



eQuate

Table of Contents

Overview.....	1
The Basic Steps	3
Quick Start.....	5
Publishing and Runtime Options	6
Runtime Modes	9
Screen Mode.....	9
Command Mode	9
Components	11
Action Editor	11
Administration	11
Application Manager	11
Capture.....	11
QPort T27 and UTS Config	11
Script Editor	11
Session Manager	11
Unpackage	12
WinQ T27 and UTS Config	12
User Sign On	13
User Id.....	13
Current Password	13
New Password (if changing)	13
Administration	15
eQuate Administrator.....	15
File menu	15
User Registration tab	15
Database Management tab	17
Runtime Publish	18
Runtime Packager tab	18
User or General Profile	19
Application Access	19
Add Application Access.....	19
Database.....	19
Application Name.....	19
Application Sign-On Script	19
Edit Application Access.....	20
Database	20
Application Name.....	20
User Application Options (User Defined)	20
Application Sign-On Script	20
Network User Station Name	21
Select General Profile.....	21
eQuate Database Backup.....	21
Backup File Directory	21
Backup.....	21
Cancel.....	21
eQuate Restore	21
Backup File Directory	21
Available Backup Files	21
Restore Options.....	21

Selected Files	21
Restore	21
Cancel	21
Application Manager	23
eQuate Application Manager	23
File menu	23
Database tab	23
Applications tab	23
Forms tab	24
Pick Values tab	24
Help Text tab	24
Standard Messages tab	25
Host Error Ids tab	25
eQuate 3.5 Language Translations	27
How it Works	27
How to Use Translation in an Application	29
Translation Usage Reporting	30
Activating Translations	33
Using Translations	33
Applications	35
eQuate Applications	35
Application Name	35
Terminal Type	35
Connection Route	35
Method of Starting Application	35
Default Form	36
When Form Id String Not Found	36
Select Form	36
Current Selection	36
List of Available Forms for the Application	36
Finder	36
Forms	37
eQuate Form Manager	37
Form Name	37
Data Fields	37
Form Description (optional)	37
Form Id String	37
Form Mode	37
Date Last Changed	37
Import Form	37
Form Designer	39
eQuate Form Designer Toolbar	39
General Features	39
The eQuate Form	39
The eQuate Form Designer Tool Bar	39
Adding Controls and Fields	39
Moving and Sizing Controls	39
Aligning and Sizing Controls	39
Automatic Properties Assignment	40
Automatic Alignment to Client Area	40
Save and Close button	40

Edit menu.....	40
View menu.....	41
Tools menu.....	41
Cancel button.....	41
Help menu.....	41
The Designer buttons.....	41
Property Editor.....	44
Controls.....	45
Form Properties.....	45
Form Initial Action.....	45
Form Activate Action.....	45
Host Message Action.....	45
Host Error Action.....	45
Screen Complete Check Action.....	45
Form Properties.....	46
Text Labels.....	46
Edit Boxes.....	47
Command Buttons.....	48
Speed Buttons.....	48
Check Boxes.....	49
Option Buttons.....	50
List Boxes.....	51
Drop-down List Boxes.....	52
Multi-column List Boxes.....	53
Memo.....	54
Bevels.....	55
Button Groups.....	55
Group Boxes.....	56
Panels.....	57
Splitters.....	58
Images.....	58
Media Players.....	59
Date/Time Labels.....	59
Browser.....	60
URL Link.....	61
Control Properties.....	63
Align Property.....	63
Alignment Property.....	63
AllowAllUp Property.....	63
AutoCenter Property.....	63
AutoSize Property.....	63
AutoSnap Property.....	63
BackColor Property.....	63
BevelInner Property.....	63
BevelOuter Property.....	63
BevelWidth Property.....	63
Bitmap Property.....	64
BitmapPosition Property.....	64
BlinkBackColor Property.....	64
BlinkColor Property.....	64
BlinkForeColor Property.....	64

Blinking Property	64
BlinkIntervalOff Property	64
BlinkIntervalOn Property	64
Border Property	64
BorderStyle Property	64
BorderWidth Property	64
ButtonStyle Property	64
Cancel Property	65
Caption Property	65
Center Property	65
CharCase Property	65
Checked Property	65
ColumnHeaders Property	65
Columns Property	65
Ctl3d Property	65
DataSource Property	65
Default Property	65
Down Property	65
EditMask Property	66
Enabled Property	66
FillFormWhenMaximized Property	66
Flat Property	66
FlatColor Property	66
Font Property	66
ForeColor Property	66
Format Property	66
FrameStyle Property	66
GroupIndex Property	66
Height Property	66
Help Property	67
Hint Property	67
ItemIndex Property	67
Items Property	67
Left Property	67
Lines Property	67
MaximizedButton Property	67
MaxLength Property	67
MinimizedButton Property	67
MinSize Property	67
MonoChromeButtons Property	67
Name Property	67
NumBitMaps Property	68
NumericSort Property	68
ParentColor Property	68
ParentCtl3d Property	68
ParentFont Property	68
ParentShowHint Property	68
PassWordChar Property	68
PickList Property	68
Picture Property	68
ReadOnly Property	68

ScrollBars Property	69
Shape Property	69
ShowAccelChar Property.....	69
ShowHint Property.....	69
SortColumn Property	69
SortDescending Property	69
Sorted Property.....	69
Strech Property	69
Style Property	69
TabOrder Property	69
TabStop Property	69
Text Property.....	69
Top Property	70
Transparent Property	70
TransparentColor Property.....	70
TransparentMode Property.....	70
URL Property	70
Visible Property	70
WantsReturns Property	70
Width Property.....	70
WindowState Property.....	70
WinHelpFile Property.....	70
WordWrap Property	70
Property Dialogs	71
Parent Controls	71
Color Selection.....	71
Standard Color	71
Custom.....	71
Select Screen Data Field Source	71
Field Name	71
Copy Selected to Clipboard	71
Select	71
Clear.....	71
Select Repeating Field Index.....	71
Select Desired Repeat Index	71
Help Selector	71
Current Selection.....	71
Clear.....	71
Use Windows Help	71
WinHelp Context Id.....	72
Available Help	72
Finder	72
Edit Mask	72
Input Edit Mask	72
Pre-defined Standard Edit Mask.....	73
Character for Blanks	73
Save Literal Characters	73
Test Input.....	73
Pick List Selector	73
Current Selection.....	73
Clear.....	73

Available Pick Lists.....	73
Finder	73
Multi-Column List Setup	74
Number of Columns	74
Row height	74
Column Header Options.....	74
Dividers.....	74
Column Size and Header Data	74
Column n Display Alignment	74
Date Time Format Editor	74
Date Time Format.....	74
Predefined Formats.....	75
Global Controls	77
eQuate Menu Designer.....	77
Menu Item Caption	77
Name.....	77
Short Cut.....	77
Insert Item	77
Indent Level.....	77
Move.....	77
Checked	77
Visible	77
Enabled	77
Preview	77
Menu Items	77
Action Key Assignments	77
Available Action Keys	78
Right arrow button.....	78
Left arrow button.....	78
Left double arrow button	78
Assigned Action Keys	78
Action	78
Data Fields.....	79
eQuate Form Data Fields	79
Data Field List	79
Name.....	79
Location	79
Repeating Field.....	79
Data Type.....	80
Other Attributes	80
Justified (in Screen)	80
Apply	80
Undo.....	80
Form Import.....	81
eQuate Form Definition Import.....	81
Import Source.....	81
Import File Name.....	81
Convert Background Text to Labels	81
Import	81
Form Import View and Adjust	81
Import Data Fields	81

Generated Text Labels.....	81
Preview Layout.....	81
Edit Data Field Name	81
Enter a Unique Control Name	81
Form Merge/Update.....	82
Merge/Update Options.....	82
Field View/Edit	82
Imported Label Edit	82
Text.....	82
Row	82
Col.....	82
Host Error Identification	83
Host Error Detection String Edit	83
Error Id. String.....	83
Error Action	83
Standard Messages	85
Standard Messages.....	85
Message Number	85
Message Text	85
Example:.....	85
Pick Lists.....	87
Pick List Edit	87
Available Pick Lists.....	87
Pick List Values and Descriptions	87
Edit Value.....	87
Edit Description	87
Up	87
Down	87
Update	87
Append	87
Remove.....	87
Sort List By.....	87
Re-sort.....	87
Uppercase Descriptions	87
Uppercase Values.	87
Save to File.....	87
Load from File	87
Print List.....	88
Help Text	89
eQuate Help Text Edit	89
Help Text.....	89
File menu	89
Edit menu.....	89
eQuate Action Script Editor.....	91
eQuate Action Script Editor.....	91
File menu	91
Edit menu.....	92
Search menu.....	92
Bookmarks	93
Options menu	93
Window	93

Help.....	93
Editor Properties	94
Edit Window Font.....	94
Tab Size	94
Highlight Colors.....	94
OK.....	94
Cancel.....	94
Help.....	94
Language Elements	95
Comments.....	95
Statements.....	95
Line Continuation Character.....	95
Numbers	95
Variable and Constant Names	95
Variables.....	97
Variable Types	97
Variant.....	97
Variants and Concatenation	97
Other Data Types	97
Scope of Variables.....	98
Declaration of Variables.....	98
Flow of Control	99
Control Structures	99
The GoTo	99
The Do Loops.....	99
The While Loop	99
The For ... Next Loop	99
The If and Select Statements.....	99
Subroutines and Functions	101
Subroutine and Function Naming Conventions.....	101
ByRef and ByVal.....	101
Calling Procedures in DLLs.....	102
Files	103
File Input/Output.....	103
Arrays	105
Arrays.....	105
User Defined Types	107
User Defined Types.....	107
Dialogs and Dialog Controls.....	109
Dialog Support.....	109
OK and Cancel	109
List Boxes and Drop-down List Boxes.....	110
Check Boxes in Dialog.....	111
Text Boxes and Text	112
Option Buttons and Group Boxes.....	113
The Dialog Function	114
The Dialog Box Controls	114
The Dialog Function Syntax	114
OLE Automation	117
What is OLE Automation?	117
Accessing an Object.....	117

What is an OLE Object?	118
OLE Fundamentals	118
OLE Automation and Word example	119
Data Types, Operators and Precedence	121
Data Types, Operators and Precedence	121
Data Types	121
Arithmetic Operators	121
Operator Precedence	121
Relational Operators	121
Comparison Operators	121
Functions, Statements, Subroutines and Events	123
Abs Function	123
AppActivate Statement	123
Asc Function	123
Atn Function	123
Beep Statement	124
Begin Dialog Statement	124
CalendarDialog Function (eQuate)	125
Call Statement	125
CancelButton Statement	125
CBool Function	126
CDate Function	126
CDBl Function	126
ChangeCursorStyle Subroutine (eQuate)	127
ChDir Statement	127
ChDrive Statement	128
CheckBox Statement	128
Choose Function	129
Chr Function	129
CInt Function	129
ClearDisplay Subroutine (eQuate)	129
ClearFields Subroutine (eQuate)	129
ClearScreenChanged Subroutine (eQuate)	130
CLng Function	130
Close Statement	130
CloseApp Subroutine (eQuate)	130
ClosePage Subroutine (eQuate)	130
ComboBox Statement	131
Const Statement	131
Cos Function	132
CreateObject Function	132
CSng Function	133
CStr Function	133
CurDir Function	133
CVar Function	134
Date Function	134
DateSerial Function	134
DateValue Function	135
Day Function	135
Declare Statement	135
Dialog Function	136

Dim Statement.....	137
Dir Function.....	138
DlgEnable Statement	138
DlgText Statement	139
DlgVisible Statement	139
Do...Loop Statement.....	140
DoTerminalKey Subroutine (eQuate).....	140
T27 Constants:.....	140
UTS Constants:	141
DropListBox Statement	142
End Statement.....	143
EOF Function	143
eQuateNavigate Function.....	144
Erase Statement	144
Exit Statement.....	145
Exp Function	145
FileCopy Function	145
FileLen Function	145
FileDialog Function (eQuate).....	145
FileDialog Function (eQuate)	146
Fix Function.....	146
For Each...Next Statement.....	146
For...Next Statement	147
Format Function	147
FreeFile Function	152
Function Statement	152
Get Statement	153
GetColor Function (eQuate)	153
GetCurrentLanguage Function (eQuate)	153
GetCursorCol Function (eQuate)	154
GetCursorField Function (eQuate).....	154
GetCursorFieldRep Function (eQuate)	154
GetCursorRow Function (eQuate)	154
GetMemoSelection Function (eQuate)	154
GetNumericProp Function (eQuate).....	154
GetObject Function.....	154
GetPage Function (eQuate).....	155
GetPages Function (eQuate)	155
GetPassword Function (eQuate)	155
GetReservedString Function (eQuate)	155
GetScreenAttribute Function (eQuate).....	156
GetScreenData Function (eQuate)	157
GetScreenLine Function (eQuate)	158
GetSessionVar Function (eQuate)	158
GetState Function (eQuate)	158
GetString Function (eQuate)	159
GetStringProp Function (eQuate).....	159
GetTranslation Function (eQuate)	159
GetUser Function (eQuate)	159
GetUserAppOptions Function (eQuate)	160
GetUserOptions Function (eQuate).....	160

GetUserType Function (eQuate)	160
Global Statement	160
GroupBox Statement	161
GoTo Statement.....	161
Hex Function.....	162
Hour Function	162
If...Then...Else Statement	162
ImageOpenDialog Function (eQuate)	163
ImageSaveDialog Function (eQuate).....	163
Input Function	164
InputBox Function	164
InStr Function.....	165
Int Function.....	165
IsArray Function.....	165
IsDate Function.....	165
IsEmpty Function	166
IsNull Function.....	166
IsNumeric Function.....	166
IsObject Function	166
Kill Statement	167
LBound Function.....	167
LCase Function.....	168
Left Function.....	168
Len Function	169
Let Statement.....	170
Line Input # Statement.....	170
ListBox Statement	170
ListClear Subroutine (eQuate).....	171
ListColHeader Subroutine (eQuate).....	171
ListCount Function (eQuate)	171
ListGetColText Function (eQuate)	171
ListGetIndex Function (eQuate).....	172
ListGetItem Function (eQuate)	172
ListItemAdd Subroutine (eQuate)	172
ListItemRemove Subroutine (eQuate)	172
ListSetColText Subroutine (eQuate)	173
ListSetIndex Subroutine (eQuate).....	173
ListSetItem Subroutine (eQuate).....	173
LoadImage Function (eQuate).....	173
LoadMMFile Function (eQuate)	173
LoadUserList Function (eQuate).....	174
LOF Function.....	174
Log Function	175
Mid Function	175
Minute Function.....	175
MkDir Statement	176
Month Function	176
MsgBox Function, MsgBox Statement.....	176
Name Statement	178
Now Function.....	178
Oct Function	178

OKButton Statement.....	178
On Error Statement	179
Open Statement.....	181
Option Base Statement	182
Option Explicit Statement.....	182
OptionButton Statement.....	182
OptionGroup Statement	183
Print # Statement	183
Print Statement.....	184
PrintBeginDoc Subroutine (eQuate)	185
PrintDlg Function (eQuate)	185
PrintDraw Subroutine (eQuate)	185
PrintEndDoc Subroutine (eQuate)	185
PrintMoveTo Subroutine (eQuate).....	186
PrintNewPage Subroutine (eQuate).....	186
PrintPageHeight Function (eQuate)	186
PrintPageWidth Function (eQuate)	186
PrintRect Subroutine (eQuate)	186
PrintSetFont Subroutine (eQuate).....	187
PrintSetFontSize Subroutine (eQuate).....	187
PrintSetFontStyle Subroutine (eQuate).....	187
PrintSetOrientation Subroutine (eQuate)	187
PrintTextHeight Function (eQuate).....	187
PrintTextWidth Function (eQuate).....	188
PushButton Statement	188
Put Statement.....	189
Randomize Statement.....	189
ReDim Statement.....	190
Rem Statement.....	190
Right Function.....	190
Rmdir Statement	191
Rnd Function	191
ScreenChanged Function (eQuate).....	192
Second Function.....	192
Seek Function	192
Seek Statement	192
Select Case Statement	193
Send Subroutine (eQuate)	194
SendKey Subroutine (eQuate).....	194
SendKeys Statement	194
SendMail Subroutine (eQuate)	195
Set Statement	196
SetColor Subroutine (eQuate)	196
SetCurrentLanguage Function (eQuate).....	198
Related Topics: SetCursor Subroutine (eQuate).....	198
SetCursorField Subroutine (eQuate).....	198
SetExecState Subroutine (eQuate)	198
SetFocus Subroutine (eQuate)	198
SetMemoSelection Subroutine (eQuate)	198
SetNumericProp Subroutine (eQuate)	199
SetPage Subroutine (eQuate).....	199

SetScreenData Subroutine (eQuate)	199
SetSessionVar Subroutine (eQuate)	200
SetState Subroutine (eQuate)	200
SetString Subroutine (eQuate)	200
SetStringProp Subroutine (eQuate)	201
Sgn Function	201
Shell Function	201
ShowFormById Subroutine (eQuate)	202
ShowFormByName Subroutine (eQuate)	202
ShowHelp Subroutine (eQuate)	202
ShowPickList Function (eQuate)	202
Sin Function	202
Space Function	202
Sqr Function	203
Static Statement	203
Stop Statement	203
Str Function	204
StrComp Function	204
String Function	204
Sub Statement	205
Tan Function	205
Text Statement	205
TextBox Statement	206
Time Function, Time Statement	206
Timer Event	207
TimeSerial Function	207
TimeValue Function	207
Trim, LTrim, RTrim Functions	208
Type Statement	208
UBound Function	209
UCase Function	210
UserName Function (eQuate)	210
UserType Function (eQuate)	210
Val Function	211
VarType Function	211
Wait Subroutine (eQuate)	211
Weekday Function	211
While...Wend Statement	212
With Statement	212
Write # - Statement	213
XmitCursor Subroutine (eQuate)	214
XmitFrom Subroutine (eQuate)	214
Year Function	214
Predefined Constants	215
Predefined Constants	215
Color Types (also see GetColor and SetColor):	215
Defined Colors (also see SetColor):	215
Message Box Constants (also see MsgBox):	216
Print Font Styles (also see PrintSetFont):	216
Screen Attributes (also see GetScreenAttribute):	216
Scrollbar Types (see Memo control):	217

Transparent Mode	217
State Types (also see GetState and):	217
T27 Constants (see DoTerminalKey):	217
UTS Constants (see DoTerminalKey):	218
Form Capture	221
eQuate Form Capture.....	221
Open Terminal	221
Capture Screen	221
WinQUTS32: routename.....	221
File menu	221
Edit menu.....	221
Options menu	222
Input Recall window	223
WinQT27: routename.....	223
File menu	223
Edit menu.....	223
Options menu	224
Input Recall window	224
Edit Capture	225
File menu	225
Edit menu.....	225
Option menu	225
Window menu	225
Modifying Field Definitions	225
Repeat Capture	226
Repeat Count	226
Rows in Repeat	226
Session Manager.....	227
eQuate Session Manager	227
Available eQuate Applications.....	227
Run Application	227
Close Application	227
File menu	227
Options menu	227
Run menu.....	228
Profiles.....	228
Application.....	228
User Route.....	228
Save and Close.....	228
User Sign On	228
User Id.....	228
Current Password	228
New Password (if changing).....	228
Change eQuate Runtime Settings Location.....	229
Delete Source Settings	229
Set	229
Cancel.....	229
Unpackage	231
eQuate Runtime File UN-Packager	231
Runtime package distribution file	231
Destination database path	231

Global Maintenance	233
eQuate Global Maintenance Utility	233
Global Maintenance.....	233
Global Maintenance Script Syntax	233
Global Maintenance Script Editor	234
Command Mode Commands	237
ECM Command Format and Conventions	237
Command Mode Record Format	237
Conventions Illustrated	237
Enclosure Characters	237
CTL.....	237
DTAFLD.....	238
DTAREC	238
ERRLST.....	239
FLDCTL	239
LOCKFRM	240
LSTDTA.....	240
NEWFRM	240
POSCURS	240
REFRESH.....	241
UAKCTL.....	241
UMNCTL	241
UNLOCKFRM	241
Defining ECM Control Strings in COBOL.....	241
Constructing ECM Data Records in COBOL	242
An eQuate Command Mode Example.....	243
The Complete Set of ECM Actions	250
ECMSTART Form.....	250
ECMCUST1 Form	251
ECMCUST2 Form	253
ECMCYST3 Form.....	253
The Complete COBOL Program	254
How to	273
Initialize eQuate User Databases	273
Assign User Privilege.....	273
Create an Application Database.....	273
Configure a Host Connection.....	274
Capture a Screen	275
Capture.....	275
Delete Unwanted Fields	275
Field Naming and Attributes.....	275
Selecting a Form Id.....	275
Setting Repeating Rows.....	275
Generating the Form Definition.....	276
Create an Equate Application	276

Overview

eQuate provides a Windows-based graphical front-end to legacy host applications. eQuate replaces text screens with enhanced windows. eQuate not only presents data in a graphical user interface (GUI), but also provides the additional functionality required to make applications easier to use. With eQuate, the developer can add command buttons, check boxes, option buttons, button bars, user menus, action keys, standard list boxes, drop-down list boxes, multi-column list boxes, and even pictures that can trigger automatic actions. Powerful eQuate Actions, or mini-programs, allow manipulation of all application screen data and eQuate Windows controls. eQuate Actions can access local PC databases and other Windows applications using a flexible scripting language, Enable. Selection lists for specific input data fields can be created and maintained within eQuate, as can context sensitive on-line help by field and window. eQuate also provides powerful input edit masking for input fields using either one of several standard input formats or customized input formats. eQuate allows the reengineering of the output screen or window without the necessity of altering legacy, host application programs.

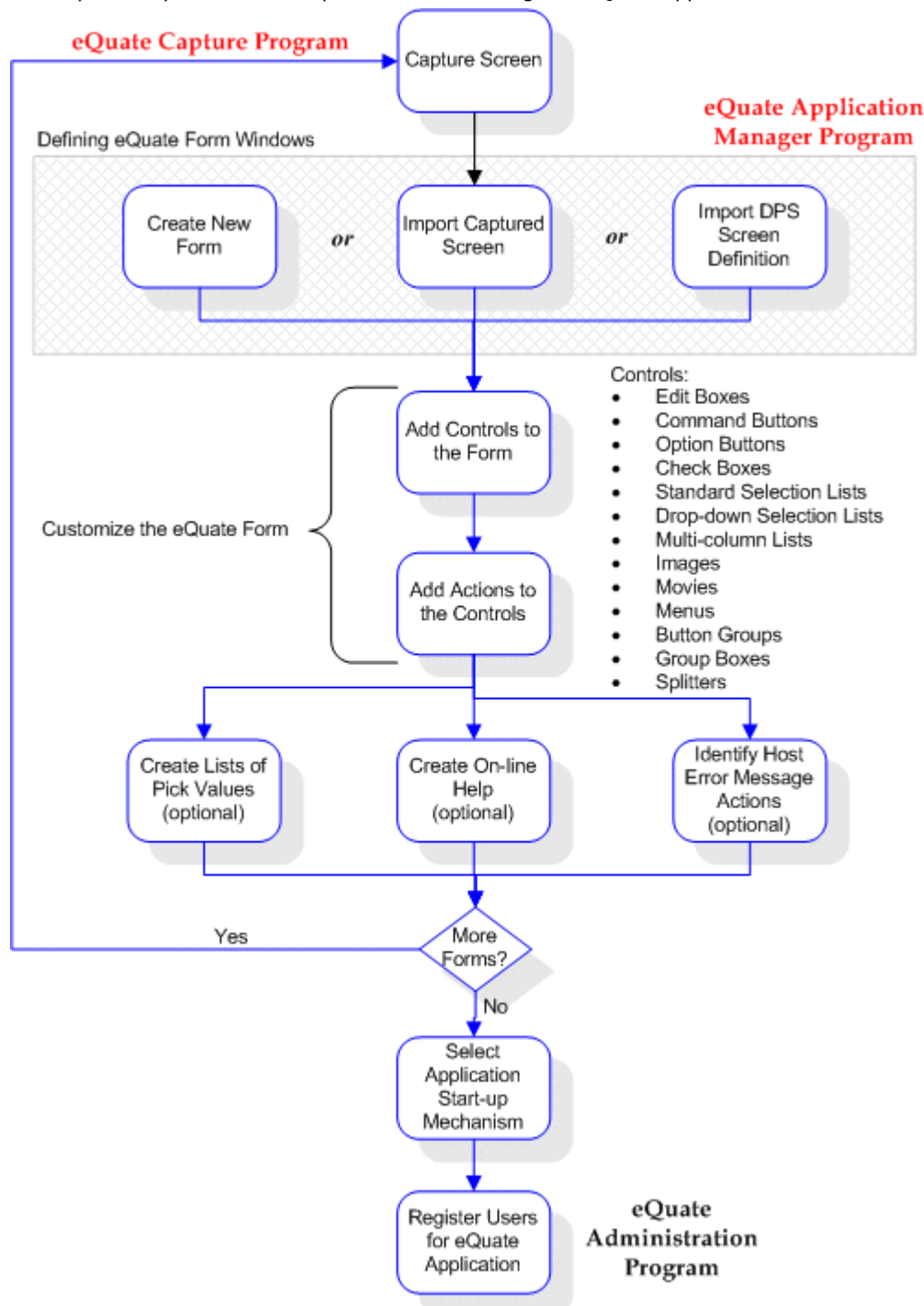
An additional facility, eQuate command mode, provides the next step to moving applications to a full client-server environment. Command mode provides a much more efficient use of eQuate, as it is not dependent on host application formatted screens.

The Enable Scripting Language (included with eQuate) allows the eQuate Developer to create scripts that can launch and manipulate other applications via Object Linking and Embedding (OLE) automation, access Dynamic Link Libraries (DLL) and automate complex or repetitious tasks.

See also, The Basic Steps and Quick Start.

The Basic Steps

This topic briefly illustrates the process used to design an eQuate application:



See also, Quick Start, eQuate Runtime Modes, Equate Components.

Quick Start

This is a step-by-step guide to assist you when using eQuate for the first time. This guide will not show the full application or features of the product, but will help in getting started. If you have any questions or problems, please contact us.

(770) 635-6363

support@kmsys.com

Use the following procedure to create an eQuate database and application initially:

1. Run the administration program from the Windows Start menu (from the Windows Taskbar, select Start | Programs | KMSYS Worldwide | eQuate | Administrator). The default User Id and Current Password are ADMIN and EQUATE, respectively. Once the administration program is open, you can add a new user id for yourself and delete or change the default ADMIN user id. Note: Any new user id added will have an initial password of EQUATE but may be changed with the Administrator program.

The next step is to create an Application Database. Choose the Database Management tab and select Application Database as the Database Type. Next, click the New Database button to initialize the database files necessary to create applications. You will be prompted to name the database directory. This name is only visible to eQuate Administrators and Developers allowing them to group databases with logical names. Once this step is completed, you will see a list of eQuate application database files that were initialized and you can close the Administrator program.

2. Now, run a WinQ Configuration program from the Windows Start menu. There are two different programs. If your host is a Unisys Clearpath Plus OS 2200 Server, run the WinQ UTS Config program. If your host is a Unisys Clearpath Plus MCP Server, select WinQ T27 Config. You will need to set up at least one connection to your host at this point. Also, note the Route Name, as it will be needed when the eQuate Session Manager is first run.

From the main WinQ Configuration window, select Open from Default Location from the File menu. Next, click the Configure Connections button. From the Configure Connections window, use the control to setup routes to your particular host(s). Note: Select Contents from the Help menu to receive instructions on how to use these controls. When you have completed the necessary routes, click the Save and Close button.

From the File menu, select Save To and choose a location where you wish to maintain the WinQ configuration files. This location should be in a non-shared location. Later in this procedure, the publish function will copy these files to a shared file server. The location chosen must have a sub directory that is either "T27" or "UTS" depending on the type of host connection required. It is into one of these two directories that you save the WinQ configuration. You may use the defaults %APPDATA%\KMSYS Worldwide\equate\3.50\Connections\T27 or %APPDATA%\KMSYS Worldwide\equate\3.50\Connections\UTS, where the system variable, %APPDATA%, represents the drive and path to the Windows "Application Data" folder.

3. Run the Capture program from your Windows Start menu. This program will allow the eQuate Administrator or Developer to capture screens from the built-in emulator and create Windows-like forms. The first step is to open the terminal to establish your connection to the host. If you used the "Save To" option suggested in the previous step, select the Alternate location option prior to clicking the Open Routes button. When the Select Alternate Connections Directory dialog appears, you will be able to select the directory where you saved the WinQ Configuration. Next, select the configured route and click the OK button. Once connected and signed on, navigate to the screen you want to capture and click the Capture Screen button on the eQuate Form Capture window. The Capture program will read the current screen and create a basic form. Next, click the Generate Form Definition button and save it to a file to be used later. You can repeat these steps to capture as many screens as you want before exiting.

Note: If your host is a 2200 and your screen definitions are stored in a DPS form file, you may skip the capture step and import the form definitions from downloaded files created by DPS's Form Language Manipulation Utility (@FLMU,F). These files may be imported directly into the eQuate Application Manager. The following is an example of the execution of the FLMU processor and editing the result:

```

▶@ASG,A TEMP.
▶I:002333 ASG complete.
▶@USE SCREEN$OMN.,TEMP.
▶I:002333 USE complete.
▶@FLMU,F your-form-file.
▶FLMU 6R4NQ1 6-8-56 Form Language Manipulation Utility 2007 Feb 15 Thu 1008:03
▶ENTER FORM NUMBER
▶75
▶          1 FORMS CONVERTED TO FLDP SOURCE.
▶ENTER FORM NUMBER
▶@EOF

```

```
►FLMU COMPLETE
►@ED,R TEMP.FLDP-75
```

4. Open the Application Manager from the Windows Start menu. Select the database that was created in the Administration program. Next, choose the Applications tab and add a new application. You will need to give the application a name that both the administrator and users will access. This will also prompt you for the terminal type (UTS or T27) and other information that is used for that application.

Once the Application is created, you can start to import the forms that were created in the Capture program or downloaded from a DPS form file. Move to the Forms tab and add a form. If you are importing forms captured or downloaded previously, click the Import Form button and open the saved file.

Important: Next, give the form a Form Name and select a Form Id String. A Form Id is needed for each form to allow eQuate to recognize which form is needed by comparing a string on the terminal screen to the one you define for the form.

Click the Form Design button to customize your forms, add and delete controls and menu items, add actions and more. The eQuate Form Designer is a VB-like development environment where you add or select a Windows control and specify what action is to take place when the user selects that control at run time. Note: For tips and examples, be sure to read the on-line help for the eQuate Form Designer.

The final step in creating your application for the user's to access is to create runtime files. When you exit the Application Manager, the runtime files are automatically saved, or you may select File | Create Runtime Files Now from the eQuate Application Manager Window. Creating runtime files separate from the design database files allow simultaneous development and execution of eQuate applications. Furthermore, runtime files are the only files accessed by the users when they run the Session Manager. The database design files will never be accessed by the user for reasons of security.

5. Once again, return to the Administration program. Each user will need to be granted access to the applications. Select the user and press the Edit Profile button to add the new application to the user's profile.

After the user has been assigned the proper applications, you will also need to assign a sign-on script to each application. The sign-on script allows the user to connect to the host when the application is initially started by the eQuate Session Manager. Two general-purpose scripts are provided by KMSYS Worldwide: soT27.bas for MCP access and soUTS.bas for 2200 access. Initially, they reside in the "Scripts" directory on the local PC where the eQuate Developers' Edition was installed (see %APPDATA%\KMSYS Worldwide\eQuate\3.50\Scripts, where the system variable, %APPDATA%, represents the drive and path to the Windows "Application Data" folder). You may alter a copy of these scripts to fit the specific requirements of your site; however, they were written in a manner to function properly without modification at most sites.

Once all the applications have been assigned to the users, the runtime files should be automatically created when exiting the Administration program.

6. The Session Manager is ready to be run. The installation of the eQuate Developers' Edition will install all the necessary files to run the Session Manager.

The Session Manager has two directories that will need to be configured from the File menu upon first use. The Runtime Directory is the directory from which you will test the eQuate application (default, %APPDATA%\eQuate\3.50\Data, where the system variable, %APPDATA%, represents the drive and path to the Windows "Application Data" folder). For this "Quick Start" procedure, it is initially on the developer's PC. Later, when the eQuate application is ready for production, the eQuate Administrator program will be used to "publish" the application which will distribute it to a shared file server (see below). The Script Directory is the location of the sign-on scripts that are used to connect to the host. This directory can also be on the computer's hard drive or a file server. Both of these directories are created during installation for first time installs.

Finally, select Profiles from the File menu and assign a Route to the eQuate Application. This step is not required if you assigned a Connection Route to the application in the Application Manager, or if you simply wish to override that setting.

Publishing and Runtime Options

When it is time to give users access to an eQuate application, use the eQuate Administrator program to distribute all the runtime files to a shared server. Use the following procedure to publish eQuate Application Databases to a shared server:

On the eQuate Administrator dialog, select the Runtime Publish tab. Use the select buttons to set the source and destination locations where you maintain your eQuate Database (includes user registration configuration and eQuate Application Databases) and runtime scripts. Click the copy buttons to distribute the runtime files to the network server.

Note: Users must have read access to the four "Destination" directories.

Note2: Regardless of whether your users are to execute the standalone eQuate Session Manager or run it from a Web page utilizing eQuate Web, always use this step to distribute the directories.

Runtime Modes

eQuate provides two modes that may be used to run an eQuate application. The following is a brief description of the two modes:

Screen Mode

This mode is used where eQuate applications will have to run on real terminals, or terminal emulators, as well as under eQuate control. In screen mode, the host application program does not know whether the user is working at a terminal or is in an eQuate session. eQuate relies on data being positioned on the same screen layout that the user would see on a terminal. No changes to the host application program are required in screen mode of operation; however, there are some features available in screen mode that may be enhanced with some program changes.

In screen mode, the eQuate Session Manager receives host messages as formatted screens, just like a terminal or terminal emulator. Each message must be decoded and mapped as if a display screen were being produced. eQuate keeps the display screen in an internal buffer not visible to the end user. Once the message is mapped to a display format, eQuate checks to determine if a new eQuate Form is needed by checking a specified location on the screen for a screen identifier string. If a form change is required, eQuate dynamically displays the corresponding eQuate form window. Once the correct form window is displayed, eQuate moves required data fields from the mapped display buffer to the eQuate form window. The eQuate form window is then available for user interaction.

When a window is sent from the eQuate Session Manager to the host application, data fields are moved from the form window to the mapped display buffer. The mapped display buffer is then transmitted to the host in the same format as it would be transmitted from a terminal, thus, the host application need not be aware of the presence of eQuate.

Command Mode

This mode provides a much more efficient use of eQuate as it is not dependent upon host application formatted screens. In command mode, the host program does not process input and output based on display screen formats. Instead, all communication between the host application and the eQuate session is based on eQuate command records. This method of operation eliminates the transmission of unnecessary screen control code and relieves the eQuate Session Manager of the task of mapping the host message to a screen format.

eQuate command records specify when a new eQuate form window is to be created rather than requiring eQuate to scan every input message. Instead of mapping data to a screen format, the eQuate Session Manager can move data fields directly from input records to the eQuate form window.

Components

The eQuate Development Edition comes with eleven (11) programs described below:

Action Editor

The eQuate Action Editor program provides a text editor that may be used to create and maintain eQuate actions that will be executed when the user selects an assigned control. Actions may be easily syntax checked using the editor. Actions may be saved in binary form for reasons of security.

Administration

The eQuate Administration program provides a mechanism to register users for access to various eQuate components. There are four levels of eQuate usage: Administrator, Developer, End User and Profile. An Administrator can access all eQuate components, including the management of the eQuate database and the registration of other users. A Developer can access all components that deal with applications setup. An End User can access only the components that are required for running applications. Profiles may be established to allow access to different groups of eQuate applications, thus making the user registration process easier.

In addition to user registration, the Administration program is used to manage application databases doing backups, restores and compacts. The program has the ability to package (zip) application files in preparation for distribution. An Unpackage program is also provided (see below).

Application Manager

The eQuate Application Manager is used to create and maintain eQuate application objects and relationships. eQuate application objects are briefly described below:

1. Applications: Applications define logical starting points into a series of application related functions.
2. Forms: Forms usually define the mapping of terminal screens to eQuate windows; however, you may create forms that are not related to host terminal screens such as "start-up forms" or forms used in command mode. Additionally, the forms object defines field level relationships to Pick Value Lists, Help and Host Error Action.
3. Pick Value Lists: Pick Value Lists define data values that may be picked for an eQuate window field. If a Pick Value List is linked to a field, the user can pick an available input value from a drop-down list. Pick Value Lists can be related to many eQuate window fields.
4. Help: Help defines help text for eQuate window fields or eQuate forms. If help is linked to a field or form, the user can view context sensitive help in a separate text window. Help text can be related (linked) to many eQuate window fields.
5. Host Error Actions: Error Actions define an action or sequence of actions to take place when the eQuate Session Manager receives an error string from the host. Host Error Actions can be related to many eQuate forms.

Capture

The eQuate Capture program is used to scrape a host program's terminal screen and generate a definition of that screen that can subsequently be imported into the eQuate Application Manager's form design process.

For DPS 2200 applications, the main source of input for the generation of eQuate forms is the Form Language Definition Processor (FLDP) text. Using the @FLMU processor, DPS 2200 can produce FLDP text for any DPS 2200 screen. This text file can be downloaded and directly used as input in the forms design process of the eQuate Application Manager, thus eliminating the need to use the eQuate Capture program.

QPort T27 and UTS Config

These two programs are required to configure the information necessary to connect to the host mainframe. One of these programs must be run, and connection configured, before you can use the eQuate Capture program.

Script Editor

The Script Editor program is used for developing and maintaining sign-on scripts that will connect the user to the host when running a developed application with the eQuate Session Manager.

Session Manager

The eQuate Session Manager is the only portion of eQuate normally available to the end user. Driven by the eQuate database, the eQuate Session Manager controls all transaction processing under eQuate, form loading and form display.

Unpackage

This program, along with the packaging function of the eQuate Administration program, can be used to distribute developed eQuate applications.

WinQ T27 and UTS Config

These two programs can be used to set terminal behavior and keyboard functionality that may affect the execution of eQuate applications. Note: The color settings in these programs do not affect the appearance of the forms developed by the eQuate Application Manager.

User Sign On

This dialog is used to enter the user id and password of the Administrator of eQuate or the eQuate Application Developer. The default user id and password supplied by KMSYS Worldwide, Inc., is ADMIN and EQUATE, respectively. For security, it is recommended that these be changed upon first use.

User Id.

In this box, enter the user id of the eQuate Administrator or Application Developer.

Current Password

In this box, enter the password of the eQuate Administrator or Application Developer.

New Password (if changing)

When changing the password, enter the new password. You will be asked to confirm by entering the new password a second time.

Administration

eQuate Administrator

This dialog is used to register new users, assign user privileges, create and assign access profiles, manage user registration and application databases, and package and distribute runtime version of developed eQuate applications.

File menu

The selections on this menu allows you to establish the location for a new user registration database, select the administrative script directory and edit network user setup settings.

Create Runtime Files Now

Click this selection to save all .RTF, .ACT and .BFM files immediately. Note: This save will also occur when you exit the program.

Select New User Registration (Main) Database

This selection allows you to choose a directory for a new eQuate user registration database. This selection also initializes the USER and USERAPP files.

Set Admin. Script Directory

Use this selection to locate the default script directory for the administrator.

User Registration tab

Use this tab to register and maintain eQuate users and profiles.

User Id.

In this text box, enter a user id. User Ids may consist of 1-12 characters. Upon selecting a user with the Find or selection buttons, you may change the user id in this box and click the Apply button to up the database.



First
Move to the first user Id in the database.



Previous
Move to the previous user id in the database relative to the current user id.



Next
Move to the next user id in the database relative to the current user id.



Last
Move to the last user id in the database.

Find

Find a user id using the data entered in the user id entry. You may enter a partial name and click the Find button to locate the matching user.

Apply

Apply changes made to a user id.

Add New

Click this button to add a new user to the registration database. Initially, the User Id value will be "NEW_USER_1", but may be changed by typing a new value in the User Id box and clicking the Apply button.

Delete

Use this button to delete a selected user id.

User Type

There are three levels of access to eQuate: Administrator, Developer and End-User.

There is an additional user type, User Profile, which may be selected when establishing a profile user id (e.g., ACCOUNTING, PAYROLL, etc.). Profile user ids may be created to define which eQuate

applications are part of that profile (see Edit Profile, below). Registered users may then be linked to the appropriate profile user id. In this respect, profiles can make maintenance of long lists of users simpler.

Note: When User Profile is selected, the controls in the General Profile and Default Sign-On groups are not available. Scripts and Profiles may only be assigned to users.

General Profile

Use the buttons in this group to assign (Select) a profile to a user or detach (Clear) a user from a profile.

Default Sign-On Script

Use the Select button in this group to assign a default sign-on script to the user. Two sample scripts are supplied with eQuate: soUTS.bas for UTS (2200) connections and soT27.bas for T27 (MCP).

The file must reside in the SCRIPTS subdirectory of the eQuate installation directory (normally, %APPDATA%\KMSYS Worldwide\eQuate\3.0\Scripts, where the system variable, %APPDATA%, represents the drive and path to the Windows "Application Data" folder).

The script will be executed whenever the user signs on using the eQuate Session Manager. Also, the "Default" Sign-On Script may be overridden by the script name assigned to a particular application to which a user has access

Default Network Station Name

For eQuate sites using the network server implementation of eQuate, use this text box to assign a default station name for the selected user.

[Note: Separate station names may be assigned for each eQuate application to which the user has access \(see, Edit Application Access\). For users allowed to run only one eQuate application at a time, configuring the Default Network Station Name is appropriate.](#)

Current Password

In this edit box, you may assign a password to an administrator, developer or user. For user password enforcement (see eQuate Applications), this is the only place where the password may be assigned.

Use Windows User Logon Name for Session Manager SignOn

Check this box if the Windows logon name is to be used when signing on with the eQuate Session Manager. When this box is checked, the Session Manager gets the current user's Windows login name then looks at the UserApp file to see if the user (registered under that name) can use the Windows user name for sign on. If so, the Session Manager bypasses the sign-on dialog.

User Options (User Defined)

User Options strings are included in eQuate's user registration information to provide additional, user-defined, security. Content and use of User Options is determined by the eQuate administrator. User Options strings are available at runtime in Action Scripts using the "GetUserOptions" function.

This alphanumeric text box may be used to assign options to a user's registration entry. In this manner, actions can check for specific options to give or deny access to certain controls or processes. For example, unless an option is set a button might not be visible to a user.

User Options Map Editing

An additional feature of the eQuate Administrator program is User Options Map Editing. This feature is only used in conjunction with the optional User Options strings that can be associated with users and/or users as related to specific applications.

User Option Map Editing provides a mechanism for the administrator to define a Map (layout) of a User Option string, and the use that Map to edit the string. In this manner, each character or group of characters in the User Options string may be documented as to their use and purpose.

To use this feature, right click on the User Option edit box in the main Administrator window or the Edit Application Access window. This will bring up a one-line popup menu containing "Use option map for editing". Click the popup to bring up the eQuate User Option Editor.

The eQuate User Option Editor is used for editing User Options strings using a pre-defined map. The content of the User Options text box are automatically transferred to the grid in the User Option Editor using the currently selected map. If no map yet exists, select "Create New Option Map" from the "Option Map Management" menu. There are also menu items to select a different option map, edit the current map and delete the current map.

When done editing user options, click OK. The updated options are automatically reconstituted into a string and put back into the User Options edit box. Where options are defined as more than one character, entries containing fewer characters will be right-padded with spaces. The Option Map Editor will not allow entry of more characters than are defined for an option.

When selecting a new Map or Editing a Map from the User Option Editor window, the current content of the User Option string is not changed, but is simply re-partitioned according to the new or changed map.

The eQuate Option Map Editor provided is used to create or change User Option Maps. Simply edit descriptions, start positions and lengths. There must be no intervening blank lines. It is not necessary to specify start positions — they will automatically be recalculated from the length entries. Each item is defined as it appears in the string from left to right with no gaps. If for some reason a gap is needed, a dummy item must be defined at the point of the gap.

User or Profile Name

In this text box, enter any name that is descriptive of the user or profile; e.g., Jane Doe, Accounting Department, etc.

Comments

Use this list box to add additional information that may be useful when maintaining user registrations; e.g., location of the user, telephone numbers, etc.

Edit Profile

Click this button to establish which eQuate applications a user may access. The button may also be used to define which eQuate applications are included in a profile.

Database Management tab

Use this tab to create and maintain application and user registration databases.

Database Type

Select the option button for the type of database you wish to maintain.

The Application Database type includes the following files:

APP.DBF
FORMS.DBF
PICK.DBF
HELP.DBF
STDMSG.DBF
ERRIDS.DBF

The User Registration Database includes the following files:

USER.DBF
USER.MDX
USERAPP.DBF
USERAPP.MDX

Select Application Database

From this list box, select the application database to maintain.

Database Utility Functions

The buttons in this group are used to create a new application database or backup, restore and compact existing application data.

Backup

Click this button to backup the selected eQuate application or the user registration database.

All eQuate components in an eQuate Application database are backed up together in a compressed archive file. Additionally, up to five cycles of backup archive files (EQUATEBACK01.ARC through EQUATEBACK05.ARC for the application and user registration databases) are automatically maintained by eQuate. The EQUATEBACK01.ARC file will always contain the latest backup. A database restore can be taken from any available backup cycle.

Note: The eQuate User Registration database (USER.DBF, USER.MDX, USERAPP.DBF and USERAPP.MDX) is backed up automatically when an application is backed up; however, they may be restored separately.

Pressing the Backup Database button will start the backup immediately. During the backup, a progression dialog will appear displaying the name of each file and percentage completion.

Restore

Click this button to restore a database from a backup. The restore allows you to restore application or user registration databases together or independently.

Compact

Click this button to compress the selected database.

New Database

Use this button to create a new database in an existing or new directory. Note: The directory name becomes the application database name.

Delete Database

Click this button to delete the selected application.

Runtime Publish

Use this tab to copy the eQuate runtime database, scripts, connection configurations, programs and NetSet program to a shared file server.

Runtime Database

Use the controls in this group to publish an eQuate database to a network server.

Select Source

Click this button to select the location containing a developed eQuate database. The location must be the folder containing USERAPP.RTF and the eQuate application subfolders.

Select Destination

Click this button to select a location on a shared file server to receive the eQuate database runtime files (.RTFs, ACTs and .BFMs).

Copy

Click this button to copy just the eQuate database.

Compiled Actions Only

Check this box to publish only the compiled action scripts (.ACX). .ACT files will not be copied.

Runtime Scripts

Use the controls in this group to copy eQuate runtime scripts to a file server.

Select Source

Click this button to select the location containing scripts to be used at run time including those used to sign on to the host.

Select Destination

Click this button to select a location on a shared file server to receive the runtime scripts.

Copy

Click this button to copy the scripts.

Compiled Scripts Only

Check this box to publish only the compiled runtime scripts (.BAX). .BAS files will not be copied.

Copy All

Click this button to copy all runtime files.

Runtime Packager tab

Use this tab to package an eQuate application database for distribution. The output is a single .ZIP file that can contain all runtime components or only those updated since a particular date.

Runtime Package Distribution File

Before attempting to package the application, click this button to select the name of the distribution file.

Select eQuate Database (source)

From this list box, select the eQuate application to be packaged.

Include User Registration Runtime File

Check this box to include the user registration runtime file in the package. This option is useful when you need to make frequent changes to user registration.

Package ALL runtime Components

Use this option (the default) to include all runtime components (.act, .bfm, etc.) in the distribution file.

Package Runtime Components Updated on, or after Date...

Set this option to package only those components that have been updated on or beyond a certain date. When this option is set, a date control is enabled allowing you to enter a date directly in the box or through a calendar that appears when you click the down arrow.

Package It

After you have named the distribution file, selected an application and set the desired options, click this button to complete the packaging process.

User or General Profile

This dialog is used to specify which eQuate applications that a user can access. This dialog is also used to setup user profiles that can be assigned to multiple users.

Application Access

This group contains the controls necessary to add eQuate Applications to a user or general profile.

Add New

Click this button to add an eQuate application to the profile.

Edit

After selecting an application from the list box, click this button to change the access to a different, application or sign-on script.

Delete

After selecting an application from the list box, click this button to remove access to the application from the profile.

Verify

Click this button to verify that the named application still exists in the application database, verify that the database is still present and make sure the script file exists.

Add Application Access

This dialog is used to add an application to the profile. First, select the database, then the application from the appropriate list boxes. If a sign-on script is required, assign it to the application from this dialog.

Database

From this drop-down list box, select the Database in which the application resides.

Application Name

From this drop-down list box, select the application to add to the profile.

Application Sign-On Script

The following controls may be used to assign and automatic sign-on script if the eQuate application requires signing on to a host.

Select

This command button brings up a standard Windows Open dialog allowing the selection of a sign-on script file (.bas or .bax) to be associated with the selected application and executed when the user selects the application with the eQuate Session Manager. The file must reside in the SCRIPTS subdirectory of the eQuate installation directory (normally, %APPDATA%\KMSYS Worldwide\eQuate\3.0\Scripts, where the system variable, %APPDATA%, represents the drive and path to the Windows "Application Data" folder). Two general-purpose sign-on scripts are provided with the installation of eQuate, soT27.bas for MCP machines and soUTS.bas for 2200 machines, and may be found in this directory.

Clear

Remove the association of the sign-on to the selected application.

No Host

This button sets the script name to "*NOHOST*". When the "*NOHOST*" script name is encountered by the eQuate Session Manager, the host sign-on will not be performed. This will allow users to demonstrate and test without connecting to a host; however, they will have to be careful not to perform any action that attempts to transmit to the host.

Edit Application Access

This dialog is used to edit an application in the profile. Use this dialog to change to a different database, select a different application and/or select a different sign-on script.

Database

From this drop-down list box, select the Database in which the application resides.

Application Name

From this drop-down list box, select the desired application for the profile.

User Application Options (User Defined)

User Application Options strings are included in eQuate's user registration information to provide additional, user-defined, security by application. Content and use of User Application Options is determined by the eQuate administrator. User Application Options strings are available at runtime in Action Scripts using the "GetUserAppOptions" function.

This alphanumeric text box may be used to assign options to a user's registration entry for a specific application. In this manner, actions can check for specific options to give or deny access to certain controls or processes by application. For example, unless an option is set a button might not be visible to a user.

User Options Map Editing

An additional feature of the eQuate Administrator program is User Options Map Editing. This feature is only used in conjunction with the optional User Options strings that can be associated with users and/or users as related to specific applications.

User Option Map Editing provides a mechanism for the administrator to define a Map (layout) of a User Option string, and the use that Map to edit the string. In this manner, each character or group of characters in the User Application Options string may be documented as to their use and purpose.

To use this feature, right click on the User Application Options edit box on the Edit Application Access window. This will bring up a one-line popup menu containing "Use option map for editing". Click the popup to bring up the eQuate User Option Editor.

The eQuate User Option Editor is used for editing User Options strings using a pre-defined map. The content of the User Options text box are automatically transferred to the grid in the User Option Editor using the currently selected map. If no map yet exists, select "Create New Option Map" from the "Option Map Management" menu. There are also menu items to select a different option map, edit the current map and delete the current map.

When done editing user options, click OK. The updated options are automatically reconstituted into a string and put back into the User Options edit box. Where options are defined as more than one character, entries containing fewer characters will be right-padded with spaces. The Option Map Editor will not allow entry of more characters than are defined for an option.

When selecting a new Map or Editing a Map from the User Option Editor window, the current content of the User Option string is not changed, but is simply re-partitioned according to the new or changed map.

The eQuate Option Map Editor provided is used to create or change User Option Maps. Simply edit descriptions, start positions and lengths. There must be no intervening blank lines. It is not necessary to specify start positions — they will automatically be recalculated from the length entries. Each item is defined as it appears in the string from left to right with no gaps. If for some reason a gap is needed, a dummy item must be defined at the point of the gap.

Application Sign-On Script

The following controls may be used to assign and automatic sign-on script if the eQuate application requires signing on to a host.

Select

This command button brings up a standard Windows Open dialog allowing the selection of a sign-on script file (.bas or .bax) to be associated with the selected application and executed when the user selects the application with the eQuate Session Manager. The file must reside in the SCRIPTS subdirectory of the eQuate installation directory (normally, %APPDATA%\KMSYS Worldwide\eQuate\3.0\Scripts, where the system variable, %APPDATA%, represents the drive and path to the Windows "Application Data" folder). Two general-purpose sign-on scripts are provided with the installation of eQuate, soT27.bas for MCP machines and soUTS.bas for 2200 machines, and may be found in this directory.

Clear

Remove the association of the sign-on to the selected application.

No Host

This button sets the script name to "*NOHOST*". When the "*NOHOST*" script name is encountered by the eQuate Session Manager, the host sign-on will not be performed. This will allow users to demonstrate and test without connecting to a host; however, they will have to be careful not to perform any action that attempts to transmit to the host.

Network User Station Name

For eQuate sites using the network server implementation of eQuate, use this text box to assign a station name to be used when the user runs the selected eQuate application.

Select General Profile

From the list box on this dialog, select the profile that you wish the user to use and click the OK button.

eQuate Database Backup

Use this window to select the directory where the eQuate backup file will be placed.

Backup File Directory

This information box shows the drive and path where the last backup took place. Use the **Select** button to change the location of the backup files. Note: The directory must previously exist.

Backup

Once the Backup File Directory has been selected, click this button to complete the backup.

Cancel

Click this button to close the dialog after the backup is complete.

eQuate Restore

This dialog is used to restore the user application database and/or the user registration database. You may also restore selected file from an eQuate application database.

Backup File Directory

This information box shows the drive and path where the last backup took place. Use the **Select** button to change the location of the backup files.

Available Backup Files

From this list box, select the backup file from which to restore. The file at the top of the list (EQUATEBAK01.ARC) has the most recent backup.

Restore Options

The restore options allow you to restore an eQuate application database, the user registration database or selected files associated with an application.

Application Database

Select this option to restore only the selected application.

User Registration Database

Choose this option to restore only the user registration database.

Both Application and User Database

If this option is set, both the application and the user registration databases will be restored.

Selected Files

Set this option to selectively restore components of the eQuate application. When this option is set, the check boxes in the Selected Files list are enabled.

Selected Files

Check the boxes of the files that are to be restored.

Restore

Once a backup file has been selected from the list of Available Backup Files, click this button to complete the restore.

Cancel

Click this button to close the dialog after the restore is complete.

Application Manager

eQuate Application Manager

The eQuate Application Manager is used to create and maintain the various components that comprise an eQuate application. This program is used for ninety percent of the design process; from initial input from the form capture process and the placement of control on the form, to the assignment of control properties and event actions to be performed when controls are activated by users.

The first dialog allows you to select the particular component that you wish to configure. The following steps would be used to set up an eQuate application, initially:

1. From the Database tab, select the database that was created using the eQuate Administration program. Note: If you have not created the initial database, do so now by running the Administration program and selecting the Database Management tab and click the New Database button. You will need to exit the Application Manager program and rerun it before the database will appear in the Select Application Database drop-down list box shown below.
2. You will need to give the application a name by clicking the Add button on the Applications tab. This name should be meaningful to the end users, as it will be selected by the users for execution.
3. Once the database and application have been created, you are ready to begin the Forms Design process by clicking the Add button on the Forms tab. Here you may import forms that you have previously captured with the eQuate Capture program or forms that you have extracted from a DPS forms file. It is also from this dialog that you always reach the eQuate Forms Designer by clicking the Forms Design button. The Forms Designer is a VB-like development environment that allows you to customize your forms with the Windows controls that will be most useful to your users.
4. All the other tabs on this dialog are optional but may be extremely useful when developing a fully functional and finished application.

File menu

Create Runtime Files

Click this button to create the runtime files (.RTF) required to execute the eQuate application with the eQuate Session Manager. Note: Runtime files will be saved automatically when you close this dialog.

Language Translations

Click this selection to provide language translations.

Database tab

The Database tab is used to select an eQuate database created by the eQuate Administration program. This selection is the first step to maintaining applications within a database.

eQuate Database Directory

This display-only text box shows the directory where the eQuate databases reside for development. This directory may only be changed by executing the InitReg.exe program.

