not empty, data item values are delimited by the character in the Delimiter property. If the Delimiter property is empty, data is formatted to a fixed length.

ItemFormatLength (String ItemName), type integer, read access
Contains the length in characters used to format a data item value. The data item is specified by the ItemName parameter. Data item values are formatted to this length when the Delimiter property is empty.

ItemLength (String ItemName), type integer, read access
Contains the data item length as specified by the AddItem method. The

data item is specified in the ItemName parameter. The value of the ItemLength property depends on the ItemType property. Both properties are used to determine the internal storage format for the item. See the AddItem method for more information.

ItemList, type String, read access
Contains a comma-delimited list of all the items of the dataset.

ItemSigned (String ItemName), type Boolean, read/write access
Indicates if data item of type P or Z is signed or not. The data item is specified by the ItemName parameter. This property is set automatically when data is read from a dataset. If you are not reading data before writing data to a dataset, you should set this property to make sure that data of type P or Z is written correctly. The default value for this property is false.

ItemType (String ItemName), type String, read access
Contains the TurboImage data item type as specified in the AddItem method. The data item is specified in the ItemName parameter. The value of the ItemType property determines the meaning of the ItemLength property. Both properties are used to determine the internal storage format for the item. See the AddItem method for more information.

RecordNumber, type long, read access
Contains the record number of the record read by the last ReadDirect method, pointed at by the last PointAt method, or written with the last WriteDirect method.

SubItems (String ItemName), type integer, read access
Contains the data item sub-item count as specified in the AddItem method. The data item is specified in the ItemName parameter.

Methods

AddItem (String ItemName, String Type, integer Length, integer Offset, integer SubItems, Boolean Signed, Boolean WriteAccess)
This method is used to add a data item. This allows you to define or redefine the record layout of the file.

- The ItemName parameter specifies the name of the new data item.
- The Type parameter specifies the TurboImage type of the new data item. Valid types are I, J, K, E, R, U, X, Z, and P.
- The Length parameter specifies the TurboImage length of the new data item.
- The Offset parameter specifies the number of bytes from the beginning of each record that the item is to start on. The first Item will have an Offset of zero.
- The SubItems parameters specify the number of sub-items contained in the new data item.
- The Signed parameter specifies if the new data item will contain a sign (type P or Z only).
- The WriteAccess parameter specifies if there is write access to the new data item.

The following table describes how the Type and Length parameters determine the internal storage format of the item:

Length Unit	Type	Type description
16 bits	E	A real (IEEE floating point) number.
16 bits	I and J	A signed binary integer in 2's complement form.
16 bits	K	An absolute binary quantity (no negative values).
16 bits	R	A real (HP 3000 floating point) number.
4 bits	P	A packed decimal number.

8 bits	X	An unrestricted ASCII character string.
8 bits	U	An ASCII character string containing no lowercase alphabetic characters.
8 bits	Z	A zoned decimal format number.

For more information on TurboImage data types and lengths, refer to the TurboImage/XL manual from Hewlett-Packard.

Boolean Close (String CloseDomain, SpaceDisposition)

This method closes an open MPE file.

- The CloseDomain parameter determines the domain of the file after it is closed. Valid values are default, permanent, temporary, and delete.
- The SpaceDisposition parameter determines what to do with unused space after the EOF. Valid values are noreturn, change-limit, and return.

A return of True indicates the file was successfully closed.

Boolean Delete (long RecordNumber)

This method deletes the specified record from a MPE file. A return of True indicates the record was successfully deleted.

Boolean FClose (integer Disposition)

This method directly calls MPE/iX's FClose intrinsic. The Disposition parameter of the FClose method corresponds directly to the disposition parameter of the FClose intrinsic. For more information on the FClose intrinsic, refer to the MPE/iX Intrinsics reference manual from Hewlett-Packard.

Boolean FControl (integer ControlCode)

This method directly calls MPE/iX's FControl intrinsic. The ControlCode parameter of the FControl method corresponds directly to the itemnum parameter of the FControl intrinsic. For more information

on the FControl intrinsic, refer to the MPE/iX Intrinsics reference manual from Hewlett-Packard.

Boolean FDelete (long RecordNumber)

This method directly calls MPE/iX's FDelete intrinsic. The RecordNumber parameter corresponds directly to the lrecnum parameter of the FDelete intrinsic. For more information on the FDelete intrinsic, refer to the MPE/iX Intrinsics reference manual from Hewlett-Packard.

Boolean FLock (Boolean WaitFlag)

This method directly calls MPE/iX's FLock intrinsic. The WaitFlag parameter corresponds directly to the lockflag parameter of the FLock intrinsic. For more information on the FLock intrinsic, refer to the MPE/iX Intrinsics reference manual from Hewlett-Packard.

Boolean FOpen (String FormalDesignator, integer FOptions, integer AOptions, integer RecordSize, long FileSize)

This method directly calls MPE/iX's FOpen intrinsic. The FormalDesignator, FOptions, AOptions, RecordSize, and FileSize parameters correspond directly to the formaldesignator, foptions, aoptions, recsize, and filesize parameters of the FOpen intrinsic. For more information on the FOpen intrinsic, refer to the MPE/iX Intrinsics reference manual from Hewlett-Packard.

Boolean FPoint (long RecordNumber)

This method directly calls MPE/iX's FPoint intrinsic. The RecordNumber parameter corresponds directly to the lrecnum parameter of the FPoint intrinsic. For more information on the FPoint intrinsic, refer to the MPE/iX Intrinsics reference manual from Hewlett-Packard.

Boolean FRead (integer Length)

This method directly calls MPE/iX's FRead intrinsic. The Length parameter specifies a byte count used as the length parameter of the

FRead intrinsic. For more information on the FRead intrinsic, refer to the MPE/iX Intrinsics reference manual from Hewlett-Packard.

Boolean FReadDir (integer Length, long RecordNumber)

This method directly calls MPE/iX's FReadDir intrinsic.

- The Length parameter specifies a byte count used as the length parameter of the FReadDir intrinsic.
- The RecordNumber parameter corresponds directly to the lrecnum parameter of the FReadDir intrinsic.