Select Application Database

From this drop-down list box, select a database that you created with the Administration program.

Select Last Database Accessed on Start-up

Check this box to automatically reopen, on the next session, the last database accessed during this session.

Applications tab

The Application tab is used to maintain all the applications within one eQuate database. From this tab, you may add new applications, edit existing applications and delete applications no longer needed. It is from this tab that you begin the process to name the application, specify the method for starting the application, etc.

Applications

This list box shows the applications located in this database. If this list is blank, use the Add button to create the first application.

To edit an existing application, select it with the mouse or by using the down and up arrows. Once selected, click the Edit button to edit the application.

Edit

After selecting an application, click this button to edit the application name, start-up method, terminal type and other controls relating to the functioning of the application as a whole.

Add

Use this button to add a new application to the database.

Delete

After selecting an application from the Applications list, click this button to delete the application.

Finder

In this text box, type the beginning characters of the application name to locate the application, quickly.

Forms tab

The Forms tab is used to maintain the various forms in the selected eQuate database. It is from this tab that you begin the forms design process by pressing either the Add or Edit buttons.

Forms

This list box contains the forms that are located in the selected database. If the list is blank, use the Add button to create the first form.

To edit an existing form, select it with the mouse or by using the down and up arrows. Once selected, click the Form Designer or Edit button to edit the form.

Form Designer

Use this button to initiate the eQuate Form Designer. The Form Designer is made up of five windows or dialogs that are used to design an eQuate form, assign properties to the form and the controls on the form and assign actions to be taken when controls are used at runtime.

Edit

After selecting a form, click this button to edit the form name, the form id string, form id location, form mode and other controls related to the form.

Add

Use this button to add a new form to the database.

Delete

After selecting a form from the Forms list, click this button to delete the form.

Finder

In this text box, type the beginning characters of the form name to locate the form, quickly.

Pick Values tab

The Pick Values tab is used to maintain lists of values from which users may select. These lists may be placed on a form as list boxes and drop-down list boxes.

Pick Values Lists

This list contains all the Pick Values Lists that are maintained in the database. If the list box is blank, use the Add button to create the first Pick Values List.

To edit an existing Pick Value List, select it with the mouse or by using the down and up arrows. Once selected, click the Edit button to edit the Pick Values List.

Edit

After selecting a Pick Values List from the list box, click this button to edit the selected list.

Add

Use this button to add a new Pick Values List to the database.

Delete

After selecting a Pick Values List from the list box, click this button to delete the list.

Finder

In this text box, type the beginning characters of the Pick Values List name to locate the list, quickly.

Help Text tab

The Help Text tab is used to store help strings in the database that may be referenced on the Help property of any field.

Help Text

This list contains all the help text strings that are maintained in the database. If the list box is blank, use the Add button to create the first text string.

To edit an existing text string, select it with the mouse or by using the down and up arrows. Once selected, click the Edit button to edit the string.

Edit

After selecting a name from the list box, click this button to edit the help text name and/or the text string.

Add

Use this button to add a new help text string to the database.

Delete

After selecting a help text name from the list box, click this button to delete that help text from the database.

Finder

In this text box, type the beginning character of the help text name to locate the help text, quickly.

Standard Messages tab

The Standard Messages tab is used to store message text in the database that may subsequently be referenced by the host program. The Message Number associated with each message is returned by the host program using eQuate Command Mode. The eQuate Session Manager will substitute the Standard Message Text stored in the eQuate database.

Msg. Num. Msg. Text.

This list box contains the message text entries and their associated message numbers that are maintained in the database. If the list box is blank, use the Add button to create the first text string.

To edit an existing message, select it with the mouse or by using the down and up arrows. Once selected, click the Edit button to edit the message.

Edit

After selecting a message from the list box, click this button to edit the message text and/or number.

Add

Use this button to add a new message to the database.

Delete

After selecting a message from the list box, click this button to delete the message and associated number.

Host Error Ids tab

The Host Error Ids tab is used to maintain a list of error strings and associated actions to be taken when the strings are received from the host.

Error Msg. Text/Action

This list box contains the text strings of error messages and their associated actions that are maintained in the database. If the list box is blank, use the Add button to create the first text string.

To edit an existing error string, select it with the mouse or by using the down and up arrows. Once selected, click the Edit button to edit the error message or assign a different action.

Edit

After selecting an error message from the list box, click this button to edit the message text and/or number.

Add

Use this button to add a new error message to the database.

Delete

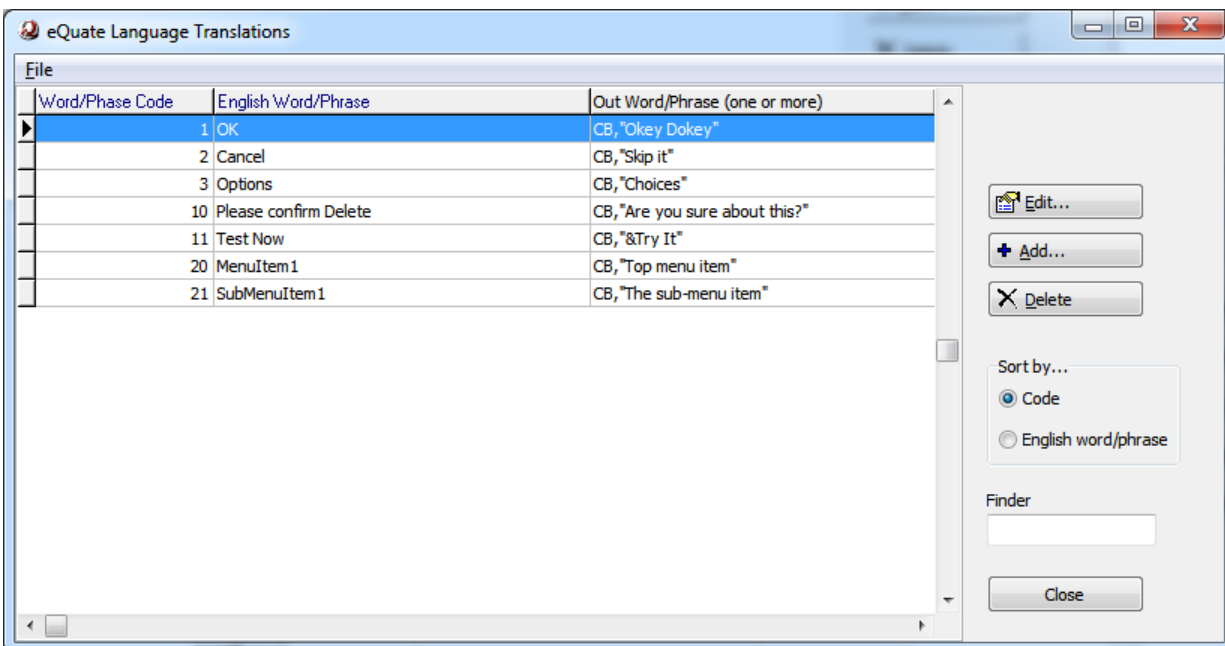
After selecting an error message from the list box, click this button to delete the message.

eQuate 3.5 Language Translations

The language translation feature added to eQuate 3.5 (January 2013) is a simple way to localize eQuate applications. It provides a simple way to translate single words or phrases from one language to another. It should work with any language that does not require double-byte characters.

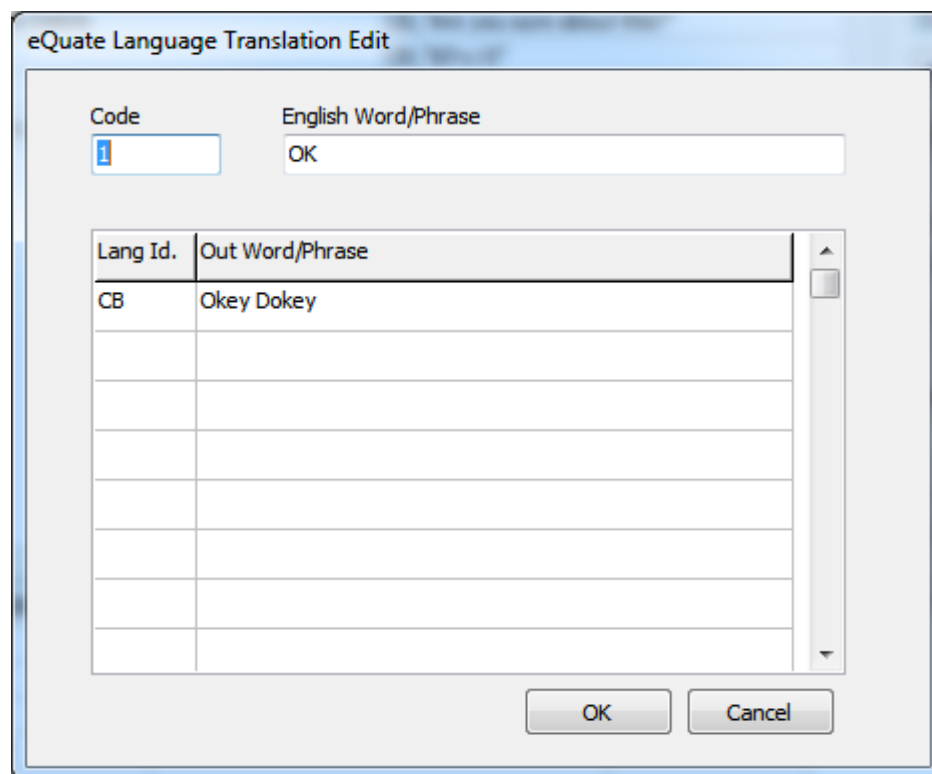
How it Works

A new item, named "Language Translations", has been added to the File menu of the Application Manager's main form. Clicking "Language Translations", shows the new window titled "eQuate Language Translations" shown here.



Each translation database entry contains a Word/Phrase Code, an English version of the word or phrase and up to 10 translations of the word or phrase. The translation consists of a Language Id and the output or translated word or phrase. Language Ids are arbitrary and defined by user. The data may be sorted by either the word/phrase code, or the English word/phrase. Clicking the sort by selection changes the sort order immediately. Use the finder to type in a sequence to automatically position the selection to the first matching entry based on the currently selected sort.

Entries may be edited, added or deleted here also. Edit or Add will bring up the Edit Window shown below:



The dialog box is titled "eQuate Language Translation Edit". It contains two input fields at the top: "Code" with a value of "1" and "English Word/Phrase" with a value of "OK". Below these is a table with two columns: "Lang Id." and "Out Word/Phrase". The table has 10 rows. The first row contains "CB" and "Okey Dokey". The remaining 9 rows are empty. At the bottom of the dialog are "OK" and "Cancel" buttons.

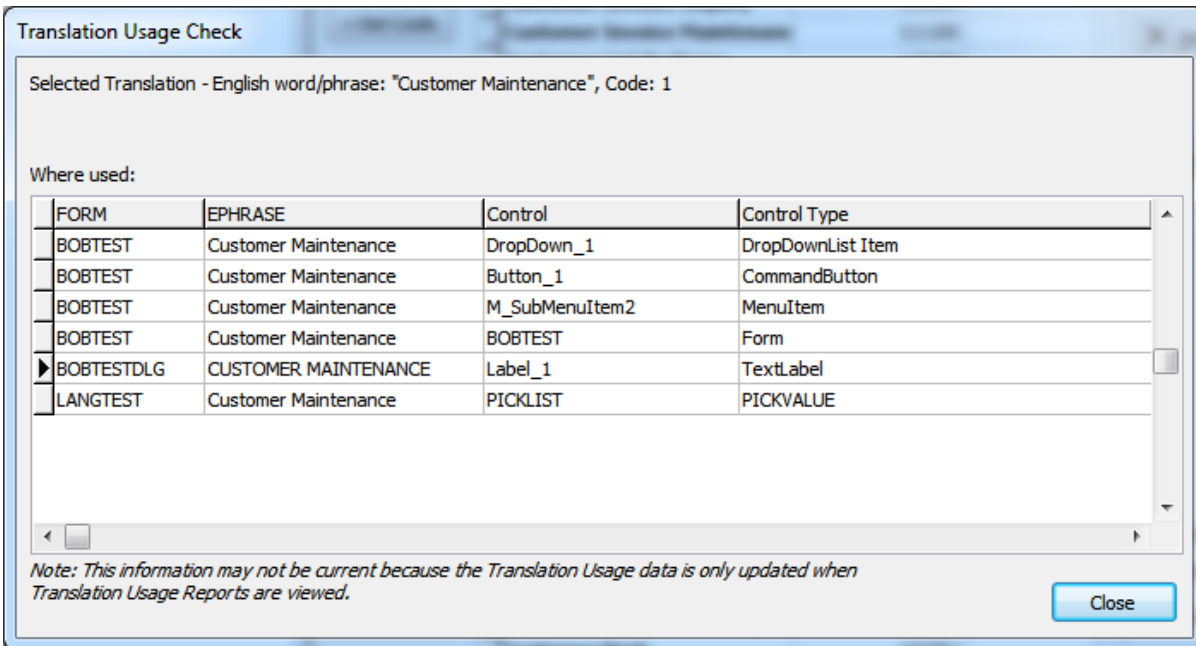
Lang Id.	Out Word/Phrase
CB	Okey Dokey

If this were an Add, the Code would contain 0 and the English Word/Phrase would contain "New"; otherwise, the entries will be filled in with current values as shown here.

The Code must be unique and can be any positive integer value greater than 0. The code is used to look up translations at run-time. The English word/phrase, may contain anything other than all blank. It has two purposes. First, it's a reference to the developer so he knows what he's translating. The second purpose is it can be used as an alternate way to look up the translation.

The lower portion of the Edit window contains up to 10 translations. Each translation consists of a 2-character language id and an Out Word/Phrase. Again, language Ids are arbitrary and are chosen by the user. Both entries must be included for each translation. To delete a translation, simply delete the text in both entries. Blank lines between translations are permitted and will be automatically removed. Translations may be entered in any sequence, and the sequence does not have to be the same for all translations.

A "Check Usage" button on the eQuate Language Translations window (from the main window) and the Language Translation For Form's Controls window (from the form designer) brings up a dialog titled "Translation Usage Check" showing where the currently selected translation is used.



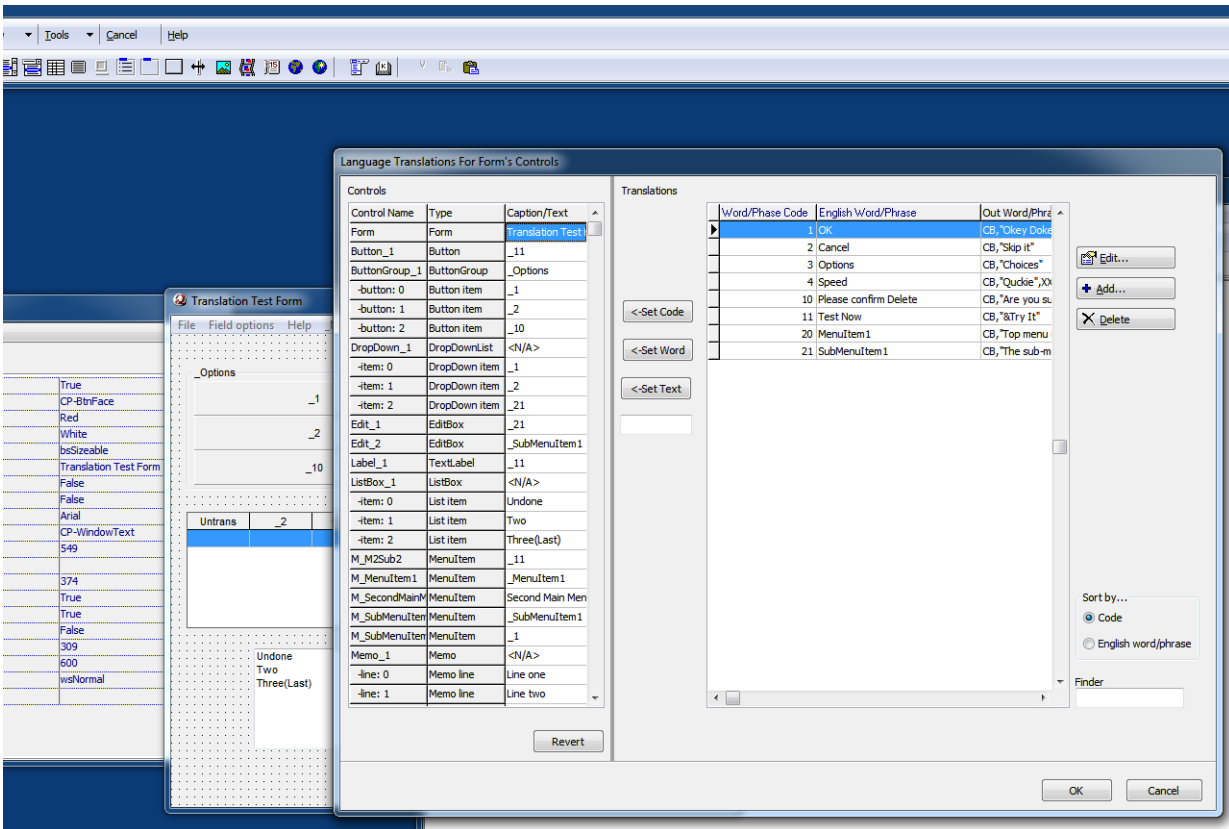
This window uses the database created when the Translation Usage Reporting window is opened (see **Translation Usage Reporting**, below). It should be noted that because it can take a very long time to scan all forms for translation usage, the usage database is only created and/or maintained when the Translation Usage Reporting window is opened (this note is displayed in the window above, also). It is recommended that the Translation Usage Reporting window be opened after making changes to translations or applying translations to control within Forms or Pick Lists.

How to Use Translation in an Application

Translations are applied in three ways. First, they (code or English word/phrase) may be placed in eQuate forms using the Form Designer. To do this, simply enter a control's caption to be translated as either the word/phrase code or the English word/phrase prefixed by the under bar ("_") character (for example, "_1" or "_OK"). eQuate will automatically determine whether to look up the translation by code or English word/phrase. Using the code will be a little faster, but using the word/phrase may be easier for the developer.

This method also works for list items entered in the form designer and entries in pick lists.

The second way to apply translations is also available from the Form Designer. From the "Tools" menu select "Caption Translations..." which will show the following dialog window:



This window is a convenient way to set up all translations for a Form in a single dialog. The window is broken into two sides—the left side shows a list of controls on the current Form that have captions or text entries that may be translated. List-type controls, ListBox, DropDownList and ButtonGroup, also include items that are currently set in their lists. The controls list is sorted by control name and contains the control's name, control type and current content. The right side of the window contains a display of the current translation database, and looks and works much like the "Language Translations" window described earlier. Translations may be edited, added and deleted here as well. There is a splitter between the two halves, which allows the user to resize as necessary.

To assign a translation to a control or control's list item, simply select it in the control list. If the selected control already has a translation set (indicated by the "_" prefix), its corresponding translation will automatically be selected in the translations list to the right. Once selected, there are three options to apply the translation using one of the three buttons just to the left of the translation list. The first two, "Set Code" and "Set Word", allow the user to select the word/phrase code or the English word/phrase. The third, "Set Text", applies the text entered in the text box just below the button, which allows the user to enter a non-translation text string.

The "Revert" button under the controls list restores controls to the state they were in when this dialog was entered. "Click "OK" to apply all control changes. Translation database changes are immediate and cannot be automatically reverted or cancelled.

The third way translations may be used is in the Form's action code itself using the new "GetTranslation" function. This function simply returns the appropriate translation for a given string value.

The format of the GetTranslation function is:

```
Function GetTranslation (ByVal InputPhrase As String) As String
```

Where *InputPhrase* is the word or phrase to be translated with the under bar prefix.

The following are examples of using the GetTranslation function:

```
Button1.Caption = GetTranslation("_Go")
MsgBox(GetTranslation("_123"), mb_IconExclamation, "Error")
```

If any translations cannot be found, the input word/phrase (or code) is used as is.

Translation Usage Reporting

In order to aid developers in managing translations used throughout several forms contained in an eQuate application, the Translation Usage Reporting window has been provided. This window is accessed from the Application Manager's main menu.

When the Translation Usage Reporting window opens it builds a database of all usages of translations in the current application. Each control and menu item in every Form in the application is scanned for translations. Additionally, all Pick Lists are scanned for translations. Translations found (test with the “_” prefix) are also matched to the current translation database. If translation items are found in Forms that do not exist in the translation database they are listed as errors in the Activity Log. All other translations are shown in the Usage report.

The following is an example for the Translation Usage Report window:

The Usage report can be sequenced by English Word/Phrase, Word/Phrase Code or Form Name by clicking the radio buttons in the “Report By” group box or by clicking one of the first 3 the column titles.

English Word/Phrase (EPHRASE)	W/P Code (CODE)	Form/Pick list (FORM)	Control Name (CTRL)	Control type (CTYPE)
Active	10	BOBTEST	MCList	Multi-Column List Head
Active	10	BOBTEST	ButtonGroup_1	ButtonGroup Item
Customer Maintenance	1	BOBTEST	DropDown_1	DropDownList Item
Customer Maintenance	1	BOBTEST	Button_1	CommandButton
Customer Maintenance	1	BOBTEST	M_SubMenuItem2	MenuItem
Customer Maintenance	1	BOBTEST	BOBTEST	Form
Customer Maintenance	1	BOBTESTDLG	Label_1	TextLabel
Customer Maintenance	1	LANGTEST	PICKLIST	PICKVALUE
Customer Name	6	BOBTEST	MCList	Multi-Column List Head
Early Ship	21	BOBTEST	DropDown_1	DropDownList Item
Early Ship	21	BOBTEST	Edit_1	EditBox

The selected sequence column is highlighted and bold.

Filtering the Usage Report

The usage report can be filtered to include only selected information (Filter check box checked). The standard filter lets the user specify single specific search values which are combined by ANDs. For example, if the user enters a code of 10 and an English Word of “Active” the resulting filter will be “CODE = 10 .AND. UPPER(EPHRASE) = “ACTIVE””. The “UPPER()” function is used to make the filter NOT case sensitive. Standard filtering is updated are you type in values.

To provide more flexibility in filtering the user can enter a custom filter. The custom filter requires some knowledge of dBASE style expressions. This is an example of a custom filter: “(code >= 6 .AND. code <=22) .AND. UPPER(from) = “CUST” .AND. CTYPE = “TextLabel””. That will select all records with codes in the range 6 through 22 with form names beginning with “CUST” and with a control type of “TextLabel”. Custom filters are not applied until the “Apply” button is clicked.

Note that the actual field names to be used in custom filter expressions are shown in the column headers in parenthesis.

Filter Expression Operators and Functions

Relational operators:

=	Equal to
<>	Not Equal to
<	Less than

>	Greater than
<=	Less than or equal to
>=	Greater than or equal to
\$	Contains string

Logical Operators:

.NOT.	Logical not
.AND.	Logical and
.OR.	Logical or

Note: the periods surrounding the logical operators are required.

Useful Functions:

LEFT (value, number-of-chars)	This function returns a specified number of characters from a character expression, beginning at the first character on the left. The parameter 'NUM_CHARS' must be constant. e.g. "LEFT('SEQUITER', 3)" returns "SEQ".
LTRIM(value)	The function trims any spaces from the left side of the value
STR(number, Length, decimals)	The string function converts a numeric value into a character value. "Length" is the number of characters in the new string, including the decimal point. "Decimals" is the number of decimal places desired. The parameters 'LENGTH' and 'DECIMALS' must be constant. If the number is too big for the allotted space, *'s will be returned. e.g. " STR(5.7, 4, 2) " returns " '5.70' " The number 5.7 is converted to a string of length 4. In addition, there will be 2 decimal places. e.g. " STR(5.7, 3, 2) " returns " '****' " The number 5.7 cannot fit into a string of length 3 if it is to have 2 decimal places. Consequently, *'s are filled in.
TRIMCHAR(value)	This function trims any spaces from the right side of the value.
UPPER(value)	The string is converted to all uppercase characters.

Here are some custom filter examples:

"Maint" \$ EPHRASE

Selects all records where the English Word/Phrase contains the string "Maint" (case sensitive).

"MAINT" \$ UPPER(EPHRASE)

Same as above except not case sensitive.

LEFT(UPPER(EPHRASE), 3) = "ACC" .OR. LEFT(UPPER(EPHRASE), 1) = 'X'

Selects all records where the English Word/Phrase begins with "ACC" or "X" (not case sensitive).

LEFT(LTRIM(STR(CODE, 10, 0)),1) = "1"

Selects all records where the first digit of the translation code equals "1". Code is converted to a 10 character string with leading spaces with the STR function. Then, the leading spaces are removed by the LTRIM function. Finally, the first character is isolated by the LEFT function.

(FROM = "JOE" .OR. FORM = "LARRY") .AND. CTYPE = "TextLabel"

Selects all records where the Form name is either "JOE" or "LARRY" and the Control type is "TextLabel".

Printed Reports

A printed report can be produced from the currently displayed report at any time. The printed report will differ in that the Report-by field will be shown as the first column and group-indicated (the same entry is not repeated on each line).

Activating Translations

There are three ways to activate or change language translations at run time (Session Manager – SessionMgr.exe).

Command line

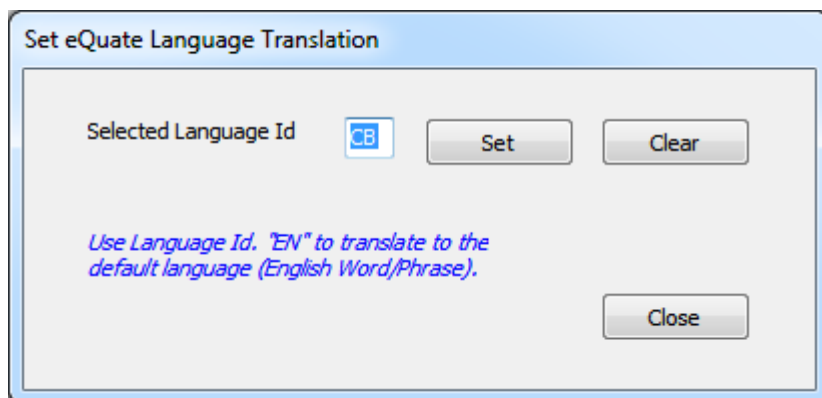
Translations can be set in the SessionMgr command line. To do this simply add the following parameter to the command line:

/LANGID:xx

Where xx is the two-character language id. This parameter should be placed after the user id, password, database and application name parameters if they are present.

From the Start Window

Language translations may be selected from Session Manager's "File" menu by selection the "Set Translations..." submenu which will open the "Set eQuate Language Translations" dialog which looks like this:



Here, you may select and set a language Id which also activates translations, or you may deactivate language translation. If a Language Id is currently selected, it will be displayed when the dialog is opened. To activate translations enter a language Id that is already available (set up in Application Manager) then click the "Set" button. If you are not using, or wish to discontinue using, the Language Translation feature, click the "Clear" button. Both Set and Clear take effect immediately. Click "Close" to return to the Session Manager main window.

From a Form's Action Script

Translations may be set by the eQuate application itself. Two functions are available for this purpose: SetCurrLanguage and GetCurrLanguage. GetCurrLanguage simply returns the two-character language Id currently in use. SetCurrLanguage sets the current translation language using the two-character language Id.

Formats:

```
Function GetCurrLanguage () As String
```

```
Function SetCurrLanguage (ByVal LangId As String) As Integer
```

If SetCurrLanguage fails it returns 0 otherwise it returns 1 indicating success.

Using Translations

If the application is set up to use the translation feature, you cannot simply turn it off. Any prefixed code or word/phrase will appear in the form at run-time. To avoid this, enter the Selected Language Id as "EN"

(English) which will cause the translation mechanism to return the English word/phrase (without the under bar prefix).

Applications

eQuate Applications

The eQuate Applications screen is used to define the application and its method of starting.

Application Name

The Application Name is used to uniquely identify the Application within the eQuate application database. The application name is the name that will be displayed to the end user when they run the eQuate Session Manager. The name may be up to 32 characters long.

Terminal Type

The Terminal Type defines the type of host connection that is to be made when the application is executed. Note: Both of these options are ignored if application access is set to "*NOHOST" (select User Registration | Edit Profile | *application-name* | Edit in the eQuate Administration program or File | Profiles | User Route in the eQuate Session Manager).

UTS

Set this option if the application is to connect to a 2200 host.

T27

Set this option if the application is to connect to an A Series host.

Connection Route

The Connection Route may be used to set a default route for all users of the application. The controls in this group are optional and if set, may be overridden by profile settings in the eQuate Session Manager.

Specify Connection Route

Check this box to enable the Route Name text box below.

Route Name

In this box, enter the name of the default route connection route for the application.

Method of Starting Application

The controls in this group are used to tell the eQuate Session Manager how to begin a user session when an application is selected with the eQuate Session Manager.

Enforce User Id and Password Validation

Check this box to require the user to enter a password when running an eQuate application with the session manager. If this box is unchecked (the default), only the user id is checked. Note: Passwords are assigned in the eQuate Administration program (see eQuate Administrator).

Send Transaction

Set this option if the application is to begin by sending a string to the host. The string might be a 2200 TIP transaction code, a MAPPER run or any 2200 or A Series program call. Set the transaction or string to start the host program/run in the Form Name or Transaction box below.

Display Start Form

Set this option if an eQuate "start" form has been designed to initiate the host program/run. Note: When this option is taken, a control on the start form will issue an action that will in turn start the host process or program.

Form Name or Transaction

If the Send Transaction option is set, enter the transaction string to be sent to the host. If the option, Display Start Form, is set, enter the start form name or click the Select the Start Form button to select from a list.

Start in Command Mode

Check this box if the application is developed for "command" mode as opposed to "screen" mode. In command mode, the host program determines which eQuate forms are to be displayed using eQuate commands sent to the eQuate Session Manager. In screen mode, information displayed on the terminal screen by the host program identifies which eQuate form is to be displayed.

No Form Id Detection

Check this box to inhibit the eQuate Session Manager from performing form id detection. This option is required when combining multiple screens on one form using eQuate screen mode.

Default Form

This text box can be used to specify a default form that will be displayed when form id detection is enabled and the eQuate Session Manager cannot detect the specified form id string in the eQuate application database.

When Form Id String Not Found

The options in this group specify what the eQuate Session Manager should do if form identification is enabled and no match occurs.

Keep Current Form

If this option is set, the form last displayed will remain open.

Show Default Form

If this option is set, the Default Form specified above will be displayed.

Select Form

This common dialog is used to select a Start Form or a Default Form for the eQuate Application.

Current Selection

This information-only text box shows the currently selected form, if any.

List of Available Forms for the Application

Select the form to be used as either the Start Form or the Default Form.

Finder

In this text box, type the beginning characters of the form name to locate the form quickly.

Forms

eQuate Form Manager

The eQuate Form Manager is used to set the form properties and is where the form design process takes place.

Form Name

The Form Name is used to uniquely identify the form within the eQuate application. The name may be up to 32 characters long. The Form Name is transparent to the end user and may be anything meaningful to the eQuate developer.

Data Fields

Use this button to make changes to the way data fields are received from the host on the terminal screen.

Form Description (optional)

Use this text box to add a meaningful description of the form.

Form Id String

This string uniquely identifies this form for the eQuate Session Manager. When a screen is received from the host, this string will be used to locate and load this form from the eQuate application database. The eQuate Session Manager uses the value in this entry (along with the Row, Column and Width in Screen Mode) to recognize and load an eQuate form.

Id Col

In this box, enter the beginning column of the Form Id String. Note: This value is only used when in Screen Mode.

Id Row

In this box, enter the row on which the Form Id String is located. Note: This value is only used when in Screen Mode.

Id Length

In this box, enter the length of the Form Id String. Note: This value is only used when in Screen Mode.

Form Mode

The Form Mode determines how the eQuate Session Manager will identify and locate this form in the application database.

Screen

If this option is selected, the eQuate Session manager will load the form when the supplied Form Id String appears on the screen in the location specified.

Command

In Command Mode, the host program sends a command containing the name of the form to load.

Date Last Changed

This information box contains the date and time that this form was last updated.

Import Form

Click this button to import a form definition from the Windows clipboard, text file (e.g., FLDP source file from DPS 2200) or binary form file from another eQuate application.

Form Designer

eQuate Form Designer Toolbar

The eQuate Form Designer is the most powerful part of eQuate development. Use the eQuate Form Designer Tool Bar and the companion dialogs (eQuate Form, Property Editor, eQuate Action Script Editor, etc.) to alter the appearance of the basic terminal screen (as initially captured or converted), add new controls and automate end-user interface functions.

General Features

The eQuate Form Designer is very similar in appearance to the development environment of several object-oriented visual languages. The eQuate Form is the canvas upon which you place Windows controls (buttons, list boxes, etc.). The eQuate Form Designer Tool Bar is the pallet from which you select the controls that you want to place on the form. Each control has its own unique set of Properties, dependent upon the type of control selected, that allow you to specify how the control will appear and behave on the form.

The eQuate Form

The initial design form contains a menu bar at the top that has three disabled (grayed out) menu names. These menus are enabled at runtime and used to access standard options (e.g., Print Setup) at runtime. You may add other menus specific to the form and they will be displayed here, but they will not cause any actions to be invoked during the design phase.

The eQuate form is a Windows control, and like any Windows control, has a set of properties associated with that control. For properties and actions, see Form Properties.

The eQuate Form Designer Tool Bar

The toolbar (initially at the top of the monitor viewing area) can be moved anywhere on the window that is convenient. To move, first select Float Tool Bar from the toolbar's View menu. Next, place the mouse cursor over the caption bar (over the words, "eQuate Form Designer Tool Bar"). While holding down the left mouse button, drag the toolbar to the desired location.

Adding Controls and Fields

To add form fields (actually, an Edit Box control) or other Windows-like controls to the eQuate Form, click the appropriate control button on the tool bar. Note: Adding menu controls are done by selecting Menu Designer from the Tools menu or clicking the corresponding button on the toolbar.

With a control button selected on the toolbar, move the mouse cursor over an empty area of the form window and click the left mouse button. The new control will be placed on the form where you clicked. Alternately, you may hold down the left mouse button and drag the mouse to create the initial size and location of the control. A box will grow and shrink as you drag in any direction. When the left mouse button is released, the control will appear, already selected and ready to move or edit.

To edit, click the left mouse button over the control and use the Property Editor window to change the desired property.

Moving and Sizing Controls

eQuate controls may be moved and resized using the mouse. Select the desired control by moving the mouse cursor over the control and clicking the left mouse button. When selected, the control will have sizing handles around it.

To size, move the mouse cursor over one of the sizing handles (the mouse cursor will change to sizing arrows), then press and hold the left mouse button and drag the sizing handle in the desired direction. When the mouse button is released, the control will snap to the new size.

To change the location on the form, move the mouse cursor into the center area of the control and press and hold the left mouse button. Using the mouse, drag the control to the desired location, and then release the mouse button.

Aligning and Sizing Controls

The alignment and sizing feature allows any group of eQuate's controls to be automatically aligned with or sized to a "base" control (any single eQuate control). The process is made simple by first selecting the base control with the left mouse button. Next, while holding down the **Shift** key, other controls to be aligned/sized are selected with the left mouse button (notice that the sizing handles become gray). Finally, right click on any of the selected controls and choose an alignment or sizing option from the popup menu.

Alternately, you may select a group of controls by moving the mouse cursor to a point on the eQuate form away from any control (the gray part). While holding the left mouse button, drag the mouse to encompass the controls you want to align or size. Note: The last control selected becomes the base control.

Alignment and sizing are based on the original alignment and size of the base control. Alignment allows vertical alignment to be left justified, centered or right justified to the base control; horizontal alignment to be aligned

across the top, middle or bottom of the base control. Sizing changes the width or height of the selected controls to match that of the base control.

Automatic Properties Assignment

Much like the ability to automatically size and align groups of controls (see above), controls may have their properties (font, color, etc.) set as a group. The procedure is practically the same. While holding down the left mouse button, drag the mouse over the controls whose properties are to change. Next, change the property or properties desired on the Property Editor window.

Most controls have "Parent" properties (ParentCtl3D, ParentFont and/or ParentShowHint). If a parent property is set to True, then the control takes on the property of the parent control. Currently, there are only two types of parent controls: the eQuate form itself and the panel control. For example, if a Button on the form has the ParentFont property set to True, the Button caption will use the same font assigned to the form. Note: Changing the Font property on the Button will automatically set ParentFont property to False.

A Panel can have controls placed on it so that whenever you move the panel during the design phase, all the controls on the panel move with it. Likewise, if an eQuate action disables the Panel at runtime, all the controls on the Panel are also disabled since the Panel is Parent to all the controls placed on the panel.

Automatic Alignment to Client Area

Some controls (panels, list boxes, text labels, etc.) can be aligned to all or part of the client area of a parent control. The Align property allows the automatic alignment of the control to the top ("alTop"), bottom ("alBottom"), left ("alLeft") or right ("alRight") edge of the parent control. If the Align property is set to "alNone" (the default), the control remains wherever it is placed on the parent control. A sixth setting, "alClient", can be used to align the control within the client area of the control. If multiple controls are aligned on the parent control, they are aligned in the precedence they were placed on the control.

An example of using aligned controls would be when you wanted to place a splitter between two list boxes thus giving the user the ability to determine how much of the list boxes would be visible at a given time.

Save and Close button

Save all changes made to the form and return to the eQuate Form Manager dialog.

Edit menu

Cut (Ctrl+X)

Use this selection to cut the selected control(s) to the Windows clipboard.

Copy (Ctrl+C)

Use this selection to copy the selected control(s) to the Windows clipboard.

Paste (Ctrl+V)

Use this selection to paste the contents of the Windows clipboard to the form.

Delete (Del)

Use this selection to delete the selected control(s).

Delete All Controls

Use this selection to clear the form.

Cut to File

Use this selection to cut the selected control(s) to a file (.clp).

Copy to File

Use this selection to cut the selected control(s) to a file (.clp).

Paste from File

Use this selection to paste the contents a file to the form.

Select All

Use this command to select all controls on the form.

Form Edits...

Use this selection to display the alignment, sizing, tab order and grid options popup menu. This selection is the same as doing a right mouse click over a selected control.

View menu**Tool Bar (F6)**

Use this selection to return emphasis to the eQuate Form Designer Tool Bar from the Form Design window...

Properties Window (F7)

Use this selection to open the eQuate Property Editor window.

Design Window (F8)

Use this selection to return emphasis to the eQuate Form Design window from the Form Designer Tool Bar.

Action Edit Window (F9)

Use this selection to open the eQuate Action Script Editor.

Data Field Names (F10)

Use this selection to open the Select Screen Data Field Source window.

Float Tool Bar

Use this selection to unlock the tool bar. This selection allows you to drag the tool bar by holding down the left mouse button over the tool bar caption and dragging it to another location.

Tools menu**Menu Designer**

Use this selection to open the eQuate Menu Designer window to add or edit menus on the form.

Action Key Assignment

Use this selection to open the Action Key Assignments window where you can assign a series of eQuate actions to a keystroke or combination of keystrokes.

Cancel button

Use this selection to cancel all form design changes and return to the eQuate Form Manager dialog.

Help menu**Contents**

This selection starts the Windows Help program and displays all help topics for eQuate.

About

Use this selection to display eQuate version and copyright notification.

The Designer buttons

The buttons along the bottom of the eQuate Form Designer Tool Bar are used to select the type of control you want to place on the form.

**Pointer button**

Use this selection to cancel adding a control, thus returning the mouse to a pointer tool. Use the pointer tool to select control(s) on the form.

**Text Label**

Use this selection to add a text label to a parent control (form or panel). This control is useful to label controls that have no caption property (e.g., Edit Boxes).

For properties, see Text Labels.

**Edit Box**

Use this selection to add an edit box to a parent control. In screen mode, only add data fields if corresponding references are made to the DPS screen and host program.

For properties, see Edit Boxes.

**Command Button**

Use this selection to add a command button to a parent control (e.g., OK button, Cancel button, Query button, etc.).

For properties, see Command Buttons.

**Speed Button**

Use this selection to add a speed button to a parent control. A speed button is a non-windowed control meaning it takes less system resources. Since it is a non-windowed control, you may not tab to it.

You can make a group of speed buttons act like radio buttons by setting the group index to something other than 0.

Speed buttons can also have that flat property that makes the outline show on mouse fly-over.

For properties, see Speed Buttons.

**Check Box**

Use this selection to add a check box to a parent control. Use check boxes when multiple options may be selected by the user.

For properties, see Check Boxes.

**Option Button**

Use this selection to add option or radio buttons to a parent control. Use option buttons when the options are mutually exclusive for selection by the user.

Option buttons placed on a form are mutually exclusive for the entire form; i.e., only one may be selected by the user on a given form. Use the Button Group control when option buttons need to be mutually exclusive within a group (see below).

For properties, see Option Buttons.

**List Box**

Use this selection to add a single column list box to a parent control that can be used for display or data value selection by the user. If more items are displayed than can be contained by the size of the list box on the, scroll bars will appear. If a list is to contain many values, it might be desirable to use a drop-down list box (see below) to conserve space on the form.

For properties, see List Boxes.

**Drop-Down List Box**

Use this selection to add a drop-down list box to a parent control that can be used for display or data value selection by the user. The drop-down list box occupies only the amount of space on the form that it takes to display a single item in the list. Other items are made visible by clicking the drop-down arrow on the box.

For properties, see Drop-down List Boxes.

**Multi-Column List Box**

Use this selection to add a multi-column list box to a parent control. Unlike the single column list box, the multi-column list box control supports header options that are displayed as part of the form.

For properties, see Multi-column List Boxes.

**Memo**

Use this selection to add a memo control to a parent control. A memo control is basically a multi-line edit box.

For properties, see Memo Controls.

**Bevel**

Use this selection to add a bevel to a parent control. The bevel has very few properties. It is primarily used to for appearance.

You can place controls within a bevel but those controls, unlike those on a panel or button group, may not be moved as a group.

For properties, see Bevels.



Button Group

Use this selection to add an option button group to a parent control. When the button group is first placed on the form, no option buttons appear in the group. Use the Items property to add the option buttons. Next, use the button group sizing handles to size the group to contain the option items entered. The Item Index property can be used to specify which option in the group is initially set when the form is loaded by the eQuate Session Manager (e.g., -1 means that no option is set, 0 means the first option is set, 1, the second, etc.).

For properties, see Button Groups.



Group Box

Use this selection to add a group box to a parent control. Group boxes are generally used to logically group controls on the form (e.g., a group of controls that presents the user the Account Number, Balance, Amount Due, etc. might be placed in a group box captioned, Account Information).

You can place controls within a group box but those controls, unlike those on a panel or button group, may not be moved as a group.

For properties, see Group Boxes.



Panel

Use this control to add a panel to a parent control. Controls can be placed on a panel and moved as a group when you select the panel and drag it to another location on the form. To place a control on a panel, select the panel on the form with the mouse. Next, click the Panel button on the eQuate Form Designer Tool Bar and then click the mouse again over the panel.

To cut a control from another location (on the form or other panel) and paste it on a panel, select the control and enter Ctrl+X on the keyboard. Next, with the mouse, select the panel that is to receive the control and press Ctrl+V. Note: The control will be placed on the panel in the same relative location on which it was originally located; therefore, it may be necessary to increase the size of the panel in order to see the control.

For properties, see Panels.



Splitter

Use this control to add a splitter to a parent control. A splitter, like those seen in popular Web browsers, allows the user to adjust the panels on a form to their own liking.

Splitters must be aligned and placed between other aligned controls. For example, if a splitter is to be placed vertically between two panels the left panel could be aligned left, followed by aligning the splitter left also. The right-hand panel then could be aligned to fill the remaining portion of the client area.

For properties, see Splitters.



Image

Use this control to add an image to a parent control. Only pictures with .bmp, .wmf, .emf or .ico extensions are supported.

For properties, see Images.



Media Player

Use this selection to add a media player control bar to a parent control. To use the Media Player control, use the LoadMMFile function in an action.

For properties, see Media Players.



Date/Time Label

Use this selection to add a date/time stamp to a parent control. It is useful in a status bar (a bottom aligned panel).

For properties, see Date/Time Labels.

**Menu Designer**

Use this selection to open the eQuate Menu Designer window to add or edit menus on the form.

**Action Key Assignment**

Use this selection to open the Action Key Assignments window where you can assign a series of eQuate actions to a keystroke or combination of keystrokes.

**Cut (Ctrl+X)**

Use this selection to cut the selected control(s) to the Windows clipboard.

**Copy (Ctrl+C)**

Use this selection to copy the selected control(s) to the Windows clipboard.

**Paste (Ctrl+V)**

Use this selection to paste the contents of the Windows clipboard to a parent control.

Property Editor

The Property Editor window is where the properties and actions for individual controls placed on the eQuate form are set. Correspondingly, the window contains two tabs: Properties and Actions. The Properties tab is where you name the control as it will be referenced in eQuate actions, initially set how the control will appear to the user at runtime, attach runtime help for the control, etc. The Action tab is used to assign an action to the control.

The eQuate form can also be considered a control and, as such, may have properties assigned or changed by first selecting the form (left click on any empty space on the form).

Note: The eQuate form is a "parent" control. Any form property (Font, Ctl3d, etc.) will be assigned to a control placed on the form if that control has its corresponding "parent" control set to True (see Parent Controls).

To change a property, first select the control with the mouse on the eQuate form. With a control selected, the properties for the control will appear on the Properties tab. The left-hand column contains the property name; the right-hand column, the property value. Next, select a property with the mouse. A control (button, text box, etc.) will appear in the cell to the right of the property name that will allow you to change the property value.

Controls

Form Properties

Form properties are set in the same manner as any control, from the Property Editor. Select form properties in one of two ways: 1) click the mouse on a blank portion of the form or 2) select the form from the drop-down list box at the top of the Property Editor dialog.

There are five separate eQuate Actions that can be associated with a form: 1) form initial, 2) form activate, 3) host message 4) host error and 5) screen complete check.

Form Initial Action

The Form Initial Action is executed each time the form is loaded.

Form Activate Action

The Form Activate Action takes place when the form regains focus.

Host Message Action

The Host Message Action is activated when any message is received from the host (see Screen Complete Action, below).

Host Error Action

The Host Error Action is executed whenever an error message is received from the host that matches error text stored in the eQuate application database (see Host Error Ids tab in eQuate Application Manager and Host Error Detection String Edit) and is set to, "Run HOSTERROR Sub in Action Script

Screen Complete Check Action

In most applications, a screen is transmitted to the host and a single message containing everything needed to paint the screen is received in response; however, in some cases, the response is contained in more than one host message. For example, the first host message paints the screen format and data. Then, a second host message adds a status message at the bottom of the screen. In such a case, it is possible that the Host Message Action of the form will be executed twice, which may cause invalid results.

The Screen Complete Check Action is provided to allow a screen made up of multiple host messages to be completed before other processing takes place under control of the eQuate developer. Checking for screen completion can be very complex and may require analyzing many conditions, which is why eQuate cannot do completion checks automatically. Using action scripts, the developer can perform whatever analysis is necessary.

Typically, a Screen Complete Check Action flow would be coded something like the following:

```
Sub ScreenCompleteCheck()
    Dim Done as Integer
    Dim Cnt as Integer

    Done = False
    Cnt = 0
    While Done = False
        If Trim$(GetString("F_Code")) = "" Then
            Done = True
        ElseIf Trim$(GetString("STATUSLINE")) <> "" Then
            Done = True
        Else
            Cnt = Cnt + 1
            If Cnt > 5 Then
                MsgBox "Did not get complete screen."
                Exit Sub
            End If
            Wait 500 ' Wait a half second, so messages can come in.
        End If
    Wend
End Sub
```

This above Screen Complete Check Action checks for two conditions that indicate the screen is complete. When the F_Code data field is blank, the screen is blank and completed in a single message. If the F_Code field has something in it, then something is expected in the STATUSLINE data field, which is always received as a separate message from the host.

Failure to detect a complete screen will not cause eQuate to take any action. The condition must be handled by the developer.

Form Properties

Forms have the following defined properties, some of which can be manipulated under the control of your application:

<u>Property</u>	<u>Type</u>	<u>Design-time</u>	<u>Run-time</u>
Action	N/A	Y	
AutoCenter	Integer	Y	Y
BackColor	Integer	Y	Y
BlinkBackColor	Integer	Y	Y
BlinkForeColor	Integer	Y	Y
BorderStyle	Integer	Y	Y
Caption	String	Y	Y
Ctl3d	Integer	Y	Y
Font	N/A	Y	
FillFormWhenMaximized	Integer	Y	Y
ForeColor	Integer	Y	Y
Height	Integer	Y	Y
Help	N/A	Y	
Left	Integer	Y	Y
MaximizedButton	Integer	Y	Y
MinimizedButton	Integer	Y	Y
ShowHint	Integer	Y	Y
Top	Integer	Y	Y
Width	Integer	Y	Y
WindowState	Integer	Y	Y
WinHelpFile	N/A	Y	

Text Labels

A Text label is used to place non-data text on the form or "output only" data fields. Text labels can be used to create form titles, captions for data fields (edit boxes), captions for groups of controls, areas to receive messages from eQuate Actions, and in general, a more Windows-like appearance to the form.

eQuate Actions associated with a text label are executed when the text label is clicked.

Related eQuate Action Statements:

```
GetNumericProp
GetState
GetString
GetStringProp
SetNumericProp
SetState
SetString
SetStringProp
```

Text labels have the following defined properties, some of which can be manipulated under the control of your application:

<u>Property</u>	<u>Type</u>	<u>Design-time</u>	<u>Run-time</u>
Action	N/A	Y	
Align	Integer	Y	Y
Alignment	Integer	Y	Y
AutoSize	Integer	Y	Y
BackColor	Integer	Y	Y
Border	Integer	Y	Y
Caption	String	Y	Y
Ctl3d	Integer	Y	Y
DataSource	N/A	Y	
Enabled	Integer	Y	Y
Font	N/A	Y	

<u>Property</u>	<u>Type</u>	<u>Design-time</u>	<u>Run-time</u>
ForeColor	Integer	Y	Y
Height	Integer	Y	Y
Help	N/A	Y	
Hint	String	Y	Y
Left	Integer	Y	Y
Name	String	Y	
ParentColor	Integer	Y	Y
ParentCtl3d	Integer	Y	Y
ParentFont	Integer	Y	Y
ParentShowHint	Integer	Y	Y
ShowAccelChar	Integer	Y	Y
ShowHint	Integer	Y	Y
Top	Integer	Y	Y
Transparent	Integer	Y	Y
Visible	Integer	Y	Y
Width	Integer	Y	Y

Edit Boxes

Input/output screen fields are automatically converted to Edit Boxes and placed on the form when the captured screen or form file is imported by the eQuate Application Manager. Normally, you do not add edit boxes unless you also change the host program and/or screen definition.

eQuate Actions associated with an edit box are executed when the user clicks in the edit box.

Related eQuate Action Statements:

```
GetNumericProp
GetState
GetString
GetStringProp
SetNumericProp
SetState
SetString
SetStringProp
```

Edit Boxes have the following defined properties, some of which can be manipulated under the control of your application:

<u>Property</u>	<u>Type</u>	<u>Design-time</u>	<u>Run-time</u>
Action	N/A	Y	
Alignment	Integer	Y	Y
AutoSize	Integer	Y	Y
BackColor	Integer	Y	Y
CharCase	Integer	Y	
Ctl3d	Integer	Y	Y
DataSource	N/A	Y	
EditMask	String	Y	Y
Enabled	Integer	Y	Y
Font	N/A	Y	
ForeColor	Integer	Y	Y
Height	Integer	Y	Y
Help	N/A	Y	
Hint	String	Y	Y
Left	Integer	Y	Y
MaxLength	String	Y	Y
Name	String	Y	
ParentColor	Integer	Y	Y
ParentCtl3d	Integer	Y	Y

<u>Property</u>	<u>Type</u>	<u>Design-time</u>	<u>Run-time</u>
ParentFont	Integer	Y	Y
ParentShowHint	Integer	Y	Y
PassWordChar	String	Y	Y
PickList	N/A	Y	
ShowHint	Integer	Y	Y
TabOrder	String	Y	Y
TabStop	Integer	Y	Y
Text	String	Y	Y
Top	Integer	Y	Y
Visible	Integer	Y	Y
Width	Integer	Y	Y

Command Buttons

A command button is used to carry out a command or action when a user clicks the button. Any number of command buttons may be placed on a form or panel (see also, Panels).

eQuate actions associated with a command button are executed when the button is clicked.

Related eQuate Action Statements:

```
GetNumericProp
GetState
GetString
GetStringProp
SetNumericProp
SetState
SetString
SetStringProp
```

Command buttons have the following defined properties, some of which can be manipulated under the control of your application:

<u>Property</u>	<u>Type</u>	<u>Design-time</u>	<u>Run-time</u>
Action	N/A	Y	
Bitmap	N/A	Y	
BitmapPosition	Integer	Y	Y
Cancel	Integer	Y	Y
Caption	String	Y	Y
Default	Integer	Y	Y
Enabled	Integer	Y	Y
Font	N/A	Y	
Height	Integer	Y	Y
Hint	String	Y	Y
Left	Integer	Y	Y
Name	String	Y	
NumBitMaps	Integer	Y	Y
ParentFont	Integer	Y	Y
ParentShowHint	Integer	Y	Y
ShowHint	Integer	Y	Y
TabOrder	String	Y	Y
TabStop	Integer	Y	Y
Top	Integer	Y	Y
Visible	Integer	Y	Y
Width	Integer	Y	Y

Speed Buttons

A speed button is a non-windowed control taking fewer system resources.

You can make a group of speed buttons act like radio buttons by setting the GroupIndex property to something other than 0.

The Down property specifies whether the button is down or up. The Down property is only in effect when the GroupIndex is not zero, otherwise it will remain False. The AllowAllUp property affects this property also.

Speed buttons can also have a flat property that makes the outline show on mouse fly-over.

eQuate actions associated with a speed button are executed when the button is clicked.

Related eQuate Action Statements:

GetNumericProp
 GetState
 GetString
 GetStringProp
 SetNumericProp
 SetState
 SetString
 SetStringProp

Speed buttons have the following defined properties, some of which can be manipulated under the control of your application:

<u>Property</u>	<u>Type</u>	<u>Design-time</u>	<u>Run-time</u>
Action	N/A	Y	
AllowAllUp	Integer	Y	Y
Bitmap	N/A	Y	
BitmapPosition	Integer	Y	Y
Caption	String	Y	Y
Down	Boolean	Y	Y
Enabled	Integer	Y	Y
Flat	Integer	Y	Y
Font	N/A	Y	
GroupIndex	Integer	Y	Y
Height	Integer	Y	Y
Hint	String	Y	Y
Left	Integer	Y	Y
Name	String	Y	
NumBitMaps	Integer	Y	Y
ParentShowHint	Integer	Y	Y
ShowHint	Integer	Y	Y
Top	Integer	Y	Y
Visible	Integer	Y	Y
Width	Integer	Y	Y

Check Boxes

A check box is used to display a true/false, on/off or yes/no option that the user can set or clear by clicking. A "3" in a check box indicates that it is selected, set to on/true/yes. Any number of check boxes on a form can be checked at one time.

eQuate actions associated with a check box are executed after the check box's state has changed (i.e., from unchecked to checked, or checked to unchecked). If an action changes the state of a check box, the check box's action will be triggered.

Related eQuate Action Statements:

GetNumericProp
 GetState
 GetString
 GetStringProp
 SetNumericProp
 SetState
 SetString
 SetStringProp

Check boxes have the following defined properties, some of which can be manipulated under the control of your application:

<u>Property</u>	<u>Type</u>	<u>Design-time</u>	<u>Run-time</u>
Action	N/A	Y	
Alignment	Integer	Y	Y
BackColor	Integer	Y	Y
Caption	String	Y	Y
Checked	Integer	Y	Y
Ctl3d	Integer	Y	Y
Enabled	Integer	Y	Y
Font	N/A	Y	
ForeColor	Integer	Y	Y
Height	Integer	Y	Y
Hint	String	Y	Y
Left	Integer	Y	Y
Name	String	Y	
ParentColor	Integer	Y	Y
ParentCtl3d	Integer	Y	Y
ParentFont	Integer	Y	Y
ParentShowHint	Integer	Y	Y
ShowHint	Integer	Y	Y
TabOrder	String	Y	Y
TabStop	Integer	Y	Y
Top	Integer	Y	Y
Visible	Integer	Y	Y
Width	Integer	Y	Y

See also, Edit Mask.