For more information on the FReadDir intrinsic, refer to the MPE/iX Intrinsics reference manual from Hewlett-Packard.

Boolean FReadSeek (long RecordNumber)

This method directly calls MPE/iX's FREADSEEK intrinsic. The RecordNumber parameter corresponds directly to the lrecnum parameter of the FREADSEEK intrinsic. For more information on the FREADSEEK intrinsic, refer to the MPE/iX Intrinsics reference manual from Hewlett-Packard.

Boolean FRename (String FormalDesignator)

This method directly calls MPE/iX's FRENAME intrinsic. The FormalDesignator parameter corresponds directly to the formaldesig parameter of the FRENAME intrinsic. For more information on the FRENAME intrinsic, refer to the MPE/iX Intrinsics reference manual from Hewlett-Packard.

Boolean FSetMode (integer ModeFlags)

This method directly calls MPE/iX's FSetMode intrinsic. The ModeFlags parameter corresponds directly to the modeflags parameter of the FSetMode intrinsic. For more information on the FSetMode intrinsic, refer to the MPE/iX Intrinsics reference manual from Hewlett-Packard.

Boolean FSpace (integer Displacement)

This method directly calls MPE/iX's FSpace intrinsic. The Displacement parameter corresponds directly to the displacement parameter of the FSpace intrinsic. For more information on the FSpace intrinsic, refer to the MPE/iX Intrinsics reference manual from Hewlett-Packard.

Boolean FUnlock ()

This method directly calls MPE/iX's FUnlock intrinsic. For more information on the FUnlock intrinsic, refer to the MPE/iX Intrinsics reference manual from Hewlett-Packard.

Boolean FUpdate (integer Length)

This method directly calls MPE/iX's FUpdate intrinsic. The Length parameter specifies a byte count used as the length parameter of the FUpdate intrinsic. For more information on the FUpdate intrinsic, refer to the MPE/iX Intrinsics reference manual from Hewlett-Packard.

Boolean FWrite (integer Length)

This method directly calls MPE/iX's FWrite intrinsic. The Length parameter specifies a byte count used as the length parameter of the FWrite intrinsic. For more information on the FWrite intrinsic, refer to the MPE/iX Intrinsics reference manual from Hewlett-Packard.

Boolean FWriteDir (integer Length, long RecordNumber)

This method directly calls MPE/iX's FWRITEDIR intrinsic.

- The Length parameter specifies a byte count used as the length parameter of the FWRITEDIR intrinsic.
- The RecordNumber parameter corresponds directly to the lrecnum parameter of the FWRITEDIR intrinsic.

For more information on the FWRITEDIR intrinsic, refer to the MPE/iX Intrinsics reference manual from Hewlett-Packard.

Boolean Open (String FileName, String OpenGroup, String OpenAccount, String FopDomain, String AopAccess)
This method opens a MPE file.

- The FileName parameter specifies the file to be opened.
- The OpenGroup parameter specifies the group that the file is located in.
- The OpenAccount parameter specifies the account that the file is located in.
- The FopDomain parameter specifies the domain of the file. Valid values are New for a new file, Permanent for an old permanent file, Temporary for an old temporary file, and Old for an old permanent or temporary file.
- The AopAccess parameter specifies the access mode for the file. Valid values are Read, Write, Write-Save, Append, Read-Write, and Update.

A return of True indicates the file was opened successfully.

Boolean PointAt (long RecordNumber)
This method sets the record pointer for a MPE file. The RecordNumber parameter specifies the record to set the pointer to. A return of True indicates the record pointer was set successfully.

Boolean ReadDirect (long RecordNumber, String RWItemList)
This method reads a record from a MPE file by record number.

- The RecordNumber parameter specifies the record number of the record to read.
- The RWItemList parameter is a comma-delimited list of the items to read.

A return of True indicates a record was found and read.

Boolean ReadNext (String RWItemList)
This method reads the next record in a MPE file. The RWItemList

parameter is a comma-delimited list of the items to read. A return of True indicates a record was found and read.

RemoveItem (String ItemName)
This method is used to remove a data item. This allows you to redefine the record layout of the file. The ItemName parameter specifies the name of the data item to remove.

Boolean Rewind ()
This method sets the record pointer to the beginning of a MPE file. A return of True indicates the file was rewound successfully.

Boolean Update (String RWItemList)
This method updates the current record of a MPE file. The RWItemList parameter is a comma-delimited list of the items to update. A return of True indicates the record was updated successfully.

Boolean Write (String RWItemList)
This method writes a new record to a MPE file. The RWItemList parameter is a comma-delimited list of the items to write. A return of True indicates the record was written successfully.

Boolean WriteDirect (long RecordNumber, String RWItemList)
This method writes a record to a MPE file by record number.

- The RecordNumber parameter specifies the record number of the record to write.
- The RWItemList parameter is a comma-delimited list of the items to write.

A return of True indicates a record was found and written.

The RWItemList parameter

Special symbols can be used in the RWItemList parameter in the above properties and methods:

- Specifying a RWItemList of @ indicates all items.
- Specifying a RWItemList of * indicates the RWItemList last used should be used again.

For MPE files, the items listed in the RWItemList only determine the length of the operation, since data must be read, updated, and written in continuous blocks. The length of the operation is determined by finding the item in the list with the largest value for the sum of its offset and storage length. All items in the record layout up to and including that item are included in the operation.

Obsolete methods

These methods are from pre-2.1.a versions of MiddleMan. They will continue to work, but should not be used for new code.

Boolean CloseFile (String CloseDomain, SpaceDisposition)
This method operates exactly like the Close method. It is only needed for use in Visual Basic 3.0, where the word *Close* is a reserved word.

Boolean UpdateRec (String RWItemList)
This method operates exactly like the Update method. It is only needed for use in Visual Basic 3.0, where the word *Update* is a reserved word.

Boolean WriteRec (String RWItemList)
This method operates exactly like the Write method. It is only needed for use in Visual Basic 3.0, where the word *Write* is a reserved word.

File transfer engine

The file transfer engine objects are used by the client program to communicate with the file server program on the host. This gives the client

the ability to transfer files to/from the host as well as execute MPE commands, MPE command files, and programs.

Object creation

In order to access the file transfer engine from a client program, the client program must create the session object of the file transfer engine as follows:

Examples

Visual Basic (a complete Visual Basic example is on page 2-58):

```
Dim FTSession As Object
 ...
Set FTSession = CreateObject("NetFT.Session")
```

Visual C++:

```
CreateDispatch("NetFT.Session");
```

Object destruction

When the client program wishes to stop using the file transfer engine, it must delete the session object as follows:

Examples

Visual Basic:

```
Set FTSession = Nothing
```

Visual C++:

```
ReleaseDispatch( );
```

Note: A more complete example of using the file transfer engine from Visual Basic is shown by the FEDIT.MAK project in the VB subdirectory of the MiddleMan installation directory.

Session object

Once the session object has been created, the client can use its properties and methods to access the host's file transfer server program. Typically the properties specifying the host's IP address, TCP/IP port number, and login information are set, then the Connect method is called to connect the host's file transfer server program.

Properties

AccountPassword, type String, write access
The string used for the account password when connecting to the host's data server program.

ConnectStatus, type Boolean, read access
Contains True if the last connection attempt was successful, else False.

CtrlZIsEof, type Boolean, write access
This property specifies whether the end-of-file of a PC file is marked with a CTRL-Z character or not.

- A value of True indicates a CTRL-Z found in the file will be interpreted as the EOF on uploads, and a CTRL-Z will be written at the end of the file on downloads.
- A value of False indicates a CTRL-Z found in the file will be interpreted as normal data on uploads, and that a CTRL-Z will not be written at the end of the file on downloads.

This property only applies to ASCII transfers. The default is False.

Direction, type integer, write access
This property specifies the direction of the next file transfer. A value of 0 indicates uploading, while a value of 1 indicates downloading. The default is uploading.

DisplayStats, type Boolean, write access
This property specifies whether a dialog box showing the status of the transfer should be displayed during the file transfer. The default is False.

ErrorMessage, type String, read access
Contains the last error message generated.

ErrorNumber, type integer, read access
Contains the last error number generated.

ExistsAction, type integer, write access
This property specifies what action to take if the destination file already exists.

- A value of 0 indicates the transfer should be canceled.
- A value of 1 indicates the destination file should be deleted and the transfer should continue.
- A value of 2 indicates the user should be prompted whether to delete the file or cancel the transfer.

The default is 0.

FileTransferStatus, type integer, read access
This property indicates the status of an in-progress or completed file transfer.

- A value of 0 indicates the last transfer was completed successfully.
- A value of 1 indicates the last transfer failed.

- Any other value indicates a file transfer or some other operation is in progress.

GroupPassword, type String, write access
The string used for the group password when connecting to the host's data server program.

HostAddress, type String, write access
The IP address used when attempting to connect to the host.

HostFile, type String, write access
This property specifies the name of the host file involved in the transfer. It may be fully qualified with the group and account if necessary. If the file is in or should be created in the temporary domain, the string ,TEMP (note the comma) should be appended to its name.

HostFileList (String ListMask), String, read access
This property returns a comma-delimited list of files on the host. The ListMask parameter specifies a valid ListF filename argument used to compile the list.

JobName, type String, write access
The string used for the job name when connecting to the host's data server program.

KeepTrailingSpaces, type Boolean, write access
This property specifies what to do with the trailing spaces on each record during ASCII downloads. A value of True indicates trailing spaces should be kept, while a value of False indicates trailing spaces should be deleted. The default is False.

LocalFile, type String, write access
This property specifies the name of the local file involved in the file transfer. It may be fully qualified with the drive letter and path if necessary.

LoginAccount, type String, write access
The string used for the account name when connecting to the host's data server program.

LoginGroup, type String, write access
The string used for the group name when connecting to the host's data server program.

LoginInfo, type String, read access
This property contains the current user name, group name, and account name the server is running from. The first 8 characters contain the user name, the next 8 characters contain the group name, and the last 8 characters contain the account name.

LoginStatus, type Boolean, read access
Contains True if the login information supplied when connecting to the host's data server program was valid, else False.

LoginUser, type String, write access
The string used for the user name when connecting to the host's data server program.

Port, type integer, write access
A number that uniquely identifies the program on the host that the client wishes to connect to. This number may be any number between 1 and 32767, though a number above 30000 is recommended.

RecordSize, type integer, write access
This property specifies the record length of the host file for uploads. Valid values are between 1 and 4096. The default is 80 for ASCII transfers or 256 for binary transfers.

SpacesPerTab, type integer, write access
This property sets the distance between tab stops and enables changing tab characters to spaces to fill to the next tab stop. It is used only on

ASCII uploads. A value of 0 disables the replacement of tab characters. Valid values are between 0 and 9. The default is 0.

TransferMode, type integer, write access
This property specifies the mode of the file transfer. Valid values are 0 for an ASCII file transfer, 1 for a binary file transfer, and 2 for a binary transfer that saves the file information in a file header.

UserPassword, type String, write access
The string used for the user password when connecting to the host's data server program.

Methods

Boolean Connect ()
This method attempts to connect to the host's data server program at the IP address specified by the HostAddress property. The Port property is used to identify the requested host program. Connect sets the ConnectStatus property and returns True if the connection request succeeded, otherwise it returns False.

void Disconnect ()
This method disconnects from the host's data server program.

integer HostCommand (String CommandString)
This method executes a MPE command or MPE command file. The CommandString parameter specifies the command or command file to be executed. A return of 0 indicates the command was executed successfully, while a return of 1 indicates the command failed.

Boolean HostLogin ()
This method changes the host server login to that specified by the LoginUser, UserPassword, LoginGroup, GroupPassword, LoginAccount, AccountPassword, and JobName properties. A return of True indicates the login was changed successfully.

void SaveConfig ()

This method saves the writeable session object properties to a file. This file can later be used to initialize a session object when created.

Boolean LoadConfig (String FileName)

This method loads the session object properties from the named file. A return of True indicates the properties were loaded successfully.

Boolean SetSessionConfig ()

This method displays the Session Configuration dialog of the File Transfer Access Engine. This dialog allows the user to set these properties: HostAddress, Port, LoginUser, UserPassword, LoginGroup, GroupPassword, LoginAccount, AccountPassword, and JobName. A return of True indicates the user clicked OK to exit the dialog; a return of False indicates the user clicked Cancel to exit the dialog.