Option Buttons

An option button (frequently referred to as a Radio button) is used in conjunction with other option buttons to offer multiple choices, from which the user can select only one. Option buttons may be placed directly on the form, on a panel or in a button group. The buttons will be mutually exclusive upon the control on which they are placed.

eQuate actions associated with an option button are executed when the option button's state has changed. If an action changes the state of an option button, the option button's action will be triggered.

Related eQuate Action Statements:

```
GetNumericProp
GetState
GetString
GetStringProp
SetNumericProp
SetState
SetString
SetStringProp
```

Option buttons have the following defined properties, some of which can be manipulated under the control of your application:

<u>Property</u>	<u>Type</u>	<u>Design-time</u>	<u>Run-time</u>
Action	N/A	Y	
Alignment	Integer	Y	Y
BackColor	Integer	Y	Y
Checked	Integer	Y	Y
Ctl3d	Integer	Y	Y
Enabled	Integer	Y	Y
Font	N/A	Y	
ForeColor	Integer	Y	Y

<u>Property</u>	<u>Type</u>	<u>Design-time</u>	<u>Run-time</u>
Height	Integer	Y	Y
Hint	String	Y	Y
Left	Integer	Y	Y
Name	String	Y	
ParentColor	Integer	Y	Y
ParentCtl3d	Integer	Y	Y
ParentFont	Integer	Y	Y
ParentShowHint	Integer	Y	Y
ShowHint	Integer	Y	Y
TabOrder	String	Y	Y
TabStop	Integer	Y	Y
Top	Integer	Y	Y
Visible	Integer	Y	Y
Width	Integer	Y	Y

List Boxes

A list box is used to display a list of items from which a user may view or select (to select from the list requires that an eQuate action be associated with the list box). The list box size on the form is not dependent upon the number of values to be placed in the box. If a list box contains more values than can be accommodated by the size of the list box, a scroll bar will appear making all items in the list accessible. If a list is to contain many values, it might be desirable to use a drop-down list box to conserve space on the form.

eQuate actions associated with a list box are executed when an item is clicked or double-clicked (two separate actions) from the list.

Related eQuate Action Statements:

GetNumericProp
 GetState
 GetStringProp
 ListClear
 ListCount
 ListGetIndex
 ListGetItem
 ListSetIndex
 ListSetItem
 ListItemAdd
 ListItemRemove
 SetNumericProp
 SetState
 SetStringProp

List boxes have the following defined properties, some of which can be manipulated under the control of your application:

<u>Property</u>	<u>Type</u>	<u>Design-time</u>	<u>Run-time</u>
Action	N/A	Y	
Align	Integer	Y	Y
BackColor	Integer	Y	Y
Ctl3d	Integer	Y	Y
Enabled	Integer	Y	Y
Font	N/A	Y	
ForeColor	Integer	Y	Y
Height	Integer	Y	Y
Hint	String	Y	Y
Items	String	Y	Y
Left	Integer	Y	Y
Name	String	Y	Y
ParentColor	Integer	Y	Y
ParentCtl3d	Integer	Y	Y

<u>Property</u>	<u>Type</u>	<u>Design-time</u>	<u>Run-time</u>
ParentFont	Integer	Y	Y
ParentShowHint	Integer	Y	Y
PickList	String	Y	Y
ShowHint	Integer	Y	Y
Sorted	Integer	Y	Y
TabOrder	String	Y	Y
TabStop	Integer	Y	Y
Top	Integer	Y	Y
Visible	Integer	Y	Y
Width	Integer	Y	Y

Drop-down List Boxes

Like the list box, the drop-down list box is used to display a list of items from which a user can select; however, the drop-down list box takes up less room on the form, and as such, is useful when the selection list contains many values. The drop-down list box only occupies one line on the form. In addition, the drop-down list box is limited to a single column.

eQuate actions associated with a drop-down list box are executed when an item is selected (clicked) from the list.

Related eQuate Action Statements:

GetNumericProp
 GetState
 GetStringProp
 ListClear
 ListCount
 ListGetIndex
 ListGetItem
 ListSetIndex
 ListSetItem
 ListItemAdd
 ListItemRemove
 SetNumericProp
 SetState
 SetStringProp

Drop-down list boxes have the following defined properties, some of which can be manipulated under the control of your application:

<u>Property</u>	<u>Type</u>	<u>Design-time</u>	<u>Run-time</u>
Action	N/A	Y	
BackColor	Integer	Y	Y
Ctl3d	Integer	Y	Y
Enabled	Integer	Y	Y
Font	N/A	Y	
ForeColor	Integer	Y	Y
Height	Integer	Y	Y
Hint	String	Y	Y
Items	String	Y	Y
Left	Integer	Y	Y
Name	String	Y	Y
ParentColor	Integer	Y	Y
ParentCtl3d	Integer	Y	Y
ParentFont	Integer	Y	Y
ParentShowHint	Integer	Y	Y
PickList	String	Y	Y
ShowHint	Integer	Y	Y
Sorted	Integer	Y	Y
TabOrder	String	Y	Y

<u>Property</u>	<u>Type</u>	<u>Design-time</u>	<u>Run-time</u>
TabStop	Integer	Y	Y
Top	Integer	Y	Y
Visible	Integer	Y	Y
Width	Integer	Y	Y

Multi-column List Boxes

A multi-column list box is similar to the standard list box in usage and behavior; however, in addition to supporting multiple columns it allows optional column headings.

eQuate actions associated with a multi-column list box are executed when an item is clicked or double-clicked (two separate actions) from the list.

Related eQuate Action Statements:

GetNumericProp
 GetState
 GetStringProp
 ListClear
 ListColHeader
 ListGetColText
 ListCount
 ListGetIndex
 ListSetColText
 ListSetIndex
 ListItemAdd
 ListItemRemove
 SetNumericProp
 SetState
 SetStringProp

Multi-column list boxes have the following defined properties, some of which can be manipulated under the control of your application:

<u>Property</u>	<u>Type</u>	<u>Design-time</u>	<u>Run-time</u>
Action	N/A	Y	
Align	Integer	Y	Y
BackColor	Integer	Y	Y
ColumnHeaders	Integer	Y	Y
Columns	N/A	Y	
Ctl3d	Integer	Y	Y
Enabled	Integer	Y	Y
Font	N/A	Y	
ForeColor	Integer	Y	Y
Height	Integer	Y	Y
Hint	String	Y	Y
Left	Integer	Y	Y
Name	String	Y	Y
NumericSort	Boolean	Y	Y
ParentColor	Integer	Y	Y
ParentCtl3d	Integer	Y	Y
ParentFont	Integer	Y	Y
ParentShowHint	Integer	Y	Y
PickList	String	Y	Y
ShowHint	Integer	Y	Y
Sort Column	Integer	Y	Y
SortDescending	Boolean	Y	Y
Sorted	Integer	Y	Y
TabOrder	String	Y	Y
TabStop	Integer	Y	Y

<u>Property</u>	<u>Type</u>	<u>Design-time</u>	<u>Run-time</u>
Top	Integer	Y	Y
Visible	Integer	Y	Y
Width	Integer	Y	Y

See also, Multi-Column List Setup.

Memo

A memo control is basically a multi-line edit box. The user can type and edit large amounts of text in a Memo control. Access to a Memo from an Action Script is similar to handling items in a List Box. Use the following eQuate, list oriented, Action statements to manipulate lines in a memo control: ListClear, ListCount, ListItemAdd and ListGetItem.

If GetString and SetString are used with Memo control, the following should be considered:

- With SetString, text is moved into the memo and split into multiple lines depending upon the setting of the WordWrap property. For line breaks, carriage returns (Chr\$(13)) may be inserted into the text before using SetString. If WordWrap is false and no carriage returns are present, the memo will contain a single line of text.
- With GetString, all lines of the memo are concatenated into a single string with spaces added between lines replacing carriage return characters. Only lines ending with a carriage return are modified. The carriage return is replaced with a single space. Lines automatically wrapped due to the control's width do not contain carriage returns.

Related eQuate Action Statements:

```
GetString
ListClear
ListCount
ListGetItem
ListItemAdd
ListItemRemove
SetString
```

Memo controls have the following defined properties, some of which can be manipulated under the control of your application:

<u>Property</u>	<u>Type</u>	<u>Design-time</u>	<u>Run-time</u>
Action	N/A	Y	
Align	Integer	Y	Y
BackColor	Integer	Y	Y
BorderStyle	Integer	Y	Y
Ctl3d	Integer	Y	Y
Enabled	Integer	Y	Y
Font	N/A	Y	
ForeColor	Integer	Y	Y
Height	Integer	Y	Y
Hint	String	Y	Y
Left	Integer	Y	Y
Lines	String	Y	Y
MaxLength	Integer	Y	Y
Name	String	Y	Y
ParentCtl3d	Integer	Y	Y
ParentFont	Integer	Y	Y
ParentShowHint	Integer	Y	Y
PickList	String	Y	Y
ReadOnly	Integer	Y	Y
ScrollBars	Integer	Y	Y
ShowHint	Integer	Y	Y
TabOrder	String	Y	Y
TabStop	Integer	Y	Y
Top	Integer	Y	Y

<u>Property</u>	<u>Type</u>	<u>Design-time</u>	<u>Run-time</u>
Visible	Integer	Y	Y
WantsReturns	Integer	Y	Y
Width	Integer	Y	Y
WordWrap	Integer	Y	Y

Bevels

A bevel may be used to enhance the appearance of the form.

eQuate actions may NOT be associated with a bevel.

Related eQuate Action Statements:

GetNumericProp
 GetStringProp
 SetNumericProp
 SetStringProp

Bevels have the following defined properties, some of which can be manipulated under the control of your application:

<u>Property</u>	<u>Type</u>	<u>Design-time</u>	<u>Run-time</u>
Align	Integer	Y	Y
Name	String	Y	Y
Shape	Integer	Y	Y
Style	Integer	Y	Y
Visible	Integer	Y	Y

Button Groups

Button groups are used to group option buttons logically onto a single control. When options need to be mutually exclusive where only one may be selected, the button group should be used. Note: Option buttons may be placed on a panel when tab control is required between buttons.

When the button group is first placed on the form, no option buttons appear in the group. Use the Items property to add the option buttons. Next, use the button group sizing handles to size the group to contain the option items entered. The Item Index property can be used to specify which option in the group is initially set when the form is loaded by the eQuate Session Manager (e.g., -1 means that no option is set, 0 means the first option is set, 1, the second, etc.).

eQuate actions associated with a button group are executed when the state of any option button in the group has changed. If an action changes the state of an option button, the button group's action will be triggered.

Button Groups may be populated at runtime just like list boxes.

Related eQuate Action Statements:

GetNumericProp
 GetState
 GetStringProp
 ListClear
 ListCount
 ListGetIndex
 ListGetItem
 ListSetIndex
 ListSetItem
 ListItemAdd
 ListItemRemove
 SetNumericProp
 SetState
 SetStringProp

Button groups have the following defined properties, some of which can be manipulated under the control of your application:

<u>Property</u>	<u>Type</u>	<u>Design-time</u>	<u>Run-time</u>
Action	N/A	Y	
Align	Integer	Y	Y

<u>Property</u>	<u>Type</u>	<u>Design-time</u>	<u>Run-time</u>
BackColor	Integer	Y	Y
ButtonStyle	Integer	Y	Y
Caption	String	Y	Y
Columns	Integer	Y	Y
Ctl3d	Integer	Y	Y
Enabled	Integer	Y	Y
Font	N/A	Y	
ForeColor	Integer	Y	Y
Height	Integer	Y	Y
Hint	String	Y	Y
ItemIndex	Integer	Y	Y
Items	String	Y	Y
Left	Integer	Y	Y
Name	String	Y	Y
ParentCtl3d	Integer	Y	Y
ParentFont	Integer	Y	Y
ParentShowHint	Integer	Y	Y
PickList	String	Y	Y
ShowHint	Integer	Y	Y
TabOrder	String	Y	Y
TabStop	Integer	Y	Y
Top	Integer	Y	Y
Visible	Integer	Y	Y
Width	Integer	Y	Y

Group Boxes

A group box may be used to logically group or frame a set of eQuate controls. The group box may not be used to establish option groups, and as such, has no physical functionality.

eQuate actions may NOT be associated with group boxes.

Related eQuate Action Statements:

```
GetNumericProp
GetState
GetString
GetStringProp
SetNumericProp
SetState
SetString
SetStringProp
```

Group boxes have the following defined properties, some of which can be manipulated under the control of your application:

<u>Property</u>	<u>Type</u>	<u>Design-time</u>	<u>Run-time</u>
BackColor	Integer	Y	Y
Caption	String	Y	Y
Ctl3d	Integer	Y	Y
Enabled	Integer	Y	Y
Font	N/A	Y	
ForeColor	Integer	Y	Y
Height	Integer	Y	Y
Hint	String	Y	Y
Left	Integer	Y	Y
Name	String	Y	Y
ParentColor	Integer	Y	Y
ParentCtl3d	Integer	Y	Y

<u>Property</u>	<u>Type</u>	<u>Design-time</u>	<u>Run-time</u>
ParentFont	Integer	Y	Y
ParentShowHint	Integer	Y	Y
ShowHint	Integer	Y	Y
Top	Integer	Y	Y
Visible	Integer	Y	Y
Width	Integer	Y	Y

Panels

A panel may be used to house one or more controls in a logical group. The controls behave on the panel as if they were placed on a form. In other words, if you move the panel the controls on the panel will move with the panel. In addition, if a "parent" property of a control is set to True, the control will take the property of the panel, the parent.

eQuate actions associated with a panel are executed when the panel is clicked. The panel must be enabled and visible to click it. A panel can be used as a big button since its caption text can wrap from line to line. Panel text can also be aligned left, right or centered. The caption of regular buttons can contain only a single line of text centered in the client area.

Related eQuate Action Statements:

```
GetNumericProp
GetState
GetString
GetStringProp
SetNumericProp
SetState
SetString
SetStringProp
```

Panels have the following defined properties, some of which can be manipulated under the control of your application:

<u>Property</u>	<u>Type</u>	<u>Design-time</u>	<u>Run-time</u>
Action	N/A	Y	
Align	Integer	Y	Y
Alignment	Integer	Y	Y
BackColor	Integer	Y	Y
BevelInner	Integer	Y	Y
BevelOuter	Integer	Y	Y
BevelWidth	Integer	Y	Y
BorderStyle	Integer	Y	Y
BorderWidth	Integer	Y	Y
Caption	String	Y	Y
Ctl3d	Integer	Y	Y
Enabled	Integer	Y	Y
Font	N/A	Y	
ForeColor	Integer	Y	Y
Height	Integer	Y	Y
Hint	String	Y	Y
Left	Integer	Y	Y
Name	String	Y	Y
ParentColor	Integer	Y	Y
ParentCtl3d	Integer	Y	Y
ParentFont	Integer	Y	Y
ParentShowHint	Integer	Y	Y
ShowHint	Integer	Y	Y
Top	Integer	Y	Y
Visible	Integer	Y	Y
Width	Integer	Y	Y

Splitters

A splitter is used to allow the end user to adjust the viewing area of controls like list boxes and panels at runtime.

eQuate actions may NOT be associated with splitters.

Related eQuate Action Statements:

GetNumericProp
GetStringProp
SetNumericProp
SetStringProp

Splitters have the following defined properties, some of which can be manipulated under the control of your application:

<u>Property</u>	<u>Type</u>	<u>Design-time</u>	<u>Run-time</u>
Align	Integer	Y	Y
AutoSnap	Integer	Y	Y
BackColor	Integer	Y	Y
Height	Integer	Y	Y
Hint	String	Y	Y
Left	Integer	Y	Y
MinSize	Integer	Y	Y
Top	Integer	Y	Y
Visible	Integer	Y	Y
Width	Integer	Y	Y

Images

Images not only can be used to enhance the appearance of a form but can be functional Windows controls that when selected perform a prescribed action.

eQuate actions associated with an image are executed when the image is clicked.

Related eQuate Action Statements:

GetNumericProp
GetStringProp
LoadImage
SetNumericProp
SetStringProp

Images have the following defined properties, some of which can be manipulated under the control of your application:

<u>Property</u>	<u>Type</u>	<u>Design-time</u>	<u>Run-time</u>
Action	N/A	Y	
Align	Integer	Y	Y
Auto Size	Integer	Y	Y
Border	Integer	Y	Y
Center	Integer	Y	Y
Enabled	Integer	Y	Y
Height	Integer	Y	Y
Hint	String	Y	Y
Left	Integer	Y	Y
Name	String	Y	
ParentShowHint	Integer	Y	Y
Picture	N/A	Y	
ShowHint	Integer	Y	Y
Stretch	Integer	Y	Y
Top	Integer	Y	Y
<u>Transparent</u>	Boolean	Y	Y
<u>TransparentColor</u>	Integer	Y	Y
<u>TransparentMode</u>	Integer	Y	Y

<u>Property</u>	<u>Type</u>	<u>Design-time</u>	<u>Run-time</u>
Visible	Integer	Y	Y
Width	Integer	Y	Y

Media Players

The media player control bar can be placed on a form to activate and play a movie. Normally, the form on which the control is placed loads the multi-media file when the form is first displayed. The control itself has no action associated with it; therefore, some other event, like the initial form action or command button, must be used to load the media file. For Example:

```
Sub FormInitial()  
  ' Action for FormInitial  
  Rslt = LoadMMFile("Player", "C:\eQuateData\TESTx\SPEEDIS.AVI")  
End Sub
```

"Player" is the name of the media player control as set in the Name property.

eQuate actions may NOT be associated with the media player control.

Related eQuate Action Statements:

```
GetNumericProp  
GetStringProp  
LoadMMFile  
SetNumericProp  
SetStringProp
```

The media player control has the following defined properties, some of which can be manipulated under the control of your application:

<u>Property</u>	<u>Type</u>	<u>Design-time</u>	<u>Run-time</u>
Enabled	Integer	Y	Y
Height	Integer	Y	Y
Hint	String	Y	Y
Left	Integer	Y	Y
MonoChromeButtons	Integer	Y	Y
Name	String	Y	
ParentShowHint	Integer	Y	Y
ShowHint	Integer	Y	Y
TabOrder	String	Y	Y
TabStop	Integer	Y	Y
Top	Integer	Y	Y
Visible	Integer	Y	Y
Width	Integer	Y	Y

Date/Time Labels

The data/time label is used to supply the current date and time (machine time) to the user on the form or control. For example, this control could be placed on a panel as a part of a status bar.

eQuate actions may NOT be associated with a date/time label.

Related eQuate Action Statements:

```
GetNumericProp  
GetState  
GetString  
GetStringProp  
SetNumericProp  
SetState  
SetString  
SetStringProp
```

Date/Time labels have the following defined properties, some of which can be manipulated under the control of your application:

<u>Property</u>	<u>Type</u>	<u>Design-time</u>	<u>Run-time</u>
Align	Integer	Y	Y
Alignment	Integer	Y	Y
BackColor	Integer	Y	Y
BlinkColor	Integer	Y	Y
Blinking	Integer	Y	Y
BlinkIntervalOff	Integer	Y	Y
BlinkIntervalOn	Integer	Y	Y
Enabled	Integer	Y	Y
FlatColor	Integer	Y	Y
Font	N/A	Y	
ForeColor	Integer	Y	Y
Format	String	Y	Y
FrameStyle	Integer	Y	Y
Height	Integer	Y	Y
Hint	String	Y	Y
Left	Integer	Y	Y
Name	String	Y	Y
ParentCtl3d	Integer	Y	Y
ParentFont	Integer	Y	Y
ParentShowHint	Integer	Y	Y
ShowHint	Integer	Y	Y
Top	Integer	Y	Y
Visible	Integer	Y	Y
Width	Integer	Y	Y

See also, Date Time Format Editor.

Browser

The browser control can be placed on a form to embed the browser on a form. The URL is set at runtime by an action using the "SetString" statement.

```
Sub FormInitial()  
  ' Action for FormInitial  
  SetString "Browser_1", "http://www.kmsys.com"  
End Sub
```

"Browser_1" is the name of the media player control as set in the Name property.

eQuate actions may NOT be associated with the browser control.

Related eQuate Action Statements:

```
GetNumericProp  
GetStringProp  
SetNumericProp  
SetString  
SetStringProp
```

The browser control has the following defined properties, some of which can be manipulated under the control of your application:

<u>Property</u>	<u>Type</u>	<u>Design-time</u>	<u>Run-time</u>
Enabled	Integer	Y	Y
Height	Integer	Y	Y
Hint	String	Y	Y
Left	Integer	Y	Y
Name	String	Y	
ParentShowHint	Integer	Y	Y
ShowHint	Integer	Y	Y

<u>Property</u>	<u>Type</u>	<u>Design-time</u>	<u>Run-time</u>
TabOrder	String	Y	Y
TabStop	Integer	Y	Y
Top	Integer	Y	Y
Visible	Integer	Y	Y
Width	Integer	Y	Y

URL Link

URL links can be used to place a URL link on a form. A URL link is similar to a Text Label in that its caption appears on the form, but it differs in that when clicked, it opens a separate browser window to the URL specified in the URL property. Furthermore, a URL link differs from a Browser control in that no browser is embedded on the form.

Actions associated with a URL link are executed when the URL link is clicked.

Related Action Statements:

```
GetNumericProp
GetState
GetString
GetStringProp
SetNumericProp
SetState
SetString
SetStringProp
```

Text labels have the following defined properties, some of which can be manipulated under the control of your application:

<u>Property</u>	<u>Type</u>	<u>Design-time</u>	<u>Runtime</u>
Action	N/A	Y	
Align	Integer	Y	Y
Alignment	Integer	Y	Y
AutoSize	Integer	Y	Y
BackColor	Integer	Y	Y
Caption	String	Y	Y
Enabled	Integer	Y	Y
Font	N/A	Y	
ForeColor	Integer	Y	Y
Height	Integer	Y	Y
Hint	String	Y	Y
Left	Integer	Y	Y
Name	String	Y	
ParentColor	Integer	Y	Y
ParentFont	Integer	Y	Y
ParentShowHint	Integer	Y	Y
ShowAccelChar	Integer	Y	Y
ShowHint	Integer	Y	Y
Top	Integer	Y	Y
Transparent	Integer	Y	Y
URL	String	Y	Y
Visible	Integer	Y	Y
Width	Integer	Y	Y

Control Properties

Align Property

Use this property to align and automatically size the control on the parent control. Predefined constants are alNone (default), alTop, alBottom, alLeft, alRight and alClient.

alTop and alBottom will align the control at the top or bottom edge of the parent control, respectively. The control will take the height of the parent control.

alLeft and alBottom will align the control at the left or right edge of the parent control, respectively. The control will take the width of the parent control.

alClient will align at the width and height of the parent control.

Alignment Property

This property is used to align the text in a control. Predefined constants are alLeftJustify, alRightJustify and alCenter (not applicable for edit boxes).

AllowAllUp Property

The AllowAllUp property is used to force all speed buttons in a group to be up by default.

AllowAllUp is used in conjunction with GroupIndex. If you want a group of speed buttons to toggle like car radio buttons you can set the GroupIndex of all of all buttons in the group to the same number. By default, one of the group will always be down. AllowAllUp changes the default behavior so that all buttons in such a group can be up.

AutoCenter Property

Use this control to cause the form to be centered on the desktop at runtime.

AutoSize Property

This property is used to size the control according to the size of the text or image placed in the control. The default is False.

AutoSnap Property

The AutoSnap property is used to determine if the splitter will automatically snap to the edge of a parent control. The default is True.

BackColor Property

Use this control to set the background (non-text) color of a control. The default for most controls is CP_BtnFace. The default for edit boxes is CP_Window.

For forms, there is a rule regarding the form's Ctl3d and BackColor properties. If the BackColor is CP_BtnFace, when you toggle Ctl3d from True to False, the BackColor will toggle from CP_BtnFace to CP_Window. If the BackColor is something else, the current color is not change when Ctl3d is toggled.

See also Predefined Constants.

BevelInner Property

Use this property along with the BorderWidth property to add an inner bevel to a Panel control. The BorderWidth is the number of pixels from the outer edge of the panel that the inner bevel will begin. Predefined constants are bvNone (default), bvLowered and bvRaised. See also, BevelOuter Property, BevelWidth, BorderStyle Property and BorderWidth.

BevelOuter Property

Use this property to apply a lowered or raised bevel on the outer edge on a Panel control. Predefined constants are bvNone, bvLowered and bvRaised (default). See also, BevelInner Property, BevelWidth, BorderStyle Property and BorderWidth.

BevelWidth Property

The BevelWidth property is used to specify the width of any bevel on a Panel control. The default for this property is 1 pixel. See also, BevelInner Property, BevelOuter Property, BorderStyle Property and BorderWidth.

Bitmap Property

Use this property to select an optional bitmap to place on a button face.

BitmapPosition Property

Use the BitmapPosition property to specify the position of the bitmap on the button face. Predefined constants are blLeft (default), blRight, blTop and blBottom.

BlinkBackColor Property

This property is used to set the background (non-text) color for any edit boxes on the form that are linked to blinking data. See also, BlinkForeColor.

BlinkColor Property

Use this property to set the color for blinking text on the DateTimeLabel control when the Blinking property is set to True. The default color is CP-Highlight. This color is only visible on the control if the Blinking property is set to True. See also, Blinking Property, BlinkIntervalOff Property and BlinkIntervalOn Property.

BlinkForeColor Property

This property is used to set the foreground (text) color for any edit boxes on the form that are linked to blinking data. See also, BlinkBackColor.

Blinking Property

Use the Blinking property property to make the text on the DateTimeLabel control blink. The default is False. See also, BlinkColor Property, BlinkIntervalOff Property and BlinkIntervalOn Property.

BlinkIntervalOff Property

This property is used to set the interval (in milliseconds) that the text in the DateTimeLabel control will NOT be displayed in BlinkColor property color. The default is 500 milliseconds. This value is only used if the Blinking property is set to True. See also, BlinkColor Property, Blinking Property and BlinkIntervalOn Property.

BlinkIntervalOn Property

Use this property to set the interval (in milliseconds) that the text in the DateTimeLabel control will be displayed in the BlinkColor property color. The default is 500 milliseconds. This value is only used if the Blinking property is set to True. See also, BlinkColor Property, Blinking Property and BlinkIntervalOff Property.

Border Property

This property may be used to place a border around a Text Label or Image control. The default is False. For Text Labels, this control is impacted by the setting of the Ctl3d property.

BorderStyle Property

Use this property to set the border style for Forms, Memo and Panel controls. Predefined constants are bsNone, bsSingle, bsSizeable (default) and bsDialog. The bsSizeable and bsDialog settings only apply to forms.

The bsSizeable value displays the form as a standard resizable dialog.

The bsDialog value displays the form as a non-resizable dialog with a standard border. Also, with the bsDialog value, the system menu (upper-left) will not be shown, and the Minimize, Maximize and Close buttons (upper-right) will not be shown.

The bsSingle value causes a non-resizable dialog with a single-line border to be displayed.

The bsNone value displays a non-resizable dialog with no border.

For Panels, also see, BevelInner Property, BevelOuter Property, BevelWidth and BorderWidth.

BorderWidth Property

Use the BorderWidth property in conjunction with the BevelInner property to place a border on a Panel control. The BorderWidth is the number of pixels from the outer edge of the panel that the inner bevel will begin. The default is 1 pixel. See also, BevelInner Property, BevelOuter Property, BevelWidth and BorderStyle Property.

ButtonStyle Property

This property may be used to select the button style for the buttons in a Button Group control. Predefined constants are bsRadio (default) and bsPush.

Cancel Property

Use the Cancel property to specify whether a button's Action executes when the Escape key is pressed. If Cancel is True, the button's Action executes when the user presses Esc. Although a Form can have more than one Cancel button, the form calls the Action only for the first visible button in the tab order. The default value is False. To set the initial tab order, select Form Edits | Tab Order from the Edit menu on the eQuate Form Designer Toolbar or right-click anywhere on the form and select Tab Order. You may also order the tab order at runtime by using the TabOrder property.

Caption Property

This property is used to set text that will appear as a caption on a control; e.g., the caption in blue at the top of a dialog, the text in a text label, etc.

To add an accelerator key, add an ampersand (&) in front of any character in the caption. That character will appear underlined and becomes the accelerator key for that control. Accelerator key values should be unique amongst all accelerator key values on the form. When an accelerator key is pressed at runtime, the control's action is executed. See also, ShowAccelChar Property.

Note: On Text Labels, the accelerator character has no effect, other than how it appears in the text.

Note: Most control captions are a single line of characters; however, Text Labels allow for multiple lines in the caption. In a Text Label, use the Carriage Return key (`chr$(13)`) to begin the next line.

Center Property

Use this property to center the picture in an Image control. The default is False.

CharCase Property

This property may be used to specify the case of the characters to be typed into an Edit Box. Predefined constants are `ecNormal` (default, allowing both uppercase and lowercase), `ecUpperCase` and `ecLowerCase`.

Checked Property

Use this property to initially or programmatically check a checkbox or set an option button. The default is False.

ColumnHeaders Property

Use this property to establish column headers on a Multi-column List control. The default is False. When set to True, use the Columns property to create header text.

Columns Property

This property may be used to alter the number of columns (default is 5), change the row height (default is 17 pixels), specify headers and dividers. See also, the ColumnHeaders property.

Ctl3d Property

Use the Ctl3d property to set disable or enable a three-dimensional look on a control. The default is True. See also, the notes for the BackColor property. Also see, ParentCtl3d Property.

DataSource Property

This property may be used to alter the data source for an Edit Box or Text Label control. The datasource must be a valid field name in the screen associated with the current form.

Default Property

Use the Default property to give focus to a Command Button control when the form is first displayed. The default is False. If True and the user hits the Enter key, the button will be clicked.

Down Property

This property is used to initially at design time or programmatically at runtime press down a Speed Button control. The default is False. When used in conjunction with the GroupIndex property and when multiple Speed Buttons have the same group index value, setting this property to True will raise another Speed Button in the same group, like car radio buttons.

See also Predefined Constants.

EditMask Property

Use this property to change the input edit mask for the Edit Box control. See Edit Mask dialog.

Enabled Property

This property is used to enable or disable a control. The default is True (enabled).

FillFormWhenMaximized Property

The FillFormWhenMaximized property is a True/False value with False being the default. If the property is set to True and the form's WindowState property is set to wsMaximized all, controls placed on the form will be proportionally spaced to fill the form's client area. Only controls that are on the base form are moved. For example, controls within container (parent) controls, like a panel, are not moved within their container. Also, controls aligned to any side are not moved.

This feature is designed for use where applications will be run on screens of various resolutions. It is recommended that forms be designed using a width and height of the lowest resolution screen on which the application will be run.

Flat Property

Use this property to give a speed button a flat instead of raised appearance. The default is False. If this property is set to True, the outline of the button will appear upon mouse flyover.

FlatColor Property

This property may be used to change the color of the frame on a Date/Time Label control when the FrameStyle is set to fsFlat. The default is Black.

Font Property

Use the Font property to change the font on a control's caption or the font of a parent control (see Parent Controls). See also, ParentFont Property.

ForeColor Property

Use this property to set the foreground (text) color of a control. The default for most controls is CP_BtnFace. The default for edit boxes is CP_Window.

For forms, there is a rule regarding the form's Ctl3d and BackColor properties. If the BackColor is CP_BtnFace, when you toggle Ctl3d from True to False, the BackColor will toggle from CP_BtnFace to CP_Window. If the BackColor is something else, the current color is not change when Ctl3d is toggled.

See also Prdefined Constants.

Format Property

This property is used to change to format of the date and time on the Date/Time Label control. See the Date Time Format Editor.

FrameStyle Property

Use this property to change the frame style on a Date/Time Label control. The available styles are fsNone, fsFlat, fsGrove, fsBump, fsLowered, fsButtonDown, fsRaised, fsButtonUp, fsStatus (default) and fsPopup. If the fsFlat value is selected, the color of the frame may be changed with the FlatColor property.

GroupIndex Property

This property is used to group Speed Buttons on a parent control. When used in conjunction with the Down property and when multiple Speed Buttons have the same group index value, setting this property to True will raise another Speed Button in the same group, car radio buttons.

Height Property

Use this property to change the height of the control. It is used in conjunction with the Top property. The value is specified in pixels. For design purposes and ease, all controls have sizing handles and may be sized/aligned with other controls by right clicking on a group of selected controls (see "Aligning and Sizing Controls" on the [eQuate Form Designer Toolbar](#) help page).

Help Property

This property may be used to assign help for a control from help text stored in an eQuate application database or from a HTML windows help file (.chm).

Hint Property

Use this property to add a hint to a control. At runtime, when the user holds the mouse cursor over the control, the hint will appear briefly. See ShowHint and ParentShowHint properties.

ItemIndex Property

The ItemIndex property is used to specify which button in a Button Group is to be on initially at runtime. The default is -1 (no button initially set). 0 is the first button, 1 the second, etc. The Items property must be set prior to setting this property.

Items Property

Use this property to name the buttons in a Button Group. Each line entered is a separate button name. Use the Return key between lines.

Left Property

This property may be used to change the horizontal starting position of the control. It is used in conjunction with the Width property. The value is specified in pixels. For design purposes and ease, all controls have sizing handles and may be sized/aligned with other controls by right clicking on a group of selected controls (see "Aligning and Sizing Controls" on the [eQuate Form Designer Toolbar](#) help page).

Lines Property

Use this property to enter or edit in a Memo control at design time.

MaximizedButton Property

This property is used to add or remove the standard Windows maximize button to a form. The default is True.

MaxLength Property

This property is used to change the maximum length (in characters) of an Edit Box or Memo control. The default is 0 (no maximum). For a Memo control, this represents the total number of characters in the control including carriage returns (chr\$(13)). See also, Memo control.

MinimizedButton Property

Use this property to add or remove the standard Windows minimize button to a form. The default is True.

MinSize Property

The MinSize property is used to specify the minimum size of the panes (in pixels) on either side of the Splitter control. The default is 30.

Set MinSize to provide a minimum size the splitter must leave when resizing its neighboring control. For example, if the Align property is alLeft or alRight, the splitter cannot resize the regions to its left or right any smaller than MinSize pixels. If the Align property is alTop or alBottom, the splitter cannot resize the regions above or below it any smaller than MinSize pixels.

Note: Always set MinSize to a value less than half the client width of its parent. When MinSize is half the client width of the splitter's parent, the splitter cannot move because to do so would be to resize one of the panes less than MinSize pixels.

MonoChromeButtons Property

Use this property to change the buttons on the Media Player control to appear in monochrome as opposed to color. The default is False.

Name Property

This property is used to assign a label to a control that can be referenced in an action script.

NumBitMaps Property

Use this property to specify the number of bitmaps that are to be used on a Command Button or Speed Button when the button is enabled or disabled. The default is 1. The maximum is 4. This property is used in conjunction with the Bitmap property.

Buttons can show different images depending on the state of the button: Up, Disabled, Clicked and Down. The Bitmap property can reference a bitmap that is divided into four images of equal size, side-by-side in a row. The first image will be shown when the button is up or has focus; the second if the button is disabled, the third when the button is pushed and the fourth if the button remains down.

If there is only one image in the bitmap, this image is used for all four states.

Note: The lower left pixel of the bitmap is reserved for the "transparent" color. Any pixel in the bitmap which matches that lower left pixel will be transparent.

NumericSort Property

Use this property in conjunction with the SetNumericProp subroutine when sorting multi-column list boxes.

The property is Boolean, specifying that the data in the SortColumn property is to be compared numerically. The default is False (alpha compare).

This property can be changed in the Form Designer or, dynamically, at runtime.

See also, SortColumn and SortDescending properties. For an example, see the SetNumericProp subroutine.

ParentColor Property

If this property is set to True, the control will use the same color property of the parent control. The default is False. Also see, Parent Controls.

ParentCtl3d Property

This property is used to specify if a control is to take on the three-dimensional look (Ctl3d property) of the parent control. The default is True. Also see, Parent Controls.

ParentFont Property

If this property is set to True, the control will use the same Font property of the parent control. The default is True. Also see, Parent Controls.

ParentShowHint Property

Use this property to specify if the control is to use the ShowHint property of the parent control. The default is True. Also see, Parent Controls.

PassWordChar Property

This property is used to specify a password character for an Edit Box control. If a password character is specified, the user will only see password characters as they type into the edit box. The default is no password character in which case all characters typed are visible.

PickList Property

Use this property to assign a list of values from which the user picks. The following controls have a PickList property: the Edit Box, List Box, Drop-down List Box, Multi-column List Box, Memo and Button Group.

For List Boxes, Drop-down List Boxes, Memos and Button Groups, the PickList values will be appended to any Items property values at runtime.

Picture Property

This property is used to place a bitmap into an Image control. The type of files that may be loaded are Windows or OS/2 bitmap (.bmp), icon (.ico), Windows metafile (.wmf) Windows enhanced metafile (.emf) or JPEG compliant files (.jpg and .jpeg).

ReadOnly Property

The ReadOnly property is used to determine whether the Memo control may have data entered at runtime or will be "read only." The default is False (read/write).

ScrollBars Property

Use this property to add or remove scrollbars to a Memo control. Predefined constants are ssNone (default), ssHorizontal, ssVertical and ssBoth.

Shape Property

This property determines the shape of a Bevel control. Predefined constants are bsBox (default), bsFrame, bsTopLine, bsBottomLine, bsLeftLine, bsRightLine and bsSpacer.

ShowAccelChar Property

Use this property to display or not display accelerator characters in a Text Label control. The default is True.

The accelerator character is an ampersand (&). If the ShowAccelChar property is set to true and an ampersand is entered to the left of a character, the character will appear underlined in the text label. For example, "S&le" would appear as "Sample".

Note: On Text Labels, the accelerator character has no effect, other than how it appears in the text.

ShowHint Property

The ShowHint property along with the Hint property determines if a brief pop-up is to be displayed when the user moves the mouse cursor over the control. The default is True. See also, ParentShowHint Property.

SortColumn Property

Use this property in conjunction with the SetNumericProp subroutine when sorting multi-column list boxes.

The property is an Integer in the range of 0 to columns -1. The default is 0 (the first column).

This property can be changed in the Form Designer or, dynamically, at runtime.

See also, NumericSort and SortDescending properties. For an example, see the SetNumericProp subroutine.

SortDescending Property

Use this property in conjunction with the SetNumericProp subroutine when sorting multi-column list boxes.

The property is Boolean, specifying that the sort order is descending. The default is False (ascending sort).

This property can be changed in the Form Designer or, dynamically, at runtime.

See also, NumericSort and SortColumn properties. For an example, see the SetNumericProp subroutine.

Sorted Property

Use this property to determine if the items in a List Box, Drop-down List Box or Multi-column List Box are to be sorted. The default is False.

Stretch Property

This property may be used to force a picture to fit the size of the Image control. The default is False.

Style Property

Use this property to specify the style of a Bevel control. Predefined constants are bsLowered (default) and bsRaised.

TabOrder Property

This property may be used to set the tab order of the controls when the user clicks the tab key. The tab order is like an index beginning with 1 and incremented by 1. It is used in conjunction with the TabStop property. An alternate and quick way to set tab order on multiple controls is to select Form Edits | Tab Order from the Edit menu on the eQuate Form Designer Toolbar or right-click anywhere on the form and select Tab Order.

TabStop Property

Use this property to specify whether a control is to be a tab stop when the user presses the tab key. The default is True.

Text Property

The Text property may be used to place text into an Edit Box control. The text is for documentation purposes only and will not be displayed at runtime.

Top Property

Use the Top property to change the vertical starting position of the control. It is used in conjunction with the Height property. The value is specified in pixels. For design purposes and ease, all controls have sizing handles and may be sized/aligned with other controls by right clicking on a group of selected controls (see "Aligning and Sizing Controls" on the [eQuate Form Designer Toolbar](#) help page).

Transparent Property

This property is used to determine if the background of a Text Label or Image control is see-through. The default is False.

TransparentColor Property

Use TransparentColor to determine how to draw the bitmap transparently. If TransparentMode is Fixed, pixels that match the TransparentColor are transparent. TransparentColor is ignored if TransparentMode is auto.

TransparentMode Property

When TransparentMode is set to tmAuto (the default), the TransparentColor property returns the color of the bottom-leftmost pixel of the bitmap image. When TransparentMode is set to tmFixed, the TransparentColor property refers to the color stored in the bitmap object.

URL Property

This property is used to set the URL link to be accessed by the browser. When this control is clicked at runtime, it automatically opens the browser to the URL specified. No user code is needed.

Visible Property

Use this property to make a control visible or hidden. The default is True (visible).

WantsReturns Property

This property determines how the return character is to be handled in a Memo control. If True, the application accepts return characters into the text and does not pass the return character to any other control. If false, return characters are not accepted by the Memo control and may be passed to another control such as a default button. The default is True.

Width Property

Use this property to change the width of the control. It is used in conjunction with the Left property. The value is specified in pixels. For design purposes and ease, all controls have sizing handles and may be sized/aligned with other controls by right clicking on a group of selected controls (see "Aligning and Sizing Controls" on the [eQuate Form Designer Toolbar](#) help page).

WindowState Property

This property is used to set the window state of the form. Predefined constants are wsNormal (default), wsMinimized and wsMaximized.

WinHelpFile Property

This property may be used to enter the name of an externally developed help file (.chm). Help files are normally placed in the eQuate database directory (set by the InitReg.exe program) and the eQuate application database directory (created by the eQuate Administration program). Alternately, they may be located in the eQuate installation directory. Note: Do not enter the .chm extension. See also, Help Selector.

WordWrap Property

Use this property to specify if words will wrap to the next line when a line is full on a Memo control. The default is True. If True, lines automatically wrap based on the width of the control.

Property Dialogs

Parent Controls

Most controls have "Parent" properties (ParentCtl3D, ParentFont and/or ParentShowHint). If a parent property is set to True, then the control takes on the property of the parent control. Currently, there are only two types of parent controls other than the form itself: the group box and the panel control. For example, if a button on the form has the ParentFont property set to True, the button caption will use the same font assigned to the form. Note: Changing the Font property on the button will automatically set ParentFont property to False.

A panel or a group box can have controls placed on it so that whenever you move the panel during the design phase, all the controls on the panel move with it. Likewise, if an eQuate action disables the panel at runtime, all the controls on the panel are also disabled since the panel is parent to all the controls placed on the panel.

Color Selection

This dialog is used to assign a color property of a control.

Standard Color

From the list box on the left, select the desired Windows color.

Custom...

To select a non-standard color, click this button to initiate the Color dialog palette.

Select Screen Data Field Source

This dialog shows the data fields, their location on the screen, field length and repeating count. The dialog is used to view and copy field definitions to the clipboard when opened from the eQuate Form Designer Tool Bar. It is also used to assign data fields to edit boxes and text labels.

Field Name

Select one or more fields from this list. When accessing this dialog from the text label Data Source property, only one field may be selected.

Copy Selected to Clipboard

This button allows you to copy field definitions to the Windows clipboard. The button only appears when the dialog is opened from the eQuate Form Designer Tool Bar.

Select

Click this button to select the highlighted field. This button only appears when assigning data field values to a text label.

Clear

Click this button to clear the property value. This button only appears when assigning data field values to a text label.

Select Repeating Field Index

This dialog is used to select the occurrence of a repeating field. The top of the dialog tells you how many times the field repeats.

Select Desired Repeat Index

From the drop-down list box, select the occurrence or repeat index to be used for the data source of the field.

Help Selector

With this dialog, you can supply help for a control from help text stored in an eQuate application database or from a HTML windows help file (.chm).

Current Selection

This box contains the name of the help text entry assigned to the control.

Clear

Click this button to clear the entry.

Use Windows Help

Check this box to assign a context id from the Windows help file assigned to the form.

WinHelp Context Id.

In this text box, enter the context id of the help topic contained in the Windows help file assigned to the form. Note: To assign the help file, set the WinHelpFile property for the form. In addition, the help file (.chm) must be in the same directory as the runtime form files.

Available Help

This list box contains the inventory of help text entries stored in the eQuate application database. To select from the list, use the mouse or arrow keys. The OK button will assign the help text to the control.

To create new help text, click the Add button on the Help Text tab of the eQuate Application Manager.

Finder

In this text box, type the beginning character of the help text name to quickly locate the help text entry.

Edit Mask

This dialog is used to apply the EditMask property to an edit box.

Input Edit Mask

The Input Edit Mask is the mask that is used to limit the data that can be put into a masked edit box. A mask restricts the characters the user can enter to valid characters and formats. If the user attempts to enter a character that is not valid, the edit box does not accept the character. Validation is performed on a character-by-character basis.

If no edit mask is specified, the end-user is not restricted, except by maximum length if specified.

If a Custom Edit Mask is selected, the Input Edit Mask text box may be used to specify a mask other than the standard field edit masks supplied with eQuate. The Input Edit Mask is a case-sensitive text box used to specify the type of input that will be allowed, and the position in the field where each character will appear.

A mask consists of three fields with semicolons (;) separating the fields. The first part of the mask is the mask itself. The second part is the character that determines whether the literal characters of a mask are saved as part of the data. The third part of the mask is the character used to represent a blank in the mask.

Part 1:

The first part of the Edit Mask can contain any of the following characters:

<u>Character</u>	<u>Meaning in Mask</u>
!	If an exclamation (!) character appears in the mask, leading blanks do not appear in the data. If an exclamation character is not present, trailing blanks do not appear in the data.
>	If a greater than (>) character appears in the mask, all characters that follow are in uppercase until the end of the mask or until a greater than character is encountered.
<	If a less than (<) character appears in the mask, all characters that follow are in lowercase until the end of the mask or until a less than character is encountered.
<>	If these two characters appear together in a mask, no case checking is done and the data is formatted with the case the user uses to enter the data.
\	The character that follows a back slash (\) character is a literal character. Use this character when you want to allow any of the mask special characters as a literal in the data.
L	The "L" character requires an alphabetic character only in this position. For the US, this is A-Z, a-z.
l	The "l" character permits only an alphabetic character in this position, but does not require it.
A	The "A" character requires an alphanumeric character only in this position. For the US, this is A-Z, a-z, and 0-9.
a	The "a" character permits an alphanumeric character in this position, but does not require it.
C	The "C" character requires a character in this position.
c	The "c" character permits a character in this position, but does not require it.
0	The zero (0) character requires a numeric character only in this position.
9	The nine (9) character permits a numeric character in this position, but does not require it.

<u>Character</u>	<u>Meaning in Mask</u>
#	The pound (#) character permits a numeric character or a plus or minus sign in this position, but does not require it.
:	The colon (:) character is used to separate hours, minutes, and seconds in times. If the character that separates hours, minutes, and seconds is different in the International settings of the Control Panel utility on your computer system, that character is used instead of the colon.
/	The slash (/) character is used to separate months, days, and years in dates. If the character that separates months, days, and years is different in the International settings of the Windows Control Panel utility on your computer system, that character is used instead of the slash.

Part 2:

In the second part of the edit mask, the "0" character means that the mask is not saved as part of the data. The "1" character means that the mask is saved as part of the data. For example, a telephone number could have parentheses around the area code as part of the mask. If the second part of the edit mask is "0", the parentheses do not become part of the data, making the size of the field slightly smaller.

Part 3:

In the third part of the Edit Mask, the underscore (_) character may be used to automatically insert underscores in the edit box for positions that are not yet filled. You may change this character to any desired fill character or a space.

Examples:

<u>Edit Mask</u>	<u>Display</u>	<u>Internal</u>
\(999\)999\ -9999;0;	(770)635-6363	7706356363
\(999\)999\ -9999;1;	(770)635-6363	(770)635-6363
999-999;0;_	123-4__	1234

Pre-defined Standard Edit Mask

This list box contains pre-defined edit masks from which you may select one. If you wish a customized edit mask, type directly into the Input Edit mask text box.

Character for Blanks

In this text box, enter the character that will appear in the edit box in place of a blank value.

Save Literal Characters

Check this box to save mask characters as part of the data. See, "Part 2," above.

Test Input

Use this text box to test type input.

Pick List Selector

With this dialog, you can supply values for an edit box or list control from pick lists stored in an eQuate application database.

Current Selection

This box contains the name of the pick list assigned to the control.

Clear

Click this button to clear the entry.

Available Pick Lists

This box contains the inventory of pick list entries stored in the eQuate application database. To select from the list, use the mouse or arrow keys. The OK button will assign the pick list to the control.

To create new pick list, click the Add button on the Pick Lists tab of the eQuate Application Manager.

Finder

In this text box, type the beginning character of the pick list name to quickly locate the pick list entry.

Multi-Column List Setup

This dialog is used to specify the number of columns, row height, column headers, column alignment and vertical and horizontal dividers of a multi-column list box.

Number of Columns

Use this spin box to set the number of columns in the list.

Row height

Use this spin box to specify the height of each row in the list. The value specified is in pixels.

Column Header Options

The options in this group are used to set column headers if any.

Show Column Headers

Check this box if column headers are to be shown at the top of the list.

Use Preset Column Headers

Select this option if you set the column headers with the Column Size and Header Data control below.

Use 1st Data Line from Host as Column Headers

Select this option if the first line of data from the host is to be used as the column header. This option is primarily intended to be used in eQuate Command Mode where the host program would establish the header with the first LSTDTA command. Note: The Show Column Headers option must be set.

Dividers

The options in this group determine if dividers should be shown between rows and columns.

Horizontal Dividers

Set this option to show dividers between rows.

Vertical Dividers

Set this option to show dividers between columns.

Column Size and Header Data

This control allows you to label column headers and size columns.

To label a column, click the button at the top of a column. A text cursor will appear in the row below the button. Type the desired header name. Note: The Show Column Headers and Use Preset Column Headers options must also be set.

To size a column, place the mouse pointer on a column divider. The mouse pointer will change to a splitter (left/right-sizing arrow). Hold the left mouse button down and drag the splitter to the left to reduce the size of the column or to the right to increase the size of the column.

Column n Display Alignment

To justify a column heading, select the column with the mouse and click the Left, Right or Center button.

Date Time Format Editor

This dialog is used to establish the format of the date/time stamp shown in the DateTimeLabel control.

Date Time Format

This table shows the characters you can use to create user-defined date/time formats:

<u>Character</u>	<u>Meaning</u>
c	Display the date as dddd and display the time as tt, in that order.
d	Display the day as a number without a leading zero (1-31).
dd	Display the day as a number with a leading zero (01-31).
ddd	Display the day as an abbreviation (Sun-Sat).
dddd	Display the day as a full name (Sunday-Saturday).
dddddd	Display the date as m/d/yy.
dddddd	Display the date as dddd, mmmm d, yyyy (e.g., Friday, October 27, 2000).
m	Display the month as a number without a leading zero (1-12). If m immediately follows h or hh, the minute rather than the month is displayed.

<u>Character</u>	<u>Meaning</u>
mm	Display the month as a number with a leading zero (01-12). If mm immediately follows h or hh, the minute rather than the month is displayed.
mmm	Display the month as an abbreviation (Jan-Dec).
mmmm	Display the month as a full month name (January-December).
y	Display the day of the year as a number (1-366).
yy	Display the day of the year as a two-digit number (00-99)
yyyy	Display the day of the year as a four-digit number (0000-9999).
h	Display the hour as a number without leading zeros (0-23).
hh	Display the hour as a number with leading zeros (00-23).
n	Display the minute as a number without leading zeros (0-59).
nn	Display the minute as a number with leading zeros (00-59).
s	Display the second as a number without leading zeros (0-59).
ss	Display the second as a number with leading zeros (00-59).
t	Display the time as h:n AM/PM.
tt	Display the time as h:n:s AM/PM.
AM/PM	Use the 12-hour clock and display an uppercase AM/PM.
am/pm	Use the 12-hour clock display a lowercase am/pm.
A/P	Use the 12-hour clock display an uppercase A/P.
a/p	Use the 12-hour clock display a lowercase a/p

The following are examples of user-defined date and time formats:

<u>Format</u>	<u>Display</u>
m/d/yy	2/26/65
d-mmmm-yy	26-February-65
d-mmmm	26-February
mmmm-yy	February-65
hh:mm AM/PM	06:45 PM
h:mm:ss a/p	6:45:15 p
h:mm:ss	18:45:15
m/d/yy h:mm	2/26/65 18:45

Predefined Formats

Select from this list if you want to use a predefined date/time format. See Date Time Format, above, for a description of the various predefined formats.

Global Controls

eQuate Menu Designer

This dialog is used to add user menu items to the form.

Once a menu has been updated with the eQuate Menu Designer, the action for the menu item may be updated by selecting the menu item from the eQuate form displayed in the eQuate Form Designer. See eQuate Form Designer Toolbar.

Menu Item Caption

This entry specifies the menu item's caption. Captions may include accelerator keys by typing an ampersand character (&) in front of the desired letter within the caption text.

Menu separators may be indicated by entering a hyphen (-) in the caption.

Blank menu items are not permitted.

Name

In this text box, enter the name of the menu or menu selection. This name can be referenced in an eQuate action.

Short Cut

From the drop-down list box, select a short cut key for the menu selection. The shot cut key is optional.

Insert Item

Insert a blank menu item before the currently selected menu item. All menu items below and including the current menu item will be pushed down one position.

Indent Level

Use the arrow controls in this group to raise or lower the level of the selected menu or menu item in the Menu Items list. A menu item without a preceding asterisk represents a menu on the form's menu bar. Clicking the right arrow lowers the item level as indicated by an added asterisk (*) preceding the item. Clicking the left arrow raises the item level. Using the left and right arrow keys on the keyboard will also raise or lower the item level.

An item of a higher level with items below it at the next lower level becomes a cascading menu. For example, clicking an item called "Updates" (denoted by a single asterisk preceding it) might reveal a cascading menu containing "Add", "Replace" and "Delete" (each preceded by double asterisks).

Move

Use the arrow controls in this group to position the menu selection up or down in the list. You may also move an item with the keyboard by holding down the shift key and pressing the up and down arrow keys.

Checked

Check this control to preset the menu item to its selected state. Note: Only menu items may be checked – menus may not.

Visible

Check this box to make the menu or menu selection initially visible.

Enabled

Check this box to enable the menu or menu item initially.

Preview

Click this button to open a small dialog that reveals the designed menu bar. Test the menu and menu selection by clicking the menu.

Menu Items

This list box contains the menu and menu items defined. To work with an existing item, select it with the mouse or up and down arrows. Use the Indent and Move controls to rearrange the menus and menu items.

Action Key Assignments

This dialog is used to assign keyboard keys or key combinations to actions.

NOTE: F2 defaults to Show Pick List (if one is present), but it may be overridden by assigning a user action to it. Also, F1 defaults to application help, but may also be overridden.

Available Action Keys

Use the mouse or arrow keys on the keyboard to highlight a key and press the *right arrow button* to select it for mapping to an action by placing it in the Assigned Action Keys list. Alternately, you may double-click on a key to select it.

Right arrow button

Click this button to add the selected key in the Available Action Keys list to the Assigned Action Keys list.

Left arrow button

Click this button to remove the selected button from the Assigned Action Keys list.

Left double arrow button

Click this button to clear the Assigned Action Key list.

Assigned Action Keys

Highlight a key in this list and click the Action button or *left arrow button*.

Action

Use this button to give focus to the eQuate Action Script Editor allowing you to apply an action to the selected key in the Assigned Action Keys list.

Data Fields

eQuate Form Data Fields

This window is used to make changes to the way a data field is received from the host on the terminal screen. There are also settings used to override host screen, field properties for display on the eQuate Form window.

Data Field List

This group shows all the fields currently on the form. To view or change a field's attributes, select the field name with the mouse and view/change the attributes in the Data Field Attributes group (see below).

Field Name

Field Name is the name of the field as specified in the host program, or as specified by the eQuate Developer with the Add Field function.

ECM Data Field Sequence

Use the controls in this group to change the order of the data fields as they will sent by the host application using the eQuate Control Mode data record command (see DTAREC). A control, "Gen COBOL Defs," is also available in this group to generate COBOL data definitions that may be used in the host program for formatting ECM records.

List Sort Order

Use the controls in this group to change the order of the fields listed in the multi-column list box shown on the dialog.

Add Field

Click this button to add a field to the form (see Field Name, above).

Delete Field

After selecting a field from the list with the mouse, click this button to delete the field.