Boolean SetTransferConfig ()

This method displays the File Transfer Configuration dialog of the File Transfer Access Engine. This dialog allows the user to set default file transfer options. A return of True indicates the user clicked OK to exit the dialog; a return of False indicates the user clicked Cancel to exit the dialog.

integer StartTransfer ()

This method initiates a file transfer. A return of 0 indicates the file transfer was successful, while a return of 1 indicates the file transfer failed.

Example

Here is a Visual Basic sample for creating, using, and deleting the session object of the file transfer engine:

```
Private Sub cmdUpload_Click( )
Const gcUpload = 0
```

```
Const gcDownload = 1
Const gcASCII = 0
Const gcBinary = 1
Dim FTSession As Object
Dim ii As Integer
Dim TempStr As String
 On Error GoTo Egress1
 Screen.MousePointer = 11
StatusLine = "Begin..."
 Set FTSession = CreateObject("NetFT.Session")
StatusLine = "Object Created - Request Connection..."
 TempStr = tHostAddr.Text
 If (Len(TempStr) > 1) Then
  FTSession.HostAddress = TempStr
 End If
 FTSession.Port = 30001
 FTSession.LoginUser = "MGR"
 TempStr = tPassword.Text
 If (Len(TempStr) > 1) Then
  FTSession.UserPassword = TempStr
 End If
 FTSession.LoginGroup = "PUB"
 FTSession.GroupPassword = ""
 FTSession.LoginAccount = "MINISOFT"
 FTSession.AccountPassword = ""
 FTSession.JobName = "SAMPLE"
 If (Not FTSession.Connect) Then
  MsgBox "1 - " & FTSession.ErrorMessage
  GoTo Egress1
 End If
 ii = 0
StatusLine = "Connection Established - Logging on..."
 FTSession.HostLogin
 TempStr = FTSession.LoginInfo
 If (Len(TempStr) <> 24) Then
  MsgBox "2 - " & FTSession.ErrorMessage
  GoTo Egress1
 End If
 FTSession.DisplayStats = True
 FTSession.Direction = gcUpload
 FTSession.ExistsAction = 1
 TempStr = tLocalFN.Text
```

```
 If (Len(TempStr) > 1) Then
  FTSession.LocalFile = TempStr
 End If
 TempStr = tRemoteFN.Text
 If (Len(TempStr) > 1) Then
  FTSession.HostFile = TempStr
 End If
 FTSession.RecordSize = 80
 FTSession.TransferMode = gcASCII
StatusLine = "Logon Successful - Transfering..."
 If (FTSession.StartTransfer <> 0) Then
  MsgBox "3 - " & FTSession.ErrorMessage
  GoTo Egress1
 End If
StatusLine = "Transfer Complete."
 GoTo Egress2
Egress1:
 MsgBox "Error Exit"
Egress2:
 If FTSession.ConnectStatus = True Then
  FTSession.Disconnect
 End If
 Set FTSession = Nothing
StatusLine = "Ready"
 Screen.MousePointer = 0
End Sub
```

Stream access engine

The stream access engine objects are used by the client program to communicate with a custom server program on the host. The custom server is any application that interfaces to the client through the Network Interface Library API on the host. The stream access engine allows the client to send and receive data to/from the custom server program.

Object creation

In order to access the stream access engine from a client program, the client program must create the session object of the stream access engine as follows:

Examples

Visual Basic:

```
Dim SASession As Object
  ...
Set SASession = CreateObject("MdmSA.Session")
```

Visual C++:

```
CreateDispatch("MdmSA.Session");
```

Object destruction

When the client program wants to stop using the stream access engine, it must delete the session object as follows:

Examples

Visual Basic:

```
Set SASession = Nothing
```

Visual C++:

```
ReleaseDispatch( );
```

Note: A more complete example of using the stream access engine from Visual Basic is shown by the FEDIT.MAK project in the VB subdirectory of the MiddleMan installation directory.

Session object

Once the session object has been created, the client can use its properties and methods to access the host's server program. Typically the properties specifying the host's IP address, TCP/IP port number, and login information are set, then the Connect method is called to connect the host's server program.

Properties

ConnectStatus, type Boolean, read access
Contains True if the last connection attempt was successful, else False.

ErrorMessage, type String, read access
Contains the last error message generated.

ErrorNumber, type integer, read access
Contains the last error number generated.

HostAddress, type String, write access
The IP address used when attempting to connect to the host.

Port, type integer, write access
A number that uniquely identifies the program on the host that the client wishes to connect to. This number may be any number between 1 and 32767, though a number above 30000 is recommended.

TraceLevel, type integer, write access
A number representing the tracing options activated. It is the sum of the individual values of each option. The options are:

1 - trace errors,
2 - trace network traffic,
4 - trace server transactions, and
8 - trace client transactions.

Methods

Boolean Connect ()
This method attempts to connect to the host's server program at the IP address specified by the HostAddress property. The Port property is used to identify the requested host program. Connect sets the ConnectStatus property and returns True if the connection request succeeded, else it returns False.

void Disconnect ()
This method disconnects from the host's server program.

String NetRead (integer Length, Boolean Block)
This method reads data from the host's server program.

- The Length parameter specifies the number of bytes to read.
- The Block parameter specifies whether to wait until the number of bytes specified by the Length parameter are available (True), or to return immediately with only the data currently available (False).

void NetWrite (String Data)
This method writes data to the host's server program. The Data parameter specifies the data to write to the host's server program.

void SaveConfig ()
This method saves the writeable session object properties to a file. This file can later be used to initialize a session object when created.

Boolean LoadConfig (String FileName)

This method loads the session object properties from the named file. A return of True indicates the properties were loaded successfully.

Boolean SetSessionConfig ()

This method displays the Session Configuration dialog of the stream access engine. This dialog allows the user to set these properties: HostAddress and Port. A return of True indicates the user clicked OK to exit the dialog; a return of False indicates the user clicked Cancel to exit the dialog.

Boolean SetTraceConfig ()

This method displays the Trace Options dialog of the stream access engine. This dialog allows the user to set tracing options used for debugging. A return of True indicates the user clicked OK to exit the dialog; a return of False indicates the user clicked Cancel to exit the dialog.

File transfer protocol engine

The file transfer protocol (ftp) engine objects are used by the client program to communicate with the ftp server program on the host. This gives the client the ability to transfer files to/from the host.

Object creation

In order to access the file transfer engine from a client program, the client program must create the session object of the file transfer engine as follows:

Examples

Visual Basic (a complete Visual Basic example is on page 2-58):

```
Dim FTSession As Object
  ...
Set FTSession = CreateObject("NetFT.Session")
```

Visual C++:

```
CreateDispatch("NetFT.Session");
```

Object destruction

When the client program wants to stop using the file transfer engine, it must delete the session object as follows:

Examples

Visual Basic:

```
Set FTSession = Nothing
```

Visual C++:

```
ReleaseDispatch( );
```

Note: A more complete example of using the file transfer engine from Visual Basic is shown by the FEDIT.MAK project in the VB subdirectory of the MiddleMan installation directory.

Session object

Once the session object has been created, the client can use its properties and methods to access the host's file transfer server program. Typically the

properties specifying the host's IP address, TCP/IP port number, and login information are set, then the Connect method is called to connect the host's file transfer server program.

Properties

AccountPassword, type String, write access
The string used for the account password when connecting to the host's data server program.

ConnectStatus, type Boolean, read access
Contains True if the last connection attempt was successful, else False.

CtrlZIsEof, type Boolean, write access
This property specifies whether the end-of-file of a PC file is marked with a CTRL-Z character or not.

- A value of True indicates a CTRL-Z found in the file will be interpreted as the EOF on uploads, and a CTRL-Z will be written at the end of the file on downloads.
- A value of False indicates a CTRL-Z found in the file will be interpreted as normal data on uploads, and that a CTRL-Z will not be written at the end of the file on downloads.

This property only applies to ASCII transfers. The default is False.

Direction, type integer, write access
This property specifies the direction of the next file transfer. A value of 0 indicates uploading, while a value of 1 indicates downloading. The default is uploading.

DisplayStats, type Boolean, write access
This property specifies whether a dialog box showing the status of the transfer should be displayed during the file transfer. The default is False.

ErrorMessage, type String, read access
Contains the last error message generated.

ErrorNumber, type integer, read access
Contains the last error number generated.

ExistsAction, type integer, write access
This property specifies what action to take if the destination file already exists.

- A value of 0 indicates the transfer should be canceled.
- A value of 1 indicates the destination file should be deleted and the transfer should continue.
- A value of 2 indicates the user should be prompted whether to delete the file or cancel the transfer.

The default is 0.

FileTransferStatus, type integer, read access
This property indicates the status of an in-progress or completed file transfer.

- A value of 0 indicates the last transfer was completed successfully.
- A value of 1 indicates the last transfer failed.
- Any other value indicates a file transfer or some other operation is in progress.

GroupPassword, type String, write access
The string used for the group password when connecting to the host's data server program.

HostAddress, type String, write access
The IP address used when attempting to connect to the host.

HostFile, type String, write access
This property specifies the name of the host file involved in the transfer. It may be fully qualified with the group and account if necessary. If the file is in or should be created in the temporary domain, the string ,TEMP (note the comma) should be appended to its name.

HostFileList (String ListMask), String, read access
This property returns a comma-delimited list of files on the host. The ListMask parameter specifies a valid ListF filename argument used to compile the list.

JobName, type String, write access
The string used for the job name when connecting to the host's data server program.

KeepTrailingSpaces, type Boolean, write access
This property specifies what to do with the trailing spaces on each record during ASCII downloads. A value of True indicates trailing spaces should be kept, while a value of False indicates trailing spaces should be deleted. The default is False.

LocalFile, type String, write access
This property specifies the name of the local file involved in the file transfer. It may be fully qualified with the drive letter and path if necessary.

LoginAccount, type String, write access
The string used for the account name when connecting to the host's data server program.

LoginGroup, type String, write access
The string used for the group name when connecting to the host's data server program.

LoginInfo, type String, read access
This property contains the current user name, group name, and account

name the server is running from. The first 8 characters contain the user name, the next 8 characters contain the group name, and the last 8 characters contain the account name.

LoginStatus, type Boolean, read access
Contains True if the login information supplied when connecting to the host's data server program was valid, else False.

LoginUser, type String, write access
The string used for the user name when connecting to the host's data server program.

Port, type integer, write access
A number that uniquely identifies the program on the host that the client wishes to connect to. This number may be any number between 1 and 32767, though a number above 30000 is recommended.

RecordSize, type integer, write access
This property specifies the record length of the host file for uploads. Valid values are between 1 and 4096. The default is 80 for ASCII transfers or 256 for binary transfers.

SpacesPerTab, type integer, write access
This property sets the distance between tab stops and enables changing tab characters to spaces to fill to the next tab stop. It is used only on ASCII uploads. A value of 0 disables the replacement of tab characters. Valid values are between 0 and 9. The default is 0.

TransferMode, type integer, write access
This property specifies the mode of the file transfer. Valid values are 0 for an ASCII file transfer, 1 for a binary file transfer, and 2 for a binary transfer that saves the file information in a file header.

UserPassword, type String, write access
The string used for the user password when connecting to the host's data server program.

Methods

Boolean Connect ()
This method attempts to connect to the host's data server program at the IP address specified by the HostAddress property. The Port property is used to identify the requested host program. Connect sets the ConnectStatus property and returns True if the connection request succeeded, otherwise it returns False.

void Disconnect ()
This method disconnects from the host's data server program.

integer HostCommand (String CommandString)
This method executes a MPE command or MPE command file. The CommandString parameter specifies the command or command file to be executed. A return of 0 indicates the command was executed successfully, while a return of 1 indicates the command failed.

Boolean HostLogin ()
This method changes the host server login to that specified by the LoginUser, UserPassword, LoginGroup, GroupPassword, LoginAccount, AccountPassword, and JobName properties. A return of True indicates the login was changed successfully.

void SaveConfig ()
This method saves the writeable session object properties to a file. This file can later be used to initialize a session object when created.