Name

Field Name is the name of the field as specified in the host program, or as specified by the eQuate Developer with the Add Field function. **Note: Fields added to a eQuate Form will require changes to the host.**

Location

These text boxes contain the Row, Column and Length of the field on the host program screen. Normally, they are only used to make minor adjustments when the application's screen definition is changed.

Warning: If changes are made to the size and location, and the field name is from an existing host application program, corresponding changes must also be made to the host program. Otherwise, the host program and eQuate will not be compatible.

In Command mode, only the Length may be changed on the dialog since only the width is specified in the host program. Additionally, the Data Field List shows the relative position of the field (used internally by eQuate) in the Command Mode record returned from the host program.

Row

This box contains the screen row where the field is located (only applied to Screen mode).

Column

This box contains the starting screen column where the field is located (only applied to Screen mode).

Length

This box contains the length of the field on the screen.

Repeating Field

This group contains controls dealing with fields that occur on a screen more than once.

Count

This box contains the count of the number of times that the selected field appears on the screen.

Spacing

This box contains the spacing between repeating fields. Spacing is the space between repetitions in either rows or columns. For vertical repetitions, a spacing of two (2) means every other line. For horizontal repetitions, a spacing of five (5) means fields start every fifth column.

Direction

The Direction options are used to specify whether the repeating field appears vertically down the screen or horizontally across the screen. The Down option means the field repeats from top to bottom down the screen. Right means the field appears from left to right across the screen.

Data Type

The controls in this group determine what the user can place into the selected field. The options here can completely override any properties defined for the application's terminal screen.

Any

This option allows any character to be entered into the field.

Alpha Only

This option allows only alphabetic characters (A-Z and a-z) to be entered.

Numeric Only

If this option is set, only numeric characters may be entered.

Signed

If this box is checked, numeric entries may contain a negative sign.

Blank When Zero

In Command Mode applications, fields containing zero are sent to eQuate as a string of zeros. The Blank When Zero check box tells eQuate to either display blanks when the field is zero or display the value as zeros. For Screen Mode applications, this check box has no effect because DPS or the host program handles it on the host.

Dec.

For numeric fields, this box specified the number of decimal positions.

Other Attributes

This group contains miscellaneous options.

Tab Stop

When this box is checked, a tab stop is located at the beginning of the field.

Output Only

This check box is used to specify that the field is "Output Only". If checked, the field may not be used as an input field.

Video Off

This check box, when checked, is used to hide the field when the form is displayed.

Justified (in Screen)

The controls in this group dictate the alignment of fields

Left

Select this option to left justify the data entered into the field.

Right

Select this option to right justify the data entered into the field.

Center

Select this option to center the data entered into the field.

Apply

Click this button to apply changes made on the dialog without closing the dialog.

Undo

Click this button to revert any changes made on the dialog.

Form Import

eQuate Form Definition Import

This dialog is used to import forms definitions created by the eQuate Capture program, forms stored in eQuate application databases or form definitions from DPS 2200.

Import Source

From this group, select the source of the import.

Clipboard (Capture)

Select this option to import the form definition from the Windows clipboard. This option can be used when preceded by a screen capture using the eQuate Capture program.

File (Form def. text)

Select this option to import from a text file containing the form definition. This file can be created by the eQuate Capture or be the result of downloading a definition from DPS 2200 utilizing the Form Language Manipulation Utility (@FLMU,F).

Binary Form File

Select this option to import the definition from another eQuate application database's binary form file (.bfm).

Import File Name

For the two file options, use the Browse button to locate and select the file to import.

Convert Background Text to Labels

When importing from the clipboard or a text file, check this option to convert all background text, to text labels. Using this option is a quick way to convert a screen to a form.

Import

Click this button to begin the import process.

Form Import View and Adjust

This dialog is used to review fields and labels imported. Minor adjustments may also be made via this dialog.

Import Data Fields

This multi-column list box shows the position, alignment and attributes of data fields on the form.

Edit

To change a field name, select the field from the list box and click the Edit button.

Delete

To delete a field, select the field from the list box and click the Delete button.

Generated Text Labels

This multi-column list box shows the position and text of all text labels on the form.

Edit

To change the name or location of a text label, select the row in the list box and click the Edit button.

Delete

To a text label, select the row in the list box and click the Delete button.

Preview Layout

Click this button to display a preview window based on the imported or adjusted settings.

Edit Data Field Name

This window allows you to edit a field name.

Enter a Unique Control Name

In this text box, enter a unique field name. If the field name is changed, the DataSource property for the field edit box must also be changed.

Form Merge/Update

This dialog appears if eQuate detects a duplicate form name during import. The controls on the dialog can be used to determine which data fields are matched and merged between the two forms.

Merge/Update Options

On this tab, select a match option. The first three options are automatic and instruct eQuate how to match. The fourth option allows you to manually pick the matches you want. Select an option on this tab and click the Field View/Edit tab to see and adjust the result.

Names Only

Select this option to match on field names only between the duplicate forms.

Row/Column Locations Only

Select this option to match any field whose row and column position are the same as that being imported.

Names and Row/Column Locations

Select this option to require the match to occur only when there is an exact match on field name, row and column.

Manually Match

Select this option when you wish to manually make the matches. Proceed to the Field View/Edit tab.

Field View/Edit

This tab allows you to view and edit the results of eQuate's automatic matching or an unmatched (UM) result before committing to the completion of the import when the OK button is clicked.

Import Fields

The fields shown in this list box are the fields being imported.

Resulting Field List

This list box contains the result of the match when one of the automatic options was chosen on the Merge/Update Options tab.

When the manual matching option is selected, an "<NM>" will appear in the "Mch#" column and the field attributes and location will appear as it does before the move. In this instance, use the mouse and Manual Match Function to selectively match fields.

Add →

After selecting a field from the Imported Fields list, click this button to add it to the result.

Match =

After selecting a field from each list, click this button to replace the item selected in the result list with the item selected in the import list.

← Remove

After selecting a field name in the Resulting Field List, click this button to delete the item from the result. Note: Removing a field from a form definition and resulting form normally requires a comparable change to the host application when using eQuate Screen Mode.

Restore

Click this button to revert any changes made on this tab.

Imported Label Edit

This dialog allows you to change the text of a label and its location.

Text

In this box, change the label.

Row

In this box, alter the row where the label is located.

Col

In this box, alter the beginning column of the label.

Host Error Identification

Host Error Detection String Edit

The Host Error Detection String Edit window is used to maintain a list of strings that may be received from the host indicating an application error. Along with the error detection string, an Error Action is stored which tells eQuate what to do in the event the error is detected. There are only four actions available: "Fatal - Close Application", "Warning - Display and continue", "No Action - Discard message and continue" and "Run HOSTERROR Sub in Action Script".. Some common host error strings are "SESSION PATH CLOSED", "INPUT IGNORED IN CONTROL MODE", "TIP ERROR", etc.

Error Id. String

Error Id. String is the text string received from the host.

Error Action

Specify the error action to be performed by the eQuate Session Manager if the Error Id. String is encountered.

Standard Messages

Standard Messages

The Standard Message Edit window is used to enter Standard Messages that may be maintained in the eQuate application database. The Message Number associated with each message is returned by the host program using Command Mode Protocol. The eQuate Session Manager will substitute the Standard Message Text stored in the eQuate application database.

Message Number

Message Number must be a unique number within the eQuate application database.

Message Text

Enter the standard message text to be displayed in the eQuate Error List window when the eQuate Session Manager receives the Message Number. Substitution points in standard error message text are indicated by the sequence "\\n", where "n" indicates which word from the input error statement is to be inserted. There may be from 1 to 9 substitutions in a single standard error message.

Example:

The text of Standard Messages 12 and 119 might look like the following:

Standard message 12:	"Field \\1 is \\2."
Standard message 119:	"Value entered in \\1 is out of range."

The Command Mode record received from the host program might contain the following ERRLLST command:

```
EQ$<TAB>ERRLLST<CR>{12:"ACCOUNT" "Out of Range"}{This is an independent error statement  
and will appear as is}  
{119:"LINE 24"}<TAB>\\
```

In the above example, "ACCOUNT" will replace "\\1" and "Out of Range" will replace "\\2" in standard message number 12, and "LINE 24" will replace "\\1" in standard message 119.

Pick Lists

Pick List Edit

This dialog is used to edit and maintain a list of data values and their descriptions. The pick list can be linked to a control through the control's PickList property. When the control is displayed to the user, the pick list specified will automatically populate the control. Controls that have the PickList property are Edit Boxes, List Boxes and Drop-Down List Boxes. An action can be assigned to the control so that, when the user selects from the list, the action is executed using their selection.

Available Pick Lists

The first text box in this group contains the name of the Pick List. This name is the name that will be linked to the control through PickList property on the control.

Pick List Values and Descriptions

The items in this list box are the values and descriptions of the pick list. Only the value will be displayed to the user. To edit or remove an existing value or description, first select it from this list with the mouse, or tab and arrow keys.

Edit Value

In this box, either you edit a value selected from the Pick List Values and Descriptions list box or you enter a new value and click the Append button.

Edit Description

In this box, either you edit a description selected from the Pick List Values and Descriptions list box or you enter a new description and click the Append button.

Up

After selecting an item from the Pick List Values and Descriptions list box, click this button to move the item up in the list. Note: This control is only enabled if None is selected in the Sort Items By group.

Down

After selecting an item from the Pick List Values and Descriptions list box, click this button to move the item down in the list. Note: This control is only enabled if None is selected in the Sort Items By group.

Update

Click this button to change an entry after selecting and editing an item from the Pick List Values and Descriptions list box.

Append

To add an item to the list box, type the new value and description into the Edit Value and Edit Description text box, respectively, and click this button. Note: If you want the new item to be sorted with other existing items in the list, you will need to click the Re-sort button after appending the item.

Remove

Click this button to delete a selected item from the list.

Sort List By

Choose one of three options in this list to arrange or not arrange the order of the list. None leaves the order as entered with new items appended to the end of the list and altered items in their original position.

Re-sort

Click this button to re-sort the list after appending an additional item(s).

Uppercase Descriptions

Check this box to force all description entries to uppercase.

Uppercase Values.

Check this box to force all value entries to uppercase.

Save to File

Click this button to open a Save dialog and write the list to a file. This option is useful when porting pick lists to other eQuate Application Databases.

Load from File

Click this button to import a pick list from a saved file.

Print List

Click this button to print a list of all values currently in the Print List Values and Descriptions list box.

Help Text

eQuate Help Text Edit

This window provides a simple text edit box for editing help text for a data field. The edit area supports standard copy/cut/paste operations that allows help text to be written using other text editors and pasted into this window. Also supported is the standard select, drag and drop feature that allows you to easily move text around in the edit area.

When the help text is displayed to the user, it will be in an additional window with scroll bars for viewing the text. In addition, the window may be resized by the user in the conventional fashion for easier viewing. If the text is to be viewed in a limited number of words per line, as in the above example, hard carriage returns (Enter key) must be placed at the end of each line.

Help Text

The first text box in this group contains the name of the Help Text entry. This name is the name that will be linked to the control through Help property on the control.

The edit area supports standard copy/cut/paste operations (Ctrl+C/Ctrl+X/Ctrl+V) that allows help text to be written using other text editors and pasted into this window.

When the help text is displayed to the user, it will be in an additional window with scroll bars for viewing the text. In addition, the window may be resized by the user in the conventional fashion for easier viewing. If the text is to be viewed in a limited number of words per line, as in the above example, hard carriage returns (Enter key) must be placed at the end of each line.

File menu

Use the selections on this menu to list, store and load help text maintained in eQuate.

Load from Text File

Make this selection to search for and load help from a text file.

Save to Text File

Make this selection to save the help text to a file.

Print

This selection will print the help text.

Edit menu

Use the selections on this menu to edit help text. Note: A right mouse click in the text area will reveal a popup menu of these same selections.

Undo (Ctrl+Z)

This selection will undo the previous edit.

Cut (Ctrl+X)

This selection will cut the selected text from the list box to the Windows clipboard.

Copy (Ctrl+C)

Use this selection to copy the selected text to the Windows clipboard.

Paste (Ctrl+V)

Use this selection to paste the contents of the Windows clipboard to the list box beginning at the position of the text cursor.

Find (Ctrl+F)

Use this selection to find a string of text.

Find Again (F3)

Use this selection to find the next occurrence of a string.

Replace (Ctrl+R)

Use this selection to find and replace strings of text.

eQuate Action Script Editor

eQuate Action Script Editor

This window provides the means to develop and edit eQuate actions. The script editor contains a Multiple Document Interface (MDI) that allows more than one script to be edited at a time.

Following is a description of each eQuate Action Script Editor command. All the commands may be performed by making a menu selection. Toolbar buttons, shortcut keys and right mouse click actions are available for frequently used commands. A right mouse click anywhere in the test area will product a pop-up menu of commands.

The menu items, below, show an image of the toolbar button and the shortcut key combination (in parentheses) where applicable.

File menu

The File menu contains commands to maintain script files and setup printing.



New (Ctrl+N)

Use this command to create a new script file (.ACT).



Open (Ctrl+O)

Use this command to open an existing script file (.ACT) or user library (USER.LIB). Note: Script Functions and Subroutines may be placed in a user library and become global to any script that has need to call them. The user library must reside in the SCRIPTS directory under the eQuate installation directory.



Save (Ctrl+S)

Use this command to save the current script file (.ACT).



Save As...

Use this command to save the current script file (.ACT) to another file name or user library (USER.LIB).

Mass Compile Actions...

Use this command to compile selected scripts and save them in encrypted form (.ACX). This option is useful for sites that wish to secure the contents of script files from general viewing (e.g., user-id/password). Either the text form (.ACT) or the compiled form of the script may be made available to the user for execution.

Choosing this option will present a list of all action scripts available in the eQuate application database. You may compile all or selected scripts using the checkboxes and buttons on the displayed dialog form.

Close

Close the currently selected script.

Close All

Close all open scripts.



Print (Ctrl+P)

Use this command to print the entire script.

Printer Setup...

Allow margins to be set and allow printers and printer fonts to be selected for printing.

Clear Previous File List

Remove all file names from the list of previously accessed files.



Editor Properties

Use this command to edit the properties of the script editor: window font, highlight colors and tab stops.

**Exit**

Exit the Script Editor.

Previous File List

Select (open) from the list of previous accessed script files.

Edit menu

The Edit menu contains commands to manage selected text between the editor and the Windows clipboard.

**Undo (Ctrl+Z)**

Use this command to reverse the effects of the most recent change.

**Redo (Ctrl+Shift+Z)**

Use this command to reverse the effects of the most recent **Undo** command.

**Cut (Ctrl+X)**

Use this command to place the selected text on the clipboard and delete.

**Copy (Ctrl+C)**

Use this command to copy the selected text to the clipboard.

**Paste (Ctrl+V)**

Use this command to paste the contents of the clipboard to the current cursor position.

Delete (Ctrl+D)

Use this command to delete the selected text without copying to the clipboard.

Word Wrap (Ctrl+W)

Use this command as a toggle. By default, long lines may only viewed/edited by first bringing the excess text into view with the horizontal scroll bar or by using the cursor keys (arrows). When Word Wrap is set, long lines wrap to the next line and are viewable within the confines (width) of the window.

**Check Script (F4)**

Use this command to check the syntax of the entire script.

**Compile Script**

Use this command to compile the script and save it in encrypted form (.ACX). This option is useful for sites that wish to secure the contents of script files from general viewing (e.g., user-id/password). Either the text form (.ACT) or the compiled form of the script may be made available to the user for execution.

**Dialog Designer (F10)**

Use this command to initiate the Enable Dialog Designer. To edit an existing dialog, place the dialog on the Windows Clipboard before using this command.

Search menu

The Search menu contains commands to locate and change text within the script.

**Find... (Ctrl+F)**

Use this command to enable the **Find** dialog used to locate text strings.

**Find Again (F3)**

Use this command to find the next occurrence of the same string used on the previous find.

**Replace... (Ctrl+R)**

Use this command to enable the **Replace** dialog used to locate and replace text strings.

Go to Line (Ctrl+G)

Use this command to go to a specific line.

Bookmarks

The Bookmarks menu contains commands to mark lines and navigate within the script.

Set Bookmark 1 through 5 (Shift+F1 through Shift+F5)

Use one of these commands to mark a line at the current text cursor position. A book marked line will appear with a gray background.

Go to Bookmark 1 through 5 (Ctrl+F1 through Ctrl+F5)

Use one of these commands to go to a line previously book marked by one of the five corresponding **Set** Bookmark commands.

Options menu

The Option menu contains commands to specify color, font and tab stop preferences.

Show Tool Bar

Use this command to toggle the display of the toolbar.

Show Highlight

Use this command to toggle the display of the script syntax. This command is affected by the settings of the **Editor Colors** command, below.

View Permanent Declarations

Use this command to show the permanent declarations plus any USER.LIB code that is automatically included with all Action Scripts. You browse permanent declarations in read-only mode; however, you may copy code from the browse window to the Windows clipboard using the Ctrl+C key.

Window

The **Window** menu contains commands to control the arrangement of and navigation within the script windows.

Tile Horizontal

Use this command to arrange the script windows horizontally, one above the other.

Tile Vertical

Use this command to arrange the script windows vertically, one beside the other.

Cascade

Use this command to overlap each window in a cascading fashion.

Arrange Icons

Use this command to arrange minimized windows icons.

Next Window

Use this command to make the next window the currently selected window.

Previous Window

Use this command to make the previous window the currently selected.

Help

The **Help** menu contains commands to display on-line help and information about the product.

Contents

Use this command to display the contents of the eQuate Action on-line help.

This Window

Use this command to receive on-line help for this window.

About...

Use this command to display copyright and product version information.

Editor Properties

This dialog is used to change the properties or appearance of a script window.

Edit Window Font

The controls in this group affect the font typeface, size and intensity used to display the script.

Font Name

From this drop-down list box, choose from the list of non-proportional, fixed fonts installed on your PC.

Size

With this spin wheel, increase or decrease the font size.

Bold

Check this box to increase the font intensity.

Tab Size

With this spin wheel, increase or decrease the number characters between tab characters.

Highlight Colors

Use this group to assign colors to different parts of the script text.

Set Text Color

To change color, select the type of text (Normal text, Strings, etc.) and select from the Set Text Color drop-down list box to change the foreground.

Set Background Color

To change the background color, select from the Set Background Color drop-down list box.

OK

Click this button to accept the changes made and exit the dialog.

Cancel

Click this button to discard the changes made and exit the dialog.

Help

Click this button to receive on-line help for this dialog.

Language Elements

Comments

Comments are non-executed lines of code, which are included for the benefit of the programmer. Comments can be included virtually anywhere in a script. Any text following an apostrophe or the word Rem is ignored. Rem, all other keywords, and most names in an action are not case sensitive:

```
' This whole line is a comment
rem This whole line is a comment
REM This whole line is a comment
Rem This whole line is a comment
```

Comments can also be included on the same line as executed code:

```
MsgBox Msg ' Display message.
```

Everything after the apostrophe is a comment.

Statements

There is no statement terminator. More than one statement can be put on a line if they are separated by a colon:

```
X.AddPoint( 25, 100) : X.AddPoint( 0, 75)
```

This is equivalent to:

```
X.AddPoint( 25, 100)
X.AddPoint( 0, 75)
```

Line Continuation Character

The underscore (_) is the line continuation character. There must be a space before and after the line continuation character.

```
X.AddPoint _
( 25, 100)
```

Numbers

Three representations of numbers are supported: Decimal, Octal and Hexadecimal. Most of the numbers used in this manual are decimal or base 10 numbers; however, if you need to use octal (base 8) or hexadecimal (base 16) numbers, simply prefix the number with &O or &H, respectively.

Variable and Constant Names

Variable and constant names must begin with a letter. They may be comprised of uppercase letters (A through Z), lowercase letters (a through z), underscore (_) characters, and numeric digits (0 through 9). Variable and constant names can be no longer than 40 characters and cannot be reserved words. For a table of reserved words, see Language Reference. One exception to this rule is that object member names and property names may be reserved words.

Variables

Variable Types

As is the case with Visual Basic, when a variable is introduced, it is not necessary to declare it first (see Option Explicit for an exception to this rule).

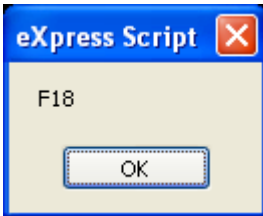
Note: Use the browse buttons (>>) (<<) at the top of this window to view the variable types.

Variant

When a variable is used but not declared, then it is implicitly declared as a variant data type. Variants can also be declared explicitly using As Variant as in Dim x As Variant. The variant data type is capable of storing numbers, strings, dates and times. When using a variant you do not have to explicitly convert a variable from one data type to another. This data type conversion is handled automatically.

Example:

```
Sub Main
  Dim x          'variant variable
  x = 10
  x = x + 8
  x = "F" & x
  print x        'prints F18
End Sub
```



A variant variable can readily change its type and its internal representation can be determined by using the function VarType. VarType returns a value that corresponds to the explicit data types. See VarType for return values.

When storing numbers in variant variables, the data type used is always the most compact type possible. For example, if you first assign a small number to the variant it will be stored as an integer. If you then assign your variant to a number with a fractional component, it will then be stored as a double.

For doing numeric operations on a variant variable, it is sometimes necessary to determine if the value stored is a valid numeric, thus avoiding an error. This can be done with the IsNumeric function.

Variants and Concatenation

If a string and a number are concatenated, the result is a string. To be sure your concatenation works regardless of the data type involved, use the ampersand (&) operator instead of the plus (+) operator. The ampersand will not perform arithmetic on your numeric values; it will simply concatenate them as if they were strings.

Example:

```
X = String_1 & Integer_2 ' Concatenate a string and a number - OK
```

Instead of:

```
X = String_1 + Integer_2 ' Error
```

The IsEmpty function can be used to find out if a variant variable has been previously assigned.

Other Data Types

The six data types available are shown below with their declaration character suffixes:

<u>Data type</u>	<u>Suffix</u>	<u>Type Declaration</u>	<u>Size</u>	<u>Range</u>
String	\$	Dim StrVar As String	String of characters	0 to 65,500 characters
Integer	%	Dim IntVar As Integer	2-byte integer	-32,768 to 32,767

<u>Data type</u>	<u>Suffix</u>	<u>Type Declaration</u>	<u>Size</u>	<u>Range</u>
Long	&	Dim LongVar As Long	4-byte integer	-2,147,483,648 to 2,147,483,647
Single	!	Dim SingVar As Single	4-byte floating-point number	-3.402823E38 to -1.401298E-45 (negative values) 1.401298E-45 to 3.402823E38 (positive values)
Double	#	Dim DbtVar As Double	8-byte floating-point number	-1.79769313486232D308 to -4.94065645841247D-324 (negative values) 4.94065645841247D-324 to 1.79769313486232D308 (positive values)
Variant		Dim X As Variant	Date/time, floating-point number or string	Date values: January 1, 0000 through December 31, 9999; numeric values: same range as Double; string values: same range as String
Currency		(Currency datatype is not supported)		

Scope of Variables

Variables can be local or global. Variables declared within a Sub or Function are local. Variables declared outside any Sub or Function are global.

Example:

```
Option Explicit
Global const Count As Integer ' (Global)
Sub BTN_1()
    Dim x As Integer           ' (Local)
    Count = count + 1
    x = count
End Sub
```

Declaration of Variables

Variables are declared with the Dim statement. To declare a variable other than a variant, the variable must be followed by As or suffixed by a type declaration character such as a percent character (%) for Integer type.

```
Sub Main
    Dim X As Integer
    Dim Y As Double
    Dim Name$, Age%
    ' multiple declaration on one line
    Dim v
End Sub
```

Flow of Control

Control Structures

The scripting language has complete process control functionality. The control structures available are Do loops, While loops, For loops, Select Case, If Then and If Then Else. In addition, one branching statement is available: GoTo.

Note: Use the browse buttons (>>) (<<) to view each control structure.

The GoTo

The GoTo statement branches to the label specified on the Goto statement.

```
Goto label1
.
.

label1:
```

The program execution jumps to the part of the program that begins with the label, "Label1:".

The Do Loops

The [Do...Loop](#) allows you to execute a block of statements an indefinite number of times. The variations of the Do...Loop are Do While, Do Until, Do Loop While and Do Loop Until.

```
Do While condition
    statement(s)...
Loop
Do Until condition
    statement(s)...
Loop
Do
    statement(s)...
Loop While condition
Do
    statement(s)...
Loop Until condition
```

Do While and Do Until check the condition before entering the loop, thus the block of statements inside the loop are only executed when those conditions are met. Do Loop While and Do Loop Until check the condition after having executed the block of statements, thereby guaranteeing that the block of statements is executed at least once.

The While Loop

The While...Wend loop is similar to the Do While loop. The condition is checked before executing the block of statements comprising the loop.

```
While condition
    statement(s)...
Wend
```

The For ... Next Loop

The For...Next loop has a counter variable and repeats a block of statements a set number of times. The counter variable increases or decreases with each repetition through the loop. The counter default is one if the Step variation is not used.

```
For counter = beginvalue To endvalue [Step increment]
    statement(s)...
Next
```

The If and Select Statements

The If...Then block has a single line and multiple line syntax. The condition of an If statement can be a comparison or an expression, but it must evaluate to true or false.

```
If condition Then statement(s) ' single line syntax

If condition Then                ' multiple line syntax
```

```
        statement(s) ...  
    End If
```

The other variation of the If statement is the If...Then...Else statement. This statement should be used when there are different statement blocks to be executed depending on the condition. A variation of the If...Then...Else is the If...Then...ElseIf.

```
    If condition Then  
        statement(s) ...  
    ElseIf condition Then  
        statement(s) ...  
    Else  
        statement(s) ...  
    End If
```

The Select Case statement tests the same variable for many different values. This statement tends to be easier to read, understand and follow and should be used in place of a complicated If...Then...ElseIf statement.

```
    Select Case variable_to_test  
        Case 1  
            statement(s) ...  
        Case 2  
            statement(s) ...  
        Case 3  
            statement(s) ...  
        Case Else  
            statement(s) ...  
    End Select
```

Subroutines and Functions

Subroutine and Function Naming Conventions

Subroutine and function names may be comprised of uppercase letters (A through Z), lowercase letters (a through z), underscore (_) and numeric digits (0 through 9). The only limitation is that subroutine and function names must begin with a letter, be no longer than 40 characters and not be a reserved word. For a list of reserved words, see the list of reserved words in Language Reference.

Script developers can create their own functions or subroutines or make DLL calls. Subroutines are created with the syntax:

```
Sub subname
.
.
End Sub
```

Functions have a similar syntax:

```
Function funcname As type
.
.
End Function
```

DLL functions are declared via the **Declare** statement.

ByRef and ByVal

ByRef gives other subroutines and functions the permission to make changes to variables that are passed as parameters. The keyword ByVal denies this permission and the parameters cannot be reassigned outside their local procedure. ByRef is the default and does not need to be used explicitly. Because ByRef is the default, all variables passed to other functions or subroutines can be changed; the only exception is when using the ByVal keyword to protect the variable or using parentheses, which indicate the variable is ByVal.

If the arguments or parameters are passed with parentheses around them, you are passing them ByVal.

```
SubOne var1, var2, (var3)
```

The var3 parameter in this case is passed by value and cannot be changed by the subroutine, SubOne.

```
Function R( X As String, ByVal n As Integer)
```

In this example, the "R" function is receiving two parameters, "X" and "n". The second parameter, "n", is passed by value and the contents cannot be changed from within the "R" function.

In the following code samples, scalar variable types and user-defined types are passed by reference:

Scalar Variables

```
Sub Main
  Dim x(5) As Integer
  Dim i As Integer
  for i = 0 to 5
    x(i) = i
  next i
  Print i
  Joe (i), x ' Parenthesis around it turn it into
              ' an expression which passes by value
  print "should be 6: "; x(2), i
End Sub
Sub Joe( ByRef j As Integer, ByRef y() As Integer )
  print "Joe: "; j, y(2)
  j = 345
  for i = 0 to 5
    print "i: "; i; "y(i): "; y(i)
  next i
  y(2) = 3 * y(2)
End Sub
```

Passing User-Defined Types by Ref to DLL's functions

```
' OpenFile() Structure
Type OFSTRUCT
  cBytes As String * 1
  fFixedDisk As String * 1
  nErrCode As Integer
```

```

        reserved As String * 4
        szPathName As String * 128
End Type
' OpenFile() Flags
Global Const OF_READ = &H0
Global Const OF_WRITE = &H1
Global Const OF_READWRITE = &H2
Global Const OF_SHARE_COMPAT = &H0
Global Const OF_SHARE_EXCLUSIVE = &H10
Global Const OF_SHARE_DENY_WRITE = &H20
Global Const OF_SHARE_DENY_READ = &H30
Global Const OF_SHARE_DENY_NONE = &H40
Global Const OF_PARSE = &H100
Global Const OF_DELETE = &H200
Global Const OF_VERIFY = &H400
Global Const OF_CANCEL = &H800
Global Const OF_CREATE = &H1000
Global Const OF_PROMPT = &H2000
Global Const OF_EXIST = &H4000
Global Const OF_REOPEN = &H8000
Declare Function OpenFile Lib "Kernel" (ByVal _
    lpFileName As String, lpReOpenBuff As OFSTRUCT, _
    ByVal wStyle As Integer) As Integer
Sub Main
    Dim ofs As OFSTRUCT
    ' Print OF_READWRITE
    ofs.szPathName = "c:\enable\openfile.bas"
    print ofs.szPathName
    ofs.nErrCode = 5
    print ofs.nErrCode
    OpenFile "t.bas", ofs
    print ofs.szPathName
    print ofs.nErrCode
End Sub

```

Calling Procedures in DLLs

DLLs or Dynamic-Link Libraries are used extensively by engineers to execute functions and subroutines located within the libraries. There are two ways scripts can be extended: 1) calling functions and subroutines in DLLs and 2) calling functions and subroutines located in the calling application. The mechanisms used for calling procedures in either place are similar (see the Declare Statement for more details).

To declare a DLL procedure or a procedure located in your calling application, place a declare statement in the global declaration section of the script. All declarations are global to the run and accessible by all subroutines and functions.

If the procedure does not return a value, declare it as a subroutine. If the procedure does have a return value, declare it as a function.

```

Declare Function GetPrivateProfileString Lib "Kernel32" _
    (ByVal lpApplicationName As String, _
    ByVal lpKeyName As String, _
    ByVal lpDefault As String, _
    ByVal lpReturnedString As String, _
    ByVal nSize As Integer, _
    ByVal lpFileName As String) As Integer
Declare Sub InvertRect Lib "User" _
    (ByVal hDC AS Integer, aRect As Rectangle)

```

Notice the line extension character is the underscore (_). If a piece of code is too long to fit on one line, a line extension character can be used when needed.

Once a procedure is declared, you can call it just as you would another function.

It is important to note that Enable cannot verify that you are passing correct values to a DLL procedure. If you pass incorrect values, the procedure may fail.

Files

File Input/Output

Enable supports full sequential and binary file I/O.

Functions and Statements that apply to file access:

Dir, EOF, FileCopy, FileLen, Seek, Open, Close, Input, Line Input, Print and Write

'File I/O Examples:

```
Sub Main
    Open "TESTFILE" For Input As #1 ' Open file.
    Do While Not EOF(1) ' Loop until end of file.
        Line Input #1, TextLine ' Read line into variable.
        Print TextLine ' Print to Debug window.
    Loop
    Close #1 ' Close file.

End Sub

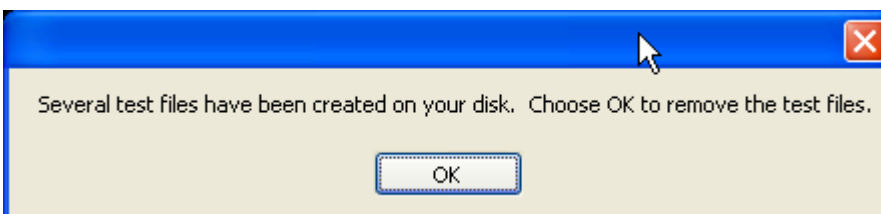
Sub test

Open "MYFILE" For Input As #1 ' Open file for input.
Do While Not EOF(1) ' Check for end of file.
    Line Input #1, InputData ' Read line of data.
    MsgBox InputData
Loop
Close #1 ' Close file.

End Sub

Sub FileIO_Example()
    Dim Msg ' Declare variable.
    Call Make3Files() ' Create data files.
    Msg = "Several test files have been created on your disk. "
    Msg = Msg & "Choose OK to remove the test files."
    MsgBox Msg
    For I = 1 To 3
        Kill "TEST" & I ' Remove data files from disk.
    Next I
End Sub

Sub Make3Files ()
    Dim I, FNum, FName ' Declare variables.
    For I = 1 To 3
        FNum = FreeFile ' Determine next file number.
        FName = "TEST" & FNum
        Open FName For Output As FNum ' Open file.
        Print #I, "This is test #" & I ' Write string to file.
        Print #I, "Here is another "; "line"; I
    Next I
    Close ' Close all files.
End Sub
```



Arrays

Arrays

Enable supports single-dimensioned and multidimensional arrays. By using arrays, you can refer to a series of variables by the same name each with a separate index. Arrays have upper and lower bounds. Enable allocates space for each index number in the array. Arrays should not be declared larger than necessary.

All the elements in an array have the same data type. Enable supports arrays of integers, singles, double and strings.

Ways to declare a fixed size array:

- Global array – use the Global or Dim statement outside the general declarations section of a module to declare the array;
- Local array – use the Dim statement inside a procedure or function.

Enable does not support dynamic arrays.

Single-dimensioned Arrays:

When declaring an array, the array name must be followed by the upper bound (boundary) in parentheses. The upper bound must be an integer.

```
Dim ArrayName (10) As Integer
Dim Sum (20) As Double
```

To create a global array, you simply use Global in place of Dim:

```
Global Counters (12) As Integer
Global Sums (26) As Double
```

The same declarations within a procedure use Static or Dim:

```
Static Counters (12) As Integer
Static Sums (26) As Double
```

The Counters declaration creates an array with 13 elements, with index numbers running from 0 to 12. The Sums declaration creates an array with 27 elements. To change the default lower bound to 1, place an Option Base statement in the declarations section of a module:

```
Option Base 1
```

Another way to specify lower bound is to provide it explicitly (as an integer, in the range -32,768 to 32,767) using the **To** key word:

```
Dim Counters (1 To 13) As Integer
Dim Sums (100 To 126) As String
```

In the preceding declarations, the index numbers of Counters run from 1 to 13, and the index numbers of Sums run from 100 to 126.

Note: Many other versions of Basic allow you to use an array without first declaring it. With Enable Basic, you must declare an array before using it.

Loops often provide an efficient way to manipulate arrays. For example, the following **For** loop initializes all elements in the array to a value of five (5):

```
Static Counters (1 To 20) As Integer
Dim I As Integer
For I = 1 To 20
    Counter ( I ) = 5
Next I
...
```

Multidimensional Arrays:

Enable supports multidimensional arrays. For example, the following example declares a two dimensional array within a procedure.

```
Static Mat(20, 20) As Double
```

Either or both dimensions can be declared with explicit lower bounds.

```
Static Mat(1 to 10, 1 to 10) As Double
```

You can efficiently process a multidimensional array with the use of **For** loops. In the following statements, the elements in a multidimensional array are set to a value.

```
Dim L As Integer, J As Integer
Static TestArray(1 To 10, 1 to 10) As Double
For L = 1 to 10
    For J = 1 to 10
```

```
        TestArray(L,J) = L * 10 J
    Next J
Next L
```

Arrays can be more than two-dimensional. Enable does not have an arbitrary upper bound on array dimensions.

```
Dim ArrTest(5, 3, 2)
```

This declaration creates an array that has three dimensions with sizes 6 by 4, by 3 unless Option Base 1 is set previously in the code. The use of Option Base 1 sets the lower bound of all arrays to 1 instead of 0.

User Defined Types

User Defined Types

Users can define their own types that are composites of other built-in or user-defined types. Variables of these new user types can be declared. Member variables of the new type can be accessed using dot notation. Variables of user-defined types cannot be passed to DLL functions expecting "C" structures.

User-defined types are created using the type statement, which must be placed outside the procedure in your Enable code. User-defined types are global. The variables that are declared as user-defined types can be either global or local. User-defined types in Enable cannot contain arrays at this time.

```

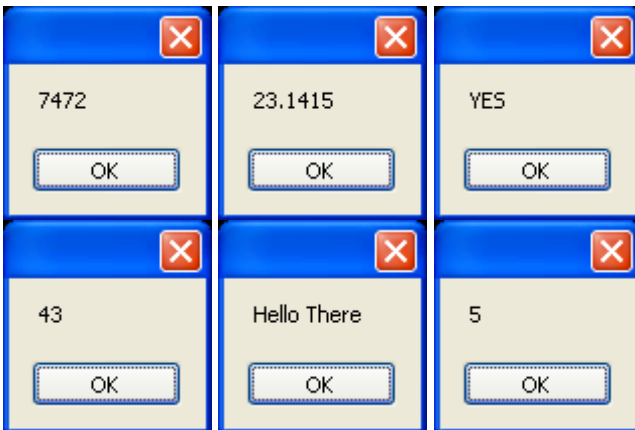
Type type1
  a As Integer
  d As Double
  s As String
End Type

Type type2
  a As Integer
  o As type1
End Type

Dim type2a As type2
Dim type1a As type1

Sub TypeExample ()
  a = 5
  type1a.a = 7472
  type1a.d = 23.1415
  type1a.s = "YES"
  type2a.a = 43
  type2a.o.s = "Hello There"
  MsgBox type1a.a
  MsgBox type1a.d
  MsgBox type1a.s
  MsgBox type2a.a
  MsgBox type2a.o.s
  MsgBox a
End Sub

```



Dialogs and Dialog Controls

Dialog Support

This topics in this book covers the code that is used to establish and maintain dialogs in a script. Written manually, this code can be very complicated; therefore, it is recommended that you use the Enable Dialog Designer, from within the eQate Action Script Editor whenever possible.

The syntax is similar to the syntax used in Microsoft Word Basic. The dialog syntax is not part of Microsoft Visual Basic or Microsoft Visual Basic for Applications (VBA). Enable has complete support for dialogs. The types of dialogs supported are outlined below.

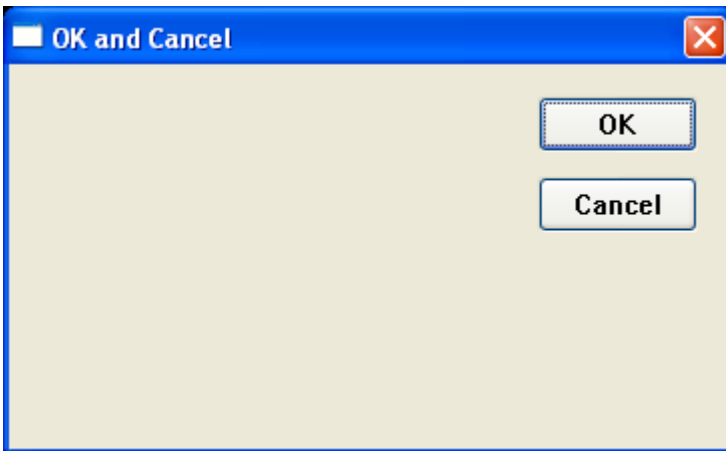
Most of the standard Windows dialog box controls are supported.

Use the browse buttons (>>) (<<) to review the controls available for custom dialog boxes and the guidelines for using them.

OK and Cancel

```
Sub Main
    Begin Dialog ButtonSample 16,32,180,96,"OK and Cancel"
        OKButton 132,8,40,14
        CancelButton 132,28,40,14
    End Dialog
    Dim Dlg1 As ButtonSample
    Button = Dialog (Dlg1)
End Sub
```

Every custom dialog box must contain at least one command button - an OK button or a Cancel button. Enable includes separate dialog box definition statements for each of these two types of buttons.



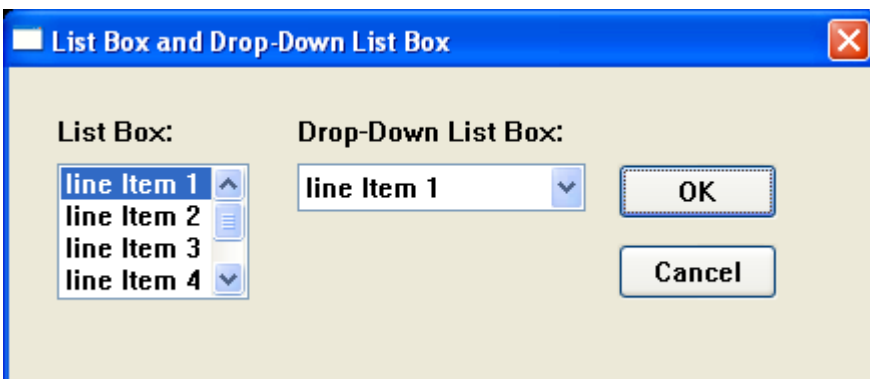
List Boxes and Drop-down List Boxes

You can use a list box or drop-down list box to present a list of items from which the user can select. A drop-down list box saves space (it can drop down to cover other dialog box controls temporarily). The items displayed in a list box or drop-down list box are stored in an array that is defined before the instructions that define the dialog box.

```
Sub Main
    Dim MyList$ (5)
    MyList (0) = "line Item 1"
    MyList (1) = "line Item 2"
    MyList (2) = "line Item 3"
    MyList (3) = "line Item 4"
    MyList (4) = "line Item 5"
    MyList (5) = "line Item 6"
Begin Dialog BoxSample 159,175, 216, 78, "List Box and Drop-Down List Box"
    OKButton 152,24,40,14
    CancelButton 152,44,40,14
    ListBox 12,24,48,40, MyList$ (), .Lstbox
    DropListBox 72,24,72,40, MyList$ (), .DrpList
    Text 12,12,32,8, "List Box:"
    Text 72,12,68,8, "Drop-Down List Box:"
End Dialog

    Dim Dlg1 As BoxSample
    Button = Dialog ( Dlg1 )

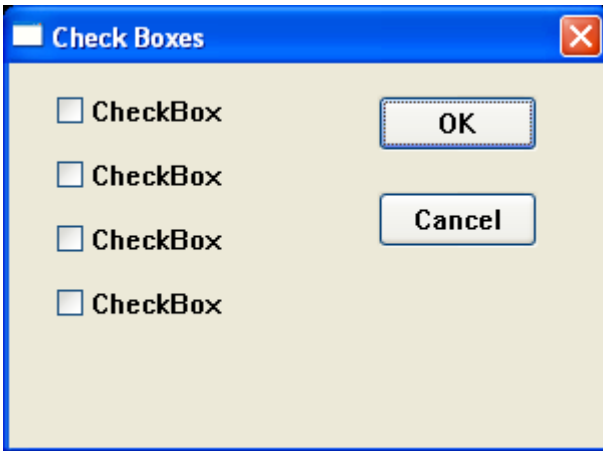
End Sub
```



Check Boxes in Dialog

```
Sub Main
    Begin Dialog CheckSample 15,32,149,96,"Check Boxes"
        OKButton 92,8,40,14
        CancelButton 92,32,40,14
        CheckBox 12,8,45,8,"CheckBox",.CheckBox1
        CheckBox 12,24,45,8,"CheckBox",.CheckBox2
        CheckBox 12,40,45,8,"CheckBox",.CheckBox3
        CheckBox 12,56,45,8,"CheckBox",.CheckBox4
    End Dialog
    Dim Dlg1 As CheckSample
    Button = Dialog ( Dlg1 )
End Sub
```

You use a check box to make a "yes or no" or "on or off" choice. For example, you could use a check box to display or hide a toolbar in your application.

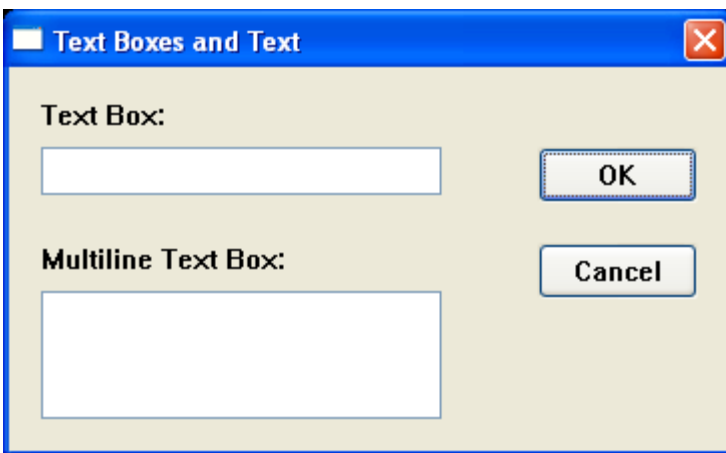


Text Boxes and Text

```
Sub Main
    Begin Dialog TextBoxSample 16,30,180,96,_
        "Text Boxes and Text"
        OKButton 132,20,40,14
        CancelButton 132,44,40,14
        Text 8,8,32,8,"Text Box:"
        TextBox 8,20,100,12,.TextBox1
        Text 8,44,84,8,"Multiline Text Box:"
        TextBox 8,56,100,32,.TextBox2
    End Dialog
    Dim Dlg1 As TextBoxSample
    Button = Dialog ( Dlg1 )

End Sub
```

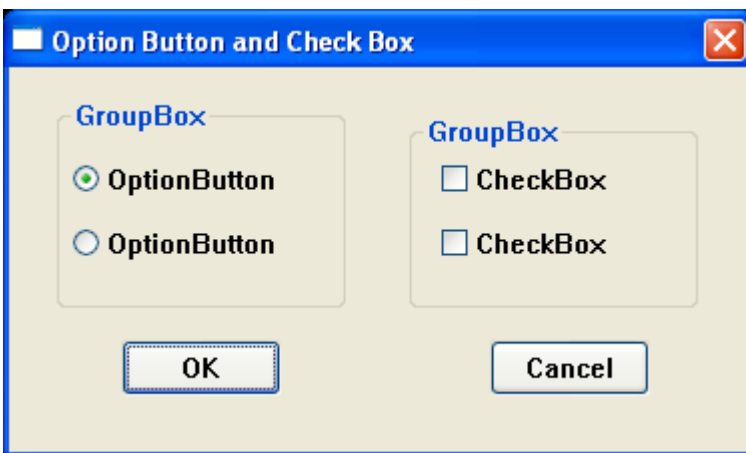
A text box control is a box in which the user can enter text while the dialog box is displayed. By default, a text box holds a single line of text. Enable does not support multiline text boxes in this version - this feature will be included in later versions.



Option Buttons and Group Boxes

You can have option buttons to allow the user to choose one option from several. Typically, you would use a group box to surround a group of option buttons, but you can also use a group box to set off a group of check boxes or any related group of controls.

```
Begin Dialog GroupSample 31,32,185,96,"Option Button and Check Box"
  OKButton 28,68,40,14
  CancelButton 120,68,40,14
  GroupBox 12,8,72,52,"GroupBox",.GroupBox1
  GroupBox 100,12,72,48,"GroupBox",.GroupBox2
  OptionGroup .OptionGroup1
  OptionButton 16,24,54,8,"OptionButton",.OptionButton1
  OptionButton 16,40,54,8,"OptionButton",.OptionButton2
  CheckBox 108,24,45,8,"CheckBox",.CheckBox1
  CheckBox 108,40,45,8,"CheckBox",.CheckBox2
End Dialog
Dim Dlg1 As GroupSample
Button = Dialog (Dlg1)
End Sub
```



The Dialog Function

Enable supports the dialog function. This function is a user-defined function that can be called while a custom dialog box is displayed. The dialog function makes nested dialog boxes possible and receives messages from the dialog box while it is still active.

When the function Dialog() is called in Enable, it displays the dialog box, and calls the dialog function for that dialog. Enable calls the dialog function to see if there are any commands to execute. Typical commands that might be used are disabling or hiding a control. By default, all dialog box controls are enabled. If you want a control to be hidden, you must explicitly make it disabled during initialization. After initialization, Enable displays the dialog box. When an action is taken by the user, Enable calls the dialog function and passes values to the function that indicate the kind of action to be taken.

The dialog box and its function are connected in the dialog definition. A function name argument (e.g., UserDialog1, below) is added to the Begin Dialog instruction, and matches the name of the dialog function located in your Enable program.

```
Begin Dialog UserDialog1 60,60, 260, 188, "3", .Enable
```

The Dialog Box Controls

A dialog function needs an identifier for each dialog box control upon which some action will take place. The dialog function uses string identifiers. String identifiers are the same as the identifiers used in the dialog record.

```
CheckBox 8, 56, 203, 16, "Check to display controls", .Chk1
```

The control's identifier and label are different. An identifier begins with a period and is the last parameter in a dialog box control instruction. In the sample code above, Check to display controls is the label and Chk1 is the identifier.

The Dialog Function Syntax

The syntax for the dialog function is as follows:

```
Function FunctionName( ControlID$, Action%, SuppValue%)
    Statement Block
FunctionName = ReturnValue
End Function
```

All parameters in the dialog function are required.

A dialog function returns a value when the user chooses a command button. Enable acts on the value returned. The default is to return zero (0) and close the dialog box. If a non-zero is assigned, the dialog box remains open. By keeping the dialog box open, the dialog function allows the user to do more than one command from the same dialog box.

ControlID\$ receives the identifier of the dialog box control.

Action identifies the action that calls the dialog function. Enable supports two actions:

Action 1	The value passed before the dialog becomes visible.
Action 2	The value passed when an action is taken (i.e. a button is pushed, checkbox is checked, etc.). The ControlID\$ is the same as the identifier for the control that was chosen.

SuppValue receives supplemental information about a change in a dialog box control. The information SuppValue receives depends upon which control calls the dialog function:

Checkbox	0 if cleared, 1 if selected.
Option Button	Number of option buttons selected, where zero is the first option button within a group.
Command Button	A value identifying the button chosen. SuppValues for push buttons are internal only.
OK Button	-1
Cancel Button	0

The following dialog function uses a Select Case control structure to check the value of Action. The SuppValue is ignored in this function.

```
' This sample file outlines dialog capabilities,
' including nesting dialog boxes.

Sub Main
    Begin Dialog UserDialog1 60,60, 260, 188, "3", .Enable
        Text 8,10,73,13, "Text Label:"
        TextBox 8, 26, 160, 18, .FText
        CheckBox 8, 56, 203, 16, "Check to display controls", .Chk1
        GroupBox 8, 79, 230, 70, "This is a group box:", .Group
        CheckBox 18,100,189,16, "Check to change button text", .Chk2
        PushButton 18, 118, 159, 16, "File History", .History
        OKButton 177, 8, 58, 21
        CancelButton 177, 32, 58, 21
    End Dialog

    Dim Dlg1 As UserDialog1
    x = Dialog( Dlg1 )
End Sub

Function Enable( ControlID$, Action%, SuppValue%)

Begin Dialog UserDialog2 160,160, 260, 188, "3", .Enable
    Text 8,10,73,13, "New dialog Label:"
    TextBox 8, 26, 160, 18, .FText
    CheckBox 8, 56, 203, 16, "New CheckBox", .ch1
    CheckBox 18,100,189,16, "Additional CheckBox", .ch2
    PushButton 18, 118, 159, 16, "Push Button", .but1
    OKButton 177, 8, 58, 21
    CancelButton 177, 32, 58, 21
End Dialog
Dim Dlg2 As UserDialog2
Dlg2.FText = "Your default string goes here"

Select Case Action%

Case 1
    DlgEnable "Group", 0
    DlgVisible "Chk2", 0
    DlgVisible "History", 0
Case 2
    If ControlID$ = "Chk1" Then
        DlgEnable "Group"
        DlgVisible "Chk2"
        DlgVisible "History"
    End If

    If ControlID$ = "Chk2" Then
        DlgText "History", "Push to display nested dialog"
    End If

    If ControlID$ = "History" Then
        Enable =1
        x = Dialog( Dlg2 )
    End If

Case Else

End Select
Enable =1
End Function
```

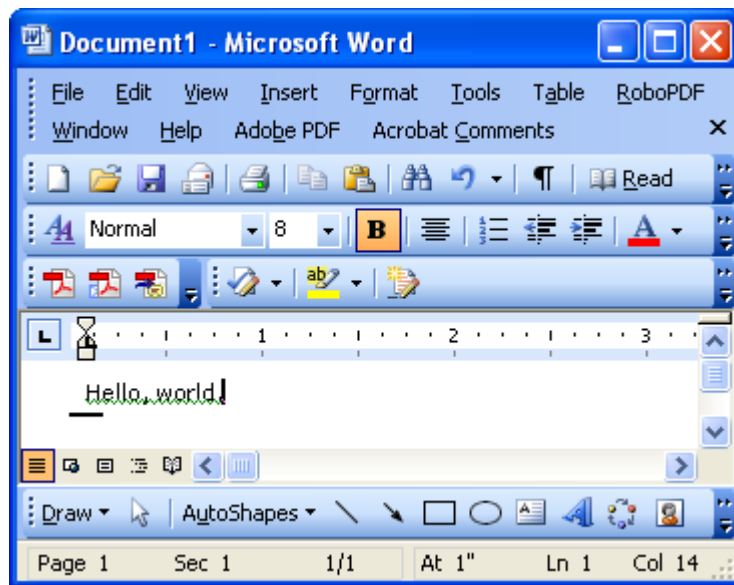

OLE Automation

What is OLE Automation?

OLE Automation is an emerging standard, promoted by Microsoft, which applications use to expose their OLE objects to development tools, Enable Basic and containers that support OLE Automation. A spreadsheet application may expose a worksheet, chart, cell or range of cells — all as different types of objects (e.g., Microsoft Excel 5.0). A word processor might expose objects such as an application, paragraph, sentence, bookmark or selection (e.g., Microsoft Word 6.0).

When an application supports OLE Automation, the objects it exposes can be accessed by Enable Basic. You can use Enable Basic to manipulate these objects by invoking methods on the object, or by getting and setting the objects properties, just as you would with the objects in Enable Basic. For example, if you created an OLE Automation object named MyObj, you might write code such as follows to manipulate the object:

```
Sub Main
  Dim MyObj As Object
  Set MyObj = CreateObject ("Word.Basic")
  MyObj.AppShow
  MyObj.FileNewDefault
  MyObj.Insert "Hello, world."
  MyObj.Bold 1
End Sub
```



The following syntax is supported for the GetObject function:

```
Set MyObj = GetObject ("", class)
```

"Class" is the parameter representing the class of the object to retrieve. The first parameter at this time must be an empty string.

The properties and methods an object supports are defined by the application that created the object. See the application's documentation for details on the properties and methods it supports.

Accessing an Object

The following functions and properties allow you to access an OLE Automation object:

<u>Name</u>	<u>Description</u>
CreateObject Function	Creates a new object of a specified type.
GetObject Function	Retrieves an object pointer to a running application.

What is an OLE Object?

An OLE Automation Object is an instance of a class within your application that you wish to manipulate programmatically. These instances may be new classes whose sole purpose is to collect and expose data and functions in a way that makes sense to your customers.

The object becomes programmable when you expose those member functions. OLE Automation defines two types of members that you may expose for an object:

- Methods are member functions that perform an action on an object. For example, a Document object might provide a Save method.
- Properties are member function pairs that set or return information about the state of an object. For example, a Drawing object might have a style property.

Microsoft suggests the following objects could be exposed by implementing the listed methods and properties for each object.

Applications:

OLE Automation

Object	Methods	Properties
Application	Help Quit AddData Repeat Undo	ActiveDocument Application Caption DefaultFilePath Documents Height Name Parent Path Printers StatusBar Top Value Visible Width

Documents:

OLE Automation

Object	Methods	Properties
Document	Activate Close NewWindow Print PrintPreview RevertToSaved Save SaveAs	Application Author Comments FullName Keywords Name Parent Path ReadOnly Saved Subject Title Value

To provide access to more than one instance of an object, expose a collection object. A collection object manages other objects. All collection objects support iteration over the objects they manage. For example, Microsoft suggests an application with a multiple document interface (MDI) might expose a Documents collection object with the following methods and properties:

Collection Object	Methods	Properties
Documents	Add Close Item Open	Application Count Parent

OLE Fundamentals

Object Linking and Embedding (OLE) is a technology that allows programmers of Windows-based applications to create applications that can display data from many different applications. OLE allows the user to edit that data

from within the application in which it was created. In some cases, the user can even edit the data from within their application.

The following terms and concepts are fundamental to understanding OLE.

OLE Object

An OLE object refers to a discrete unit of data supplied by an OLE application. An application can expose many types of objects. For example, a spreadsheet application can expose a worksheet, macro sheet, chart, cell or range of cells all as different types of objects. You use the OLE control to create linked and embedded objects. When a linked or embedded object is created, it contains the name of the application that supplied the object, its data (or, in the case of a linked object, a reference to the data) and an image of the data.

OLE Automation

Some applications provide objects that support OLE Automation. You can use Enable Basic to programmatically manipulate the data in these objects. Some objects that support OLE Automation also support linking and embedding. You can create an OLE Automation object by using the CreateObject function.

Class

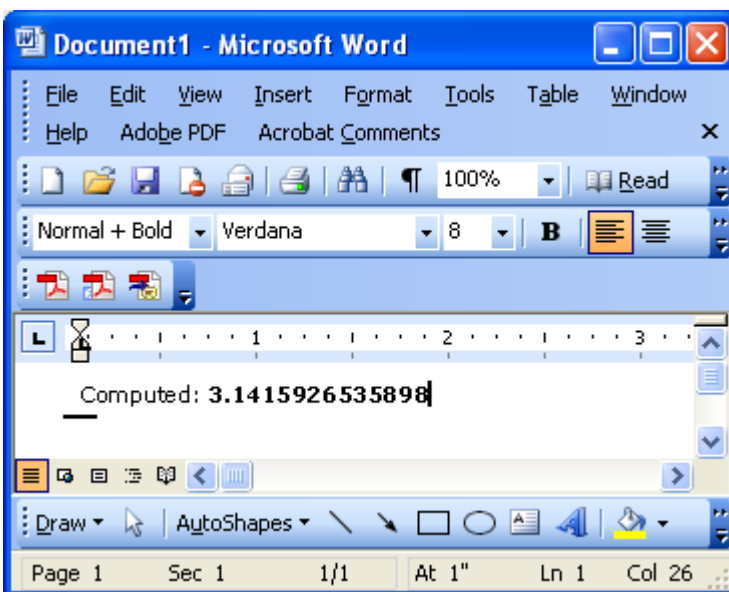
The class of an object determines the application that provides the object's data and the type of data the object contains. The class names of some commonly used Microsoft applications include MSGraph, MSDraw, WordDocument, and ExcelWorksheet.

OLE Automation and Word example

```
Sub OLEexample()
    Dim word6 As Object
    Dim myData As String

    myData = 4 * Atn(1)
    ' Demonstrates Automatic type conversion
    Set word6 = CreateObject("Word.Basic")
    word6.AppShow
    word6.FileNewDefault
    word6.Insert "The following was computed in Cypress Enable: "
    word6.Bold 1          ' Show value in boldface
    word6.Insert myData
    word6.Bold 0

    MsgBox "Done"
End Sub
```



Data Types, Operators and Precedence

Data Types, Operators and Precedence

This topic is a quick reference of the data types, operators and precedence used in eQuate scripting.

Data Types

<u>Variable</u>		<u>Type Declaration</u>	<u>Size</u>
String	\$	Dim Str_Var As String	0 to 65500 char
Integer	%	Dim Int_Var As Integer	2 bytes
Long	&	Dim Long_Var As Long	4 bytes
Single	!	Dim Sing_Var As Single	4 bytes
Double	#	Dim Dbl_Var As Double	8 bytes
Variant		Dim X As Variant	
Currency		(Currency datatype is not supported)	

Arithmetic Operators

<u>Operator</u>	<u>Function</u>	<u>Usage</u>
^	Exponentiation	$x\% = y\%^2$
-	Negation	$x\% = -2$
*	Multiplication	$x\% = 2 * 3$
/	Division	$x\% = 10/2$
Mod	Modulo	$x\% = y\% \text{ Mod } z\%$
+	Addition	$x\% = 2 + 3$
-	Subtraction	$x\% = 6 - 4$

*Arithmetic operators follow mathematical rules of precedence

* "+" or "&" can be used for string concatenation.

Operator Precedence

<u>Operator</u>	<u>Description</u>	<u>Order</u>
()	Parenthesis	Highest
^	Exponentiation	
-	Unary Minus	
/, *	Division/Multiplication	
mod	Modulo	
+, -, &	Addition, subtraction, concatenation	
=, <>, >, <, <=, >=	Relational	
not	Logical negation	
and	Logical conjunction	
or	Logical disjunction	
xor	Logical exclusion	Lowest
eqv	Logical equivalence	
imp	Logical implication	

Relational Operators

<u>Operator</u>	<u>Function</u>	<u>Usage</u>
<	Less than	$x < y$
<=	Less than or equal to	$x <= y$
=	Equals	$x = y$
>=	Greater than or equal to	$x >= y$
>	Greater than	$x > y$
<>	Not equal to	$x <> y$

Comparison Operators

<u>Operator</u>	<u>Function</u>	<u>True Tests</u>	<u>Binary Result</u>
And	Logical And	If 9 And 8 = 8	1001 And 1000 → 1000
Eqv	Logical Equivalence	If 9 Eqv 8 = -2	1001 Eqv 1000 → 1110
Imp	Logical Implication	If 8 Imp 9 = -1	1000 Imp 1001 → 1111

<u>Operator</u>	<u>Function</u>	<u>True Tests</u>	<u>Binary Result</u>
Is	Compare Two Objects Refs	If Obj Is Obj	
Not	Negation	If Not 0 = -1	
Mod	Divide and Return Remainder	If 9 Mod 8 = 1	
Or	Logical Or	If 9 Or 8 = 9	1001 Or 1000 → 1001
Xor	Logical Exclusion	If 9 Xor 8 = 1	1001 Xor 1000 → 0001

Functions, Statements, Subroutines and Events

Abs Function

Return the absolute value of a *number*.

Format:

Abs(number)

The data type of the return value is the same as that of the *number* argument. If the *number* argument is a variant of VarType (String) and can be converted to a number, the return value will be a variant of VarType (Double). If the numeric expression results in a null, Abs returns a null.

Example:

```
Sub Main
    Dim Msg, X, Y

    X = InputBox("Enter a Number:")
    Y = Abs(X)

    Msg = "The number you entered is " & X
    Msg = Msg + ". The Absolute value of " & X & " is " & Y
    MsgBox Msg ' Display Message.

End Sub
```

AppActivate Statement

Activate an application.

Format:

AppActivate "application"

The *application* parameter is a string expression and is the name that appears in the title bar of the application window to be activated.

Related Topics: Shell, SendKeys

Example:

```
Sub Main ()
    AppActivate "Microsoft Word"
    SendKeys "%F,%N,Enable"
    Msg = "Click OK to close Word"
    MsgBox Msg
    AppActivate "Microsoft Word" ' Focus back to Word
    SendKeys "%F,%C,N" ' Close Word
End Sub
```

Asc Function

Return a numeric value that is the ASCII code for the first character in a string.

Format:

Asc(string)

Example:

```
Sub Main ()
    Dim I, Msg ' Declare variables.
    For I = Asc("A") To Asc("Z") ' From A through Z.
        Msg = Msg & Chr(I) ' Create a string.
    Next I
    MsgBox Msg ' Display results.
End Sub
```

Atn Function

Return the arctangent of a number.

Format:

Atn(string)

The *rad* argument can be any numeric expression. The result is expressed in radians.

Related Topics: Cos, Tan, Sin

Example:

```
Sub AtnExample ()
    Dim Msg, Pi           ' Declare variables.
    Pi = 4 * Atn(1)       ' Calculate Pi.
    Msg = "Pi is equal to " & Str(Pi)
    MsgBox Msg            ' Display results.
End Sub
```

Beep Statement

Sound a tone through the computer's speaker.

Format:

Beep

The frequency and duration of the beep depends on hardware, which may vary among computers.

Example:

```
Sub BeepExample ()
    Dim Answer, Msg       ' Declare variables.
    Do
        Answer = InputBox("Enter a value from 1 to 3.")
        If Answer >= 1 And Answer <= 3 Then ' Check range.
            Exit Do        ' Exit Do...Loop.
        Else
            Beep            ' Beep if not in range.
        End If
    Loop
    MsgBox "You entered a value in the proper range."
End Sub
```

Begin Dialog Statement

Mark the beginning of a dialog and the overall dimensions of the dialog box.

Note: Dialogs when written manually can be very difficult; therefore, it is recommended that you use the Dialog Designer, from within the Action Script Editor whenever possible.

Format:

Begin Dialog *name starting-x-pos, starting-y-pos, width, height, "caption"* [, . *functionname*]

Related Topics: Dialog Support, CancelButton, CheckBox, Dialog, DropListBox, GroupBox, ListBox, OKButton, OptionButton, OptionGroup, PushButton, Text, TextBox

Example:

```
Sub LST()
    Begin Dialog DIALOG_1 0,0, 128, 60, "Order Amount Update"
        OKButton 80,12,36,12
        CancelButton 80,28,36,12
        Text 4,8,36,8, "Amount"
        TextBox 4,16,64,12, .TXT_AMT
    End Dialog

    Dim Dlg As Dialog_1
    Dim amt As Single
    Dim x As integer

    if ListCount("LST") > 0 then
        x = Val(ListGetIndex("LST")) + 1
        Amt = Val(ListGetColText("LST", 1))
        Dlg.Txt_Amt = Format$(Amt, "#0.00")
        if Dialog(Dlg) then
            Amt = Val(Dlg.Txt_Amt)
            ListSetColText "LST", 1, Format$(Amt, "#0.00")
            SetString "TOTAL_CHARGES:" + Str$(x), Format$(Amt, "#####.00")
        end if
    end if
```

```

else
    beep
end if
End SUB

```

CalendarDialog Function (eQuate)

Display a calendar from which a user can select a date.

Format:

CalendarDialog (*InitialDate*, *ControlName*)

The *InitialDate* parameter is any string variable containing date on the calendar to initially select. The date is entered as a string in the YYYYMMDD format. It must be exactly eight characters in length. If an empty string is used, the current date is selected.

The *ControlName* parameter is any string variable containing the name of the control in the current form over which the calendar dialog will be positioned. If an empty string is used, the dialog will be positioned in the upper left corner of the form.

The dialog simply displays a calendar with which the user can select a date. Initially, the calendar displays a single month, but the dialog may be expanded to show up to an entire year. The date is returned as a string in the format "YYYYMMDD". Canceling returns what ever was supplies as an initial date.

Call Statement

Activate an Enable subroutine or DLL function.

Formats:

Call *name* [(*parameters(s)*)]

or

name [*parameter(s)*]

Activate an Enable or DLL subroutine called *name*. The first parameter is the *name* of the function or subroutine to call, and the second is the list of *parameter(s)* to pass to the called function or subroutine.

Note: Script Functions and Subroutines may be placed in a user library (USER.LIB) and become global to any script that has need to call them. The user library must reside in the SCRIPTS directory under the installation directory.

You are never required to use the Call statement when calling an Enable or DLL subroutine. Parentheses must be used in the argument list if the Call statement is being used.

Example:

```

Sub Main ()
    Call Beep
    MsgBox "Returns a Beep"
End Sub

```

CancelButton Statement

Use to close a dialog when omitting changes.

Format:

CancelButton *starting-x-pos*, *starting-y-pos*, *width*, *height*

Related Topics: Begin Dialog, CheckBox, Dialog, DropListBox, GroupBox, ListBox, OKButton, OptionButton, OptionGroup, PushButton, Text, TextBox

Example:

```

Sub BTN_1()
Begin Dialog Dialog_1 0,0, 252, 136, "Dialog Title"
    OptionGroup .GRP_1
        OptionButton 32,48,80,12, "1st Radio - Group 1"
        OptionButton 32,64,80,12, "2nd Radio - Group 1"
    OptionGroup .GRP_2
        OptionButton 144,48,84,12, "1st Radio - Group 2"
        OptionButton 144,64,84,12, "2nd Radio - Group 2"
    OKButton 24,96,68,20
    CancelButton 156,96,52,20
    GroupBox 24,36,92,52, "Option Group 1"
    GroupBox 136,36,100,52, "Option Group 2"
    GroupBox 24,4,212,28, "Check Group"
End Dialog

```

```

        CheckBox 36,16,76,12, "Check_Box_1", .CHECKBOX_1
        CheckBox 124,16,76,12, "Check_Box_1", .CHECKBOX_2
    End Dialog
    Dim Dlg1 As Dialog_1
    Dlg1.Grp_1 = 0           ' Set 1st button - group 1
    Dlg1.Grp_2 = 1           ' Set 2nd button - group 2
    button = Dialog ( Dlg1 )
    If button = 0 Then Return
    MsgBox "Grp1: " + Dlg1.Grp_1 + ", Grp2: " + Dlg1.Grp_2
    Dialog Dlg1
End Sub

```

CBool Function

Convert expression from one data type to a Boolean expression.

Format:

CBool (*expression*)

The *expression* parameter must be a valid string or numeric expression.

Example:

```

Sub Main

    Dim A, B, Check

    A = 5: B = 5
    Check = CBool(A = B)
    Print Check

    A = 0
    Check = CBool(A)
    Print Check

End Sub

```

CDate Function

Converts any valid expression to a Date variable with a vartype of 7.

Format:

CDate (*expression*)

The parameter *expression* must be a valid string or numeric date expression and can represent a date from January 1, 30 through December 31, 9999.

Example:

```

Sub Main

    Dim MyDate, MDate, MTime, MStime
    MybDate = "May 29, 1959"      ' Define date.
    MDate = CDate(MybDate)        ' Convert to Date data type.

    MTime = "10:32:27 PM" ' Define time.
    MStime = CDate(MTime) ' Convert to Date data type.

    Print MDate
    Print MStime

End Sub

```

CDBl Function

Convert expressions from one data type to a double.

Format:

CDBl (*expression*)

The *expression* parameter must be a valid string or numeric expression.

Example:

```
Sub Main ()
  Dim y As Integer
  y = 25
  If VarType(y) = 2 Then
    Print y
    x = CDbl(y)
    'Converts integer value of y to a double value in x
    Print x
  End If
End Sub
```

ChangeCursorStyle Subroutine (eQuate)

ChangeCursorStyle allows the eQuate developer to control the cursor style from within an eQuate Action script.

Format:

ChangeCursorStyle (*CursorStyle*)

The following are available cursor styles:

<u>Name</u>	<u>Value</u>	<u>Comment</u>
crDefault	0	Usually crArrow
crNone	-1	
crArrow	-2	
crCross	-3	
crIBeam	-4	
crSizeNESW	-6	
crSizeNS	-7	
crSizeNWSE	-8	
crSizeWE	-9	
crUpArrow	-10	
crHourGlass	-11	
crDrag	-12	
crNoDrop	-13	
crHSplit	-14	
crVSplit	-15	
crMultiDrag	-16	
crSQLWait	-17	
crNo	-18	
crAppStart	-19	
crHelp	-20	
crHandPoint	-21	
crSizeAll	-22	

ChDir Statement

Change the default directory.

Format:

ChDir *pathname*

Pathname syntax:

[*drive:*][\] *dir*[*dir*]...

The *pathname* parameter is a string limited to fewer than 128 characters. The *drive* parameter is optional. The *dir* parameter is a directory name. ChDir changes the default directory on the current drive if the *drive* is omitted.

Related Topics: ChDrive, CurDir, Dir, Mkdir, Rmdir

Example:

```
Sub Main ()
  Dim Answer, Msg, NL      ' Declare variables.
```

```

NL = Chr(10)           ' Define newline.
CurPath = CurDir()    ' Get current path.
ChDir "\"              ' Change to the root directory.
Msg = "The current directory has been changed to "
Msg = Msg & CurDir() & NL & NL & "Press OK to change "
Msg = Msg & "back to your previous default directory."
Answer = MsgBox(Msg)   ' Get user response.
ChDir CurPath          ' Change back to user default.
Msg = "Directory changed back to " & CurPath & "."
MsgBox Msg             ' Display results.
End Sub

```

ChDrive Statement

Change the current drive.

Format:

ChDrive *drivename*

The *drivename* parameter is a string and must correspond to an existing drive. If *drivename* contains more than one letter, only the first character is used.

Example:

```

Sub Main ()
    Dim Msg, NL          ' Declare variables.
    NL = Chr(10)         ' Define newline.
    CurPath = CurDir()   ' Get current path.
    ChDir "\"            ' Change to the root directory.
    ChDrive "G:"         ' Change to G drive.
    Msg = "The current directory has been changed to "
    Msg = Msg & CurDir() & NL & NL & "Press OK to change back "
    Msg = Msg & "to your previous default directory."
    MsgBox Msg           ' Get user response.
    ChDir CurPath        ' Change back to user default.
    Msg = "Directory changed back to " & CurPath & "."
    MsgBox Msg           ' Display results.
End Sub

```

Related Topics: ChDir, CurDir, Dir, Mkdir, Rmdir

CheckBox Statement

Use a checkbox in a dialog for selecting one or more in a series of choices.

Format:

CheckBox *starting-x-pos, starting-y-pos, width, height, "caption", .name*

Related Topics: Begin Dialog, CancelButton, Dialog, DropListBox, GroupBox, ListBox, OKButton, OptionButton, OptionGroup, PushButton, Text, TextBox

Example:

```

Sub Main ()
    Begin Dialog DialogName1 60, 60, 160, 70, "ASC - Hello"
        TEXT 10, 10, 28, 12, "Name:"
        TEXTBOX 42, 10, 108, 12, .nameStr
        TEXTBOX 42, 24, 108, 12, .descStr
        CHECKBOX 42, 38, 48, 12, "&CHECKME", .checkInt
        OKBUTTON 42, 54, 40, 12
    End Dialog
    Dim Dlg1 As DialogName1
    Dialog Dlg1

    MsgBox Dlg1.nameStr
    MsgBox Dlg1.descStr
    MsgBox Dlg1.checkInt
End Sub

```

Choose Function

Return a value from a list of arguments.

Format:

Choose (*number*, *choice1*, [*choice2*], [*choice3*],...)

Choose will return a null value if number is less than one or greater than the number of choices in the list. If *number* is not an integer, it will be rounded to the nearest integer.

Example:

```
Sub Main
    number = 2
    GetChoice = Choose(number, "Choice1", "Choice2", "Choice3")
    Print GetChoice
End Sub
```

Chr Function

Return a one-character string whose ASCII number is the argument.

Format:

Chr(*integer*)

Chr returns a String

Example:

```
Sub ChrExample ()
    Dim X, Y, Msg, NL
    NL = Chr(10)
    For X = 1 to 2
        For Y = Asc("A") To Asc("Z")
            Msg = Msg & Chr(Y)
        Next Y
        Msg = Msg & NL
    Next X
    MsgBox Msg
End Sub
```

CInt Function

Convert any valid expression to an integer.

Format:

CInt(*expression*)

Example:

```
Sub Main ()
    Dim y As Long

    y = 25
    If VarType(y) = 2 Then
        Print y
        x = CInt(y)
        'Converts long value of y to an integer value in x
        Print x
    End If
End Sub
```

ClearDisplay Subroutine (eQuate)

Clear the internal screen to spaces and clears all field controls characters (FCC).

Format:

ClearDisplay

ClearFields Subroutine (eQuate)

Clear all eQuate form data fields (both in the form and the internal screen) to spaces. ClearFields does not affect the FCCs in the internal screen.

Format:

ClearFields

ClearScreenChanged Subroutine (eQuate)

Reset the Screen Changed Flag. Note: The Screen Changed Flag is automatically cleared on receipt of any output from the host.

Format:

ClearScreenChanged

Related Topics: ScreenChanged

CLng Function

Convert any valid expression into a Long.

Format:

CLng(*expression*)

Example:

```
Sub Main ()
    Dim y As Integer

    y = 25
    If VarType(y) = 2 Then
        Print y
        x = CLng(y)
        'Converts integer value of y to a long value in x
        Print x
    End If

End Sub
```

Close Statement

Close active file.

Format:

Close [*filenumber*]

If the optional *filenumber* parameter is omitted, all open files will be closed.

Related Topics: EOF, Input, Line Input, Open, Print #, Write #

Example:

```
Sub Make3Files ()
    Dim I, FNum, FName           ' Declare variables.
    For I = 1 To 3
        FNum = FreeFile           ' Determine next file number.
        FName = "TEST" & FNum
        Open FName For Output As FNum ' Open file.
        Print #I, "This is test #" & I ' Write string to file.
        Print #I, "Here is another "; "line"; I
    Next I
    Close                         ' Close all files.
End Sub
```

CloseApp Subroutine (eQuate)

Close the current eQuate application immediately upon exiting the current action.

Format:

CloseApp

ClosePage Subroutine (eQuate)

Close the current screen page. This subroutine only applies when developing applications utilizing T27 connections.

Format:

ClosePage

ComboBox Statement

Use a combo box in a dialog to allow the user to make a selection either by typing text into the combo box or by selecting an item from its list. If there are more items in the list than will fit in the combo box, a scroll bar will appear allowing access to all items in the list.

Format:

ComboBox *starting-x-pos, starting-y-pos, width, height, listsource, .name*

Related Topics: Begin Dialog, CancelButton, CheckBox, Dialog, DropListBox, GroupBox, ListBox, OKButton, OptionButton, OptionGroup, PushButton, Text, TextBox

Example:

```
Sub BTN_3()
Dim LISTSRC$(2)
LISTSRC(0) = "Item 1"
LISTSRC(1) = "Item 2"
LISTSRC(2) = "Item 3"
Begin Dialog DIALOG_2 16,16, 204, 96, "Test ComboBox"
  ComboBox 36,4,128,32, LISTSRC(), .COMBOBOX_1
  TextBox 36,40,128,20, .TEXTBOX_2
  OKButton 8,64,76,20
  CancelButton 108,64,72,20
End Dialog
Dim Dlg2 As DIALOG_2
Dlg2.COMBOBOX_1 = "Combo List"
button = Dialog(Dlg2)
If button = 0 Then Return
x = Dlg2.COMBOBOX_1
Dlg2.TEXTBOX_2 = LISTSRC(x)
MsgBox "Text box is set to: " + Dlg2.TEXTBOX_2
Dialog Dlg2
End SUB
```

Const Statement

Assign a symbolic name to a constant value.

Format:

[Global] Const *name* = *expression*

A constant must be defined before it is used.

The Global statement must be outside the procedure section (i.e., not in a Sub or Function) of Enable. Global variables are available to all functions and subroutines.

The definition of a Const in Enable outside the procedure is global. The syntax, Global Const and Const, used below, outside the procedure, are identical.

A type declaration character may be used (see Other Data Types). If no type declaration character is used, Enable will automatically assign one of the following data types to the constant by evaluating *expression*:

Long (if *expression* evaluates to a long or integer),
 Double (if a decimal place is present) or
 String (if *expression* evaluates to a string).

Example:

```
Global Const GloConst = 142
Const MyConst = 122

Sub Main ()
  Dim Answer, Msg, NL
  Const PI = 3.14159
  NL = Chr(10)
  CurPath = CurDir()
  ChDir "\"
  Msg = "The current directory has been changed to "
  Msg = Msg & CurDir() & NL & NL & "Press OK to change "
```

```

    Msg = Msg & "back to your previous default directory."
    Answer = MsgBox(Msg)      ' Get user response.
    ChDir CurPath             ' Change back to user default.
    Msg = "Directory changed back to " & CurPath & "."
    MsgBox Msg                ' Display results.
    Myvar = myConst + PI + GloConst
    Print MyVar
End Sub

```

Cos Function

Return the cosine of an angle.

Format:

Cos (expression)

The *radian* argument must be expressed in radians and must be a valid numeric expression. Cos will, by default, return a Double unless a Single or Integer is specified as the return value.

Example:

```

Sub Main()
    Dim J As Double
    Dim I As Single           ' Declare variables.
    Dim K As Integer
    For I =1 To 10
        Msg = Msg & Cos(I) & ", " 'Cos function call
        J=Cos(I)
        Print J
        K=Cos(I)
        Print K
    Next I
    MsgBox Msg                ' Display results.
    MsgBox Msg1
End Sub

```

CreateObject Function

Create an OLE automation object.

Format:

CreateObject (class)

The *class* parameter has the following Format:

"appname.objecttype"

The *class* parameter has the following parts:

Part	Description
<i>appname</i>	Name of the application providing the object.
<i>Objecttype</i>	Type or class of object to create.

Example:

```

Sub BUTTON_BAR_02()
    Dim word6 as object
    Set word6 = CreateObject("Word.Basic")
    word6.AppShow
    word6.FileNewDefault
    word6.ViewPage
    word6.InsertPara
    word6.Insert DF_CUST_NAME + Chr$(13) + DF_ADDRESS1 + Chr$(13)
    if Left$(DF_ADDRESS2,1) <> " " then
        word6.Insert DF_ADDRESS2 + Chr$(13)
    end if
    if Left$(DF_ADDRESS3,1) <> " " Then
        word6.Insert DF_ADDRESS3 + Chr$(13)
    end if
    word6.insert DF_CITY + ", " + DF_STATE + " " + DF_ZIP
    word6.InsertPara

```

```

word6.InsertPara
word6.Insert "Dear Valued Customer:"
word6.InsertPara
word6.InsertPara
word6.Insert "The following is your account number and "
word6.Insert "telephone number as stored in our database:"
word6.InsertPara
word6.Bold 1
word6.Insert "Account:" + Chr$(9) + DF_ACCOUNT + Chr$(13)
word6.insert "Phone number:" + Chr$(9) + DF_PHONE
word6.bold 0
word6.InsertPara
word6.Insert "If either of the above is incorrect, please "
word6.Insert "contact us immediately."
word6.InsertPara
word6.InsertPara
word6.Insert "Sincerely,"
word6.InsertPara
word6.Insert " John J. Doe"
word6.Insert " Customer Services Rep."
word6.InsertPara
MsgBox "Your letter is ready to print. Make adjustments " _
    & "in Word and print before pressing the OK button below."
End Sub

```

CSng Function

Convert any valid expression to a Single.

Format:

CSng (*expression*)

Example:

```

Sub Main ()
- Dim y As Integer

y = 25
If VarType(y) = 2 Then
    Print y
    x = CSng(y)
    'Converts the integer value of y to a single value in x
    Print x
End If

End Sub

```

CStr Function

Convert any valid expression to a String.

Format:

CStr (*expression*)

Use CStr to force the result to be expressed as a String. Use the CStr function instead of Str to provide internationally aware conversions from any other data type to a String; e.g., different decimal separators are properly recognized depending on the local setting of your system.

The data in *expression* determines what is returned according to the following:

<u>If expression is</u>	<u>CStr returns</u>
Boolean	A string containing true (-1) or false (0).
Other numeric	A string containing the number.

Related Topics: Str

CurDir Function

Return the current path for the specified drive.

Format:

CurDir (*drive*)

CurDir returns a Variant.

Related Topics: ChDir, ChDrive, Dir, Mkdir, Rmdir

Example:

```
'Declare Function CurDir Lib "NewFuns.dll" ()-As String
Sub Form_Click ()
    Dim Msg, NL           ' Declare variables.
    NL = Chr(10)          ' Define newline.
    Msg = "The current directory is: "
    Msg = Msg & NL & CurDir()
    MsgBox Msg             ' Display message.
End Sub
```

CVar Function

Convert any valid expression to a Variant.

Format:

CVar (*expression*)

Example:

```
Sub Main

    Dim MyInt As Integer
    MyInt = 4534
    Print MyInt
    MyVar = CVar(MyInt & "0.23") 'makes MyInt a Variant + 0.32
    Print MyVar

End Sub
```

Date Function

Return the current system date.

Format:

Date

or

Date()

Date returns a Variant of VarType 8 (String) containing a date.

Related Topics: Day, Format, Hour, Minute, Month, Now, Second, Weekday, Year

Examples:

```
x = Date()
Print Date
Print x
Print VarType(Date)

SysDate = Date
MsgBox Sysdate,0,"System Date"

' Returns current system date in the system-defined long
' date format.
MsgBox Format(Date, "Short Date") & " Short Date"
MsgBox Format(Date, "Long Date") & " Long Date"
```

DateSerial Function

Return a variant (Date) corresponding to the year, month and day specified.

Format:

DateSerial (*year, month, day*)

All three parameters for the DateSerial Function are required and must be valid.

Related Topics: DateValue, Day, Month, TimeSerial, TimeValue, Year

Example:

```
Sub Main

    Dim MDate
    MDate = DateSerial(1959, 5, 29)
    Print MDate

End Sub
```

DateValue Function

Return a variant (Date) corresponding to the year, month and day specified.

Format:

DateValue (*dateexpression*)

The *dateexpression* parameter can be a string or any expression that can represent a date, time or both a date and a time.

Related Topics: DateSerial, Day, Month, TimeSerial, TimeValue, Year

Example:

```
Sub Main()
Dim v As Variant
Dim d As Double
    d = Now
    Print d
    v = DateValue("1959/05/29")
    MsgBox (VarType(v))
    MsgBox (v)
End Sub
```

Day Function

Return an integer between 1 and 31 that is the portion of the *date* parameter representing the day of the month.

Format:

Day (*date*)

The *date* parameter is any string expression.

The returned integer represents the day of the *date* parameter.

If *date* is a Null, this function returns a Null.

Related Topics: Date, Format, Hour, Minute, Month, Now, Second, Weekday, Year

Example:

```
Sub Main
    MyDate = "03/03/96"
    print MyDate
    x = Day(MyDate)
    print x

End Sub
```

Declare Statement

Refer to an external procedure in a Dynamic Linking Library (DLL).

Formats:

Declare Sub *procedure* Lib "*library*" [Alias "*aliasname*"] [(*argumentlist*)]
or

Declare Function *procedure* Lib "*library*" [Alias "*aliasname*"] [(*argumentlist*)] [As *type*]

The *procedure* parameter is the name of the function or subroutine being called.

Supply the name of the DLL that contains the *procedure* on the *library* parameter.

The optional Alias *aliasname* clause is used to supply the procedure name in the DLL if different from the name specified on the *procedure* parameter.

When the optional *argumentlist* needs to be passed, the format is as follows:

```
([ [ByVal] variable [As type],[ByVal] variable [As type] ]...)
```

The optional ByVal parameter specifies that the *variable* is passed by value instead of by reference (see ByRef and ByVal).

The optional As *type* parameter is used to specify the data type. Valid types are String, Integer, Single, Double, Long and Variant (see Other Data Types).

If a procedure has no arguments, use double parentheses () only to assure that no arguments are passed. For Example:

```
Declare Sub OneTime Lib "Check" ()
```

The following syntax is an Enable extension to the Declare statement but is not supported by Microsoft Visual Basic.

```
Declare Function procedure App [Alias "aliasname"] [(argumentlist)][As Type]
```

This form of the Declare statement refers to a function located in the executable file located in the application where Enable is embedded.

Related Topics: Call

Example:

```
Declare Function GetFocus Lib "User" () As Integer
Declare Function GetWindowText Lib "User" (ByVal hWnd%, _
    ByVal Mess$, ByVal cbMax%) As Integer

Sub Main
    Dim hWndWindow%
    Dim str1 As String *51
    Dim str2 As String * 25

    hWndWindow% = GetFocus()
    print "GetWindowText returned: ", _
        GetWindowText( hWndWindow%, str1,51 )
    print "GetWindowText2 returned: ", _
        GetWindowText( hWndWindow%, str2, 25)
    print str1
    print str2

End Sub
```

Dialog Function

Return a value corresponding to the button the user chooses.

Format:

```
Dialog [(dialogrecord)]
```

The Dialog(**)** function is used to display the dialog box specified by *dialogrecord*. The *dialogrecord* parameter is the name of the dialog and must be defined in a preceding Dim statement.

The return value or button:

<u>Value</u>	<u>Description</u>
-1	OK button.
0	Cancel button.
>0	A command button where 1 is the first push button in the definition of the dialog, 2 is the second, and so on.

Related Topics: Begin Dialog, CancelButton, CheckBox, DropListBox, GroupBox, ListBox, OKButton, OptionButton, OptionGroup, PushButton, Text, TextBox

Example:

```
' This sample shows all of the dialog controls on one
' dialog and how to vary the response based on which
' PushButton was pressed.

Sub Main ()
    Dim MyList$(2)
    MyList(0) = "Banana"
    MyList(1) = "Orange"
    MyList(2) = "Apple"
```

```

Begin Dialog DIALOGNAME1 59,111, 240, 147, "Test Dialog"
  OptionGroup .GRP1
    OptionButton 42,56,48,12, "Option&1"
    OptionButton 42,66,48,12, "Option&2"
  OptionGroup .GRP2
    OptionButton 42,92,48,12, "Option&3"
    OptionButton 42,102,48,12, "Option&4"
  GroupBox 132,81,70,36, "Group"
  Text 10,10,28,12, "Name:"
  TextBox 40,10,50,12, .joe
  ListBox 102,10,108,16, MyList$(), .MyList1
  DropListBox 42,32,108,36, MyList$(), .DropList1$
  CheckBox 142,56,48,12, "Check&A", .Check1
  CheckBox 142,66,48,12, "Check&B", .Check2
  CheckBox 142,92,48,12, "Check&C", .Check3
  CheckBox 142,102,48,12, "Check&D", .Check4
  CancelButton 42,124,40,12
  OKButton 90,124,40,12
  PushButton 140,124,40,12, "&Push Me 1"
  PushButton 190,124,40,12, "Push &Me 2"
End Dialog
Dim Dlg1 As DialogName1
Dlg1.joe = "Def String"
Dlg1.MyList1 = 1
Dlg1.DropList1 = 2
Dlg1.grp2 = 1
' Dialog returns -1 for OK, 0 for Cancel, button #
' for PushButtons
button = Dialog( Dlg1 )
' MsgBox "button: " & button ' uncomment for return val
If button = 0 Then Return
MsgBox "TextBox: "& Dlg1.joe
MsgBox "ListBox: " & Dlg1.MyList1
MsgBox Dlg1.DropList1
MsgBox "grp1: " & Dlg1.grp1
MsgBox "grp2: " & Dlg1.grp2
Begin Dialog DialogName2 60, 60, 160, 60, "Test Dialog 2"
  Text 10, 10, 28, 12, "Name:"
  TextBox 42, 10, 108, 12, .fred
  OkButton 42, 44, 40, 12
End Dialog
If button = 2 Then
  Dim Dlg2 As DialogName2
  Dialog Dlg2
  MsgBox Dlg2.fred
ElseIf button = 1 Then
  Dialog Dlg1
  MsgBox Dlg1.MyList1
End If
End Sub

```

Dim Statement

Allocate storage for, and declare the data type of, variables and arrays in a module.

Format:

```
Dim variablename[(subscripts)] [As type][,name] [As type]
```

The types currently supported are Integer, Long, Single, Double and String.

Example:

```

Sub Main
  Dim x As Long
  Dim y As Integer
  Dim z As single
  Dim a As double

```

```
Dim s As String
Dim v As Variant ' This is the same as Dim x or Dim x as any
```

Dir Function

Return a file/directory name that matches the given path and attributes.

Format:

Dir [(*path*, *attributes*)]

The *path* parameter is a string expression that contains a file name. The file name may include drive and directory specifications. In addition, standard wildcard characters, asterisk (*) and question mark (?), may be included.

The Dir function returns the first file name that matches the path specified on a previous Dir function. A Dir function with no parameters specified will return the next file name that matches the path specification. When no more files meet the specification, an empty string is returned.

The *attributes* parameter is a numeric expression containing a number equal to the sum of all required attributes. The attributes are:

<u>Value</u>	<u>Attribute</u>
0	Normal
2	Hidden
4	System file
8	Volume label
16	Directory

Related Topics: ChDir, ChDrive, CurDir, Mkdir, Rmdir

DlgEnable Statement

Enable or disable a particular control on a dialog box.

Format:

DlgEnable "*controlname*", *value*

The *controlname* parameter is the name of the control on the dialog box. The *value* parameter is the value to which it is set: 1 = Enable, 0 = Disable. "On" is equal to 1 in the example below. If the *value* parameter is omitted, the status of the control toggles.

Related Topics: DlgText, DlgVisible

Example:

```
Sub Main
    Begin Dialog UserDialog1 60,60, 260, 188, "3", .Enable
        Text 8,10,73,13, "Text Label:"
        TextBox 8, 26, 160, 18, .FText
        CheckBox 8, 56, 203, 16, "Check to display controls",. Chk1
        GroupBox 8, 79, 230, 70, "This is a group box:", .Group
        CheckBox 18,100,189,16, "Check to change button text", .Chk2
        PushButton 18, 118, 159, 16, "File History", .History
        OKButton 177, 8, 58, 21
        CancelButton 177, 32, 58, 21
    End Dialog

    Dim Dlg1 As UserDialog1
    x = Dialog( Dlg1 )
End Sub

Function Enable( ControlID$, Action%, SuppValue%)

Begin Dialog UserDialog2 160,160, 260, 188, "3", .Enable
    Text 8,10,73,13, "New dialog Label:"
    TextBox 8, 26, 160, 18, .FText
    CheckBox 8, 56, 203, 16, "New CheckBox",. ch1
    CheckBox 18,100,189,16, "Additional CheckBox", .ch2
    PushButton 18, 118, 159, 16, "Push Button", .but1
    OKButton 177, 8, 58, 21
    CancelButton 177, 32, 58, 21
End Dialog
```

```

Dim Dlg2 As UserDialog2
Dlg2.FText = "Your default string goes here"

Select Case Action%

Case 1
    DlgEnable "Group", 0
    DlgVisible "Chk2", 0
    DlgVisible "History", 0
Case 2
    If ControlID$ = "Chk1" Then
        DlgEnable "Group"
        DlgVisible "Chk2"
        DlgVisible "History"
    End If

    If ControlID$ = "Chk2" Then
        DlgText "History", "Push to display nested dialog"
    End If

    If ControlID$ = "History" Then
        Enable = 1
        x = Dialog( Dlg2 )
    End If

Case Else

End Select
Enable = 1

End Function

```

DlgText Statement

Set or change the text of a dialog control.

Format:

`DlgText "controlname", string`

The *controlname* parameter is the name of the control on the dialog box. The *string* parameter is the value to which it is set.

Related Topics: DlgEnable, DlgVisible

Example:

```

If ControlID$ = "Chk2" Then
    DlgText "History", "Push to display nested dialog"
End If

```

DlgVisible Statement

Hide or make visible a particular control on a dialog box.

Format:

`DlgVisible "controlname", value`

The *controlname* parameter is the name of the control on the dialog box. The *value* parameter is the value to which it is set: 1 = Visible, 0 = Hidden. "On" is equal to 1. If the *value* parameter is omitted, the status of the control toggles.

Related Topics: DlgEnable, DlgText

Example:

```

If ControlID$ = "Chk1" Then
    DlgEnable "Group", On
    DlgVisible "Chk2"
    DlgVisible "History"
End If

```

See the DlgEnable Statement for a complete example.

Do...Loop Statement

Repeat a group of statements while a condition is true or until a condition is met.

Formats:

```
Do [While | Until condition]
    [statement(s)]
[Exit Do]
    [statement(s)]
Loop [While | Until condition]
```

The *condition* parameter may be a numeric or string expression.

Related Topics: While..Wend, With

See also, Data Types, Operators and Precedences

Example:

```
Sub Main ()
    Dim Value, Msg          ' Declare variables.
    Do
        Value = InputBox("Enter a value from 5 to 10.")
        If Value >= 5 And Value <= 10 Then
            Exit Do          ' Exit Do...Loop.
        Else
            Beep              ' Beep if not in range.
        End If
    Loop
End Sub
```

DoTerminalKey Subroutine (eQuate)

Issue any of the supported T27 or UTS keystrokes.

Format:

```
DoTerminalKey (key)
```

The *key* parameter is an integer expression representing the specific T27 or UTS key to be issued. The *key* may be specified as an Integer or Constant:

T27 Constants:

<u>Constant</u>	<u>Integer</u>
TK_ARROWDN	249
TK_ARROWLEFT	247
TK_ARROWRIGHT	248
TK_ARROWUP	246
TK_BACKSPACE	8
TK_BACKTAB	196
TK_BOUND	218
TK_CARRIAGERTN	13
TK_CLRALLVTAB	16442
TK_CLREOL	134
TK_CLREOP	135
TK_CLRFORMS	159
TK_CLRHOME	128
TK_COPY	16432
TK_CTRL	164
TK_CUT	16431
TK_DBLZERO	234
TK_DELCHAR	132
TK_DELCHARPAGE	16425
TK_DELLINE	133
TK_HOME	174
TK_INSCHAR	130
TK_INSCHARPAGE	16424

<u>Constant</u>	<u>Integer</u>
TK_INLINE	131
TK_LOCAL	168
TK_LOCKCTRL	165
TK_LOGICALEOL	16415
TK_MARK	217
TK_MOVELINEDOWN	138
TK_MOVELINEUP	139
TK_NEXTPAGE	253
TK_PASTE	16434
TK_PREVPAGE	252
TK_PRINTALL	157
TK_PRINTUNPROT	156
TK_RECALL	214
TK_RECEIVE	170
TK_ROLLDN	136
TK_ROLLUP	137
TK_SETFORMS	158
TK_SPECIFY	166
TK_STORE	213
TK_TAB	198
TK_TOGGLEFORMS	141
TK_TOGGLETAB	16441
TK_TRANSMIT	172
TK_TRANSMITLINE	16428
TK_TRIPZERO	236
TK_UPPERONLYON	210
TK_UPPERONLYOFF	211
TK_WRITEESC	16426
TK_WRITEETX	3
TK_WRITEGS	16427

Note: Since variables do not have to be declared prior to first reference, use an **Option Explicit** statement (external to the procedure) when using constants to assure that they are spelled correctly.

UTS Constants:

<u>Constant</u>	<u>Integer</u>
UK_BACK_SPACE	95
UK_CURSOR_DOWN	6
UK_CURSOR_LEFT	7
UK_CURSOR_RETURN_KEY	32
UK_CURSOR_RIGHT	8
UK_CURSOR_TO_END_LINE	66
UK_CURSOR_TO_HOME	23
UK_CURSOR_TO_START_LINE	65
UK_CURSOR_UP	9
UK_DELETE_IN_DISPLAY	11
UK_DELETE_IN_LINE	12
UK_DELETE_LINE	10
UK_ERASE_CHAR	67
UK_ERASE_DISPLAY	14
UK_ERASE_TO_END_DISPLAY	15
UK_ERASE_TO_END_FIELD	16
UK_ERASE_TO_END_LINE	17
UK_FKEY_1	43
UK_FKEY_2	44

<u>Constant</u>	<u>Integer</u>
UK_FKEY_3	45
UK_FKEY_4	46
UK_FKEY_5	47
UK_FKEY_6	48
UK_FKEY_7	49
UK_FKEY_8	50
UK_FKEY_9	51
UK_FKEY_10	52
UK_FKEY_11	53
UK_FKEY_12	54
UK_FKEY_13	55
UK_FKEY_14	56
UK_FKEY_15	57
UK_FKEY_16	58
UK_FKEY_17	59
UK_FKEY_18	60
UK_FKEY_19	61
UK_FKEY_20	62
UK_FKEY_21	63
UK_FKEY_22	64
UK_INSERT_IN_DISPLAY	25
UK_INSERT_IN_LINE	26
UK_INSERT_LINE	24
UK_KEYBOARD_UNLOCK	27
UK_LINE_DUP	28
UK_MSG_WAIT	29
UK_PRINT_KEY	30
UK_PRINT_ENTIRE_SCREEN	69
UK_SOE	3
UK_TAB_BACK	33
UK_TAB_FORWARD	34
UK_TAB_SET	35
UK_TRANSMIT_KEY	36

Note: Since variables do not have to be declared prior to first reference, use an **Option Explicit** statement (external to the procedure) when using constants to assure that they are spelled correctly.

DropListBox Statement

Use a drop-down list box in a dialog to allow the user to make a selection from a drop-down list. The drop-down list is useful when you wish to conserve space on the dialog.

Format:

DropListBox starting-x-pos, starting-y-pos, width, height, listsource, .name

Related Topics: Begin Dialog, CancelButton, CheckBox, Dialog, GroupBox, ListBox, OKButton, OptionButton, OptionGroup, PushButton, Text, TextBox

Example:

```
Sub BTN_3()
Dim LISTSRC$(2)
LISTSRC(0) = "Item 1"
LISTSRC(1) = "Item 2"
LISTSRC(2) = "Item 3"
Begin Dialog DIALOG_2 0,0, 204, 96, "Test DropListBox"
  DropListBox 36,12,128,44, LISTSRC$, .DROPLISTBOX_1
  TextBox 36,36,128,20, .TEXTBOX_2
  OKButton 8,64,76,20
  CancelButton 108,64,72,20
End Dialog
```

```

Dim Dlg2 As DIALOG_2
Dlg2.DROPLISTBOX_1 = 0
button = Dialog(Dlg2)
If button = 0 Then Return
x = Dlg2.DROPLISTBOX_1
Dlg2.TEXTBOX_2 = LISTSRC(x)
MsgBox "Text box is set to: " + Dlg2.TEXTBOX_2
Dialog Dlg2
End SUB

```

End Statement

End a program or a block of statements such as a Sub procedure or a Function.

Format:

End [Dialog | Function | If | Sub | Select | Type | With]

The End statement, without any parameters, may be used anywhere within a procedure to close files opened with the Open statement and to clear variables. To suspend execution without clearing variables or closing files, use the Stop statement.

Related Topics: Dialog, Do...Loop, Exit, Function, If...Then...Else, Select Case, Stop, Sub, Type, With

Example:

```

Sub Main()

    Dim Var1 as String

    Var1 = "hello"
    MsgBox " Calling Test"
    Test Var1
    MsgBox Var1

End Sub

Sub Test(wvar1 as string)

    wvar1 = "goodbye"
    MsgBox "Use of End Statement"
    End

End Sub

```

EOF Function

Return a value during file input that indicates whether the end of a file has been reached.

Format:

EOF (*filenumber*)

Related Topics: Close, Input, Line Input, Open, Print #, Write #

Example:

```

' This example uses the Input function to read 10 characters
' at a time from a file and display them in a MsgBox. This
' example assumes that TESTFILE is a text file with a few
' lines of sample data.

Sub Main
    Open "TESTFILE" For Input As #1      ' Open file.
    Do While Not EOF(1)                  ' Loop until end of file.
        MyStr = Input(10, #1)            ' Get ten characters.
        MsgBox MyStr
    Loop
    Close #1                              ' Close file.
End Sub

```

eQuateNavigate Function

The eQuate Navigate function display a tree structure in a separate window. It is mean to create very large and complex navigation menus.

Format:

eQuateNavigate (*FileName*, *Caption*)

FileName contains the navigation tree definition. *Caption* contains the navigator window title. The returned value is the selected item in the navigation tree. Canceling the navigator returns an empty string.

The navigation tree definition text file (*FileName*) contains one line for each item in the outline.

The line format is:

CaptionText[:*Function*]

Each space preceding text in each line indicates the level of indent — one space per level. The function to be returned follows the *CaptionText* in the form "[:function]". The colon and the function name will not be displayed in the outline.

You may include separators in the outline by entering a '-' as the first or only character in the line following the indent space(s).

The following is an example of a navigation outline file.

```
System Functions
Development
  Test User Registration : UREG
  Test Other Admin :ADMIN
Productions
  Register User :UREGP
  -
  Enroll Student :ENROLL
User Functions
  Inquire Enrollments :ENQ1
  - You can put a comment here(it will not display)
  Inquire Classes :ENQ2
  Inquire Instructors :ENQ3
  -
Maintenance
  Add :ADD
  Update :UPDT
  Delete :DELETE
```

Erase Statement

Reinitialize the elements of a fixed array.

Format:

Erase *arrayname* [, *arrayname*] ...

Related Topics: Dim

Example:

```
' This example demonstrates some of the features of
' arrays. The lower bound for an array is 0 unless
' option base has been set as in this Example:

Option Base 1
Sub Main
  Dim a(10) As Double
  MsgBox "LBound: " & LBound(a) & " UBound: " & UBound(a)
  Dim i As Integer
  For i = 0 to 3
    a(i) = 2 + i * 3.1
  Next i
  Erase ( a ) ' If this line is uncommented,
              ' then the values will all be 0
  Print a(0),a(1),a(2), a(3)
End Sub
```

Exit Statement

Exit a loop or procedure.

Format:

Exit {Do | For | Function | Sub}

Related Topics: End, Stop

Example:

```
' This sample shows Do ... Loop with Exit Do to get out.
Sub Main ()
    Dim Value, Msg          ' Declare variables.
    Do
        Value = InputBox("Enter a value from 5 to 10.")
        If Value >= 5 And Value <= 10 Then      ' Check range.
            Exit Do                            ' Exit Do...Loop.
        Else
            Beep                               ' Beep if not in range.
        End If
    Loop
End Sub
```

Exp Function

Return e to a power (e^{number}).

Format:

Exp (*number*)

The value of the constant e is approximately 2.71828.

Related Topics: Log

Example:

```
Sub ExpExample ()
    ' Exp(x) is e ^x so Exp(1) is e ^1 or e.
    Dim Msg, ValueOfE      ' Declare variables.
    ValueOfE = Exp(1)      ' Calculate value of e.
    Msg = "The value of e is " & ValueOfE
    MsgBox Msg             ' Display message.
End Sub
```

FileCopy Function

Copy a file from source to destination.

Format:

FileCopy (*sourcefile*, *destinationfile*)

The *sourcefile* and *destinationfile* parameters must be valid string expressions. The *sourcefile* is the file name of the file to copy; *destinationfile* is the file name to which the *sourcefile* is to be copied.

FileLen Function

Return a Long integer that is the length of the file in bytes.

Format:

FileLen(*filename*)

Example:

```
Sub Main

    Dim MySize
    MySize = FileLen("C:\TESTFILE") ' Returns file length (bytes).
    Print MySize

End Sub
```

FileOpenDialog Function (eQuate)

Open the File Open dialog.

Format:

`FileOpenDialog (initialdirectory, filename, defaultextension, filter, title)`

The *initialdirectory* is any string expression containing the initial directory to use when the dialog is opened.

The *filename* parameter is any string expression containing the file name to initially be displayed when the dialog is opened (usually left blank).

The *defaultextension* parameter is any string expression containing the default file extension to use if one is not supplied by the user.

The *filter* parameter is any string expression containing the filter or filters used to restrict file selection. A *filter* is setup as follows:

file-type-description-1|file-filter-1|file-type-description-2|filter-2|...file-type-description-n|filter-n

For example, to show all text files or all files use the following filter string:

"Text files(*.txt)|*.txt|All files(*.*)|*.*"

"Text files(*.txt)" is the file-type-description and the second "*.txt" is the file-filter.

The *title* parameter is any string expression containing the title to display in the file dialog.

This function returns the full file name, including path, of the selected file. If a file is not selected, the dialog is cancelled, and an empty string is returned.

Related topics: FileSaveDialog Function (eQuate), ImageOpenDialog Function (eQuate), ImageSaveDialog Function (eQuate)

FileSaveDialog Function (eQuate)

Open the File Save dialog.

`FileSaveDialog (initialdirectory, filename, defaultextension, filter, title)`

The *initialdirectory* is any string expression containing the initial directory to use when the dialog is opened.

The *filename* parameter is any string expression containing the file name to initially be displayed when the dialog is opened (usually left blank).

The *defaultextension* parameter is any string expression containing the default file extension to use if one is not supplied by the user.

The *filter* parameter is any string expression containing the filter or filters used to restrict file selection. A *filter* is setup as follows:

file-type-description-1|file-filter-1|file-type-description-2|filter-2|...file-type-description-n|filter-n

For example, to show all text files or all files use the following filter string:

"Text files(*.txt)|*.txt|All files(*.*)|*.*"

"Text files(*.txt)" is the file-type-description and the second "*.txt" is the file-filter.

The *title* parameter is any string expression containing the title to display in the file dialog.

This function returns the full file name, including path, of the selected file. If a file is not selected, the dialog is cancelled, and an empty string is returned.

Related Topics: FileOpenDialog Function (eQuate), ImageOpenDialog Function (eQuate), ImageSaveDialog Function (eQuate)

Fix Function

Return the integer portion of a number.

Format:

`Fix (number)`

Related Topics: Int

For Each...Next Statement

Repeat the group of statements for each element in an array or collection.

For Each *element* in *group*

[*statement(s)*]

[Exit For]

[*statement(s)*]

Next [*element*]

For Each ... Next statements can be nested if each loop element is unique. The For Each...Next statement cannot be used with and array of user defined types.

Example:

```
Sub Main
  dim z(1 to 4) as double
  z(1) = 1.11
  z(2) = 2.22
  z(3) = 3.33
  For Each v In z
    Print v
  Next v
End Sub
```

For...Next Statement

Repeat the execution of a block of statements for a specified number of times.

Format:

```
For counter = expression1 To expression2 [Step increment]
  [statement(s)]
Next [counter]
```

See also, Data Types, Operators and Precedences

Example:

```
Sub main ()
  Dim x,y,z

  For x = 1 to 5
    For y = 1 to 5
      For z = 1 to 5
        Print "Looping" ,z,y,x
      Next z
    Next y
  Next x
End Sub
```

Format Function

Format a string, number or variant (date/time) data type to a format expression.

Format:

```
Format (expression [,fmt] )
```

Format returns a string.

The Format has two parts:

<u>Part</u>	<u>Description</u>
<i>expression</i>	Expression to be formatted.
<i>fmt</i>	A string of characters that specify how the expression is to be displayed or the name of a commonly used predefined. Do not mix different type format expressions in a single <i>fmt</i> parameter.

If the *fmt* parameter is omitted or is zero-length and the *expression* parameter is a numeric, Format provides the same functionality as the Str function by converting the numeric value to the appropriate return data type. Positive numbers converted to strings using Format lack the leading space reserved for displaying the sign of the value, whereas those converted using Str retain the leading space.

To format dates and times, you can either use the predefined formats or create formats containing characters that have special meaning when used in a format expression.

Predefined numeric format names:

<u>Format Name</u>	<u>Description</u>
General Number	Display the number as is – without thousands separators.
Fixed	Display at least one digit to the left and two digits to the right of the decimal separator.
Standard	Display the number with thousands separators, if appropriate; display two digits to the right of the decimal separator.

<u>Format Name</u>	<u>Description</u>
Percent	Display the number multiplied by 100 with a percent sign (%) appended to the right; display two digits to the right of the decimal separator.
Scientific	Use standard scientific notation.
True/False	Display False if number is 0; otherwise, display True.

The following shows the characters you can use to create user-defined number formats:

<u>Character</u>	<u>Meaning</u>
Null string	Display the number with no formatting.
0	Digit placeholder. Display a digit or a zero. If the number being formatted has fewer digits than there are zeros (on either side of the decimal) in the format expression, leading or trailing zeros are displayed. If the number has more digits to the right of the decimal separator than there are zeros to the right of the format expression's decimal separator, the number is rounded to as many decimal places as there are zeros. If the number has more digits to left of the decimal separator than there are zeros to the left of the decimal separator in the format expression, the extra digits are displayed without modification.
#	Digit placeholder. Displays a digit or nothing. If there is a digit in the expression being formatted in the position where the # appears in the format string, displays it; otherwise, nothing is displayed.
.	Decimal placeholder. The decimal placeholder determines how many digits are displayed to the left and right of the decimal separator.
%	Percentage placeholder. The percent character (%) is inserted in the position where it appears in the format string. The expression is multiplied by 100.
,	Thousand separators. The thousands separator separates thousands from hundreds within a number that has four or more places to the left of the decimal separator. Use of this separator as specified in the format statement contains a comma surrounded by digit placeholders (0 or #). Two adjacent commas or a comma immediately to the left of the decimal separator (whether or not a decimal is specified) means "scale the number by dividing it by 1000, rounding as needed."
E-E+e-e+	Scientific format. If the format expression contains at least one digit placeholder (0 or #) to the right of E-, E+, e- or e+, the number is displayed in scientific format and E or e is inserted between the number and its exponent. The number of digit placeholders to the right determines the number of digits in the exponent. Use E- or e- to place a minus sign next to negative exponents. Use E+ or e+ to place a plus sign next to positive exponents.
:	Time separator. The actual character used as the time separator depends on the Time Format specified in the International section of the Control Panel.
/	Date separator. The actual character used as the date separator in the formatted output depends on Date Format specified in the International section of the Control Panel.
- \$ () space	Display a literal character. To display a character other than one of those listed, precede it with a backslash (\).
\	Display the next character in the format string. The backslash itself is not displayed. To display a backslash, use two backslashes (\\).