Boolean LoadConfig (String FileName)
This method loads the session object properties from the named file. A return of True indicates the properties were loaded successfully.

Boolean SetSessionConfig ()
This method displays the Session Configuration dialog of the file transfer access engine. This dialog allows the user to set these properties: HostAddress, Port, LoginUser, UserPassword,

LoginGroup, GroupPassword, LoginAccount, AccountPassword, and JobName. A return of True indicates the user clicked OK to exit the dialog; a return of False indicates the user clicked Cancel to exit the dialog.

Boolean SetTransferConfig ()

This method displays the File Transfer Configuration dialog of the file transfer access engine. this dialog allows the user to set default file transfer options. A return of True indicates the user clicked OK to exit the dialog; a return of False indicates the user clicked Cancel to exit the dialog.

integer StartTransfer ()

This method initiates a file transfer. A return of 0 indicates the file transfer was successful, while a return of 1 indicates the file transfer failed.

Example

Here is a Visual Basic sample for creating, using, and deleting the session object of the file transfer engine:

```
Private Sub cmdUpload_Click( )
Const gcUpload = 0
Const gcDownload = 1
Const gcASCII = 0
Const gcBinary = 1
Dim FTSession As Object
Dim ii As Integer
Dim TempStr As String
  On Error GoTo Egress1
  Screen.MousePointer = 11
StatusLine = "Begin..."
  Set FTSession = CreateObject("NetFT.Session")
StatusLine = "Object Created - Request Connection..."
  TempStr = tHostAddr.Text
  If (Len(TempStr) > 1) Then
   FTSession.HostAddress = TempStr
```

```
  End If
  FTSession.Port = 30001
  FTSession.LoginUser = "MGR"
  TempStr = tPassword.Text
  If (Len(TempStr) > 1) Then
   FTSession.UserPassword = TempStr
  End If
  FTSession.LoginGroup = "PUB"
  FTSession.GroupPassword = ""
  FTSession.LoginAccount = "MINISOFT"
  FTSession.AccountPassword = ""
  FTSession.JobName = "SAMPLE"
  If (Not FTSession.Connect) Then
   MsgBox "1 - " & FTSession.ErrorMessage
   GoTo Egress1
  End If
  ii = 0
StatusLine = "Connection Established - Logging on..."
  FTSession.HostLogin
  TempStr = FTSession.LoginInfo
  If (Len(TempStr) <> 24) Then
   MsgBox "2 - " & FTSession.ErrorMessage
   GoTo Egress1
  End If
  FTSession.DisplayStats = True
  FTSession.Direction = gcUpload
  FTSession.ExistsAction = 1
  TempStr = tLocalFN.Text
  If (Len(TempStr) > 1) Then
   FTSession.LocalFile = TempStr
  End If
  TempStr = tRemoteFN.Text
  If (Len(TempStr) > 1) Then
   FTSession.HostFile = TempStr
  End If
  FTSession.RecordSize = 80
  FTSession.TransferMode = gcASCII
StatusLine = "Logon Successful - Transfering..."
  If (FTSession.StartTransfer <> 0) Then
   MsgBox "3 - " & FTSession.ErrorMessage
   GoTo Egress1
  End If
```

```
StatusLine = "Transfer Complete."
  GoTo Egress2
Egress1:
  MsgBox "Error Exit"
Egress2:
  If FTSession.ConnectStatus = True Then
   FTSession.Disconnect
  End If
  Set FTSession = Nothing
StatusLine = "Ready"
  Screen.MousePointer = 0
End Sub
```

Programmer notes

Using manual dictionary mode

We recommend that the value of this property not be changed without extensive knowledge of your TurboImage database.

Using OLE automation engines from Visual C++

Type libraries that describe the API for the data access engine, file transfer engine, and stream access engine are included in the MSVC subdirectory of the MiddleMan installation directory. They are called MDMDA.TLB, MDMFT.TLB, and MDMSA.TLB, respectively.

They can be used with the Read Type Library option under the OLE Automation tab of the ClassWizard dialog box of Microsoft Visual C++. When a Type Library is read this way, a new class derived from COleDispatchDriver is created that will contain member functions to access the properties and methods of the requested OLE automation engine.

Notes:

_

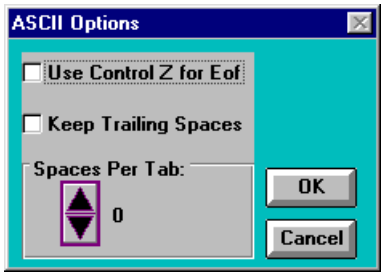
_

_

Chapter 3
Server Software

Server software structure

The MiniSoft server software consists of a master server program, a listener server, and many user-accessible server programs.



Master server program

MSSERVER.MM.MINISOFT is the master server program. It runs and manages all listeners. Streaming the job MSJOB.MM.MINISOFT starts the

master server. Once the job is running clients may connect to any of the services the master server is configured to provide. To shut down the master server, or add or delete listeners dynamically, see the sections *Server console client* and *MSJOB command utility*, below.

We recommend that the streaming of MSJOB.MM.MINISOFT be added to your normal system startup procedures. The default contents of MSJOB.MM.MINISOFT are usually sufficient for most sites. If you need to make a change, the following paragraphs explain how the master server program is configured.

The master server program is run as follows:

```
RUN MSSERVER.MM.MINISOFT;INFO=<console port> <listener prog>
```

where:

<console port> specifies a TCP/IP port number the client console program will use to connect to the master server with. This is typically 30000.

<listener prog> specifies the program to use as the listener program. This is SERVER.MM.MINISOFT.

When the master server is run, it checks its licensing information and then reads environment variables for each product it is licensed to run. The environment variables are named MSSERVER######, where ###### is the product number of the product being described. Product numbers are:

000002 - Network File Transfer

000008 - MiddleMan

000016 - FrontMan

000024 - Scout Query

000032 - Scope

The value of the environment variable has the form:

<port> <# of users> <server prog>

where:

<port> specifies the TCP/IP port number that client programs will use to connect to the server for this product.