<u>Character</u>	<u>Meaning</u>
	Examples of characters that cannot be displayed as literal characters are the date- and time-formatting characters (a, c, d, h, m, n, p, q, s, t, w, y and /:), the numeric-formatting characters (#, 0, %, E, e, comma, and period), and the string-formatting characters (@, &, <, >, and !).
"String"	Display the string inside the double quotation marks. To include a string in <i>fmt</i> from within Enable, you must use the ANSI code for a double quotation mark Chr(34) to enclose the text.
*	Display the next character as the fill character. Any empty space in a field is filled with the character following the asterisk.

Unless the *fmt* argument contains one of the predefined formats, a format expression for numbers can have from one to four sections. Each section must be separated by semicolons.

<u>If you use</u>	<u>The result is</u>
One section only	The format expression applies to all values.
Two	The first section applies to positive values, the second to negative values.
Three	The first section applies to positive values, the second to negative values, and the third to zeros.
Four	The first section applies to positive values, the second to negative values, the third to zeros, and the fourth to Null values.

The following example has two sections: the first defines the format for positive values and zeros; the second section defines the format for negative values:

"\$#,##0;(\$#,##0)"

If you include semicolons with nothing between them, the missing section is printed using the format of the positive value. For example, the following format displays positive and negative values using the format in the first section and displays "Zero" if the value is zero:

"\$#,##0;;\Z\e\r\o"

Sample format expressions for numbers are shown below (examples assume the Country is set to the United States in the International section of the Control Panel). The first column contains the format strings. The other columns contain the output that results if the formatted data has the value given in the column headings:

<u>Format (fmt)</u>	<u>Positive 3</u>	<u>Negative 3</u>	<u>Decimal 3</u>	<u>Null</u>
Null string	3	-3	0.3	
0	3	-3	1	
0.00	3.00	-3.00	0.30	
#,##0	3	-3	1	
#,##0.00;;Nil	3.00	-3.00	0.30	Nil
\$#,##0;(\$#,##0)	\$3	(\$3)	\$1	
\$#,##0.00;(\$#,##0.00)	\$3.00	(\$3.00)	\$0.30	
0%	300%	-300%	30%	
0.00%	300.00%	-300.00%	30.00%	
0.00E+00	3.00E+00	-3.00E+00	3.00E-01	
0.00E-00	3.00E00	-3.00E00	3.00E-01	

Numbers can also be used to represent date and time information. You can format date and time serial numbers using date and time formats or number formats because date/time serial numbers are stored as floating-point values.

To format dates and times, you can use either the commonly used formats that have been predefined in Enable or create user-defined date and time formats using standard characters that have special meaning when used in a format expression.

The following table shows the predefined data format names you can use and the meaning of each:

<u>Format Name</u>	<u>Description</u>
General	Display a date and/or time. For real numbers, display a date and time (e.g. 4/3/93 03:34 PM); if there is no fractional part, display only a date (e.g. 4/3/93); if there is no integer part, display time only (e.g. 03:34 PM).

<u>Format Name</u>	<u>Description</u>
Long Date	Display a Long Date, as defined in the International section of the Control Panel.
Medium Date	Display a date in the same form as the Short Date, as defined in the international section of the Control Panel, except spell out the month abbreviation.
Short Date	Display a Short Date, as defined in the International section of the Control Panel.
Long Time	Display a Long Time, as defined in the International section of the Control panel. Long Time includes hours, minutes, seconds.
Medium Time	Display time in 12-hour format using hours and minutes and the AM/PM designator.
Short Time	Display a time using the 24-hour format (e.g. 17:45)

This table shows the characters you can use to create user-defined date/time formats.

<u>Character</u>	<u>Meaning</u>
c	Display the date as dddd and display the time as tttt, in that order.
d	Display the day as a number without a leading zero (1-31).
dd	Display the day as a number with a leading zero (01-31).
ddd	Display the day as an abbreviation (Sun-Sat).
dddd	Display the day as a full name (Sunday-Saturday).
w	Display the day of the week as a number (1 for Sunday through 7 for Saturday).
ww	Display the week of the year as a number (1-53).
m	Display the month as a number without a leading zero (1-12). If m immediately follows h or hh, the minute rather than the month is displayed.
mm	Display the month as a number with a leading zero (01-12). If mm immediately follows h or hh, the minute rather than the month is displayed.
mmm	Display the month as an abbreviation (Jan-Dec).
mmmm	Display the month as a full month name (January-December).
q	Display the quarter of the year as a number (1-4).
y	Display the day of the year as a number (1-366).
yy	Display the day of the year as a two-digit number (00-99)
yyyy	Display the day of the year as a four-digit number (0000-9999).
h	Display the hour as a number without leading zeros (0-23).
hh	Display the hour as a number with leading zeros (00-23).
n	Display the minute as a number without leading zeros (0-59).
nn	Display the minute as a number with leading zeros (00-59).
s	Display the second as a number without leading zeros (0-59).
ss	Display the second as a number with leading zeros (00-59).
tttt	Display a time serial number as a complete time (including hour, minute, and second) formatted using the time separator defined by the Time Format in the International section of the Control Panel. A leading zero is displayed if the Leading Zero option is selected and the time is before 10:00 A.M. or P.M. The default time format is h:mm:ss.
AM/PM	Use the 12-hour clock and display an uppercase AM/PM.
am/pm	Use the 12-hour clock display a lowercase am/pm.
A/P	Use the 12-hour clock display a uppercase A/P.
a/p	Use the 12-hour clock display a lowercase a/p
AMPM	Use the 12-hour clock and display the contents of the 11:59 string (s1159) in the WIN.INI file with any hour before noon; display the contents of the 23:59 string (s2359) with any hour between noon and 11:59 PM. AMPM can be either uppercase or lowercase, but the case of the string displayed matches the string as it exists in the WIN.INI file. The default format is AM/PM.

The following are examples of user-defined date and time formats:

<u>Format</u>	<u>Display</u>
m/d/yy	2/26/65
d-mmmm-yy	26-February-65
d-mmmm	26-February
mmmm-yy	February-65
hh:mm AM/PM	06:45 PM
h:mm:ss a/p	6:45:15 p
h:mm:ss	18:45:15
m/d/yy h:mm	2/26/65 18:45

Strings can also be formatted with Format. A format expression for strings can have one section or two sections separated by a semicolon.

<u>If you use</u>	<u>The result is</u>
One section only	The format expression applies to all string data.
Two	The first section applies to string data, the second to Null values and zero-length strings.

The following characters can be used to create a format expression for strings:

<u>Character</u>	<u>Meaning</u>
@	Character placeholder. Displays a character or a space. Placeholders are filled from right to left unless there is an exclamation character (!) in the format string.
&	Character placeholder. Display a character or nothing.
<	Force lowercase.
>	Force uppercase.
!	Force placeholders to fill from left to right instead of right to left.

Related Topic: Str

Example:

```
' This example shows various uses of the Format function to
' format values using both named and user-defined formats.
' For the date separator (/), time separator (:), and AM/
' PM literal, the actual formatted output displayed by your
' system depends on the locale settings on which the code
' is running. When times and dates are displayed in the
' development environment, the short time and short date
' formats of the code locale are used. When displayed by
' running code, the short time and short date formats of
' the system locale are used, which may differ from the code
' locale. For this example, English/United States is
' assumed.
' MyTime and MyDate are displayed in the development
' environment using current system short time and short
' date settings.
```

```
Sub Main
```

```
MyTime = "08:04:23 PM"
MyDate = "January 27, 1993"
```

```
MsgBox Now
MsgBox MyTime
```

```
MsgBox Second( MyTime ) & " Seconds"
MsgBox Minute( MyTime ) & " Minutes"
MsgBox Hour( MyTime ) & " Hours"
```

```
MsgBox Day( MyDate ) & " Days"
MsgBox Month( MyDate ) & " Months"
MsgBox Year( MyDate ) & " Years"
```

```

' Returns current system time in the system-defined long
' time format.
MsgBox Format (Time, "Short Time")
MyStr = Format(Time, "Long Time")

' Returns current system date in the system-defined long
' date format.
MsgBox Format (Date, "Short Date")
MsgBox Format (Date, "Long Date")

MyStr = Format (MyTime, "h:n:s")      ' Returns "17:4:23".
MyStr = Format (MyTime, "hh:nn:ss AMPM") ' Returns "05:04:23 PM".

MyStr = Format (MyDate, "dddd, mmm d yyyy") ' Returns "Wednesday, Jan 27 1993".

' If format is not supplied, a string is returned.
MsgBox Format (23)                  ' Returns "23".

' User-defined formats.
MsgBox Format (5459.4, "##,##0.00") ' Returns "5,459.40".
MsgBox Format (334.9, "##0.00")     ' Returns "334.90".
MsgBox Format (5, "0.00%")          ' Returns "500.00%".
MsgBox Format ("HELLO", "<")        ' Returns "hello".
MsgBox Format ("This is it", ">")   ' Returns "THIS IS IT".
End Sub

```

FreeFile Function

Return the next valid unused file number.

Format:

FreeFile [()]

Example:

```

Sub Make3Files ()
    Dim I, FNum, FName           ' Declare variables.
    For I = 1 To 3
        FNum = FreeFile          ' Determine next file number.
        FName = "TEST" & FNum
        Open FName For Output As FNum ' Open file.
        Print #I, "This is test #" & I ' Write string to file.
        Print #I, "Here is another "; "line"; I
    Next I
    Close                        ' Close all files.
End Sub

```

Function Statement

Declare and define a procedure that can receive arguments and return a value of a specified data type.

Format:

```

Function functionname [(argumentlist)] [As type]
    [statement(s)]
        functionname = expression
[Exit Function]
    [statement(s)]
        functionname = expression
End Function

```

When the optional *argumentlist* needs to be passed, the format is as follows:

([ByVal] *variable* [As *type*][,[ByVal] *variable* [As *type*]]...)

The optional ByVal parameter specifies that the *variable* is passed by value instead of by reference (see ByRef and ByVal).

The optional *As type* parameter is used to specify the data type. Valid types are String, Integer, Single, Double, Long and Variant (see Other Data Types).

Related Topics: Dim, End, Exit, Sub

Example:

```
Sub Main
  For I = 1 to 10
    Print GetColor2(I)
  Next I
End Sub
Function GetColor2( c% ) As Long
  GetColor2 = c% * 25
  If c% > 2 Then
    GetColor2 = 255      ' 0x0000FF - Red
  End If
  If c% > 5 Then
    GetColor2 = 65280    ' 0x00FF00 - Green
  End If
  If c% > 8 Then
    GetColor2 = 16711680 ' 0xFF0000 - Blue
  End If
End Function
```

Get Statement

Read from a disk file into a variable.

Format:

Get [#] *filename*, [*recordnumber*,] *variablename*

The Get statement has three parts:

<u>Parameter</u>	<u>Description</u>
<i>filename</i>	The number used to open the file.
<i>recordnumber</i>	For files opened in Binary mode, <i>recordnumber</i> is the byte position where reading starts.
<i>variablename</i>	The name of the variable used to receive the data from the file.

Related Topics: Open, Put

GetColor Function (eQuate)

Retrieve either the foreground or the background color of an eQuate control.

Format:

GetColor (*name*, *colortype*)

This function returns a Long integer.

The *name* parameter is any string expression containing the name of a valid eQuate control. The *colortype* parameter specifies either foreground or background color. The *colortype* parameter may be specified as an Integer or Constant. Valid *colortype* entries are:

<u>Color</u>	<u>Integer</u>	<u>Constant</u>
Foreground color	0	tpForeColor
Background color	1	tpBackColor

Note: Since variables do not have to be declared prior to first reference, use an **Option Explicit** statement (external to the procedure) when using constants to assure that they are spelled correctly.

Related Topics: SetColor

GetCurrentLanguage Function (eQuate)

Return the two-character language id currently in use.

Format:

Function GetCurrLanguage () As String

Related Topics: eQuate 3.5 Language Translations.

GetCursorCol Function (eQuate)

Retrieve the column of the current cursor position within the logical screen.

Format:

GetCursorCol()

This function returns an Integer.

Related Topics: GetCursorRow, SetCursor

GetCursorField Function (eQuate)

Retrieve the name of the data field at the current cursor position. If no field is selected, an empty field is returned.

Format:

GetCursorField()

This function returns a String.

Related Topics: GetCursorFieldRep, SetCursorField

GetCursorFieldRep Function (eQuate)

Retrieve the current cursor field's repeat index. If the field is not repeating, a zero is returned.

Format:

GetCursorFieldRep()

This function returns an Integer.

Related Topics: GetCursorField

Example:

```
If GetCursorField() = "ORDER_NUMBER" Then
    x = GetCursorFieldRep()
    MsgBox "You selected line " + Str$(x)
End If
```

GetCursorRow Function (eQuate)

Retrieve the row of the current cursor position within the logical screen.

Format:

GetCursorRow()

This function returns an Integer.

Related Topics: GetCursorCol, SetCursor

GetMemoSelection Function (eQuate)

Return the text within a memo's selection.

Format:

GetMemoSelection (*name*)

The *name* parameter is any string expression containing the name of the memo control.

Related Topic: SetMemoSelection

GetNumericProp Function (eQuate)

Retrieve the current value for the specified numeric property. Numeric properties include Height, Width, Left, Top, TabOrder, etc.

Format:

GetNumericProp(*name*, *property*)

This function returns an Integer.

The *name* parameter is any string expression containing the name of the control. The *property* parameter is any string expression containing the name of the property.

Related Topics: GetStringProp, SetNumericProp, SetStringProp

GetObject Function

Retrieve an OLE Automation object from a file.

Format:

```
GetObject ("filename" [, "class"])
```

The GetObject function has these parts:

<u>Part</u>	<u>Description</u>
<i>filename</i>	Full path and name of file containing the object. If filename is an empty string (""), class is required, and the function returns the currently active object of the specified type.
<i>class</i>	String representing the class of the object.

The optional *class* parameter has the following format:

```
"appname.objecttype"
```

The *class* parameter has the following parts:

<u>Part</u>	<u>Description</u>
<i>appname</i>	Name of the application providing the object.
<i>objecttype</i>	Type or class of object to get.

If the optional *class* is not supplied, the OLE2 DLLs determine the application based on the filename provided. Use the optional *class* parameter when the file supports more than one class.

Example:

```
Dim WordObject as Object
Set WordObject = GetObject ("C:\WINWORD\LETTERS\OLETST.DOC")
```

GetPage Function (eQuate)

Retrieve the current page number that is in focus. This subroutine only applies when developing applications utilizing T27 connections.

Format:

```
GetPage()
```

This function returns an Integer.

Related Topic: GetPages

GetPages Function (eQuate)

Retrieve the number representing the maximum number of pages available for the terminal screen.

Format:

```
GetPages()
```

This function returns an Integer.

Related Topic: GetPage

GetPassword Function (eQuate)

Retrieve current user's password.

Format:

```
GetPassword()
```

This function returns a String.

Example:

```
Msgbox "Your password is " +
GetPassword()
```

GetReservedString Function (eQuate)

Retrieves one of the eQuate reserved word strings.

Format:

```
GetReservedString(reserved_word_string_type)
```

This function returns a String.

The *reserved_word_string_type* parameter may be any of the following types:

<u>Type</u>	<u>Description</u>
rwCurForm	The name of the current form.
rwCurFormId	The form id. string of the current form.

<u>Type</u>	<u>Description</u>
rwCursorFld	The current screen cursor field.
rwCursorPos	The current screen cursor position ("rrrccc").
rwCursorRow	The current screen cursor row ("rrr").
rwCursorCol	The current screen cursor column ("ccc").
rwMEPMode	The current command mode indicator.
rwMasQLev	The current eQuate release level id.
rwMsgWaiting	The Message Waiting indicator.
rwPassword	The user's eQuate Password.
rwTDTime	Last transmit-to-host to form-ready time in seconds ("nnnnn.nn").
rwUserId	The user's eQuate User Id.
rwUserType	The user's eQuate User Type (1=Administrator, 2=Developer, 3=End User).

GetScreenAttribute Function (eQuate)

Return Protected, Blink and Video Off attribute states and the specified column and row position.

Format:

GetScreenAttribute (*column*, *row*)

The *column* and *row* parameters are any integer expressions.

The attribute is a numeric expression containing a number equal to the sum of all required attributes.

The attribute may be checked using an Integer or Constant:

<u>Attribute</u>	<u>Integer</u>	<u>Constant</u>
Normal	0	ATTR_NORMAL
Start of Field	1	ATTR_FIELD
Tab Stop	2	ATTR_TAB
Data Field Changed	4	ATTR_CHANGED
Protected	8	ATTR_PROTECTED
Video Off	16	ATTR_VIDEO_OFF
Numeric Only Input	32	ATTR_NUMERIC
Alphabetic Only Input	64	ATTR_ALPHA
Blinking	128	ATTR_BLINK
Right Justified Data	256	ATTR_RIGHT
UTS Low Intensity	512	ATTR_LOWINT
Reverse Video	1024	ATTR_REV

Note: Since variables do not have to be declared prior to first reference, use an **Option Explicit** statement (external to the procedure) when using constants to assure that they are spelled correctly.

See also, Data Types, Operators and Precedences

Example:

```
x = GetScreenAttribute(12, 3)
If (x and 128) <> 0 then
    MsgBox "It's blinking"
End If
```

Example2:

```
'This script selects all typed characters in the current field.
Sub Main()
    Dim CRow as Integer
    Dim CCol as Integer
    Dim SCol as Integer
    Dim ECol as Integer
    Dim FoundBlank as Integer
    Dim s as Integer
    Dim l as Integer
    Dim EndDel as String
    ' Get the cursor row and column.
    CRow = GetCursorRow()
```

```

CCol = GetCursorCol()

If (GetScreenAttribute(CCol, CRow) and 8) = 8 Then ' Cursor must be in
                                                ' an unprotected area.
                                                ' Logical AND (and 8)
                                                ' is used to mask out
                                                ' other attributes.

    Exit Sub
End If
' Find the end of the field (or end of the line)
s = CCol
ECol = 80
Do
    If (GetScreenAttribute(s, CRow) and 8) = 8 Then
        s = s - 1
        ECol = s
        Exit Do
    End If
    If s >= 80 Then
        Exit Do
    End If
    s = s + 1
Loop
' s now points to character before next protected region
' Work backward to the first non space (or beginning of
' the field) then get the end of the selection
SCol = 1
FoundBlank = False
Do
    If (GetScreenAttribute(s, CRow) and 8) = 8 Then
        SCol = s + 1
        Exit Do
    End If
    If FoundBlank = False and GetScreenText(s, CRow, 1) <> " " Then
        FoundBlank = True
        ECol = s
    End If
    If s = 1 Then
        SCol = s
        Exit Do
    End If
    s = s - 1
Loop
' Move the cursor the start of the field.
SetCursor SCol, CRow
' Mark the selection
MarkBlock SCol, CRow, ECol, CRow
RefreshScreen
' Done.
End Sub

```

GetScreenData Function (eQuate)

Retrieve a data string from the specified positions within the logical screen. This function will retrieve any text within the specified area including protected and video off text.

Format:

GetScreenData (*col*, *row*, *len*)

The *col*, *row* and *len* parameters are any integer expression.

If *row* is specified as -1, then *col* is assumed to be the offset from the beginning of the logical screen buffer. For Example:

GetScreenData (5, 2, 5)

Is the same as:

GetScreenData (85, -1,

5)

The above example assumes the screen has 80 columns.

Related Topics: GetScreenLine, SetScreenData

Example:

```
' Check for an entry in row 1 column 80.
If GetScreenData(80, 1, 1) <> " " then
...
```

GetScreenLine Function (eQuate)

Retrieve one logical line of the mapped terminal screen buffer. This function will retrieve any text within the specified area including protected and video off.

Format:

GetScreenLine (*linenumber*)

This function returns a String.

The *linenumber* parameter is any integer expression, but must be within the range of 1 through the total number of lines of the terminal screen.

Related Topics: GetScreenData,

Example:

```
Dim Hold_Line as String
' Retrieve the second line of the screen
Hold_Line = GetScreenLine(2)
```

GetSessionVar Function (eQuate)

Retrieve the current content of an eQuate Global Session Variable. If the named Session Variable has not been set yet, an empty string is returned.

Format:

GetSessionVar (*name*)

The *name* parameter is the string expression containing the session variable.

Related Topics: SetSessionVar

Example:

```
' Get the current account for the session
variable
SetString "ACCOUNT",
GetSessionVar("SAVE_ACCOUNT")
```

GetState Function (eQuate)

Retrieve the specified state of data fields or controls. The value returned will be either True (1) or False (0).

Format:

GetState (*name, statetype*)

Returns an Integer.

The *name* parameter is any string expression containing the name of a valid eQuate Data Field or eQuate Control.

The *statetype* parameter is the property type for which the state is to be retrieved.

The *statetype* is a numeric expression containing a number equal to the sum of all required attributes.

The *statetype* parameter may be expressed as an Integer or a Constant:

<u>Effect</u>	<u>Integer</u>	<u>Constant</u>
Enabled	0	tpEnabled
Visible*	1	tpVisible
Checked**	2	tpChecked

* Visible does not apply to eQuate User Menu Items.

** Checked only applies to eQuate Option Button and Check Box controls.

Note: Since variables do not have to be declared prior to first reference, use an **Option Explicit** statement (external to the procedure) when using constants to assure that they are spelled correctly.

Related Topics: SetState

Example:

```
' If ACCOUNT is enabled, disabled it.
If GetState("ACCOUNT", tpEnabled) Then
    SetState "ACCOUNT", tpEnabled, False
End If
```

GetString Function (eQuate)

Retrieve a string value from a data field or eQuate control. For eQuate controls like buttons and text labels, the string returned is the caption. For data fields, the string returned is the actual data value.

Format:

GetString (*name*)

This function returns a String.

The *name* parameter is any string expression containing the name of a valid eQuate Data Field, an eQuate Control or eQuate Global Variable.

For repeating data fields, the *name* is specified as:

name: repeat_count

Related Topics: SetString

Example:

In this example the value of a repeating data field, ORD_AMOUNT, is accumulated and displayed in a text label, LBL_TOTAL. Note how the repeating field's name and index are constructed ("ORD_AMOUNT: Str\$(x)").

```
Dim Total as Double

Total = 0
for x = 1 to 12
    Total = Total + Val(GetString("ORD_AMOUNT:" + Str$(x)))
next x
SetString "LBL_TOTAL", Format(Total, "#0.00")
```

GetStringProp Function (eQuate)

Retrieve the current value for the specified string property. String properties include Hints, Captions, Pictures, etc.

Format:

GetStringProp (*name, property*)

This function returns a String.

The *name* parameter is any string expression containing the name of the control. The *property* parameter is any string expression containing the name of the property.

Related Topics: GetNumericProp, SetNumericProp, SetStringProp

GetTranslation Function (eQuate)

Retrieve a translation from the translation database.

Format:

Function GetTranslation (ByVal *InputPhrase* As String) As String

Where *InputPhrase* is the word or phrase to be translated with the under bar prefix.

The following are examples of using the GetTranslation function:

```
Button1.Caption = GetTranslation("_Go")
MsgBox(GetTranslaton("_123"), mb_IconExclamation, "Error")
```

If any translations cannot be found, the input word/phrase (or code) is used as is.

Related Topics: eQuate 3.5 Language Translations

GetUser Function (eQuate)

Retrieve the user id. of the current user as entered at sign-on.

Format:

GetUser()

Returns a String.

Example:

```
Sub BTN_INFO()
    Dim Utype as String
    Dim Msg as String

    Utype = GetUserType()
    Select Case Utype
        Case "1" : Utype = "Administrator"
        Case "2" : Utype = "Developer"
        Case Else : Utype = "End User"
    End Select
    MsgBox "User Id: " + GetUser() + " User Type: " + _
        Utype, MB_ICONINFORMATION
End Sub
```

GetUserAppOptions Function (eQuate)

Retrieve the user application options defined in the eQuate Administration program for the current signed-on user. User application options allow the application designer to test for options placed the application registration entry for the user. This capability is similar to that available with the GetUserOption Function, but provides a means to check options on an application-by-application basis.

Format:

```
GetUserAppOptions()
```

This function returns a String.

Related Topics: GetUserOptions

GetUserOptions Function (eQuate)

Retrieve the user options defined in the eQuate Administration program for the current signed-on user. User options allow the application designer to test for options in actions before setting properties (enable, make visible, etc.).

Format:

```
GetUserOptions()
```

This function returns a String.

Related Topics: GetUserAppOptions

GetUserType Function (eQuate)

Retrieve the user type of the current user. The user type can be one of the following:

- 1 Administrator
- 2 Developer
- 3 End-User

Format:

```
GetUserType()
```

This function returns an Integer.

Example:

(see GetUser)

Global Statement

Declare a global variable and allocate storage space.

Format:

```
[Global] Const constant
```

A constant must be defined before it is used.

The Global statement must be outside the procedure section (i.e., not in a Sub or Function) of Enable. Global variables are available to all functions and subroutines in your program.

The definition of a Const in Enable, outside the procedure, is global. The syntax, Global Const and Const (used below, outside the Sub) are identical.

A type declaration character may be used (see Other Data Types). If no type declaration character is used, Enable will automatically assign one of the following data types to the constant by evaluating *expression*:

Long (if *expression* evaluates to a long or integer),
 Double (if a decimal place is present) or
 String (if *expression* evaluates to a string).

Related Topics: Const, Dim, Type

Example:

```
Global Const GloConst = 142
Const MyConst = 122      ' Global to all procedures in a module
Sub Main ()
    Dim Answer, Msg, N    ' Declare variables
    Const PI = 3.14159
    NL = Chr(10)          ' Define newline
    CurPath = CurDir()    ' Get current path
    ChDir "\"
    Msg = "The current directory has been changed to "
    Msg = Msg & CurDir() & NL & NL & "Press OK to change "
    Msg = Msg & "back to your previous default directory."
    Answer = MsgBox(Msg)  ' Get user response
    ChDir CurPath        ' Change back to user default
    Msg = "Directory changed back to " & CurPath & "."
    MsgBox Msg           ' Display results
    myvar = myConst + PI + GloConst
    Print MyVar
End Sub
```

GroupBox Statement

Use a group box in a dialog to logically group controls.

Format:

GroupBox *starting-x-pos, starting-y-pos, width, height, "caption"*

Related Topics: Begin Dialog, CancelButton, CheckBox, Dialog, DropListBox, ListBox, OKButton, OptionButton, OptionGroup, PushButton, Text, TextBox

Example:

```
GroupBox 24,4,212,28, "Check Group"
CheckBox 36,16,76,12, "Check_Box_1", .CHECKBOX_1
CheckBox 124,16,76,12, "Check_Box_1", .CHECKBOX_2
```

GoTo Statement

Branch unconditionally and without return to a specified label in a procedure.

Format:

GoTo *linelabel*

Example:

```
Sub main ()
    Dim x,y,z

    For x = 1 to 5
        For y = 1 to 5
            For z = 1 to 5
                Print "Looping" ,z,y,x
                If y > 3 Then
                    GoTo Labell1
                End If
            Next z
        Next y
    Next x
    Labell1:

End Sub
```

Hex Function

Return the hexadecimal value of a decimal parameter.

Format:

`Hex(number)`

Hex returns a string.

The *number* parameter can be any valid number. It is rounded to the nearest whole number before evaluation.

Related Topics: Oct

Example:

```
Sub Main ()

    Dim Msg As String, x%
    x% = 10
    Msg = Str( x%) & " decimal is "
    Msg = Msg & Hex(x%) & " in hex "
    MsgBox Msg
End Sub
```

Hour Function

Return an integer between 0 and 23 that is the portion of the *time* parameter representing the hour of the day.

Format:

`Hour(time)`

The *time* parameter is any string expression that can represent a time.

Related Topics: Date, Day, Format, Minute, Month, Now, Second, Weekday, Year

Example:

```
MyTime = "08:04:23 PM"
MsgBox MyTime

MsgBox Second( MyTime ) & " Seconds"
MsgBox Minute( MyTime ) & " Minutes"
MsgBox Hour( MyTime ) & " Hours"
```

If...Then...Else Statement

Allow conditional statements to be executed in the code.

Formats:

```
If condition Then
    [statement(s)]
[ElseIf condition Then
    [statement(s)]]
[Else
    [statement(s)]]
End If

or

If condition Then statement [Else statement]
```

Related Topics: Select Case

See also, Data Types, Operators and Precedences

Example:

```
Sub IfTest
    Dim msg as String
    Dim nl as String
    Dim someInt as Integer

    nl = Chr(10)
    msg = "Less"
    someInt = 4
```

```

If 5 > someInt Then msg = "Greater" : Beep
MsgBox (msg)

If 3 > someInt Then
    msg = "Greater"
    Beep
Else
    msg = "Less"
End If
MsgBox (msg)

If someInt = 1 Then
    msg = "Spring"
ElseIf someInt = 2 Then
    msg = "Summer"
ElseIf someInt = 3 Then
    msg = "Fall"
ElseIf someInt = 4 Then
    msg = "Winter"
Else
    msg = "Salt"
End If
MsgBox (msg)

End Sub

```

ImageOpenDialog Function (eQuate)

Open the Open Picture Image dialog. ImageOpenDialog differs from the File Open dialog in that it includes an image preview window. In addition, the filter is only applied in the Image Open dialog when it contains something, because default filters (*.jpg, *.bmp, etc.) are already present (not in file dialogs).

Format:

ImageOpenDialog (*initialdirectory*, *filename*, *defaultextension*, *filter*, *title*)

The *initialdirectory* is any string expression containing the initial directory to use when the dialog is opened.

The *filename* parameter is any string expression containing the file name to initially be displayed when the dialog is opened (usually left blank).

The *defaultextension* parameter is any string expression containing the default file extension to use if one is not supplied by the user.

The *filter* parameter is any string expression containing the filter or filters used to restrict file selection. A *filter* is setup as follows:

file-type-description-1|file-filter-1|file-type-description-2|filter-2|...file-type-description-n|filter-n

For example, to show all text files or all files use the following filter string:

"Text files(*.txt)|*.txt|All files(*.*)|*.*"

"Text files(*.txt)" is the file-type-description and the second "*.txt" is the file-filter.

The *title* parameter is any string expression containing the title to display in the file dialog.

This function returns the full file name, including path, of the selected file. If a file is not selected, the dialog is cancelled, and an empty string is returned.

Related topics: FileOpenDialog Function (eQuate), FileSaveDialog Function (eQuate), ImageSaveDialog Function (eQuate)

ImageSaveDialog Function (eQuate)

Open the File Save dialog. ImageSaveDialog differs from the File Save dialog in that it includes an image preview window. In addition, the filter is only applied in the Image Save dialog when it contains something, because default filters (*.jpg, *.bmp, etc.) are already present (not in file dialogs).

FileSaveDialog (*initialdirectory*, *filename*, *defaultextension*, *filter*, *title*)

The *initialdirectory* is any string expression containing the initial directory to use when the dialog is opened.

The *filename* parameter is any string expression containing the file name to initially be displayed when the dialog is opened (usually left blank).

The *defaultextension* parameter is any string expression containing the default file extension to use if one is not supplied by the user.

The *filter* parameter is any string expression containing the filter or filters used to restrict file selection. A *filter* is setup as follows:

file-type-description-1\file-filter-1\file-type-description-2\filter-2\...file-type-description-n\filter-n

For example, to show all text files or all files use the following filter string:

"Text files(*.txt)|*.txt|All files(*.*)|*.*"

"Text files(*.txt)" is the file-type-description and the second "*.txt" is the file-filter.

The *title* parameter is any string expression containing the title to display in the file dialog.

This function returns the full file name, including path, of the selected file. If a file is not selected, the dialog is cancelled, and an empty string is returned.

Related topics: FileOpenDialog Function (eQuate), FileSaveDialog Function (eQuate), ImageOpenDialog Function (eQuate)

Input Function

Return characters from a sequential file.

Format:

Input (*n*,[#]*filename*)

The Input function has two parameters: *n* and *filename*. The *n* parameter is the number of bytes to be read from a file, and *filename* is the number used in the open statement when the file was opened.

Related Topics: Close, EOF, Line Input, Open, Print #, Write #

Example:

```
Sub Main
    Open "TESTFILE" For Input As #1 ' Open file.
    Do While Not EOF(1)             ' Loop until end of
file.
        MyStr = Input(10, #1)      ' Get ten characters.
        MsgBox MyStr
    Loop
    Close #1                        ' Close file.
End Sub
```

InputBox Function

Display a prompt in a dialog box.

Format:

InputBox (*prompt*[,*title*][,*default*][,*x-pos*,*y-pos*]])

InputBox returns a String.

The Input function has these parts:

<u>Part</u>	<u>Description</u>
<i>prompt</i>	String expression displayed as the message in the dialog box.
<i>title</i>	String expression displayed in the title bar of the dialog.
<i>default</i>	String expression displayed in the textbox as the default response if no other input is provided.
<i>x-pos</i>	Numeric expression that specifies, in twips, the horizontal distance of the left edge of the dialog from the left edge of the screen. If omitted, <i>y-pos</i> must also be omitted. If <i>x-pos</i> and <i>y-pos</i> are omitted, the dialog box is horizontally centered and vertically positioned approximately one-third the way down the screen.
<i>y-pos</i>	Numeric expression that specifies, in twips, the vertical distance of the upper edge of the dialog from the top edge of the screen.

Note: A twip is 1/20 of a printer's point (1,440 twips equal an inch and 567 twips equal a centimeter).

Example:

```
MyTime = InputBox$ ("Enter Time <hh:mm:ss>." + Chr(13) + _
    Chr(10) + "(Not military time).", "Time Entry")
MyDate = InputBox$ ("Enter Date.", "Date Entry")
MsgBox Format(MyDate, "Long Date") + " at " + _
    Format(MyTime, "Long Time")
```

InStr Function

Return the position of the first occurrence of one string within another string.

Format:

`InStr (numbegin, string1, string2)`

Return the character position of the first occurrence of *string2* within *string1*.

The *numbegin* parameter is not optional and sets the starting point of the search within *string1*. The *numbegin* parameter must be a valid positive integer, no greater than 65,535.

The *string1* parameter is the string being searched and *string2* is the string for which we are looking.

The function returns the following values:

<u>If</u>	<u>InStr returns</u>
<i>string2</i> is found within <i>string1</i>	Position at which match is found
<i>string2</i> is not found	0
<i>string2</i> is zero-length	<i>numbegin</i>
<i>string2</i> is Null	Null
<i>string1</i> is zero-length	0
<i>string1</i> is Null	Null
<i>numbegin</i> > <i>string2</i>	0

Related Topics: Len

Example:

```
Sub Main ()
    B$ = "Good Bye"
    A% = InStr(2, B$, "Bye")
    C% = Instr(3, B$, "Bye")
End Sub
```

Int Function

Return the integer portion of a number.

Format:

`Int (number)`

Related Topics: Fix

IsArray Function

Return a Boolean value True or False indicating whether the *variablename* parameter is an array.

Format:

`IsArray (variablename)`

Related Topics: IsEmpty, IsNumeric, VarType, IsObject

Example:

```
Sub Main
    Dim MArray(1 To 5) As Integer, MCheck

    MCheck = IsArray(MArray)
    Print MCheck

End Sub
```

IsDate Function

Return a value that indicates if a variant parameter can be converted to a date.

Format:

`IsDate (variant)`

The IsDate function returns True if the variant can be converted to a date; otherwise, it returns False.

Related Topics: IsEmpty, IsNull, IsNumeric, VarType

Example:

```
If IsDate(inputvar)
    MsgBox Format(inputvar, "Long Date")
Else
    MsgBox "Input not a valid date."
EndIf
```

IsEmpty Function

Return a value that indicates if a variant parameter has been initialized.

Format:

IsEmpty (*variant*)

The IsEmpty function returns True if the variant is empty; otherwise, it returns False.

Related Topics: IsDate, IsNull, IsNumeric, VarType

Example:

```
Sub BTN_5()
    Dim new, answer
    If IsEmpty(new) Then
        answer = MsgBox ("Start at the beginning?", 36)
        If answer = 6 Then
            new = 1
            MsgBox "Starting at the beginning now."
        End If
    End If
End Sub
```

IsNull Function

Return a value that indicates if a variant contains the NULL value.

Format:

IsNull (*variant*)

IsNull returns a True if *variant* contains NULL; otherwise, it returns False.

The NULL value is special because it indicates that the *variant* parameter contains no data. This is different from a null-string, which is a zero-length string and an empty string that has not yet been initialized.

Related Topics: IsDate, IsEmpty, IsNumeric, VarType

IsNumeric Function

Return a value that indicates if a variant contains a numeric value.

Format:

IsNumeric (*variant*)

IsNumeric returns a True if *variant* is recognized as a number; otherwise, it returns False.

The *variant* parameter can be any variant, numeric value, date or string (if the string can be interpreted as a numeric).

Related topics: IsDate, IsEmpty, IsNull, VarType

Example:

```
Sub Form_Click ()
    Dim TestVar
    TestVar = InputBox("Please enter a number or a letter:")
    If IsNumeric(TestVar) Then
        MsgBox "Entered data is numeric."
    Else
        MsgBox "Entered data is not numeric."
    End If
End Sub
```

IsObject Function

Return a Boolean value True or False indicating whether the *objectname* parameter is an object.

Format:

IsObject (*objectname*)

Related Topics: IsEmpty, IsNumeric, VarType, IsArray

Example:

```
Sub Main

    Dim MyInt As Integer, MyCheck
    Dim MyObject As Object
    Dim YourObject As Object
    Set MyObject = CreateObject("Word.Basic")

    Set YourObject = MyObject
    MyCheck = IsObject(YourObject)

    Print MyCheck
End Sub
```

Kill Statement

Delete files from a disk. Kill will only delete files. To remove a directory, use the Rmdir Statement.

Format:

Kill *filename*

Related Topics: Rmdir

Example:

```
Const NumberOfFiles = 3

Sub Main ()
    Dim Msg                ' Declare variable.
    Call MakeFiles()       ' Create data files.
    Msg = "Several test files have been created on your "
    Msg = Msg & "disk. You may see them by switching tasks."
    Msg = Msg & " Choose OK to remove the test files."
    MsgBox Msg
    For I = 1 To NumberOfFiles
        Kill "TEST" & I      ' Remove data files from disk.
    Next I
End Sub

Sub MakeFiles ()
    Dim I, FNum, FName     ' Declare variables.
    For I = 1 To NumberOfFiles
        FNum = FreeFile    ' Determine next file number.
        FName = "TEST" & I
        Open FName For Output As FNum    ' Open file.
        Print #FNum, "This is test #" & I
        ' Write string to file.
        Print #FNum, "Here is another "; "line"; I
    Next I
    Close                  ' Close all files.
    Kill FName
End Sub
```

LBound Function

Return the smallest available subscript for the dimension of the indicated array.

Format:

LBound (*array* [, *dimension*])

Related Topics: Dim, Global, Option Base, Static, UBound

Example:

```
' This example demonstrates some of the features of
' arrays. The lower bound for an array is 0 unless it is
```

' specified or an Option Base is set as in this example.

Option Base 1

```
Sub Main
    Dim a(10) As Double
    MsgBox "LBound: " & LBound(a) & " UBound: " & UBound(a)
    Dim i As Integer
    For i = 0 to 3
        a(i) = 2 + i * 3.1
    Next i
    Print a(0), a(1), a(2), a(3)
End Sub
```

LCase Function

Return a string in which all *string* parameter letters have been converted to lower case.

Format:

LCase (*string*)

Related Topics: UCase

Example:

```
' This example uses the LTrim and RTrim functions to strip leading
' and trailing spaces, respectively, from a string variable. It
' uses the Trim function alone to strip both types of spaces.
' LCase and UCase are also shown in this example as well as the
' use of nested function calls.

Sub Main
    MyString = " <-Trim-> "           ' Initialize string
    TrimString = LTrim(MyString)      ' TrimString = "<-Trim-> "
    MsgBox "|" & TrimString & "|"
    TrimString = LCase(RTrim(MyString)) ' TrimString = " <-trim->"
    MsgBox "|" & TrimString & "|"
    TrimString = LTrim(RTrim(MyString)) ' TrimString = "<-Trim->"
    MsgBox "|" & TrimString & "|"      ' Using the Trim function
                                      ' alone achieves the same
                                      ' result.
    TrimString = UCase(Trim(MyString)) ' TrimString = "<-TRIM->"
    MsgBox "|" & TrimString & "|"
End Sub
```

Left Function

Return the left most *number* of characters of a string parameter.

Format:

Left (*string*, *number*)

The *string* parameter is the string expression from which the leftmost characters are returned.

The *number* parameter is the numeric expression indicating the number of characters that will be returned.

Related Topics: Len, Mid, Right

Example:

```
Sub Main ()
    Dim LWord, Msg, RWord, SpcPos, UsrInp ' Declare variables.
    Msg = "Enter two words separated by a space."
    UsrInp = InputBox(Msg)               ' Get user input.
    print UsrInp
    SpcPos = InStr(1, UsrInp, " ")      ' Find space.
    If SpcPos Then
        LWord = Left(UsrInp, SpcPos - 1) ' Get left word.
        RWord = Right(UsrInp, Len(UsrInp) - SpcPos)
                                           ' Get right word.
        Msg = "The first word you entered is <" & LWord & ">"
        Msg = Msg & RWord & "."
    End If
End Sub
```

```
Else
    Msg = "You didn't enter two words."
End If
MsgBox Msg                ' Display message.
MidTest = Mid("Mid Word Test", 4, 5)
Print MidTest
End Sub
```

Len Function

Return the number of characters in a string.

Format:

Len (*string*)

Related Topics: InStr

Example:

```
Sub Main ()
    A$ = "Enable"
    StrLen% = Len(A$)      ' the value of StrLen is 6
    MsgBox StrLen%
End Sub
```

Let Statement

Assign a value to a variable.

Format:

[Let] *variablename* = *expression*

The Let keyword is optional and only rarely used. The Let keyword is required in older versions of BASIC.

Example:

```
Sub Form_Click ()
    Dim Msg, Pi           ' Declare variables.
    Let Pi = 4 * Atn(1)    ' Calculate Pi.
    Msg = "Pi is equal to " & Str(Pi)
    MsgBox Msg             ' Display results.
End Sub
```

Line Input # Statement

Read a line from a sequential file into a String or Variant variable.

Format:

Line Input # *filenumber*, *name*

The parameter *filenumber* is used in the open statement to open the file. The *name* parameter is the name of a variable used to hold the line of text from the file.

Related Topics: Close, Input, EOF, Open, Print #, Write #

Example:

```
Sub FORM_INITIAL_ACTION()
    ' Load current part descriptions into list box from a file.
    dim fl as integer
    dim dta as string

    fl = FreeFile
    Open "H:\MSQDTA\PARTS.TXT" for Input as fl
    ListClear "LST_PARTDESC"
    While not Eof(fl)
        Line Input #fl, dta
        ListItemAdd "LST_PARTDESC", dta
    WEnd
    Close fl
End SUB
```

ListBox Statement

Use a list box in a dialog to allow the user to make a selection from a list box. If there are more items in the list than will fit in the list box, a scroll bar will appear allowing access to all items in the list.

Format:

ListBox *starting-x-pos*, *starting-y-pos*, *width*, *height*, *listsource*, *.name*

Related Topics: Begin Dialog, CancelButton, CheckBox, Dialog, DropListBox, GroupBox, OKButton, OptionButton, OptionGroup, PushButton, Text, TextBox

Example:

```
Sub BTN_3()
    Dim LISTSRC$(2)
    LISTSRC(0) = "Item 1"
    LISTSRC(1) = "Item 2"
    LISTSRC(2) = "Item 3"
    Begin Dialog DIALOG_2 0,0, 204, 96, "Test ListBox"
        ListBox 36,8,128,16, LISTSRC$, .LISTBOX_1
    End Dialog
```

```

    TextBox 36,36,128,20, .TEXTBOX_2
    OKButton 8,64,76,20
    CancelButton 108,64,72,20
End Dialog
Dim Dlg2 As DIALOG_2
Dlg2.LISTBOX_1 = 1
button = Dialog(Dlg2)
If button = 0 Then Return
x = Dlg2.LISTBOX_1
Dlg2.TEXTBOX_2 = LISTSRC(x)
MsgBox "Text box is set to: " + Dlg2.TEXTBOX_2
Dialog Dlg2
End SUB

```

ListClear Subroutine (eQuate)

Clear or removes all items from any type of list box.

Format:

ListClear *name*

The *name* parameter is any string expression containing the name of a list box.

Example:

(See ListItemAdd)

ListColHeader Subroutine (eQuate)

Set the column header value for a specified column in a multi-column list box. Columns are numbered from left to right starting with zero (0).

Format:

ListColHeader *name, column, headertext*

The *name* parameter is any string expression containing the name of a multi-column list box. The *column* parameter is any integer expression representing a column relative to 0. The *headertext* parameter is any string expression.

Related Topics: ListCount, ListGetColText, ListGetIndex, ListItemAdd, ListItemRemove, ListSetColText, ListSetIndex

Example:

(see ListItemAdd)

ListCount Function (eQuate)

Retrieve the item count for an eQuate Standard or Drop-down List Box.

Format:

ListCount (*name*)

The *name* parameter is any string expression containing the name of a list box.

This function returns an Integer.

Related Topics: ListColHeader, ListGetColText, ListGetIndex, ListItemAdd, ListItemRemove, ListSetColText, ListSetIndex

Example:

(see ListGetColText)

ListGetColText Function (eQuate)

Retrieve the text in a specified column from the currently selected item of a multi-column list box. Columns are numbered from left to right starting with zero (0).

Format:

ListGetColText (*name, column*)

This function returns a String.

The *name* parameter is any string expression containing the name of a multi-column list box. The *column* parameter is any integer expression representing a column relative to zero (0).

Related Topics: ListColHeader, ListCount, ListGetIndex, ListItemAdd, ListItemRemove, ListSetColText, ListSetIndex

Example:

```
' Get account from the second column of the selected row.
If ListCount("CUST_LIST") > 0 then
    ' Make sure list contains something.
    MsgBox "Selected account:" + ListGetColText("CUST_LIST", 1)
End If
```

ListGetIndex Function (eQuate)

Retrieve the index of the currently selected item of a standard or drop-down list box.

Format:

ListGetIndex (*name*)

This function returns an Integer.

The *name* parameter is any string expression containing the name of a list box.

The index can be in the range -1 to the value of ListCount - 1. A returned value of -1 indicates the list is either empty or nothing is currently selected.

Related Topics: ListColHeader, ListCount, ListGetColText, ListItemAdd, ListItemRemove, ListSetColText, ListSetIndex

ListGetItem Function (eQuate)

Get a row of data from the specified list box. Note: This function may not be used with multi-column list boxes (see ListGetColText).

Format:

ListGetItem (*name*, *index*)

This function returns a String.

The *name* parameter is any string expression containing the name of a list box or drop-down list box. The *index* parameter is any integer expression representing the row relative to zero (0).

Related Topics: ListColHeader, ListCount, ListGetColText, ListGetIndex, ListItemAdd, ListItemRemove, ListSetColText, ListSetIndex, ListSetItem

ListItemAdd Subroutine (eQuate)

Add an item to a standard, drop-down or multi-column list box.

Format:

ListItemAdd *name*, *item1* [, *item2* ... , *itemn*]

The *name* parameter is any string expression containing the name of a list box. An *item* parameter is any string expression.

To add a line to a multi-column list, use multiple *item* parameters separated by commas. Each item is put into the corresponding column from left to right.

Related Topics: ListColHeader, ListCount, ListGetColText, ListGetIndex, ListItemRemove, ListSetColText, ListSetIndex

Example:

```
' Add repeating data field to a multi-column list box.
ListColHeader "LST", 0, "Order Id." ' Set column headers
ListColHeader "LST", 1, "Amount"
ListClear "LST" ' Empty the list box
for x = 1 to 13
    Tmp = Rtrim$(GetString("ORDER_ID:" + Str$(x)))
    If Tmp <> "" then ' Do not add if order-id is blank
        ListItemAdd "LST", Tmp + "," + _
            GetString("TOTAL_CHARGES:" + Str$(x))
    End If
next x
```

ListItemRemove Subroutine (eQuate)

Remove the currently selected item from a standard or drop-down list box.

Format:

ListItemRemove *name*

The *name* parameter is any string expression containing the name of a list box.

If no item is selected or the list is empty, this function has no effect.

Related Topics: ListColHeader, ListCount, ListGetColText, ListGetIndex, ListItemAdd, ListSetColText, ListSetIndex

ListSetColText Subroutine (eQuate)

Set the text of a specified column in the currently selected item of a multi-column list box. Columns are numbered from left to right starting with zero (0).

Format:

ListSetColText *name, column, value*

The *name* parameter is any string expression containing the name of a multi-column list box. The *column* parameter is any integer expression representing the column relative to zero (0). The *value* parameter is any string expression.

Related Topics: ListColHeader, ListCount, ListGetColText, ListGetIndex, ListItemAdd, ListItemRemove, ListSetIndex

ListSetIndex Subroutine (eQuate)

Set the selected list item index of a standard or drop-down list box.

Format:

ListSetIndex *name, index*

The *name* parameter is any string expression containing the name of a list box. The *index* parameter is any integer expression representing the row relative to zero (0).

Related Topics: ListColHeader, ListCount, ListGetColText, ListGetIndex, ListItemAdd, ListItemRemove, ListSetColText

ListSetItem Subroutine (eQuate)

Set a row of data into the specified list box. Note: This function may not be used with multi-column list boxes (see ListSetColText Subroutine).

Format:

ListSetItem *name, index, value*

This function returns a String.

The *name* parameter is any string expression containing the name of a list box or drop-down list box. The *index* parameter is any integer expression representing the row relative to zero (0). The *value* parameter is any string expression containing the text of the row.

Related Topics: ListColHeader, ListCount, ListGetColText, ListGetIndex, ListGetItem, ListItemAdd, ListItemRemove, ListSetColText, ListSetIndex

LoadImage Function (eQuate)

Load an image file into the specified image control. The image file may be a Windows or OS/2 bitmap (.bmp), icon (.ico), Windows metafile (.wmf) Windows enhanced metafile (.emf) or JPEG compliant files (.jpg and .jpeg).

Format:

LoadImage (*name, imagefile*)

The *name* parameter is any string expression containing the name of an image control. The *imagefile* parameter is any string expression containing the complete drive, path and file specification of the image file. To clear the image and load a blank image, specify a null string for the *imagefile* property.

Examples:

```
Rslt = LoadImage(ImControl, "c:\Image.bmp")
```

or:

```
Rslt = LoadImage(ImControl, "")
```

LoadMMFile Function (eQuate)

Load a multimedia file into the specified multimedia player control. The file type may be any supported by the Windows Media Player.

Format:

LoadMMFile (*name*, *mmfile*)

The *name* parameter is any string expression containing the name of a media player control. The *mmfile* parameter is any string expression containing the complete drive, path and file specification of the multimedia file.

Example:

In this example, a button on one form creates an object with the TEQDlgFormX.ocx (supplied with eQuate), loads MOVIE.BFM into the dialog form and shows the form:

```
Sub Btn_Movie()
' Action for Btn_Movie
    set df = CreateObject("TEQDlgFormx.TEQDlgForm")
    df.LoadDialogForm("C:\eQuateData\BOBTSTx\MOVIE.BFM")
    df.ShowDialogForm
End Sub
```

A second form, MOVIE (hence, MOVIE.BFM) contains the control that actually activated the movie when the "Play" button on the control is clicked:

```
Sub FormInitial()
' Action for FormInitial
    Rslt = LoadMMFile("Player", "C:\eQuateData\BOBTSTx\SPEEDIS.AVI")
End Sub
```

Also see, Media Players.

LoadUserList Function (eQuate)

Load an internal list with all registered users. LoadUserList returns a count of the number of users in the list.

Format:

LoadUserList()

This function MUST be performed before the UserName or UserType functions are issued.

The function only produces a result if the current user is an Administrator. Also, the function is not available to eQuate Web clients.

Related Topics: UserName, UserType

Example:

```
' Put all registered user names and types in a listbox
Dim Cnt As Integer
Dim x as Integer

Cnt = LoadUserList()

For x = 1 to Cnt
    ListItemAdd "Listbox_1", UserName(x) & " : " & UserType(x)
Next x
```

LOF Function

Return a long number for the number of bytes in the open file.

Format:

LOF (*filenumber*)

The parameter *filenumber* is required and must be an integer.

Related Topics: FileLen

Example:

```
Sub Main

    Dim FileLength
    Open "TESTFILE" For Input As #1
    FileLength = LOF(1)
    Print FileLength
    Close #1

End Sub
```

Log Function

Return the natural log of a number.

Format:

Log (*number*)

The *number* parameter must be greater than zero, and must be a valid number.

Related Topics: Cos, Exp, Sin, Tan

Example:

```
Sub Form_Click ( )
    Dim I, Msg, NL
    NL = Chr(13) & Chr(10)
    Msg = Exp(1) & NL
    For I = 1 to 3
        Msg = Msg & Log(Exp(1) ^ I) & NL
    Next I
    MsgBox Msg
End Sub
```

Mid Function

Return a substring within a string.

Format:

Mid (*string,begin,length*)

Mid returns a String.

The Mid function has these parts:

<u>Part</u>	<u>Description</u>
<i>string</i>	String expression from which another string is created.
<i>begin</i>	Long expression that indicates the character position in <i>stringexpression</i> at which the part to be taken begins.
<i>length</i>	Long expression that indicates the number of characters to return.

Related Topics: Left, Len, Right

Example:

```
Sub BTN_6()
    Dim MidWord, Msg, TstStr, SpcPos1, SpcPos2, WordLen
    TstStr = "Mid Function Demo"
    SpcPos1 = InStr(1, TstStr, " ") ' Find 1st space
    SpcPos2 = InStr(SpcPos1 + 1, TstStr, " ") ' Find 2nd space
    WordLen = (SpcPos2 - SpcPos1) - 1 ' Get 2nd word length
    MidWord = Mid(TstStr, SpcPos1 + 1, WordLen) ' Get 2nd word
    Msg = "the word in the middle of Title is '" & MidWord & "'."
    MsgBox Msg, 0, TstStr
End SUB
```

Minute Function

Return an integer between 0 and 59 that is the portion of the *time* parameter representing the minute of the hour.

Format:

Minute (*time*)

The *time* parameter is any string expression that can represent a time.

Related Topics: Date, Day, Format, Hour, Month, Now, Second, Weekday, Year

Example:

```
MyTime = "08:04:23 PM"
MsgBox MyTime
MsgBox Second( MyTime ) & " Seconds"
MsgBox Minute( MyTime ) & " Minutes"
MsgBox Hour( MyTime ) & " Hours"
```

Mkdir Statement

Create a new directory.

Format:

`Mkdir path`

The *path* parameter is a string expression that must contain fewer than 128 characters.

Related Topics: ChDir, ChDrive, CurDir, Dir, Rmdir

Example:

```
Sub Main
    Dim DST As String
    DST = "t1"
    mkdir DST
    mkdir "t2"
End Sub
```

Month Function

Return an integer between 1 and 12 that is the portion of the *date* parameter representing the month of the year.

Format:

`Month (date)`

The *date* parameter is any string expression that can represent a date.

The returned integer represents the month of the *date* parameter.

If *date* is a Null, this function returns a Null.

Related Topics: Date, Day, Format, Hour, Minute, Now, Second, Weekday, Year

Example:

```
Sub Main
    MyDate = "03/03/96"
    print MyDate
    x = Month(MyDate)
    print x

End Sub
```

MsgBox Function, MsgBox Statement

Display a message in a dialog box and waits for the user to choose a button.

MsgBox Function returns a value indicating which button the user has chosen; the MsgBox statement does not.

Function Format:

`MsgBox (msg[, [type][, title]])`





Statement Format:

`MsgBox msg[, [type][, title]]`

The *msg* parameter is the string displayed in the dialog box as the message. The second and third parameters are optional and respectively designate the type of buttons and the title displayed in the dialog box.

The *type* is the sum of the values specifying the type of buttons to display, the icon style to use, the identity of the default button and the modality. The following illustrates the values and meaning of each group:

<u>Constant</u>	<u>Value</u>	<u>Meaning</u>
MB_OK	0	Display OK button only.
MB_OKCANCEL	1	Display OK and Cancel buttons.
MB_ABORTRETRYIGNORE	2	Display Abort, Retry and Ignore buttons.
MB_YESNOCANCEL	3	Display Yes, No and Cancel buttons.
MB_YESNO	4	Display Yes and No buttons.
MB_RETRYCANCEL	5	Display Retry and Cancel buttons.
-----	---	-----

<u>Constant</u>	<u>Value</u>	<u>Meaning</u>
MB_ICONSTOP	16	 Display:
MB_ICONQUESTION	32	 Display:
MB_ICONEXCLAMATION	48	 Display:
MB_ICONINFORMATION	64	 Display:
-----	---	-----
MB_DEFBUTTON1	0	First button is default.
MB_DEFBUTTON2	256	Second button is default.
MB_DEFBUTTON3	512	Third button is default.
-----	---	-----
MB_APPLMODAL	0	Application modal. The user must respond to the message box before continuing work in the current application.
MB_SYSTEMMODAL	4096	System modal. All applications are suspended until the user responds to the message box.

The first group of values (0-5) describes the number and type of buttons displayed in the dialog box. The second group (16, 32, 48, 64) describes the icon style. The third group (0, 256, 512) determines which button is the default. The fourth group (0, 4096) determines the modality of the message box. When adding numbers to create a final value for the argument type, use only one number from each group. If omitted, the default value for type is zero (0).

The *title* parameter is a string expression displayed in the title bar of the dialog box. If you omit the argument title, MsgBox has no default title.

The value returned by the MsgBox function indicates which button has been selected, as shown below:

<u>Constant</u>	<u>Value</u>	<u>Meaning</u>
IDOK	1	OK button selected.
IDCANCEL	2	Cancel button selected.
IDABORT	3	Abort button selected.
IDRETRY	4	Retry button selected.
IDIGNORE	5	Ignore button selected.
IDYES	6	Yes button selected.
IDNO	7	No button selected.

If the dialog box displays a **Cancel** button, pressing the **Esc** key has the same effect as choosing **Cancel**.

Note: Since variables do not have to be declared prior to first reference, use an **Option Explicit** statement (external to the procedure) when using constants to assure that they are spelled correctly.

Example:

This example uses MsgBox to display a "close without saving" message in a dialog box with a Yes button, a No button and a Cancel button. The Yes button is the default response. The MsgBox function returns a value based on the button chosen by the user. The MsgBox statement uses that value to display a message that indicates which button was chosen.

```
Sub Main
    Dim DgDef, Msg, Response, Title
    Title = "MsgBox Sample Question"
    Msg = "This is a sample of Close Without Saving?."
    Msg = Msg & " Do you want to save changes?"
    DgDef = MB_YESNOCANCEL + MB_ICONQUESTION + MB_DEFBUTTON1
    Response = MsgBox(Msg, DgDef, Title)
    If Response = IDYES Then
        Msg = "You chose Yes or pressed Enter."
```

```

    ElseIf Response = IDCANCEL
        Msg = "You chose Cancel or pressed Esc."
    Else
        Msg = "You chose No."
    End If
    MsgBox Msg
End Sub

```

Name Statement

Change the name of a directory or a file.

Format:

Name *oldname* As *newname*

The *oldname* and *newname* parameters are strings expressions that can optionally contain a path.

Related Topics: ChDir, Kill

Example:

```

Sub Main

    Name "testfile" As "newtest"

End Sub

```

Now Function

Return a date that represents the current date and time according to the settings in the computer's system date and time.

Format:

Now

Related Topics: Date, Day, Format, Hour, Minute, Month, Second, Weekday, Year

Example:

```

Sub Main ()
    Dim Today
    Today = Now
End Sub

```

Oct Function

Return the octal value of the decimal parameter.

Format:

Oct (*number*)

Oct returns a string.

Related Topics: Hex, Hex\$

Example:

```

Sub Main ()
    Dim Msg, Num      ' Declare variables.
    Num = InputBox("Enter a number.")    ' Get user input.
    Msg = Num & " decimal is &O"
    Msg = Msg & Oct(Num) & " in octal notation."
    MsgBox Msg        ' Display results.
End Sub

```

OKButton Statement

Use to close a dialog when accepting changes.

Format:

OKBUTTON *starting-x-pos, starting-y-pos, width, height*

Related Topics: Begin Dialog, CancelButton, CheckBox, Dialog, DropListBox, GroupBox, ListBox, OptionButton, OptionGroup, PushButton, Text, TextBox

Example:

```

Sub Main ()
  Begin Dialog DialogName1 60, 60, 160, 70, "ASC - Hello"
    TEXT 10, 10, 28, 12, "Name:"
    TEXTBOX 42, 10, 108, 12, .nameStr
    TEXTBOX 42, 24, 108, 12, .descStr
    CHECKBOX 42, 38, 48, 12, "&CHECKME", .checkInt
    OKBUTTON 42, 54, 40, 12
  End Dialog
  Dim Dlg1 As DialogName1
  Dialog Dlg1
  MsgBox Dlg1.nameStr
  MsgBox Dlg1.descStr
  MsgBox Dlg1.checkInt
End Sub

```

On Error Statement

Enable error-handling routine and specifies the line label of the error-handling routine.

Format:

On Error { GoTo *line* | Resume Next | GoTo 0 }

The *line* parameter refers to a label. That label must be present in the code or an error is generated.

Errors can be raised with the syntax:

Err.Raise x

The list below shows the corresponding descriptions for the defined values of x:

5	Invalid procedure call
6	Overflow
7	Out of memory
9	Subscript out of range
10	Array is fixed or temporarily locked
11	Division by zero
13	Type mismatch
14	Out of string space
16	Expression too complex
17	Cannot perform requested operation
18	User interrupt occurred
20	Resume without error
28	Out of stack space
35	Sub, Function or Property not defined
47	Too many DLL application clients
48	Error in loading DLL
49	Bad DLL calling convention
51	Internal error
52	Bad file name or number
53	File not found
54	Bad file mode
55	File already open
57	Device I/O error
58	File already exists
59	Bad record length
60	Disk full
62	Input past end of file
63	Bad record number
67	Too many files
68	Device unavailable
70	Permission denied

71	Disk not ready
74	Cannot rename with different drive
75	Path/File access error
76	Path not found
91	Object variable or With block variable not set
92	For loop not initialized
93	Invalid pattern string
94	Invalid use of Null

OLE Automation Messages:

429	OLE Automation server cannot create object
430	Class does not support OLE Automation
432	File name or class name not found during OLE Automation operation
438	Object doesn't support this property or method
440	OLE Automation error
443	OLE Automation object does not have a default value
445	Object doesn't support this action
446	Object doesn't support named arguments
447	Object doesn't support current local setting
448	Named argument not found
449	Argument not optional
450	Wrong number of arguments
451	Object not a collection

Miscellaneous Messages:

444	Method not applicable in this context
452	Invalid ordinal
453	Specified DLL function not found
457	Duplicate Key
460	Invalid Clipboard format
461	Specified format does not match format of data
480	Cannot create AutoRedraw image
481	Invalid picture
482	Printer error
483	Printer driver does not support specified property
484	Problem getting printer information from the system – make sure the printer is setup correctly
485	invalid picture type
520	Cannot empty Clipboard
521	Cannot open Clipboard

Example:

```

On Error GoTo dude
Dim x as object
x.draw      ' Object not set
jpe         ' Undefined function call
print 1/0   ' Division by zero
Err.Raise 6 ' Generate an "Overflow" error
MsgBox "Back"
MsgBox "Jack"
Exit Sub
dude:
MsgBox "HELLO"
Print Err.Number, Err.Description
Resume Next
MsgBox "Should not get here!"
MsgBox "What?"
End Sub

```

Open Statement

Open a file for input and output operations.

Format:

Open *file* [For *mode*] [Access *access*] As [#] *filenumber*

You must open a file before any I/O operation can be performed on it. The Open statement has these parts:

<u>Part</u>	<u>Description</u>
<i>file</i>	File name or path.
<i>mode</i>	Reserved word that specifies the file mode: Append, Input, Output.
<i>access</i>	Reserved word that specifies which operations are permitted on the open file: Read, Write.
<i>filenumber</i>	Integer expression with a value between 1 and 255, inclusive. When an Open statement is executed, <i>filenumber</i> is associated with the file as long as it is open. Other I/O statements can use the number to refer to the file.

If the file doesn't exist, it is created when the file is opened for Append or Output modes.

The *mode* argument is a reserved word that specifies one of the following file modes:

<u>Mode</u>	<u>Description</u>
Input	Sequential input mode.
Output	Sequential output mode.
Append	Sequential output mode. Append sets the file pointer to the end of the file. A Print # or Write # statement then extends (appends to) the file.

The *access* argument is a reserved word that specifies the operations that can be performed on the opened file. If the file is already opened by another process and the specified type of access is not allowed, the Open operation fails and a Permission denied error occurs. The Access clause works only if you are using a version of MS-DOS that supports networking (MS-DOS version 3.1 or later). If you use the Access clause with a version of MS-DOS that does not support networking, a "feature unavailable" error occurs. The *access* argument can be one of the following reserved words.

<u>Access</u>	<u>Description</u>
Read	Opens the file for reading only.
Write	Opens the file for writing only.
Read Write	Opens the file for both reading and writing. This access is valid only for files opened for Append mode.

Related Topics: Close, EOF, Input, Line Input, Open, Print #, Write

The following example writes data to a test file and reads it back:

```
Sub Main ()
    Dim FileData, Msg, NL          ' Declare variables.
    NL = Chr(10)                  ' Define newline.
    Open "TESTFILE" For Output As #1
                                   ' Open to write file.
    Print #2, "This is a test of the Print # statement."
    Print #2                        ' Print blank line to file.
    Print #2, "Zone 1", "Zone 2"   ' Print in two print zones.
                                   ' Print two strings together.
    Print #2, "With no space between" ; "."
    Close
    Open "TESTFILE" for Input As #2 ' Open to read file.
    Do While Not EOF(2)
        Line Input #2, FileData    ' Read a line of data.
        Msg = Msg & FileData & NL ' Construct message.
        MsgBox Msg
    Loop
    Close                          ' Close all open files.
    MsgBox "Testing Print Statement" ' Display message.
    Kill "TESTFILE"               ' Remove file from disk.
End Sub
```

Option Base Statement

Declare the default lower bound for array subscripts.

Format:

Option Base *number*

The Option Base statement is never required. If used, the Option Base statement can appear only once in a module and must be used before declaring the dimensions of any arrays. The Option Base can occur only in the Declarations section.

The value of *number* must be either zero (0) or one (1). The default base is zero.

The To clause in the Dim, Global, and Static statements provides a more flexible way to control the range of an array's subscripts. If you do not explicitly set the lower bound with a To clause, you can use Option Base to change the default lower bound to one (1).

Related Topics: Dim, Global, Lbound, Static, UBound

The following example uses the Option Base statement to override the default base array subscript value of zero (0):

```
Option Base 1                ' Module level statement.
Sub Main
    Dim A(20), Msg, NL       ' Declare variables.
    NL = Chr(10)              ' Define newline.
    Msg = "The lower bound of array A is " & LBound(A) & "."
    Msg = Msg & NL & "The upper bound is " & UBound(A) & "."
    MsgBox Msg                ' Display message.
End Sub
```

Option Explicit Statement

Require all variables referenced to be explicitly declared.

Format:

Option Explicit

The Option Explicit statement is used outside of the script in the Declarations section.

It is highly recommended that an Option Explicit statement be used in all eQuate actions.

Related Topics: Const, Global

Example :

```
Option Explicit
Sub Main
    Print y                  ' because y is not explicitly
                            ' dimmed, an error will occur.
End Sub
```

OptionButton Statement

Use an option button (radio button) in a dialog for selecting one, and only one, option from a group of options.

Format:

OptionButton *starting-x-pos, starting-y-pos, width, height, "caption"*

Related Topics: Begin Dialog, CancelButton, CheckBox, Dialog, DropListBox, GroupBox, ListBox, OKButton, OptionGroup, PushButton, Text, TextBox

Example:

```
Sub BTN_1()
Begin Dialog Dialog_1 0,0, 252, 136, "Dialog Title"
OptionGroup .GRP_1
    OptionButton 32,48,80,12, "1st Radio - Group 1"
    OptionButton 32,64,80,12, "2nd Radio - Group 1"
OptionGroup .GRP_2
    OptionButton 144,48,84,12, "1st Radio - Group 2"
    OptionButton 144,64,84,12, "2nd Radio - Group 2"
OKButton 24,96,68,20
CancelButton 156,96,52,20
GroupBox 24,36,92,52, "Option Group 1"
GroupBox 136,36,100,52, "Option Group 2"
End Dialog
```

```

    GroupBox 24,4,212,28, "Check Group"
    CheckBox 36,16,76,12, "Check_Box_1", .CHECKBOX_1
    CheckBox 124,16,76,12, "Check_Box_1", .CHECKBOX_2
End Dialog
Dim Dlg1 As Dialog_1
Dlg1.Grp_1 = 0          ' Set 1st button - group 1
Dlg1.Grp_2 = 1          ' Set 2nd button - group 2
button = Dialog ( Dlg1 )
If button = 0 Then Return
MsgBox "Grp1: " + Dlg1.Grp_1 + ", Grp2: " + Dlg1.Grp_2
Dialog Dlg1

End Sub

```

OptionGroup Statement

Use an option group in a dialog for grouping mutually exclusive option buttons.

Format:

OptionGroup *.name*

Related Topics: Begin Dialog, CancelButton, CheckBox, Dialog, DropListBox, GroupBox, ListBox, OKButton, OptionButton, PushButton, Text, TextBox

Example:

(See OptionButton.)

Print # Statement

Write data to a sequential file.

Format:

Print # *filenumber*, [{Spc(*n*) | Tab(*n*)}] [*expressionlist*] [{; | ,}]

The Print statement consists of the following parts:

<u>Part</u>	<u>Description</u>
<i>filenumber</i>	Number used in an Open statement to open a sequential file. This parameter can be any numeric expression that evaluates to the number of an open file. Note that the number sign (#) preceding filenumber is not optional.
Spc(<i>n</i>)	Name of the Basic function optionally used to insert <i>n</i> spaces into the printed output. Multiple use is permitted.
Tab(<i>n</i>)	Name of the Basic function optionally used to tab to the <i>n</i> th column before printing expressionlist. Multiple use is permitted.
<i>expressionlist</i>	Numeric and/or string expressions to be written to the file.
{; ,}	Character that determines the position of the next character printed. A semicolon means the next character is printed immediately after the last character. A comma means the next character is printed at the start of the next print zone. Print zones begin every 14 columns. If neither character is specified, the next character is printed on the next line.

If you omit expressionlist, the Print # statement prints a blank line in the file, but you must include the comma. Because Print # writes an image of the data to the file, you must delimit the data so it is printed correctly. If you use commas as delimiters, Print # also writes the blanks between print fields to the file.

The Print # statement usually writes Variant data to a file the same way it writes any other data type; however, there are some exceptions:

- If the data being written is a Variant of VarType 0 (Empty), Print # writes nothing to the file for that data item.
- If the data being written is a Variant of VarType 1 (Null), Print # writes the literal #NULL# to the file.
- If the data being written is a Variant of VarType 7 (Date), Print # writes the date to the file using the Short Date format defined in the WIN.INI file. When either the date or the time component is missing or zero, Print # writes only the part provided to the file.

Related Topics: Close, EOF, Input, Line Input, Open, Write #

The following example writes data to a test file:

```

Sub Main
    Dim I, FNum, FName           ' Declare variables.
    For I = 1 To 3
        FNum = FreeFile         ' Determine next file number.
        FName = "TEST" & FNum
        Open FName For Output As FNum ' Open file.
        Print #I, "This is test #" & I ' Write string to file.
        Print #I, "Here is another "; "line"; I
    Next I
    Close                       ' Close all files.
End Sub

```

The following example writes data to a test file and reads it back.

```

Sub Main ()
    Dim FileData, Msg, NL       ' Declare variables.
    NL = Chr(10)                ' Define newline.
    Open "TESTFILE" For Output As #1 ' Open to write file.
    Print #2, "This is a test of the Print # statement."
    Print #2

    Print #2, "Zone 1", "Zone 2" ' Print blank line to file.
    Print #2, "With no space between" ; "." ' Print in two print zones.
    Print #2, "With no space between" ; "." ' Print two strings together.

    Close
    Open "TESTFILE" for Input As #2 ' Open to read file.
    Do While Not EOF(2)
        Line Input #2, FileData ' Read a line of data.
        Msg = Msg & FileData & NL ' Construct message.
        MsgBox Msg
    Loop
    Close ' Close all open files.
    MsgBox "Testing Print Statement" ' Display message.
    Kill "TESTFILE" ' Remove file from disk.
End Sub

```

Print Statement

Print a string to the default printer (specified by the user in the Windows Control Panel).

Format:

Print *expression*

Print text on the current printer at the current x, y coordinates. The x and y coordinates are set using the PrintMoveTo subroutine.

Use the following technique for printing text and graphics to the default printer:

- Send text and graphics to the Printer Object and print them using the PrintNewPage and PrintEndDoc subroutines.

The Printer Object is a device-independent drawing space that supports Print, PrintBeginDoc, PrintEndDoc, PrintMoveTo, PrintNewPage, PrintPageHeight, PrintPageWidth, PrintRect, PrintSetFont, PrintSetFontSize, PrintSetFontStyle, PrintTextHeight and PrintTextWidth subroutines. When you finish placing the information on the Printer Object, you use the PrintEndDoc or PrintNewPage subroutines to send the output to the printer.

Related Topics: PrintBeginDoc, PrintEndDoc, PrintMoveTo, PrintNewPage, PrintPageHeight, PrintPageWidth, PrintRect, PrintSetFont, PrintSetFontSize, PrintSetFontStyle, PrintTextHeight, PrintTextWidth

Example:

```

Sub Main()
    dim x as integer
    dim LineHeight as integer
    dim Margin as integer

    PrintBeginDoc
    ' Set printer font
    PrintSetFont "Courier New"
    PrintSetFontStyle fsBold
    ' Calculate margin as one inch
    Margin = PrintPageHeight / 11

```

```

' Calculate line height based on current font
LineHeight = PrintTextHeight("X")

'Print a title
PrintMoveTo Margin, Margin - LineHeight
Print "List of Orders for Account " + GetScreenText(10, 5, 9)

'Print order ids and amounts
PrintSetFontStyle fsNormal
PrintMoveTo Margin, Margin
For x = 1 to 13
    PrintMoveTo Margin, Margin + x * LineHeight
    Print GetScreenText(18, 7 + Str(x), 11), _
        GetScreenText(44, 7 + Str(x), 12)
Next x

' Print a box around orders ids and amounts
PrintRect Margin, Margin, Margin + (Margin * 6), _
    Margin + (LineHeight * (x + 1)), 5
PrintEndDoc
MsgBox "Printing Complete", MB_ICONINFORMATION
End Sub

```

PrintBeginDoc Subroutine (eQuate)

Initialize the Printer Object (page) context.

Format:

```
PrintBeginDoc
```

Related Topics: Print, PrintEndDoc, PrintMoveTo, PrintNewPage, PrintPageHeight, PrintPageWidth, PrintRect, PrintSetFont, PrintSetFontSize, PrintSetFontStyle, PrintTextHeight, PrintTextWidth

Example:

(See Print Statement).

PrintDlg Function (eQuate)

Display a standard print dialog.

Format:

```
PrintDlg()
```

This function returns an integer. Returns True (1) if OK, else returns False (0).

Related topics: PrintDraw Subroutine (eQuate), PrintSetOrientation Subroutine (eQuate)

PrintDraw Subroutine (eQuate)

Draw an image from the named image control on the print page within the specified boundary. The image is stretched to fit the boundary.

Format:

```
PrintDraw ImageCtlName, left, top, right, bottom
```

The *ImageCtlName* parameter is any string expression containing a valid image control name. The *left*, *top*, *right*, *bottom* and *width* parameters are any integer expression and are specified in pixels.

To maintain the aspect ratio of image, *bottom* or *right* may be set to zero (0) indicating the following. If *right* is set to zero, the image will be scaled for height only and the width will be scaled proportionally. If *bottom* is set to zero, the image will be scaled for width only and the height will be scaled proportionally.

Related topics: PrintDlg Function (eQuate), PrintSetOrientation Subroutine (eQuate)

PrintEndDoc Subroutine (eQuate)

Terminate printing. If print is in the current printer context, it is printed.

Format:

```
PrintEndDoc
```

Related Topics: Print, PrintBeginDoc, PrintMoveTo, PrintNewPage, PrintPageHeight, PrintPageWidth, PrintRect, PrintSetFont, PrintSetFontSize, PrintSetFontStyle, PrintTextHeight, PrintTextWidth

Example:

(See Print Statement).

PrintMoveTo Subroutine (eQuate)

Move the current x- and y-coordinates of the print object. The new x- and y-coordinates will be used as the upper left position of text printed using the Print statement.

Format:

`PrintMoveTo x-coordinate, y-coordinate`

The *x-coordinate* and *y-coordinate* parameters are any integer expression.

Related Topics: Print, PrintBeginDoc, PrintEndDoc, PrintNewPage, PrintPageHeight, PrintPageWidth, PrintRect, PrintSetFont, PrintSetFontSize, PrintSetFontStyle, PrintTextHeight, PrintTextWidth

Example:

(See Print Statement).

PrintNewPage Subroutine (eQuate)

Cause the current content of the Printer Object to be printed immediately.

Format:

`PrintNewPage`

Related Topics: Print, PrintBeginDoc, PrintEndDoc, PrintMoveTo, PrintPageHeight, PrintPageWidth, PrintRect, PrintSetFont, PrintSetFontSize, PrintSetFontStyle, PrintTextHeight, PrintTextWidth

Example:

(See Print Statement).

PrintPageHeight Function (eQuate)

Retrieve the current page height in pixels. The pixel height and width of the page will vary depending on the currently selected printer.

Format:

`PrintPageHeight()`

Related Topics: Print, PrintBeginDoc, PrintEndDoc, PrintMoveTo, PrintNewPage, PrintPageWidth, PrintRect, PrintSetFont, PrintSetFontSize, PrintSetFontStyle, PrintTextHeight, PrintTextWidth

Example:

(See Print Statement).

PrintPageWidth Function (eQuate)

Retrieve the current page width in pixels. The pixel height and width of the page will vary depending on the currently selected printer.

Format:

`PrintPageWidth()`

Related Topics: Print, PrintBeginDoc, PrintEndDoc, PrintMoveTo, PrintNewPage, PrintPageHeight, PrintRect, PrintSetFont, PrintSetFontSize, PrintSetFontStyle, PrintTextHeight, PrintTextWidth

Example:

(See Print Statement).

PrintRect Subroutine (eQuate)

Print a rectangle using the specified left, top, right and bottom coordinates, and line width.

Format:

`PrintRect left, top, right, bottom, width`

The *left*, *top*, *right*, *bottom* and *width* parameters are any integer expression and are specified in pixels.

Related Topics: Print, PrintBeginDoc, PrintEndDoc, PrintMoveTo, PrintNewPage, PrintPageHeight, PrintPageWidth, PrintSetFont, PrintSetFontSize, PrintSetFontStyle, PrintTextHeight, PrintTextWidth

Example:

(See Print Statement).

PrintSetFont Subroutine (eQuate)

Set the current print font.

Format:

PrintSetFont *name*

The *name* parameter is any string expression containing a valid font name. If the font does not exist on the user's system, Windows will substitute a default font of the selected printer.

Related Topics: Print, PrintBeginDoc, PrintEndDoc, PrintMoveTo, PrintNewPage, PrintPageHeight, PrintPageWidth, PrintRect, PrintSetFont, PrintSetFontStyle, PrintTextHeight, PrintTextWidth

Example:

(See Print Statement).

PrintSetFontSize Subroutine (eQuate)

Set the size of the current print font in points.

Format:

PrintSetFontSize *size*

The *size* parameter is any integer expression.

Related Topics: Print, PrintBeginDoc, PrintEndDoc, PrintMoveTo, PrintNewPage, PrintPageHeight, PrintPageWidth, PrintRect, PrintSetFont, PrintSetFontStyle, PrintTextHeight, PrintTextWidth

Example:

(See Print Statement).

PrintSetFontStyle Subroutine (eQuate)

Set the style of the current print font.

Format:

PrintSetFontStyle *style*

The *style* parameter is a numeric expression containing a number equal to the sum of all required attributes.

The *style* parameter may be stated as an Integer or a Constant. Available styles are:

<u>Effect</u>	<u>Integer</u>	<u>Constant</u>
Normal	0	fsNormal
Bold	1	fsFontBold
Italic	2	fsFontItalic
Underline	4	fsFontUnderline
Strikethrough	8	fsFontStrikeThru

Styles may be ORed together to create combined effects. For example, to set bold and italic, use "fsBold or fsItalic".

Note: Since variables do not have to be declared prior to first reference, use an **Option Explicit** statement (external to the procedure) when using constants to assure that they are spelled correctly.

Related Topics: Print, PrintBeginDoc, PrintEndDoc, PrintMoveTo, PrintNewPage, PrintPageHeight, PrintPageWidth, PrintRect, PrintSetFont, PrintSetFontSize, PrintTextHeight, PrintTextWidth

Example:

(See Print Statement).

PrintSetOrientation Subroutine (eQuate)

Set print orientation to portrait or landscape.

Format:

PrintSetOrientation *orientation*

The *orientation* parameter is any integer expression, where Portrait = zero (0) and Landscape = one (1).

Related topics: PrintDlg Function (eQuate), PrintDraw Subroutine (eQuate)

PrintTextHeight Function (eQuate)

Retrieve the height in pixels of a specified text string. The text height can be used to determine vertical spacing between lines.

Format:

`PrintTextHeight (textstring)`

The *textstring* parameter is any string expression.

Related Topics: `Print`, `PrintBeginDoc`, `PrintEndDoc`, `PrintMoveTo`, `PrintNewPage`, `PrintPageHeight`, `PrintPageWidth`, `PrintRect`, `PrintSetFont`, `PrintSetFontSize`, `PrintSetFontStyle`, `PrintTextWidth`

Example:

(See `Print Statement`).

PrintTextWidth Function (eQuate)

Retrieve the width in pixels of a specified text string. The text width can be used to determine horizontal spacing.

Format:

`PrintTextWidth (textstring)`

The *textstring* parameter is any string expression.

Related Topics: `Print`, `PrintBeginDoc`, `PrintEndDoc`, `PrintMoveTo`, `PrintNewPage`, `PrintPageHeight`, `PrintPageWidth`, `PrintRect`, `PrintSetFont`, `PrintSetFontSize`, `PrintSetFontStyle`, `PrintTextHeight`

Example:

(See `Print Statement`).

PushButton Statement

Use a push button in a dialog for assigning a button to a command.

Format:

`PushButton starting-x-pos, starting-y-pos, width, height, "caption" .name`

Related Topics: `Begin Dialog`, `CancelButton`, `CheckBox`, `Dialog`, `DropListBox`, `GroupBox`, `ListBox`, `OKButton`, `OptionButton`, `OptionGroup`, `Text`, `TextBox`

Example:

```
Sub Main
  Begin Dialog UserDialog1 60,60, 260, 188, "3", .Enable
    Text 8,10,73,13, "Text Label:"
    TextBox 8, 26, 160, 18, .FText
    CheckBox 8, 56, 203, 16, "Check to display controls", . Chk1
    GroupBox 8, 79, 230, 70, "This is a group box:", .Group
    CheckBox 18,100,189,16, "Check to change button text", .Chk2
    PushButton 18, 118, 159, 16, "File History", .History
    OKButton 177, 8, 58, 21
    CancelButton 177, 32, 58, 21
  End Dialog

  Dim Dlg1 As UserDialog1
  x = Dialog( Dlg1 )
End Sub

Function Enable( ControlID$, Action%, SuppValue%)

  Begin Dialog UserDialog2 160,160, 260, 188, "3", .Enable
    Text 8,10,73,13, "New dialog Label:"
    TextBox 8, 26, 160, 18, .FText
    CheckBox 8, 56, 203, 16, "New CheckBox", .ch1
    CheckBox 18,100,189,16, "Additional CheckBox", .ch2
    PushButton 18, 118, 159, 16, "Push Button", .but1
    OKButton 177, 8, 58, 21
    CancelButton 177, 32, 58, 21
  End Dialog
  Dim Dlg2 As UserDialog2
  Dlg2.FText = "Your default string goes here"

  Select Case Action%

  Case 1
```

```

    DlgEnable "Group", 0
    DlgVisible "Chk2", 0
    DlgVisible "History", 0
Case 2
    If ControlID$ = "Chk1" Then
        DlgEnable "Group"
        DlgVisible "Chk2"
        DlgVisible "History"
    End If

    If ControlID$ = "Chk2" Then
        DlgText "History", "Push to display nested dialog"
    End If

    If ControlID$ = "History" Then
        Enable =1
        x = Dialog( Dlg2 )
    End If

Case Else

End Select
Enable =1

End Function

```

Put Statement

Write to a disk file from a variable.

Format:

Put [#] *filename*, [*recordnumber*,] *variablename*

The Put statement has three parts:

<u>Parameter</u>	<u>Description</u>
<i>filename</i>	The number used to open the file.
<i>recordnumber</i>	For files opened in Binary mode, <i>recordnumber</i> is the byte position where writing starts.
<i>variablename</i>	The name of the variable containing the data to be written to the file.

Related Topics: Open, Get

Randomize Statement

Initialize the random number generator.

Format:

Randomize [*number*]

The Randomize statement has one optional parameter: *number*. This parameter can be any valid number and is used to initialize the random number generator. If you omit the parameter, then the value returned by the Timer function is used as the default parameter to seed the random number generator.

Example:

```

Sub Main

    Dim MValue

    Randomize ' Initialize random-number generator.
    MValue = Int((6 * Rnd) + 1)
    Print MValue

End Sub

```

ReDim Statement

Declare dynamic arrays and reallocate storage space.

Format:

`ReDim varname (subscripts) [As type][, varname(subscripts) [As type]] ...`

The ReDim statement is used to size or resize a dynamic array that has already been declared using the Dim statement with empty parentheses. You can use the ReDim statement to repeatedly change the number of elements in an array, but not to change the number of dimensions in an array or the type of the elements in the array.

Related Topics: Dim, Option Base, Set, Static

Example:

```
Sub Main

    Dim TestArray() As Integer
    Dim I
    ReDim TestArray(10)
    For I = 1 To 10
        TestArray(I) = I + 10
        Print TestArray(I)
    Next I

End Sub
```

Rem Statement

Include explanatory remarks in a program.

Format:

`Rem remark`

Or anywhere after a statement on a line:

`' remark`

The *remark* parameter is the text of any comment you wish to include in the code.

Example:

```
Rem This is a remark

Sub Main()

    Dim Answer, Msg          ' Declare variables.
    Do
        Answer = InputBox("Enter a value from 1 to 3.")
        Answer = 2
        If Answer >= 1 And Answer <= 3 Then      ' Check range.
            Exit Do                               ' Exit Do...Loop.
        Else
            Beep                                  ' Beep if not in range.
        End If
    Loop
    MsgBox "You entered a value in the proper range."
End Sub
```

Right Function

Return the right most *number* of characters of the string parameter.

Format:

`Right (string, number)`

The *string* parameter is the string expression from which the rightmost characters are returned.

The *number* parameter is the numeric expression indicating the number of characters that will be returned.

Related Topics: Left, Len, Mid

Example:

```
' The example uses the Right function to return the first
' of two words input by the user.
```

```

Sub Main ()
    Dim LWord, Msg, RWord, SpcPos, UsrInp ' Declare variables
    Msg = "Enter two words separated by a space."
    UsrInp = InputBox(Msg) ' Get user input
    SpcPos = InStr(1, UsrInp, " ") ' Find space
    If SpcPos Then
        LWord = Left(UsrInp, SpcPos - 1) ' Get left word
        RWord = Right(UsrInp, Len(UsrInp) - SpcPos) ' Get right word
        Msg = "The first word you entered is <" & LWord & ">"
        Msg = Msg & RWord & "."
    Else
        Msg = "You didn't enter two words."
    End If
    MsgBox Msg ' Display message
End Sub

```

Rmdir Statement

Remove an existing directory.

Format:

Rmdir *path*

The *path* parameter is a string that is the name of the directory to be removed.

Related Topics: ChDir, ChDrive, CurDir, Dir, Mkdir

Example:

```

' This sample shows the functions mkdir (Make Directory)
' and rmdir (Remove Directory)

Sub Main
    Dim dirName As String

    dirName = "t1"
    mkdir dirName
    mkdir "t2"
    MsgBox "Directories: t1 and t2 created. Press OK to " _
        & "remove them"
    rmdir "t1"
    rmdir "t2"
End Sub

```

Rnd Function

Return a random number.

Format:

Rnd [(*number*)]

The *number* parameter must be a valid numeric expression.

The Rnd function returns a Single value less than one (1) but greater than or equal to zero (0).

The value of *number* determines how Rnd generates a random number:

<u>Value of <i>number</i></u>	<u>Number returned</u>
< 0	The same number every time as determined by <i>number</i> .
> 0	The next random number in the sequence.
= 0	The number most recently generated.
<i>number</i> omitted	The next random number in the sequence.

Example:

```

' The example uses the Rnd function to simulate rolling a
' pair of dice by generating random values from 1 to 6.

Sub Main ()
    Dim Dice1, Dice2, Msg ' Declare variables.

```

```

Dice1 = CInt(6 * Rnd() + 1)      ' Generate first die value.
Dice2 = CInt(6 * Rnd() + 1)      ' Generate second die value.
Msg = "You rolled a " & Dice1
Msg = Msg & " and a " & Dice2
Msg = Msg & " for a total of "
Msg = Msg & Str(Dice1 + Dice2) & "."
MsgBox Msg                        ' Display message.
End Sub

```

ScreenChanged Function (eQuate)

Return the current state of the Screen Changed Flag (0 = False, 1 = True). The Screen Changed Flag indicates when a user or action has updated the screen. The Screen Changed flag is automatically cleared on receipt of any output from the host.

Format:

ScreenChanged()

Related Topic: ClearScreenChanged

Second Function

Return an integer between 0 and 59 that is the portion of the *time* parameter representing the second of the minute.

Format:

Second (*time*)

The *time* parameter is any string expression that can represent a time.

Related Topics: Date, Day, Format, Hour, Minute, Month, Now, Weekday, Year

Example:

```

MyTime = "08:04:23 PM"
MsgBox MyTime

MsgBox Second( MyTime ) & " Seconds"
MsgBox Minute( MyTime ) & " Minutes"
MsgBox Hour( MyTime ) & " Hours"

```

Seek Function

Return a number that represents the byte position where the next operation is to take place. The first byte in the file is at position 1.

Format:

Seek (*filenumber*)

The *filenumber* parameter is used on an Open statement and must be a valid numeric expression.

Related Topics: Open

Example:

```

Sub Main
  Open "TESTFILE" For Input As #  ' Open file for reading.
  Do While Not EOF(1)             ' Loop until end of file.
    MyChar = Input(1, #1)         ' Read next character of data.
    Print Seek(1)                 ' Print byte position .
  Loop
  Close #1                       ' Close file.
End Sub

```

Seek Statement

Set the position in a file for the next read or write.

Format:

Seek *filenumber*, *position*

The *filenumber* parameter is used in the open statement and must be a valid numeric expression. The *position* parameter is the number that indicates where the next read or write is to occur. The *position* parameter is the byte position relative to the beginning of the file.

Related Topics: Open

Example:

```
Sub Main
  Open "TESTFILE" For Input As #1 ' Open file for reading.
  For i = 1 To 24 Step 3           ' Loop until end of file.

    Seek #1, i                    ' Seek to byte position
    MyChar = Input(1, #1)         ' Read next character of data.
    Print MyChar                  ' Print character of data.
  Next i
  Close #1                        ' Close file.
End Sub
```

Select Case Statement

Execute one of the sets of statement(s) in the case, based on the test variable.

Format:

```
Select Case testexpression
  [Case expressionlist1
    [statement(s)]]
  [Case expressionlist2
    [statement(s)]]
  [Case Else
    [statement(s)]]
End Select
```

The Select Case statement has these parts:

<u>Part</u>	<u>Description</u>
Select Case	Begins the Select Case decision control sequence.
<i>testexpression</i>	Any numeric or string expression. If <i>testexpression</i> matches the <i>expressionlist</i> associated with the Case clause, the <i>statement(s)</i> following the Case clause are executed.
Case	Sets apart a group of statements to be executed if an expression in <i>expressionlist</i> matches the <i>testexpression</i> .
<i>expressionlist</i>	The <i>expressionlist</i> consists of a comma-delimited list of one or more of the following forms: <i>expression</i> <i>expression</i> To <i>expression</i> Is <i>compare-operator</i> <i>expression</i>
<i>statement(s)</i>	Any number of statements on one or more lines.
Case Else	Begins the statement(s) to be executed if no match is found between the <i>testexpression</i> and an <i>expressionlist</i> in any of the other Case selections.
End Case	Ends the Select Case.

The *expression* parameter may be any numeric or string expression; however, it must be compatible with the type of *testexpression*.

The *compare-operator* may be any valid comparison operator, except Is and Like.

Related Topics: If...Then...Else

See also, Data Types, Operators and Precedences

Example:

```
Sub Test ()
  For x = 1 to 5
    print x
    Select Case x
      Case 2
        Print "Outer Case Two"
      Case 3
        Print "Outer Case Three"
    '
    Exit For
    Select Case x
      Case 2
```

```

        Print "Inner Case Two"
    Case 3
        Print "Inner Case Three"
        Exit For
    Case Else
        ' Must be something else.
        Print "Inner Case Else:", x
    End Select

    Print "Done with Inner Select Case"
    Case Else
        ' Must be something else.
        Print "Outer Case Else:",x
    End Select
Next x
Print "Done with For Loop"
End Sub

```

Send Subroutine (eQuate)

Send a specified string to the host. The string is not sent to the host until the current action is exited.

Format:

Send *textstring*

The *textstring* parameter is any string expression.

Example:

```

' Send a query using the current order id.
Send "CUST5 " + Curr_Order_Id

```

SendKey Subroutine (eQuate)

Send a specified function key sequence to the host. The function key is not sent to the host until the current action is exited. This subroutine is NOT related to the SendKeys function.

Format:

SendKey *keynumber*

The *keynumber* parameter is any integer expression. The *keynumber* parameter corresponds to the function key number. For example, on a UTS terminal key numbers 1 through 22 correspond to function keys F1 through F22.

Key number 0 corresponds to the MsgWait key on a UTS terminal.

Example:

```

Sub BTN_RESUME()
    SendKey 1 ' Send UTS F1 to host
End Sub

```

SendKeys Statement

Send one or more keystrokes to the active window as if they had been entered at the keyboard.

Format:

SendKeys *keys*

The *keys* parameter is a string and is sent to the active window.

To send a single keyboard character, use the character itself. To send the letter A, use "A". To send multiple keyboard characters, one behind the other, include them in the string in the order you want them sent. To send a D followed by an E and then followed by an F, use "DEF".

Ten keyboard characters have special significance when used with the SendKeys statement:

<u>Character(s)</u>	<u>Usage</u>
Braces { }	Used to enclose a special character or key name being sent. For example, {F4} sends function key 4.
Plus sign +	The SHIFT key.
Caret ^	The CTRL key.
Percent sign %	The ALT key.
Tilde ~	The ENTER key.

<u>Character(s)</u>	<u>Usage</u>
Parentheses ()	Used to enclose multiple keystrokes in combination with the SHIFT, CTRL and ALT keys. For example, "%(EF)" would be the same as holding down the ALT key while pressing E followed by F.
Brackets []	No special significance but must be enclosed in braces when sent; e.g., "{[]}" and "{}" .

To send any special character, enclose it in braces. For example, "{{" sends an open brace and "{+}" sends a plus sign.

To send keys that do not display when you press them, use the following substitution codes:

<u>Key</u>	<u>Substitution Code</u>
BACKSPACE	{BACKSPACE}
BREAK	{BREAK}
CAPS LOCK	{CAPSLOCK}
DEL	{DELETE} or {DEL}
DOWN ARROW	{DOWN}
END	{END}
ENTER	{ENTER} or ~
ESC	{ESC}
HELP	{HELP}
HOME	{HOME}
INS	{INSERT}
LEFT ARROW	{LEFT}
NUM LOCK	{NUMLOCK}
PAGE DOWN	{PGDN}
PAGE UP	{PGUP}
RIGHT ARROW	{RIGHT}
SCROLL LOCK	{SCROLLLOCK}
TAB	{TAB}
UP ARROW	{UP}
F1	{F1}
F2	{F2}
:	:
F16	{F16}

To repeat a key, follow the key by the number of times to repeat the keystroke. For example, "{UP 10}" is the same as pressing the UP ARROW 10 times. Note: A space is required between the key and the number.

Example:

```

Sub Main ()
    Dim I, X, Msg          ' Declare variables.
    X = Shell("Calc.exe", 1) ' Shell Calculator.
    For I = 1 To 5          ' Set up counting loop.
        SendKeys I & "{+}"
                                ' Send keystrokes to Calculator.
    Next I                  ' to add each value of I.

    Msg = "Choose OK to close the Calculator."
    MsgBox Msg              ' Display OK prompt.
    AppActivate "Calculator" ' Return focus to Calculator.
    SendKeys "%{F4}"        ' Alt+F4 to close Calculator.
End Sub

```

SendMail Subroutine (eQuate)

Create an e-mail message in an eQuate form's Action.

Format:

SendMail Recipients, Subject, CcRecipients, BccRecipients, MessageText, Attachments, NoPrompt

The *Recipients, Subject, CcRecipients, BccRecipients, MessageText* and *Attachments* parameters are any string expressions.

The *Recipients*, *CcRecipients*, *BccRecipients*, *MessageText* and *Attachments* are actually lists of strings allowing multiple items. Items must be separated by semicolons. For example, to send to two recipients, you use the following: "Alice@kmsys.com;Ralph@kmsys.com".

Empty parameters must be entered as "" (zero length strings).

When the Mail is sent, the user's e-mail handler will allow the message to be reviewed before sending. If the user does not send the message or an error occurs, a message box containing the message "Message Not Sent" will pop up.

NoPrompt is an integer value (which is also a Boolean in VB). If *NoPrompt* is non-zero (True) the mail is sent without the review prompt.

Example:

```
Sub BtnSendMail()
    SendMail
    "Ralph@KMSys.com", "TextSubject", "Alice@KMSys.Com;Steve@xys.com", "", "This
    is the message text line 1;line two", "C:\Docs\MyFile.txt", True
End Sub
```

Set Statement

Assign an object to an object variable.

Format:

Set *objectvar* = {[New] *objectexpression* | Nothing}

Related Topics: Dim, Global, Static

Example:

```
Sub Main
    Dim visio As Object
    Set visio = CreateObject( "visio.application" )
    Dim draw As Object
    Set draw = visio.Documents
    draw.Open "c:\visio\drawings\Sample1.vsd"
    MsgBox "Open docs: " & draw.Count
    Dim page As Object
    Set page = visio.ActivePage
    Dim red As Object
    Set red = page.DrawRectangle (1, 9, 7.5, 4.5)
    red.FillStyle = "Red fill"

    Dim cyan As Object
    Set cyan = page.DrawOval (2.5, 8.5, 5.75, 5.25)
    cyan.FillStyle = "Cyan fill"

    Dim green As Object
    Set green = page.DrawOval (1.5, 6.25, 2.5, 5.25)
    green.FillStyle = "Green fill"

    Dim DarkBlue As Object
    set DarkBlue = page.DrawOval (6, 8.75, 7, 7.75)
    DarkBlue.FillStyle = "Blue dark fill"

    visio.Quit
End Sub
```

SetColor Subroutine (eQuate)

Set either the foreground or the background color of a data field or eQuate control.

Format:

SetColor-*name*, *colortype*, *color*

The *name* parameter is a string expression containing the name of a valid eQuate data field or eQuate control. The *colortype* is an integer expression specifying the property type for which the color is to be set. The *color* is a long integer expression designating the desired color. Both the *colortype* and *color* are expressed as Constants.

Valid *colortype* entries are:

<u>Constant</u>	<u>Description</u>
tpForeColor	The color of text or box border.
tpBackColor	The color of control's background area.

Valid *color* entries are:

<u>Constant</u>	<u>Description</u>
clBlack	Black
clMaroon	Maroon
clGreen	Green
clOlive	Olive green
clNavy	Navy blue
clPurple	Purple
clTeal	Teal
clGray	Gray
clSilver	Silver
clYellow	Yellow
clRed	Red
clLime	Lime green
clBlue	Blue
clFuchsia	Fuchsia
clAqua	Aqua
clWhite	White

The following colors refer to the default colors set in the current Windows environment:

clScrollBar	Current color of Windows scrollbar.
clBackground	Current color of Windows background.
clActiveCaption	Current color of the title bar of the active window.
clInactiveCaption	Current color of the title bar of inactive windows.
clMenu	Current background color of menus.
clWindow	Current background color of windows.
clWindowFrame	Current color of window frames.
clMenuText	Current color of text on menus.
clWindowText	Current color of text in windows.
clCaptionText	Current color of the text on the title bar of the active window.
clActiveBorder	Current border color of the active window.
clInactiveBorder	Current border color of inactive windows.
clAppWorkSpace	Current color of the application workspace.
clHighlight	Current background color of selected text.
clHightlightText	Current color of selected text.
clBtnFace	Current color of a button face.
clBtnShadow	Current color of a shadow cast by a button.
clGrayText	Current color of text that is dimmed.
clBtnText	Current color of text on a button.
clInactiveCaptionText	Current color of the text on the title bar of an inactive window.
clBtnHighlight	Current color of the highlighting on a button.

Note: Since variables do not have to be declared prior to first reference, use an **Option Explicit** statement (external to the procedure) when using constants to assure that they are spelled correctly.

Related Topics: GetColor

Example:

```
Dim clr as long
Clr = GetColor("ACCOUNT", tpForeColor)
If Clr = clWindowText Then
    SetColor "ACCOUNT", tpForeColor, clBlue
End If
```

SetCurrentLanguage Function (eQuate)

Set the current translation language using the two-character language Id.

Format:

Function SetCurrLanguage (ByVal *LangId* As String) As Integer

If SetCurrLanguage fails it returns 0 otherwise it returns 1 indicating success.

Related Topics: eQuate 3.5 Language Translations

Related Topics: SetCursor Subroutine (eQuate)

Set the column and row position of the cursor within the logical screen.

Format:

SetCursor *col*, *row*

Related Topics: GetCursorCol, [GetCursorRow](#)

SetCursorField Subroutine (eQuate)

This is a synonym for SetFocus. See the SetFocus Subroutine for information.

SetExecState Subroutine (eQuate)

Set the eQuate Action execution state (*TRUE*/*FALSE*).

Format:

SetExecState *value*

Any non-zero *value* is assumed to indicate True, while zero indicates False.

SetFocus Subroutine (eQuate)

Set the windows focus to the specified data field or eQuate control. The only eQuate control types that may receive focus are: data field, command button, standard list box, drop-down list box, multi-column list box, check box or option button. The value returned by GetCursorField is only affected when a data field is given focus.

Format:

SetFocus *name*

The *name* parameter is any string expression containing a valid eQuate data field name or eQuate control name.

Related Topics: GetCursorField

Example:

This example sets the focus to either a data field edit box or a command button depending on whether an option button is currently checked.

```
If GetState(OPT_1, tpChecked) then
    SetFocus "ACCOUNT"
Else
    SetFocus "BTN_FIND_ACCOUNTS"
End If
```

SetMemoSelection Subroutine (eQuate)

Position the selection in a memo control. Please notice that memo selection does not use lines. It uses positions, as if the memo is one long block of text.

Format:

SetMemoSelection *name*, *starting-pos*, *length*

The *name* parameter is any string expression containing the name of a valid eQuate memo control. The *starting-pos* parameter is any integer expression representing the zero-relative position within the memo control. The *length* parameter is any integer expression that represents the length of the memo text.

Related Topic: GetMemoSelection

Example:

To move the selection the start of a memo:

```
SetMemoSelection "Memo_1", 0, 0
```

SetNumericProp Subroutine (eQuite)

Set the integer property to the value specified. Numeric properties include Height, Width, Left, Top, TabOrder, etc.

Format:

`SetNumericProp name, property, value`

The *name* parameter is any string expression containing the name of the control. The *property* parameter is any string expression containing the name of the property. The *value* parameter is any integer expression.

Related Topics: GetNumericProp, GetStringProp, SetStringProp

Example:

```
Option Explicit

Sub Btn_Load()
' Load some items into the list

    ListItemAdd "MCList", "1234, Spring, 5.12"
    ListItemAdd "MCList", "2312, Nut, .02"
    ListItemAdd "MCList", "3334, Bolt, .12"
    ListItemAdd "MCList", "2230, TBolt, .35"
    ListItemAdd "MCList", "9324, Flange, 1.23"
    ListItemAdd "MCList", "0534, Washer, 1.10"
    ListItemAdd "MCList", "7801, Lock Washer, 1.77"
    ListItemAdd "MCList", "0725, Lock Washer 2, -1.05"
    ListItemAdd "MCList", "3012, Lock Washer 3, -1.77"
    ListItemAdd "MCList", "3012, Trailing Sign Invalid, 1.77-"

End Sub

Sub Sb_SortPartNo()
' Change sort the first column and an alpha sort
    SetNumericProp "MCList", "SortColumn", 0
    SetNumericProp "MCList", "NumericSort", 0
End Sub

Sub Sb_SortDesc()
' Change sort the second column and an alpha sort
    SetNumericProp "MCList", "SortColumn", 1
    SetNumericProp "MCList", "NumericSort", 0
End Sub

Sub Sb_SortPrice()
' Change sort the third column and a numeric sort
    SetNumericProp "MCList", "SortColumn", 2
    SetNumericProp "MCList", "NumericSort", 1
End Sub

Sub Chk_Desc()
' Switch between ascending and descending sort.
    SetNumericProp "MCList", "SortDescending", GetState("Chk_Desc", tpChecked)
End Sub
```

SetPage Subroutine (eQuate)

Open a new page. The current page remains open, but the new page gains focus. This subroutine only applies when developing applications utilizing T27 connections.

Format:

`SetPage pagenum`

The *pagenum* parameter is any integer expression containing a page number.

SetScreenData Subroutine (eQuate)

Set the string value of an area within the logical screen. This subroutine will set text regardless of protected FCCs in the screen.

Format:

```
SetScreenData col, row, len, data
```

The *col*, *row* and *len* parameters are any integer expression. The *data* parameter is any string expression.

If *row* is specified as -1, then *col* is assumed to be the offset from the beginning of the logical screen buffer. For Example:

```
SetScreenData 5, 2, 5
```

Is the same as:

```
SetScreenData 85, -1, 5
```

The above example assumes the screen has 80 columns.

Related Topics: GetScreenData, GetScreenLine

Example:

```
' Put the trans code in the screen
SetScreenData 1, 1, 6, "CUST1 "
```

SetSessionVar Subroutine (eQuate)

Set the contents of an eQuate Global Session Variable. If the named variable has not yet been defined, eQuate will define it. Any changes to eQuate Global Session Variables remain available to all forms within the application session. eQuate Global Session Variables provide a mechanism to pass data between forms within the application session.

Format:

```
SetSessionVar name, value
```

The *name* parameter is any string expression containing the session variable. The *value* parameter is the string expression to be assigned to the named session variable.

Related Topics: GetSessionVar

Example:

```
' Save the current account in a session variable.
SetSessionVar "SAVE_ACCOUNT", GetString("ACCOUNT")
```

SetState Subroutine (eQuate)

Set the specified state of data fields or controls.

Format:

```
SetState name, statetype, state
```

The *name* parameter is a string expression containing an eQuate Data Field Name or an eQuate Control Name.

The *statetype* parameter is an integer expression specifying the property type for which the state is to be retrieved. The *state* parameter is an integer expression containing either **True** (non-zero) or **False** (zero). The *statetype* parameter may be expressed as an Integer or a Constant. Valid *statetype* entries are:

<u>Effect</u>	<u>Integer</u>	<u>Constant</u>
Enabled	0	tpEnabled
Visible*	1	tpVisible
Checked**	2	tpChecked

* Visible does not apply to eQuate User Menu Items.

** Checked only applies to eQuate Option Button and Check Box controls.

Note: Since variables do not have to be declared prior to first reference, use an **Option Explicit** statement (external to the procedure) when using constants to assure that they are spelled correctly.

Related Topics: GetState

Example:

(see GetState).

SetString Subroutine (eQuate)

Set an eQuate Data Field, eQuate Control or eQuate Global Session Variable string value. For data fields, the data text is set. For eQuate controls like command buttons and labels, the caption text is set. For list boxes, the currently selected line is set.

Format:

```
SetString name, value
```

The *name* parameter is any string expression containing the name of a valid eQuate Data Field Name, eQuate Control Name or eQuate Global Session Variable. The *value* parameter is the string expression to be assigned to the named data field, control or session variable.

Related Topics: GetString

Example:

(see GetString).

SetStringProp Subroutine (eQuate)

Set the string property to the value specified. String properties include Hints, Captions, Pictures, etc.

Format:

SetStringProp *name*, *property*, *value*

The *name* parameter is any string expression containing the name of the control. The *property* parameter is any string expression containing the name of the property. The *value* parameter is any string expression containing the value to be set in the property.

Note: The Caption property of Text Label controls cannot be set using SetStringProp — use the SetString subroutine, instead.

Related Topics: GetNumericProp, GetStringProp, SetNumericProp, SetString

Sgn Function

Return an integer indicating the sign (+, -, 0) of a number.

Format:

Sgn(*number*)

The number parameter can be any valid numeric expression. The **Sgn** function returns the following values:

<u>Value</u>	<u>Condition</u>
1	<i>number</i> > 0
0	<i>number</i> = 0
-1	<i>number</i> < 0

Shell Function

Run an executable program.

Format:

Shell (*app* [,*style*])

The **Shell** function has two parameters. The first one, *app*, is the name of the program to be executed. The name of the program in *app* must include a .PIF, .COM, .BAT or .EXE file extension or an error will occur. The second argument, *style*, is the number corresponding to the style of the window. The second argument is also optional, and if omitted, the program is opened minimized with focus.

<u>Value</u>	<u>Window Style</u>
1, 5, 9	Normal with focus.
2	Minimized with focus (default).
3	Maximized with focus.
4, 8	Normal without focus.
6, 7	Minimized without focus.

Return value: ID, the task ID of the started program.

Example:

```
' This example uses Shell to leave the current application
' and run the Calculator program included with Microsoft
' Windows; it then uses the SendKeys statement to send
' keystrokes to add some numbers.
Sub Main ()
    Dim I, X, Msg           ' Declare variables
    X = Shell("Calc.exe", 1) ' Shell Calculator
    For I = 1 To 5           ' Set up counting loop
        SendKeys I & "+"    ' Send keystrokes to Calculator
    Next I                  ' to add each value of I
    Msg = "Choose OK to close the Calculator."
    MsgBox Msg              ' Display OK prompt
```

```

    AppActivate "Calculator"      ' Return focus to Calculator
    SendKeys "{F4}"              ' Alt+F4 to close Calculator
End Sub

```

ShowFormById Subroutine (eQuate)

Show a new eQuate Form based on its form id. string. The form will be shown until the current action is exited.

Format:

```
ShowFormById formidstring
```

The *formidstring* is any string expression containing a valid eQuate Form Id.

ShowFormByName Subroutine (eQuate)

Show a new eQuate Form. The Form will be shown until the current action is exited.

Format:

```
ShowFormByName formname
```

The *formname* is any string expression containing a valid eQuate Form Name.

```
End Sub
```

ShowHelp Subroutine (eQuate)

Show specified help context.

Format:

```
ShowHelp HelpContextId
```

If *HelpContextId* is 0, the help contents section is shown; if -1, the help index is shown; otherwise, the specified help context is shown. The current form's WinHelpFile is used.

ShowPickList Function (eQuate)

Causes the Pick List dialog to be executed for a specified pick list

Format:

```
ShowPickList (PickListName)
```

This function returns a String.

PickListName must be the name of a valid pick list. The function returns the string value of the item selected by the user. If the user cancels the dialog a empty (zero length) string is returned. If an invalid pick list name is specified, the dialog will show an empty list and the return will be an empty string regardless of the user's choice.

Example:

```

' Show an pick list and place the result into an edit box.
SetString "Edit_1", ShowPickList("States")

```

Sin Function

Return the sine of an angle that is expressed in radians.