<# of users> specifies the maximum number of clients that can connect to this server via the specified TCP/IP port number.

<server prog> specifies the name of the server program for this product.

Typical values for <port> and <server prog> for the various MiniSoft products are:

Network File Transfer: <port> = 30001, <server prog> = NFTSRVR.MM.MINISOFT S

MiddleMan: <port> = 30002, <server prog> = DATASRVR.MM.MINISOFT S

FrontMan: <port> = 30004, <server prog> = FMSRVR.MM.MINISOFT S

Scout : <port> = 30005, <server prog> = SCOUTCSP.MM.MINISOFT

The environment variable can specify more than one <port> and <server prog> to be used for the product by using the bar (|) character to separate the multiple specifications as such:

SETVAR MSSERVER000008 30002 10 DATASRVR.MM.MINISOFT|30009 5 DATASRVR.MM.MINISOFT

Job options

Tracing

For diagnostic purposes, a tracing facility is available. There are several levels of tracing available. The more detailed tracing levels can use a substantial amount of spool file space.

```
SETVAR MSTRACE 0
SETVAR MSTRACE 1
SETVAR MSTRACE 3
SETVAR MSTRACE 7
```

- Use one to output warnings.
- Use three to output information and warnings.
- Use seven to output a detailed trace, including information and warnings.
- If no variable is set, the default behavior is 0 (zero).

The standard NIFOBJ file for custom servers does not display trace information. A diagnostic version of this library is available if needed.

TEMP file space

TEMP files are not shared between connections. Each connection by each user gets a new TEMP file space. This should correct problems encountered by applications that expect to operate in separate sessions.

```
SETVAR MSTEMPFM 0
SETVAR MSTEMPFM 1
```

The variable MSTEMPFM will cause the created processes to share TEMP file space (including $STDLIST) with the job when set to 1 (one).

If no variable is set, the default behavior is 0 (zero).

File equations

There are now four options for handling file equations.

SETVAR MSFILEEQ 0
SETVAR MSFILEEQ 1
SETVAR MSFILEEQ 2
SETVAR MSFILEEQ 3

- When this value is set to zero or two, file equations are global in scope for the job.
- When this value is set to one or three, file equations are local in scope. Each process in the job must set any needed file equations.
- When this value is set to two or three, a command of MSFILEEQ is issued after the user is 'logged in'. The normal method of using this would be to have a command file or login UDC for each user that requires individual file equations.
- If no variable is set, the default behavior is 0 (zero).

Listener server

MiddleMan supplies a listener program (SERVER.MM.MINISOFT) that listens for connection requests from client programs and executes a new copy of the appropriate server program to service that request. After a server program is created and executed, the connection made with the client is passed to a new server program, and the listener program goes back to listening for connection requests.

In this way a server program interacts with only one client, since a separate copy of the server program is spawned for each client that requests a connection to that service. A separate listener program is set up by the master server program for each product licensed.

Data access server

The data access server (DATASRVR.MM.MINISOFT) allows access to TurboImage, KSAM, and ASCII (MPE) files. You may provide a parameter of S when setting up the DATASRVR program. This parameter indicates that all clients connecting to the server must provide valid login information.

File transfer server

The file transfer server (NFTSRVR.MM.MINISOFT) allows client software to move files to or from an HP 3000. Optionally, it will store the MPE specific file information (file code, etc.) in the file on the PC. You may provide a parameter of S when setting up the NFTSRVR program. This parameter indicates that all clients connecting to the server must provide valid login information.

Note: This file transfer server is not designed for POSIX files. If you need to move POSIX files, please use the FTP protocol.

Stream server

Custom server programs can be written with the Network Interface Library. These custom applications use the network interface in a raw or unprocessed format that is historically called a *stream*.

All custom server applications run as 'sons' of a listener process. The custom server applications are passed a value for the connection they are to use in subsequent calls to the Network Interface Library.

The Network Interface Library consists of four functions that allow custom server programs to interface with the LAN. Any server program that uses these functions must be run as a son program by the MiddleMan listener

program (SERVER). These functions assume that the MiddleMan listener program has already received a connection request from a client program and is giving the connection to the program it has called as son that is calling these functions.

The Network Interface Library is supplied as a NMOBJ file named NIFOBJ. This file can be linked with other NMOBJ files with this Link Editor syntax:

```
LINK FROM= <file1>,...,<filen>,NIFOBJ.MM.MINISOFT;...
```

NIFOBJ can also be included in a RL or XL file with the Link Editor.

Function NET_OPEN

Syntax:

```
return=NET_OPEN( secure, product )
```

Use:

NET_OPEN opens a connection to a client that has initiated a server program. This function must be called successfully before any other network interface functions can be called.

return **I32**

The return value is an integer that is the ID number of the connection. This ID number is used in all of the other network interface functions. If NET_OPEN cannot get a connection, -1 is returned.

Parameters:

secure **I32V**

32-bit signed integer, by value. If non-zero the client must provide valid user login information. If it is zero, the login information sent by the client may, but is not required to, be valid. The stream access engine sends login information to the server when it connects. To make sure valid information is sent,

the client must set the appropriate properties of the session object before calling the Connect method. See *Stream access engine* in chapter 2.

product **I32V**
32-bit signed integer, by value. This number needs to correspond to the VendorID property of the session object used by the client. For MiddleMan, this value is defaulted to 8.

Function NET_CLOSE

Syntax:
NET_CLOSE(*id***)**

Use:
NET_CLOSE disconnects from a client. Once NET_CLOSE is called, no further calls can be made to any network interface functions, including NET_OPEN.

Parameters:

id **I32V**
32-bit signed integer, by value. The connection ID number is returned by a call to NET_OPEN.

Function NET_READ

Syntax:
return=NET_READ (*id*,*buffer*,*length***)**

Use:
NET_READ reads data from a connection with a client. NET_READ blocks communication until the specified number of bytes of data have been read.

return **I32**
NET_READ returns the actual number of bytes received. It may be less than the number requested if the client has closed the connection. The return value is -1 if an error occurs.

Parameters:

id **I32V**
32-bit signed integer, by value. The connection ID number is returned by a call to NET_OPEN.

buffer **CA**
Character array. This is where the received data will be stored.

length **I32V**
32-bit signed integer, by value. The number of bytes to read.

Function NET_WRITE

Syntax:
return = NET_WRITE(*id*,*buffer*,*length***)**

Use:
NET_WRITE writes data to a connection with a client. NET_WRITE blocks communication until the specified number of bytes has been sent.

return **I32**
NET_WRITE returns 0 if the data is successfully written, or it returns -1 if an error occurs.

Parameters:

id **I32V**
32-bit signed integer, by value. The connection ID number is returned by a call to NET_OPEN.

buffer **CA**
Character array. This is the data to be transferred to the host.

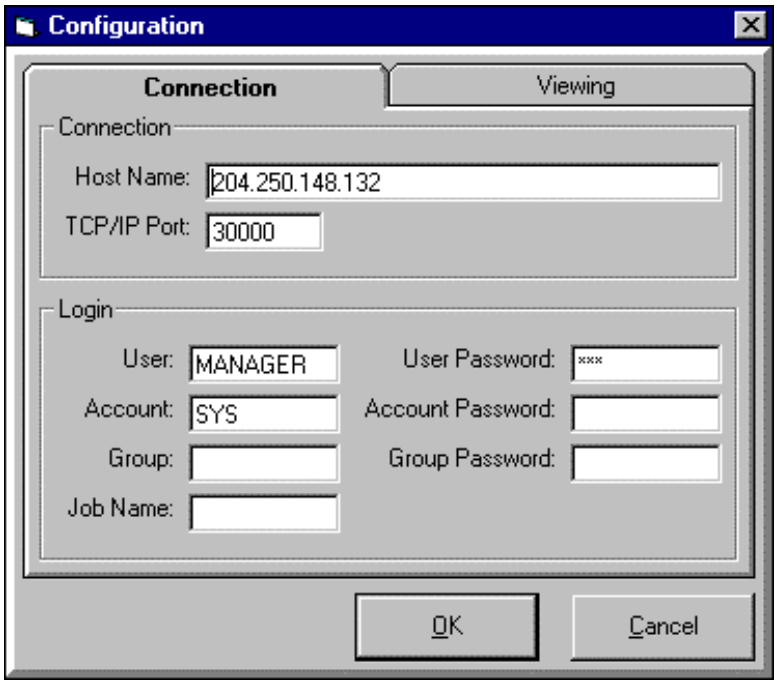
length **I32V**
32-bit signed integer, by value. The number of bytes to send.

Server console client

The server console client program provides listener and connection management for the MiddleMan master server. Since all servers started by clients are descendants of the master server program, the MPE/iX operating system does not see the individual connections as individual users. Tasks such as listing all users connected, disconnecting users, starting new listeners, deleting listeners, and shutting down the master server program are accomplished with the server console client.

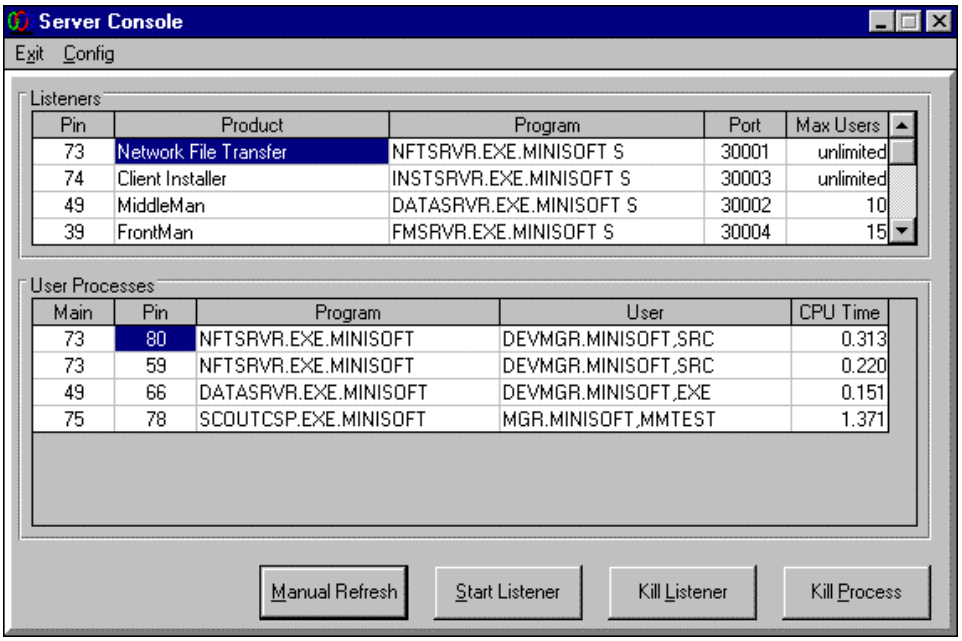
The server console client is run by double clicking on the server console icon in Windows 3.1 or selecting Server Console from the program list in Windows 95.

If the server console client has not been configured, the following configuration screen will be displayed:



The user you enter must have OP or SM capability.

When the server console client is connected to the master server program, the following screen is displayed:



The top grid lists all the listeners currently listening for connection requests. To select a listener, click on its row. You can stop the listener by pressing the Kill Listener button.

The bottom grid lists all the processes currently active. To select a process, click on its row. You can stop the connection by pressing the Kill Process button.

To exit the server console client, select the Exit Console menu selection from the Exit pulldown menu.

To stop the master server program and exit the server console client, select the Shutdown All Servers and Exit Menu Selection from the Exit pulldown menu.

To add a listener, click on the Start Listener button and the following screen is displayed:

MSJOB command utility

You may control the MSJOB from the command line of a job or session on the HP 3000 with the MSJOB command utility (MSJOBCMD). This program allows you to view the current servers and processes and optionally stop the execution of MSJOB. The program will obtain the current job or session logon information when passing the control request to the MSJOB server.

The program has three PARM options:

- PARM=0 - Lists the current servers and active users.
- PARM=1 - List as PARM=0 and if no users, will stop the job.
- PARM=2 - List as PARM=0 and will stop the job with any user count.

For example:

```
RUN MSJOBCMD;PARM=1
```

Notes:

Index

T

V

X