Format:

```
Sin (radian)
```

Example:

```

Sub Main ()
    pi = 4 * Atn(1)
    rad = 90 * (pi/180)
    x = Sin(rad)
    print x
End Sub

```

Space Function

Skip a specified number of spaces in a Print # statement.

Format:

```
Space (number)
```

The *number* parameter can be any valid integer and determines the number of blanks.

Example:

```
Sub Main
    MsgBox "Hello" & Space(20) & "There"
End Sub
```

Sqr Function

Return the square root of a number.

Format:

Sqr (number)

The *number* parameter must be a valid number greater than or equal to zero.

Example:

```
Sub Form_Click ()
    Dim Msg, Number          ' Declare variables.
    Msg = "Enter a non-negative number."
    Number = InputBox(Msg)    ' Get user input.
    If Number < 0 Then
        Msg = "Cannot determine the square root of a " _
            & "negative number."
    Else
        Msg = "The square root of " & Number & " is "
        Msg = Msg & Sqr(Number) & "."
    End If
    MsgBox Msg                ' Display results.
End Sub
```

Static Statement

Declare variables and allocate storage space. These variables will retain their value through the program run.

Format:

Static variable

Related Topics: Dim, Function, Sub

Example:

```
' This example shows how to use the static keyword to
' retain the value of the variable i in sub Joe. If Dim is
' used instead of Static then i is empty when printed on
' the second call as well as the first.

Sub Main
    For i = 1 to 2
        Joe 2
    Next i
End Sub
Sub Joe( j as integer )
    Static i
    print i
    i = i + 5
    print i
End Sub
```

Stop Statement

End the execution of the program.

Format:

Stop

The **Stop** statement can be placed anywhere in your code.

Related Topics: End, Exit

Example:

```
Sub main ()
    Dim x,y,z
```

```

    For x = 1 to 5
      For y = 1 to 5
        For z = 1 to 5
          Print "Looping" ,z,y,x
        Next z
      Next y
    Next x
  End Sub

```

Str Function

Return the value of a numeric expression.

Format:

Str (numericexpression)

Str returns a String.

Use the Format function to convert numeric values you want formatted as dates, times or in other user-defined formats.

The Str function recognizes only the period (.) as a valid decimal separator. When a possibility exists that different decimal separators may be used (e.g., in international applications), you should use CStr to convert a number to a string.

Related topics: CStr, Format, Val

Example:

```

Sub main ()
  Dim msg
  a = -1
  MsgBox "Num = " & Str(a)
  MsgBox "_Abs(Num) =" & Str(_Abs(a))

End Sub

```

StrComp Function

Return a variant that is the result of the comparison of two strings.

Format:

StrComp(string1,string2,[compare])

Example:

```

Sub Main

  Dim MStr1, MStr2, MComp
  MStr1 = "ABCD": MStr2 = "today"      ' Define variables.
  print MStr1, MStr2
  MComp = StrComp(MStr1, MStr2) ' Returns -1.
  print MComp
  MComp = StrComp(MStr1, MStr2) ' Returns -1.
  print MComp
  MComp = StrComp(MStr2, MStr1) ' Returns 1.
  print MComp

End Sub

```

String Function

String is used to create a string that consists of one character repeated repeatedly.

Formats:

String (numeric, charcode)

or

String (numeric, string)

String returns a string.

Related topics: Space

Example:

```
Sub Main

    Dim MString
    MString = String(5, "*") ' Returns "*****".
    MString = String(5, 42)  ' Returns "44444".
    MString = String(10, "Today") ' Returns "TTTTTTTTTT".
    Print MString
End Sub
```

Sub Statement

Declare and define a Sub procedure name, parameters and code.

Format:

```
Sub subname [(argumentlist)]
    [statement(s)]
        subname = expression
[Exit Sub]
    [statement(s)]
        subname = expression
End Sub
```

When the optional *argumentlist* needs to be passed, the format is as follows:

```
([ByVal] variable [As type][, [ByVal] variable [As type] ]...)
```

The optional ByVal parameter specifies that the *variable* is passed by value instead of by reference (see ByRef and ByVal).

The optional As *type* parameter is used to specify the data type. Valid types are String, Integer, Single, Double, Long and Variant (see Other Data Types).

Related Topics: Call, Dim, Function

Example:

```
Sub Main
    Dim DST As String
    DST = "t1"
    mkdir DST
    mkdir "t2"
End Sub
```

Tan Function

Return the tangent of an angle as a double.

Format:

```
Tan(angle)
```

The *angle* parameter must be a valid angle expressed in radians.

Related Topic: Atn, Cos, Sin

Example:

```
Sub Main ()
    Dim Msg, Pi ' Declare variables.
    Pi = 4 * Atn(1) ' Calculate Pi.
    Msg = "Pi is equal to " & Pi
    MsgBox Msg ' Display results.
    x = Tan(Pi/4)
    MsgBox x & " is the tangent of Pi/4"
End Sub
```

Text Statement

Create a text field for titles and labels.

Format:

```
Text starting-x-pos, starting-y-pos, width, height, label
```

Related Topics: Begin Dialog, CancelButton, CheckBox, Dialog, DropListBox, GroupBox, ListBox, OKButton, OptionButton, OptionGroup, PushButton, TextBox

Example:

```
Sub Main ()
  Begin Dialog DialogName1 60, 60, 160, 70, "ASC - Hello"
    TEXT 10, 10, 28, 12, "Name:"
    TEXTBOX 42, 10, 108, 12, .nameStr
    TEXTBOX 42, 24, 108, 12, .descStr
    CHECKBOX 42, 38, 48, 12, "&CHECKME", .checkInt
    OKBUTTON 42, 54, 40, 12
  End Dialog
  Dim Dlg1 As DialogName1
  Dialog Dlg1

  MsgBox Dlg1.nameStr
  MsgBox Dlg1.descStr
  MsgBox Dlg1.checkInt
End Sub
```

TextBox Statement

Create a Text Box for typing in numbers and text.

Format:

TextBox *starting-x-pos, starting-y-pos, width, height, .default_string*, [32]

The optional string, "32", instructs enable to provide password protection characters as the user types into the text box. The password protection character is the asterisk (*).

Related Topics: Begin Dialog, CancelButton, CheckBox, Dialog, DropListBox, GroupBox, ListBox, OKButton, OptionButton, OptionGroup, PushButton, Text

Example:

```
Sub Main ()
  Begin Dialog DialogName1 60, 60, 160, 70, "ASC - Hello"
    TEXT 10, 10, 28, 12, "Name:"
    TEXTBOX 42, 10, 108, 12, .nameStr
    TEXTBOX 42, 24, 108, 12, .descStr
    CHECKBOX 42, 38, 48, 12, "&CHECKME", .checkInt
    OKBUTTON 42, 54, 40, 12
  End Dialog
  Dim Dlg1 As DialogName1
  Dialog Dlg1

  MsgBox Dlg1.nameStr
  MsgBox Dlg1.descStr
  MsgBox Dlg1.checkInt

End Sub
```

Time Function, Time Statement

Returns the current system time or sets the system time.

Time function returns a value; the Time statement does not.

Function Format:

Time[()]

Statement Format:

Time = *time*

The time parameter is any numeric or string expression that represents a time.

```
x = Time$(Now)
Print x

' Returns current system time in the
' system-defined long time format.
MsgBox Format(Time, "Short Time")
```

```
MyStr = Format(Time, "Long Time")
```

To set the system time use the TIME statement:

```
SysTime = "8:00:00 AM"
Time = SysTime
```

Timer Event

Timer Event is used to track elapsed time. It can also be displayed as a stopwatch in a dialog. The timer's value is the number of seconds from midnight.

Format:

Timer

Related topics: DateSerial, DateValue, Hour, Minute, Now, Second, TimeSerial, TimeValue.

Example:

```
Sub Main

    Dim TS As Single
    Dim TE As Single
    Dim TEL As Single

    TS = Timer

    MsgBox "Starting Timer"

    TE = Timer

    TT = TE - TS
    Print TT

End Sub
```

TimeSerial Function

Return the time serial for the supplied parameters *hour*, *minute*, *second*.

Format:

TimeSerial (*hour*, *minute*, *second*)

Related topics: DateSerial, DateValue, Hour, Minute, Now, Second, TimeValue

Example:

```
Sub Main

    Dim MTime
    MTime = TimeSerial(12, 25, 27)
    Print MTime

End Sub
```

TimeValue Function

Return a double precision serial number based of the supplied string parameter.

Format:

TimeValue (*timestring*)

Midnight = TimeValue("23:59:59")

Related topics: DateSerial, DateValue, Hour, Minute, Now, Second, TimeSerial

Example:

```
Sub Main

    Dim MTime
    MTime = TimeValue("12:25:27 PM")
    Print MTime

End Sub
```

```
End Sub
```

Trim, LTrim, RTrim Functions

Return a copy of a string with leading, trailing or both leading and training spaces removed.

Format:

[L | R]Trim (*string*)

LTrim removes leading spaces. RTrim removes trailing spaces. Trim removes leading and trailing spaces.

Example:

```
' This example uses the LTrim and RTrim functions to strip leading
' and trailing spaces, respectively, from a string variable. It
' uses the Trim function alone to strip both types of spaces.
' LCase and UCase are also shown in this example as well as the
' use of nested function calls

Sub Main
  MyString = " <-Trim-> "          ' Initialize string
  TrimString = LTrim(MyString)      ' TrimString = "<-Trim-> "
  MsgBox "|" & TrimString & "|"
  TrimString = LCase(RTrim(MyString)) ' TrimString = " <-trim->"
  MsgBox "|" & TrimString & "|"
  TrimString = LTrim(RTrim(MyString)) ' TrimString = "<-Trim->"
  MsgBox "|" & TrimString & "|"      ' Using the Trim function
                                   ' alone achieves the same
                                   ' result.
  TrimString = UCase(Trim(MyString)) ' TrimString = "<-TRIM->"
  MsgBox "|" & TrimString & "|"
End Sub
```

Type Statement

Define a user-defined data type containing one or more elements.

Format:

```
Type usertype elementname [(subscripts)] As typename
    [ elementname [(subscripts)] As typename]
```

```
...
```

```
End Type
```

The Type statement has these parts:

<u>Part</u>	<u>Description</u>
Type	Marks the beginning of a user-defined type.
<i>usertype</i>	Name of a user-defined data type. It follows standard variable naming conventions.
<i>elementname</i>	Name of an element of the user-defined data type. It follows standard variable-naming conventions.
<i>subscripts</i>	Dimensions of an array element. You can declare multiple dimensions (not currently implemented).
<i>typename</i>	One of these data types: Integer, Long, Single, Double, String (for variable-length strings), String * length (for fixed-length strings), Variant or another user-defined type. The argument <i>typename</i> cannot be an object type.
End Type	Marks the end of a user-defined type.

Once you have declared a user-defined type using the **Type** statement, you can declare a variable of that type anywhere in your script. Use **Dim** or **Static** to declare a variable of a user-defined type. Line numbers and line labels are not allowed in **Type...End Type** blocks.

User-defined types are often used with data records because data records frequently consist of a number of related elements of different data types. Arrays cannot be an element of a user-defined type.

Example:

```
' This sample shows some of the
' features of user defined types.
```

```

Type type1
    a As Integer
    d As Double
    s As String
End Type

Type type2
    a As String
    o As type1
End Type
Type type3
    b As Integer
    c As type2
End Type

Dim type2a As type2
Dim type2b As type2
Dim type1a As type1
Dim type3a As type3

Sub Form_Click ()
    a = 5
    type1a.a = 7472
    type1a.d = 23.1415
    type1a.s = "YES"
    type2a.a = "43 - forty three"
    type2a.o.s = "Yaba Daba Doo"
    type3a.c.o.s = "COS"
    type2b.a = "943 - nine hundred and forty three"
    type2b.o.s = "Yogi"
    MsgBox type1a.a
    MsgBox type1a.d
    MsgBox type1a.s
    MsgBox type2a.a
    MsgBox type2a.o.s
    MsgBox type2b.a
    MsgBox type2b.o.s
    MsgBox type3a.c.o.s
    MsgBox a
End Sub

```

UBound Function

Return the value of the largest usable subscript for the specified dimension of an array.

Format:

Ubound (*arrayname* [, *dimension*])

Related Topics: Dim, Global, Lbound, Option Base, Static

Example:

```

' This example demonstrates some of the features of
' arrays. The lower bound for an array is 0 unless it is
' specified or Option Base has set it as is done in this
' example.

Option Base 1

Sub Main
    Dim a(10) As Double
    MsgBox "LBound: " & LBound(a) & " UBound: " & UBound(a)
    Dim i As Integer
    For i = 1 to 3
        a(i) = 2 + i
    Next i
    Print a(1), a(1), a(2), a(3)

```

```
End Sub
```

UCase Function

Return a copy of a string in which all lowercase characters have been converted to uppercase.

Format:

UCase (*string*)

Related Topics: LCase

Example:

```
' This example uses the LTrim and RTrim functions to strip leading
' and trailing spaces, respectively, from a string variable. It
' uses the Trim function alone to strip both types of spaces.
' LCase and UCase are also shown in this example as well as the
' use of nested function calls

Sub Main
    MyString = " <-Trim-> "           ' Initialize string
    TrimString = LTrim(MyString)      ' TrimString = "<-Trim-> "
    MsgBox "|" & TrimString & "|"
    TrimString = LCase(RTrim(MyString)) ' TrimString = " <-trim->"
    MsgBox "|" & TrimString & "|"      ' TrimString = "<-Trim->"
    MsgBox "|" & TrimString & "|"      ' Using the Trim function
                                      ' alone achieves the same
                                      ' result.
    TrimString = UCase(Trim(MyString)) ' TrimString = "<-TRIM->"
    MsgBox "|" & TrimString & "|"
End Sub
```

UserName Function (eQuate)

Return the user name at the specified index in the user list.

Format:

UserName(*index*)

The *index* parameter is any valid integer expression.

The function only produces a result if the current user is an Administrator. Also, the function is not available to eQuate Web clients.

Related Topics: LoadUserList, UserType

Example:

```
' Put all registered user names and types in a listbox
Dim Cnt As Integer
Dim x as Integer

Cnt = LoadUserList()

For x = 1 to Cnt
    ListItemAdd "Listbox_1", UserName(x) & " : " & UserType(x)
Next x
```

UserType Function (eQuate)

Return the user type at the specified index in the user list.

Format:

UserType(*index*)

The *index* parameter is any valid integer expression.

The function only produces a result if the current user is an Administrator. Also, the function is not available to eQuate Web clients.

Related Topics: LoadUserList, UserName

Example:

```
' Put all registered user names and types in a listbox
Dim Cnt As Integer
```

```

Dim x as Integer

Cnt = LoadUserList()

For x = 1 to Cnt
    ListItemAdd "Listbox_1", UserName(x) & " : " & UserType(x)
Next x

```

Val Function

Return the numeric value of a string of characters.

Format:

Val (string)

Example:

```

Sub main
    Dim Msg
    Dim YourVal As Double
    YourVal = Val(InputBox$("Enter a number"))
    Msg = "The number you entered is: " & YourVal
    MsgBox Msg
End Sub

```

VarType Function

Return a value that indicates how the parameter varname is stored internally.

Format:

VarType (varname)

The *varname* parameter is a variant data type.

<u>VarType</u>	<u>Return Values</u>
Empty	0
Null	1
Integer	2
Long	3
Single	4
Double	5
Currency	6 (not available at this time)
Date/Time	7 (mapped to a string)
String	8

Related Topics: IsNull, IsNumeric

Example:

```

If VarType(x) = 5 Then Print "Vartype is Double"
    ' Display variable type

```

Wait Subroutine (eQuate)

Wait for a fixed amount of time. A call to this subroutine causes the script to wait for a period before continuing on to the next script statement, subroutine or function.

Format:

Wait milliseconds

The *milliseconds* parameter is an integer expression containing the amount of time to wait expressed in milliseconds.

Weekday Function

Return an integer between 1 (Sunday) and 7 (Saturday) that represents the day of the week for a date argument.

Format:

Weekday (date)

The *date* parameter is any string expression that can represent a date.

The returned integer represents the day of the *date* parameter.

If *date* is a Null, this function returns a Null.

Related Topics: Date, Day, Format, Hour, Minute, Month, Now, Second, Year

Example:

```
Sub Main
    MyDate = "03/03/96"
    print MyDate
    x = Weekday(MyDate)
    print x

End Sub
```

While...Wend Statement

Execute a series of statements as long as a condition is true.

Format:

```
While condition
    [statement(s)]
Wend
```

The While...Wend statement has these parts:

<u>Part</u>	<u>Description</u>
While	Begins the While...Wend flow of control structure.
<i>condition</i>	Any numeric or expression that evaluates to true or false. If the condition is true, the statements are executed.
<i>statement(s)</i>	Any number of valid statements.
Wend	Ends the While...Wend flow of control structure.

Related Topics: Do...Loop Statement, With

See also, Data Types, Operators and Precedences

Example:

```
Sub Main
    Const Max = 5
    Dim A(5) As String
    A(1) = "Programmer"
    A(2) = "Engineer"
    A(3) = "President"
    A(4) = "Tech Support"
    A(5) = "Sales"
    Exchange = True

    While Exchange
        Exchange = False
        For I = 1 To Max
            MsgBox A(I)
        Next I
    Wend

End Sub
```

With Statement

Execute a series of statements on a single object or user-defined type.

Format:

```
With object
    [statement(s)]
End With
```

The With statement allows you to perform a series of commands or statements on a particular object without referring to the name of that object again. With statements can be nested by putting one With block within another With block. You will need to fully specify any object in an inner With block to any member of an object in an outer With block.

Related Topics: Do...Loop, While...Wend

Example:

```
' This sample shows some of the features of
' user defined types and the with statement.

Type type1
    a As Integer
    d As Double
    s As String
End Type

Type type2
    a As String
    o As type1
End Type

Dim typela As type1
Dim type2a As type2

Sub Main ()

    With typela
        .a = 65
        .d = 3.14
    End With
    With type2a
        .a = "Hello, world"
        With .o
            .s = "Goodbye"
        End With
    End With
    typela.s = "YES"
    MsgBox typela.a
    MsgBox typela.d
    MsgBox typela.s
    MsgBox type2a.a
    MsgBox type2a.o.s

End Sub
```

Write # - Statement

Write and format data to a sequential file that must be opened in output or append mode.

Format:

Write # filename [,parameterlist]

A comma-delimited list of the supplied parameters is written to the indicated file. If no parameters are present, the "newline" character is all that will be written to the file.

Related Topics: Close, EOF, Input, Line Input, Open, Print #

Example:

```
Sub Main ()

    Open "TESTFILE" For Output As #1 ' Open to write file.
    userData1$ = InputBox ("Enter your own text here")
    userData2$ = InputBox ("Enter more of your own text here")
    Write #1, "This is a test of the Write # statement."
    Write #1,userData1$, userData2
    Close #1

    Open "TESTFILE" for Input As #2 ' Open to read file.
    Do While Not EOF(2)
        Line Input #2, FileData ' Read a line of data.
```

```

        Print FileData          ' Construct message.

    Loop
    Close #2                    ' Close all open files.
    MsgBox "Testing Print Statement" ' Display message.
    Kill "TESTFILE"             ' Remove file from disk.
End Sub

```

XmitCursor Subroutine (eQuate)

Cause the current screen to be transmitted to the host through the end of the current cursor field. The transmit is not performed until the current action is exited.

Format:

XmitCursor

XmitFrom Subroutine (eQuate)

Cause the current screen to be transmitted to the host through the end of the specified field. The transmit is not performed until the current action is exited.

Format:

XmitFrom *fieldname*

The *fieldname* parameter is any string expression containing the name of a valid eQuate Data Field.

Year Function

Return an integer between 100 and 9999 that is the portion of the *date* parameter representing a year.

Format:

Year (*date*)

The *date* parameter is any string expression that can represent a date.

The returned integer represents the year of the *date* parameter.

If *date* is a Null, this function returns a Null.

Related Topics: Date, Day, Format, Hour, Minute, Month, Now, Second, Weekday

Example:

```
ThisYear = Year(Now)
```

Predefined Constants

Predefined Constants

This topic contains all eQuate interface constants that are predefined when an eQuate action script is invoked.

Note: Since variables do not have to be declared prior to first reference, use an **Option Explicit** statement (external to the procedure) when using constants to assure that they are spelled correctly.

Color Types (also see **GetColor** and **SetColor**):

<u>Constant</u>	<u>Value</u>	<u>Description</u>
tpForeColor	0	The color of text or box border
tpBackColor	1	The color of the control's background area





Defined Colors (also see **SetColor**):

<u>Constant</u>	<u>Description</u>
clBlack	Black
clMaroon	Maroon
clGreen	Green
clOlive	Olive green
clNavy	Navy blue
clPurple	Purple
clTeal	Teal
clGray	Gray
clSilver	Silver
clYellow	Yellow
clRed	Red
clLime	Lime green
clBlue	Blue
clFuchsia	Fuchsia
clAqua	Aqua
clWhite	White

The following colors refer to the default colors set in the current Windows environment:

clScrollBar	Current color of Windows scrollbar
clBackground	Current color of Windows background
clActiveCaption	Current color of the title bar of the active window
clInactiveCaption	Current color of the title bar of inactive windows
clMenu	Current background color of menus
clWindow	Current background color of windows
clWindowFrame	Current color of window frames
clMenuText	Current color of text on menus
clWindowText	Current color of text in windows
clCaptionText	Current color of the text on the title bar of the active window
clActiveBorder	Current border color of the active window
clInactiveBorder	Current border color of inactive windows
clAppWorkSpace	Current color of the application workspace
clHighlight	Current background color of selected text
clHighlightText	Current color of selected text
clBtnFace	Current color of a button face
clBtnShadow	Current color of a shadow cast by a button
clGrayText	Current color of text that is dimmed
clBtnText	Current color of text on a button
clInactiveCaptionText	Current color of the text on the title bar of an inactive window
clBtnHighlight	Current color of the highlighting on a button

Message Box Constants (also see MsgBox):

<u>Constant</u>	<u>Value</u>	<u>Description</u>
<i>MsgBox Buttons:</i>		
MB_OK	0	OK button only
MB_OKCANCEL	1	OK and Cancel buttons
MB_ABORTRETRYIGNORE	2	Abort, Retry and Ignore buttons
MB_YESNOCANCEL	3	Yes, No and Cancel buttons
MB_YESNO	4	Yes and No buttons
MB_RETRYCANCEL	5	Retry and Cancel button
<i>MsgBox Icons:</i>		
MB_ICONSTOP	16	 Critical message
MB_ICONQUESTION	32	 Warning query
MB_ICONEXCLAMATION	48	 Warning message
MB_ICONINFORMATION	64	 Information message
<i>MsgBox Defaults:</i>		
MB_APPLMODAL	0	Application Modal Message Box - the user must respond to the message before continuing work in the current application (Default)
MB_DEFBUTTON1	0	First button is default
MB_DEFBUTTON2	256	Second button is default
MB_DEFBUTTON3	512	Third button is default
MB_SYSTEMMODAL	4096	System Modal - all applications are suspended until the user responds to the message box
<i>MsgBox return values:</i>		
IDOK	1	OK button pressed
IDCANCEL	2	Cancel button pressed
IDABORT	3	Abort button pressed
IDRETRY	4	Retry button pressed
IDIGNORE	5	Ignore button pressed
IDYES	6	Yes button pressed
IDNO	7	No button pressed

Print Font Styles (also see PrintSetFont):

<u>Constant</u>	<u>Value</u>	<u>Description</u>
fsNormal	0	Normal font
fsFontBold	1	Bold font
fsFontItalic	2	Italic font
fsFontUnderline	4	Underlined font
fsFontStrikeThru	8	Strikethrough font

Screen Attributes (also see [GetScreenAttribute](#)):

<u>Constant</u>	<u>Value</u>	<u>Description</u>
ATTR_NORMAL	0	Normal
ATTR_FIELD	1	Start of Field (set on first position of field)
ATTR_TAB	2	Tab Stop (at start of field only)
ATTR_CHANGED	4	Data Field Changed Flag
ATTR_PROTECTED	8	Protected

<u>Constant</u>	<u>Value</u>	<u>Description</u>
ATTR_VIDEO_OFF	16	Video Off
ATTR_NUMERIC	32	Numeric Only Input
ATTR_ALPHA	64	Alphabetic Only Input
ATTR_BLINK	128	Blinking
ATTR_RIGHT	256	Right Justified Data
ATTR_LOWINT	512	UTS Low Intensity
ATTR_REV	1024	Reverse Video

Scrollbar Types (see Memo control):

<u>Constant</u>	<u>Value</u>	<u>Description</u>
sbNone	0	No scrollbar.
sbHorizontal	1	Memo has a horizontal scrollbar at the bottom of the control
sbVerticle	2	Memo has a vertical scrollbar at the right of the control
sbBoth	3	Memo has both horizontal and vertical scrollbars

Transparent Mode

<u>Constant</u>	<u>Value</u>	<u>Description</u>
tmAuto	0	TransparentColor property returns the bottom-leftmost pixel of the bitmap image
tmFixed	1	TransparentColor property refers to the color stored in the bitmap object

State Types (also see GetState and):

<u>Constant</u>	<u>Value</u>	<u>Description</u>
tpEnabled	0	Whether the control is enabled or disabled (grayed) - applies to all controls
tpVisible	1	Whether the control is visible to the user or not - applies to all controls.
tpChecked	2	Whether the control is checked or unchecked - applies only to option buttons and check boxes

T27 Constants (see DoTerminalKey):

<u>Constant</u>	<u>Integer</u>
TK_ARROWDN	249
TK_ARROWLEFT	247
TK_ARROWRIGHT	248
TK_ARROWUP	246
TK_BACKSPACE	8
TK_BACKTAB	196
TK_BOUND	218
TK_CARRIAGERTN	13
TK_CLRALLVTAB	16442
TK_CLREOL	134
TK_CLREOP	135
TK_CLRFORMS	159
TK_CLRHOME	128
TK_COPY	16432
TK_CTRL	164
TK_CUT	16431
TK_DBLZERO	234
TK_DELCHAR	132
TK_DELCHARPAGE	16425
TK_DELLINE	133
TK_HOME	174
TK_INSCAR	130

<u>Constant</u>	<u>Integer</u>
TK_INSCHARPAGE	16424
TK_INSLINE	131
TK_LOCAL	168
TK_LOCKCTRL	165
TK_LOGICALEOL	16415
TK_MARK	217
TK_MOVELINEDOWN	138
TK_MOVELINEUP	139
TK_NEXTPAGE	253
TK_PASTE	16434
TK_PREVPAGE	252
TK_PRINTALL	157
TK_PRINTUNPROT	156
TK_RECALL	214
TK_RECEIVE	170
TK_ROLLDN	136
TK_ROLLUP	137
TK_SETFORMS	158
TK_SPECIFY	166
TK_STORE	213
TK_TAB	198
TK_TOGGLEFORMS	141
TK_TOGGLETAB	16441
TK_TRANSMIT	172
TK_TRANSMITLINE	16428
TK_TRIPZERO	236
TK_UPPERONLYON	210
TK_UPPERONLYOFF	211
TK_WRITEESC	16426
TK_WRITEETX	3
TK_WRITEGS	16427

UTS Constants (see DoTerminalKey):

<u>Constant</u>	<u>Integer</u>
UK_BACK_SPACE	95
UK_CURSOR_DOWN	6
UK_CURSOR_LEFT	7
UK_CURSOR_RETURN_KEY	32
UK_CURSOR_RIGHT	8
UK_CURSOR_TO_END_LINE	66
UK_CURSOR_TO_HOME	23
UK_CURSOR_TO_START_LINE	65
UK_CURSOR_UP	9
UK_DELETE_IN_DISPLAY	11
UK_DELETE_IN_LINE	12
UK_DELETE_LINE	10
UK_ERASE_CHAR	67
UK_ERASE_DISPLAY	14
UK_ERASE_TO_END_DISPLAY	15
UK_ERASE_TO_END_FIELD	16
UK_ERASE_TO_END_LINE	17
UK_FKEY_1	43
UK_FKEY_2	44
UK_FKEY_3	45

<u>Constant</u>	<u>Integer</u>
UK_FKEY_4	46
UK_FKEY_5	47
UK_FKEY_6	48
UK_FKEY_7	49
UK_FKEY_8	50
UK_FKEY_9	51
UK_FKEY_10	52
UK_FKEY_11	53
UK_FKEY_12	54
UK_FKEY_13	55
UK_FKEY_14	56
UK_FKEY_15	57
UK_FKEY_16	58
UK_FKEY_17	59
UK_FKEY_18	60
UK_FKEY_19	61
UK_FKEY_20	62
UK_FKEY_21	63
UK_FKEY_22	64
UK_INSERT_IN_DISPLAY	25
UK_INSERT_IN_LINE	26
UK_INSERT_LINE	24
UK_KEYBOARD_UNLOCK	27
UK_LINE_DUP	28
UK_MSG_WAIT	29
UK_PRINT_KEY	30
UK_PRINT_ENTIRE_SCREEN	69
UK_SOE	3
UK_TAB_BACK	33
UK_TAB_FORWARD	34
UK_TAB_SET	35
UK_TRANSMIT_KEY	36

Form Capture

eQuate Form Capture

This is the first dialog of the screen capture process. Screen capture is used to scrape a host program's terminal screen and generate a definition of that screen that can subsequently be imported into the eQuate Application Manager's form design process.

The first step in the screen capture process is to select the type of host connection to be made. The dialog offers two choices: UTS for Unisys 2200 applications and T27 for Unisys MCP or A-Series applications. When you open the terminal connection, you will pick from a list of configured routes to connect to the desired host.

If no routes are available, run the QPort UTS Config or QPort T27 Config program (in the eQuate folder) to configure a route to the appropriate host.

Open Terminal

After choosing the connection type, UTS or T27, click this button to select the connection route. When the Select Route dialog appears, select a configured route and click the OK button.

UTS

Set this option for Unisys 2200 applications.

T27

Set this option for Unisys MCP or A-Series applications.

Capture Screen

After the desired screen has been displayed in the WinQT27 or WinQUTS screen capture window, use this button to capture the screen.

WinQUTS32: routename

This window displays the host application screen as it appears during execution of the host application. The window appears during the screen capture process when executing the eQuate Capture utility and optionally when running an eQuate application with the eQuate Session Manager. When used with the eQuate Session manager, this window is extremely useful to help identify communication problems between the host application and eQuate. The status bar at the bottom of the window allows you to easily identify the row and column position of the text cursor and mouse pointer, thus letting you quickly find the starting and ending positions of anything on the screen.

File menu

The File menu contains controls for printing.

Print Screen (All) Ctrl+P

Use this selection to print an exact image of the entire screen.

Print Screen (UTS) Ctrl+U

Use this selection to produce the traditional UTS "Print Screen" operation. The selection prints all the text from the previous SOE to the current cursor position.

Close Print Ctrl+L

Use this selection to end the print operation. This selection only appears on the menu as a result of issuing a **Print Screen** command.

Edit menu

The Edit menu contains controls to manage selected text between the screen and the Windows clipboard.

Copy Ctrl+C

Use this selection to copy selected data onto the clipboard. This command is unavailable if there is no data currently selected. Note: Copying data to the clipboard replaces the contents previously stored there.

Data may be selected by placing the mouse cursor at some point on the screen, and holding down the left mouse button, drag the mouse cursor to any other location on the screen. Upon releasing the left mouse button, the data will be selected.

An alternate form of selection is with standard, keyboard key sequences. With the mouse or arrow keys, place the screen text cursor at some point on the screen and use any of the following key sequences to highlight the data being selected:

Shift+Home selects everything from the cursor position through column one of the same row.

Shift+End selects everything from the cursor position through the last column of the same row.

Shift+Right Arrow extends/contracts the selection to the right.

Shift+Down Arrow extends/contracts the selection down the screen.

Shift+Left Arrow extends/contracts the selection to the left.

Shift+UP Arrow extends/contracts the selection up the screen.

Cut Ctrl+X

Use this selection to remove the currently selected data from the document and put it on the clipboard. This command is unavailable if there is no data currently selected.

Cutting data to the clipboard replaces the contents previously stored there.

Paste Ctrl+V

Use this selection to insert a copy of the clipboard contents at the insertion point. This command is unavailable if the clipboard is empty.

Note: Use a right mouse click, anywhere on the screen, to reveal a Copy/Cut/Paste popup menu.

Input Recall Ctrl+Up

Use this selection to initiate the Input Recall window. This selection allows previous lines transmitted to the host to be recalled and retransmitted.

Options menu

Show Control Characters

Use this selection to toggle the display of control characters (Tab, Form-Feed, Line-Feed, etc.).

Screen Font Sizing...

Use this selection to display the font sizing controls. This selection allows the font size and spacing between lines to be increased decreased.

Trace to Window

Use this selection to initiate the trace facility. All communications between eQuate and the host will be displayed in a separate window.

Note: A right mouse click in the trace window will reveal a menu containing three selections that control the trace window: Stay on Top, Copy to Clipboard and Clear. Stay on Top controls whether the trace window remains on top (the default) of the emulator window during the trace, or remains in the background. Copy to Clipboard may be used to copy the contents of the trace window to the Windows clipboard for subsequent pasting into another Windows application (e.g., Notepad, WordPad, etc.). Clear may be used to clear the contents of the trace window.

Trace to File

Use this selection to initiate the trace facility. This selection is the same as Trace to Window except that the trace information will be written to a user-named file.

Trace to Both Window and File

Use this selection to initiate the trace facility. This selection is a combination of the two above.

Clear Trace Window

Use this selection to clear the trace window and/or file without terminating the trace. This selection appears after a trace has been initiated.

Close Trace

Use this selection to close the trace window and/or file and terminate the trace.

FCC Map

Use this selection to display an FCC Map. Use only at the request of KMSYS Worldwide, Inc., support personnel.

Control Page

Use this selection to display the contents of the UTS Control Page in the trace window and/or file.

Note: The last four selections only become visible once the trace has been started.

Input Recall window

The capture utility saves single-line messages sent to the host system when you press the transmit key. A maximum of 200 of the most recent messages is saved. You can display a list of these messages from the Edit, Input Recall menu selection, or by pressing the Input Recall key (Ctrl+I).

Paste

When this list is displayed in the Input Recall window, you can view the entries and optionally select an entry to be "pasted" onto the current screen when the Paste button is pressed.

Transmit

The Transmit button performs the same operation as the Paste button, but also ends by transmitting to the host (double clicking on the entry also performs the paste-and-transmit operation).

Close

Use this button to close the Input Recall window.

WinQT27: routename

This window displays the host application screen as it appears during execution of the host application. The window appears during the screen capture process when executing the eQuate Capture utility and optionally when running an eQuate application with the eQuate Session Manager. When used with the eQuate Session manager, this window is extremely useful to help identify communication problems between the host application and eQuate. The status bar at the bottom of the window allows you to easily identify the row and column position of the text cursor and mouse pointer, thus letting you quickly find the starting and ending positions of anything on the screen.

File menu

The File menu contains controls to configure the appearance and behavior of T27 Express Plus32 and to manage the print environment of T27 Express Plus32.

Print Screen

Use this selection to print an exact image of the entire screen.

Printer Setup...

Use this selection to select a printer for use in future print operations.

Edit menu

The Edit menu contains controls to manage selected text between the screen and the Windows clipboard.

Copy Ctrl+C

Use this selection to copy selected data onto the clipboard. This command is unavailable if there is no data currently selected. Note: Copying data to the clipboard replaces the contents previously stored there.

Data may be selected by placing the mouse cursor at some point on the screen, and holding down the left mouse button, drag the mouse cursor to any other location on the screen. Upon releasing the left mouse button, the data will be selected.

An alternate form of selection is with standard, keyboard key sequences. With the mouse or arrow keys, place the screen text cursor at some point on the screen and use any of the following key sequences to highlight the data being selected:

Shift+Home selects everything from the cursor position through column one of the same row.

Shift+End selects everything from the cursor position through the last column of the same row.

Shift+Right Arrow extends/contracts the selection to the right.

Shift+Down Arrow extends/contracts the selection down the screen.

Shift+Left Arrow extends/contracts the selection to the left.

Shift+UP Arrow extends/contracts the selection up the screen.

Cut Ctrl+X

Use this selection to remove the currently selected data from the document and put it on the clipboard. This command is unavailable if there is no data currently selected.

Cutting data to the clipboard replaces the contents previously stored there.

Paste Ctrl+V

Use this selection to insert a copy of the clipboard contents at the insertion point. This command is unavailable if the clipboard is empty.

Note: Use a right mouse click anywhere on the screen to reveal a Copy/Cut/Paste popup menu.

Input Recall Ctrl+I

Use this selection to initiate the Input Recall window. This selection allows previous lines transmitted to the host to be recalled and retransmitted.

Options menu

The Options menu contains controls that adjust the appearance and functional state of T27 Express Plus32.

Show Control Characters

Use this selection to toggle the display of control characters (Tab, Form-Feed, Line-Feed, etc.).

Screen Font Sizing...

Use this selection to display the font sizing controls. This selection allows the font size and spacing between lines to be increased decreased.

Trace to Window

Use this selection to initiate the T27 Express Plus32 trace facility. All communications between T27 Express Plus32 and the host will be displayed in a separate window.

Note: A right mouse click in the trace window will reveal a menu containing three selections that control the trace window: Stay on Top, Copy to Clipboard and Clear. Stay on Top controls whether the trace window remains on top (the default) of the emulator window during the trace, or remains in the background. Copy to Clipboard may be used to copy the contents of the trace window to the Windows clipboard for subsequent pasting into another Windows application (e.g., Notepad, WordPad, etc.). Clear may be used to clear the contents of the trace window.

Trace to File

Use this selection to initiate the T27 Express Plus32 trace facility. This selection is the same as Trace to Window except that the trace information will be written to a user-named file.

Trace to Both Window and File

Use this selection to initiate the T27 Express Plus32 trace facility. This selection is a combination of the two above.

Clear Trace Window

Use this selection to clear the trace window and/or file without terminating the trace. This selection appears after a trace has been initiated.

Close Trace

Use this selection to close the trace window and/or file and terminate the trace.

FCC Map

Use this selection to display an FCC Map. Use only at the request of KMSYS World Wide, Inc., support personnel.

Note: The last four selections only become visible once the trace has been started.

Input Recall window

The capture utility saves single-line messages sent to the host system when you press the transmit key. A maximum of 200 of the most recent messages is saved. You can display a list of these messages from the Edit, Input Recall menu selection, or by pressing the Input Recall key (Ctrl+I).

Paste

When this list is displayed in the Input Recall window, you can view the entries and optionally select an entry to be "pasted" onto the current screen when the Paste button is pressed.

Transmit

The Transmit button performs the same operation as the Paste button, but also ends by transmitting to the host (double clicking on the entry also performs the paste-and-transmit operation).

Close

Use this button to close the Input Recall window.

Edit Capture

The Edit Capture window displays the fields and fixed text of the captured screen.

File menu

Generate Form Definition Text

Creates form definition text that can be edited and subsequently saved in a file or placed on the Windows clipboard.

Close

Close the eQuate Edit Capture and return to the emulation window for eQuate Form Capture.

Edit menu

Add Field

Add a new field to the form.

Delete Selected Fields

Delete a field or fields from the form. To delete fields from the form, first select the field with the left mouse button, then press Ctrl+D or select Delete Selected Field from the Edit menu. You may select multiple fields by holding down the Shift key when selecting fields with the mouse. Another way to select multiple fields is to move the mouse over a teal (dark greenish blue) portion of the screen without a field, and while holding down the left mouse button, drag the cursor across a group of fields to be deleted. Any field the cursor touches will be selected.

Undelete Selected Fields

Restore previously deleted fields.

Mark Form Id String

Mark a string of screen text to be used as the unique Form Id. String. Making this selection changes the mouse cursor to a cross hair. Holding the left mouse button down, drag the mouse over the screen text the uniquely identifies the form. A maximum of 32 characters may be marked. Note: After marking a Form Id String, the characters that comprise the string will appear on the next to the last status bar at the bottom of the window, even though the field might be "hidden" on the screen.

Mark Fields

Mark all fields on the captured screen. The command acts as an "undo" of all edits made since this window was first displayed; i.e., fields marked as deleted will be restored, fields added to the form will be removed and fields data types and justification restored to their defaults (Any/Left).

Set Repeating Rows

Set the number of repeating rows. This selection will enable the Repeat Capture dialog. . It is necessary for eQuate to know if a given field is a repeating field as opposed to one that appears just one time. When a screen is captured, eQuate has no way of ascertaining if a field repeats on a given screen. There is no information on the screen that tells eQuate that the field repeats; therefore, you must identify which fields repeat.

Option menu

Ignore Adjacent Protected Fields

When option is checked (the default), Capture will not automatically mark protected areas as fields when they are immediately to the right of another field. It assumes that there is some gap between every field. This feature will work well for all T27 users and most UTS users; however, DPS does allow fields to be immediately adjacent to each other.

Window menu

Resize Window to Capture

Resize the Edit Capture window to the size of the captured screen.

Modifying Field Definitions

Since the eQuate Capture program cannot always detect certain aspect of a field (meaningful name, numeric attributes, etc.), additional controls are provided at the bottom of the dialog allowing you to assign them within the capture program before generating the definition that will be imported into the eQuate Application Manager.

To edit a field, first select it with the mouse. Notice that eQuate Capture has automatically assigned a field name. This name is comprised of the word "FIELD" and the row and starting column where the field resides on the screen. The name is the name referenced in any eQuate actions, and therefore, might be changed to a more meaningful name by editing the value in the Selected Field Name text box.

Selected Field Name

The Selected Field Name is the reference name that will be used in eQuate actions. Valid characters are A through Z and the under bar (_). Note: Striking the spacebar will cause an under bar to be entered.

Force Upper Case

Any alphabetic characters typed into the field entry will be shifted to uppercase.

Data Type

These controls determine what the user can enter into the selected field. Changing an option here will completely override any properties defined for the application's terminal screen.

Justify

Use these option buttons to effect how data is to be justified when the user types in the field.

Decimal Positions

Decimal positions may be specified for decimal alignment of Numeric Only fields.

Signed

If this box is checked, numeric fields may contain a sign character.

Repeat Capture

This dialog allows you to specify how many times a field repeats on a screen. It is necessary for eQuate to know if a given field is a repeating field as opposed to one that appears just one time. When a screen is captured, eQuate has no way of ascertaining if a field repeats on a given screen. There is no information on the screen that tells eQuate that the field repeats; therefore, you must identify which fields repeat.

Repeat Count

This value allows the specification of a Repeat count that corresponds to the number of times a particular field appears on the screen.

Rows in Repeat

Rows in repeat parameter must be used when multiple rows are used to accommodate all the fields in a single occurrence of repeating fields. If only one (1) row is used, set this field to one (1).

Session Manager

eQuate Session Manager

The eQuate Session Manager window is where you execute all your application sessions. The applications are setup by your eQuate application administrator and should be cleared through that individual. If an application fails to run properly, please contact that individual for assistance.

Available eQuate Applications

From this list box, select the application (click on the application with the mouse) and click the Run Application button.

Run Application

After selecting the desired application with the mouse, click this button to run the eQuate application.

Close Application

Click this button to terminate the eQuate Session manager.

File menu

Change Reg. Settings Location

Use this selection to change the location of the eQuate registry settings from current user (each user has their own settings) to local machine (all users use the same settings), or visa versa.

Note: This option is not available for eQuate network users.

Lock Settings/Unlock Settings

Use this selection to lock or unlock other options on this menu.

Note: This option is not available for eQuate network users.

Profiles

Use this selection to assign routes to eQuate applications. For users with the eQuate Session Manager installed on their PC, routes are assigned on the eQuate Session Manager since each user may have different routes configured for host access.

Note: This option is not available for eQuate network users. For eQuate network users, routes are assigned by the eQuate Administrator.

Runtime Directory

Use this selection to point to the runtime directory where the eQuate application databases are located. Normally, this directory would be located on a file server to which the user must have read privileges.

Note: This option is not available for eQuate network users.

Script Directory

Use this selection to point to the directory where the eQuate sign-on script(s) are located.

Note: This option is not available for eQuate network users.

Note: Normally, the above directories are maintained by the person responsible for administering eQuate on your system. If you have any questions regarding the route to use, please contact that person.

Close

Use this selection to exit the session manager.

Options menu

Force Action Reloads

Select this option to always reload an eQuate action file (.act) of a form when an action is invoked. This option allows eQuate actions to be changed while a form is displayed by the eQuate Session Manager.

Enter Equals Tab

Select this option if you are use to using the Enter key on the keyboard act as the Tab key.

Tab on Full Field

Select this option to have the eQuate Session Manager automatically move to the next field on the window when the current field is full. Note: The next field is determined by person that designed the eQuate application.

No Trace

Set this option (default) when tracing is not needed.

Trace to Window

Set this option to trace the host connection to a window.

Trace to File

Set this option to trace the host connection to a file.

Trace to Window and File

Set this option to trace the host connection to both a window and a file.

Screen Visible

Set this option to make the emulator screen visible.

Debug Logging

Set this option only when directed by the eQuate Administrator or KMSYS Worldwide, Inc..

Transport Trace

Set this option only when directed by the eQuate Administrator or KMSYS Worldwide, Inc.

Run menu**Run Application**

After selecting the desired application with the mouse, use this selection to run the eQuate application.

Terminal Screen

Use this configurable selection to select a route, open a plain terminal screen and connect to the host.

This feature was designed as a contingency plan for user access if the file server containing the eQuate applications is down or otherwise unavailable.

Note: This menu item is only visible if the "/TERMINAL" parameter is specified on the Session Manager command line.

Profiles

This window is used to assign routes to the various applications that the user is allowed to run.

Application

This is the list of applications available to the user. To assign a route to the application, first select the application with the mouse and then select the route from the User Route drop-down list box at the bottom of the window.

User Route

From this drop-down list box, select the route that is to be applied to the eQuate application.

Note: Normally, the route is configured by the person responsible for administering and maintaining eQuate on your system. If you have any questions regarding the route to use, please contact that person.

Save and Close

After assigning one or routes, click this button to save the settings and exit the window.

User Sign On

This dialog is used to enter the user id and password of the Administrator of eQuate or the eQuate Application Developer. The default user id and password supplied by KMSYS Worldwide, Inc., is ADMIN and EQUATE, respectively. For security, it is recommended that these be changed upon first use.

User Id.

In this box, enter the user id of the eQuate Administrator or Application Developer.

Current Password

In this box, enter the password of the eQuate Administrator or Application Developer.

New Password (if changing)

When changing the password, enter the new password. You will be asked to confirm by entering the new password a second time.

Change eQuate Runtime Settings Location

This dialog is used to switch the location of the QPort T27 eXpress Plus transport settings. The settings may be placed in the Windows Registry in such a manner as to allow multiple users of a single Windows system to maintain their individual configurations. When individual configurations are to be maintained (the default), the settings are placed under HKEY_CURRENT_USER in the registry. On the other hand, if it is desirable for all users of the system to use the same configuration, an option is available to move those settings to the common location, HKEY_LOCAL_MACHINE.

The dialog acts as a toggle between CURRENT_USER and LOCAL_MACHINE; i.e., if your settings are currently stored in CURRENT_USER, the message will indicate that they be moved to LOCAL_MACHINE; if they are in LOCAL_MACHINE, the move would be to CURRENT_USER.

Delete Source Settings

Check this box to delete the settings in their former location. If this box is not checked, the settings will remain in their old location but will NOT be used or updated in any way.

Set

After selecting an option above, click this button to complete the move.

Cancel

Press this button to ignore any selections and return to the previous dialog.

Unpackage

eQuate Runtime File UN-Packager

This dialog is used to select a packaged eQuate application and update an existing eQuate database.

Runtime package distribution file

Use the Select button beside this text box to select the drive, path and file to unpackage.

Destination database path

Use the Select button beside this text box to select the drive and path of the eQuate database.

Global Maintenance

eQuate Global Maintenance Utility

The eQuate Global Maintenance Utility provides a mechanism to change properties in several eQuate Forms at once without having to individually open and edit each form.

Global Maintenance

There are three windows that may be used when performing global form maintenance:

- Main Window for editing scripts;
- Error window for showing script errors;
- Form selection window for selecting forms to be affected.

Global Maintenance Script Syntax

Control Block Structure

All global maintenance is specified in a Global Maintenance Script. Global Maintenance Scripts contain one or more Control blocks. Each Control Block specifies changes to a specific named control on a form, or all controls on a form of the same type. Any number of properties may be changed in a single control block.

The following is the basic syntax of a Control Block:

```
CONTROL
    NAME= ControlName
    TYPE= ControlType
    PROPERTY= PropertySpecification
    ....
END
```

CONTROL/END

The CONTROL/END statements simply define the beginning and ending of a control block. All CONTROL statements must have a matching END statement.

NAME

The NAME statement specifies which control or controls are to be affected. If a specific name is used, only controls of that name (and Type) will be affected. If a name omitted is used, all controls of the same type will be affected. One NAME statement must be specified in a Control Block. The NAME statement must be present whether or not a control name is specified.

TYPE

The TYPE statement specifies which type of control is to be affected. A specific control type must be specified. Only controls on the specified type, and specified NAME, will be affected. One TYPE statement is must be specified in a control block.

PROPERTY

The PROPERTY statement specifies individual control property changes. There are as many PROPERTY statements are allowed within a Control Block as needed.

A PROPERTY Statement has the following addition Property Specification syntax:

```
PROPERTY= PropertyName,[ OldValue],NewValue
```

PropertyName must be a valid property name for the specified control type

OldValue is optional and specifies which Old values to change. If the Old value does not match, the change will not be applied.

NewValue specifies the new property value.

The font property requires a compound property value enclosed within double quotes.

Syntax:

```
Name,Size,Styles
```

Name must be a valid font name.

Size must be a valid font size specified in points (positive whole numbers)

Styles may contain any of the following separated by commas:

```
BOLD, ITALIC, UNDERLINE, STRIKETHRU
```

Example:

```
Property=Font,, "Arial,10,Bold,Italic"
```

Example Global Maintenance Scripts

In this example all selected Forms (a Form is a control) will be changed as follows:

The Caption property changed to "Cool Stuff".

The Ctl3d property will be set to true.

Any selected forms with a BackColor property of red will have the BackColor property changed to blue.

The form's Font property will be changed to Courier New, with a size of 9 and the bold and italic styles applied.

```
CONTROL
NAME=*
TYPE=Form
PROPERTY=Caption,, "Cool Stuff"
PROPERTY=Ctl3d,, true
PROPERTY=BackColor, Red, Blue
property=Font,, "Courier New, 9, Bold, Italic"
END
```

Global Maintenance Script Editor

File menu

The File menu contains commands to maintain script files and setup printing.



New (Ctrl+N)

Use this command to create a new global maintenance script file (.gms).



Open (Ctrl+O)

Use this command to open an existing script file (.gms).



Save (Ctrl+S)

Use this command to save the current script file (.gms).



Save As...

Use this command to save the current script file (.gms) to another file.

Select Database Directory

Use this selection to point the eQuate Global Form Maintenance Utility to the eQuate application database directory containing the forms to be altered.



Editor Properties

Use this command to edit the propertiesEditor_Properties of the script editor window: window font, highlight colors and tab stops.



Exit

Exit the Script Editor.

Edit menu

The Edit menu contains commands to manage selected text between the editor and the Windows clipboard.



Undo (Ctrl+Z)

Use this command to reverse the effects of the most recent change.



Redo (Ctrl+Shift+Z)

Use this command to reverse the effects of the most recent **Undo** command.

**Cut (Ctrl+X)**

Use this command to place the selected text on the clipboard and delete.

**Copy (Ctrl+C)**

Use this command to copy the selected text to the clipboard.

**Paste (Ctrl+V)**

Use this command to paste the contents of the clipboard to the current cursor position.

Symbolic Name Insertion

The script editor provides an easy way to get correct names for control types, property names and property values. To use this feature move the cursor to the desired position within the editing area and press Ctrl+I, click the tool bar button or click the right mouse button and select "Insert Symbolic Name" from the Popup menu. Either action will cause the Symbolic Name Selector dialog to come up.

This dialog not only provides a list of various names, but it also provides them as they apply to specific items. For example, if Controls Names is selected in column 1, only control names are shown in column 2 and, only valid properties of the control selected in column 2 are shown in column 3.

View menu

The View menu contains commands to display the form selector and error log windows.

Form Selection List

Use this command to bring up the Global Maintenance Form Selector dialog. Multiple forms may be selected by checking the appropriate box.

Error List

Use this selection to view the Global Maintenance Maintenance Log.

Process Changes button

Once the script has been written, use this button to enact the changes.

Stop Process

Use this button to stop the process.

Command Mode Commands

ECM Command Format and Conventions

The eQuate Command Mode (ECM) provides a more powerful environment than Screen Mode since Command Mode is not dependent upon terminal protocol. Instead of formatting output for presentation on a formatted terminal screen, the application program formats command records for eQuate. eQuate handles the user presentation. Data sent to the application is also in eQuate record format.

Command Mode Record Format

The basic format of all ECM records consists of an ECM header, one or more ECM commands, followed by an ECM trailer.

ECM records have the following Format:

```
EQ$<TAB>ECM-command [<CR>params] [<TAB>ECM-command [<CR>params]...]<TAB>\\
```

Where <TAB> = ASCII value 9 and <CR> = ASCII value 13 (see "Defining ECM Control Strings in COBOL").

The string "EQ\$" is the ECM header and "<TAB>\\\" is the ECM trailer. All ECM records must start with an ECM header and end with an ECM trailer. Each *ECM-command* must be preceded by "<TAB>".

An ECM record may not exceed 4000 characters in total length.

Note: Multiple ECM records can be included in a single message to eQuate. All ECM records are processed in the sequence received. eQuate will only send single record messages as replies to the host application program (this will keep host application programming simple).

Conventions Illustrated

The following additional conventions are used when illustrating record/command formats:

- All words in UPPERCASE letters (not italicized) are reserved keywords and must be entered exactly as shown (except "<TAB>" and "<CR>").
- All italicized words (mostly in lowercase letters) are to be substituted by a user supplied name or value.
- Selections appearing within brackets ([]) are optional items.
- Selections appearing within braces ({ }) that are separated by vertical bars (|), are lists of items from which one and only one must be selected. In the following example, one of either C, D or E must be selected:

```
{C | D | E}
```

Enclosure Characters

Whenever it is necessary to provide a text string of more than one word (i.e., a string with embedded spaces) or a string that contains a special character (see below), it will be necessary to bind the string with a pair of matching enclosure characters. Valid matching enclosure characters are:

" "	Open and close quotes
{ }	Braces
[]	Brackets
()	Parenthesis

A "special character" is defined as any instance when an enclosure character must be a part of the actual text string being supplied. The following string would pass the enclosure characters, parenthesis (), as special characters and discard the enclosure characters, braces { }:

```
EQ$<TAB>ERRLST<CR>{Some record(s) were not processed.}<TAB>\\
```

In certain character sets, some of these characters (braces and brackets) are translated to local language characters and cannot be used as enclosure characters.

CTL

The CTL command is used to manipulate eQuate Controls (other than the global controls of menu items and action keys).

Note: The form must be unlocked prior to using the CTL command (see UNLOCKFRM).

Format:

```
CTL<CR>control-name,property-name,value[...property-name,value]
```

The *control-name* parameter must be defined in the current form definition.

The *property-name* and *value* pairs can be one or more of the following:

<u>property-name</u>	<u>Abbreviation</u>	<u>Description</u>	<u>value</u>
CAPTION	C	Changes the control's caption.	Any text. If the text is more than a single word, it must be enclosed within enclosure characters (i.e., "Click me").
VISIBLE	V	Makes the control visible or invisible.	1 = visible 0 = not visible
ENABLED	E	Makes the control enabled or disabled (grayed).	1 = enabled 0 = not enabled
BACKCOLOR	BC	Sets the background color of the control.	0 through 16, or D for default (see "Color Values", below)
FORECOLOR	FC	Sets the foreground (text color) of the control.	0 through 16, or D for default (see "Color Values", below)

Color Values:

<u>Color</u>	<u>value</u>	<u>Color</u>	<u>value</u>
Black	0	Gray	8
Red	1	Light Red	9
Green	2	Light Green	10
Yellow	3	Light Yellow	11
Blue	4	Light Blue	12
Magenta	5	Light Magenta	13
Cyan	6	Light Cyan	14
White	7	Bright White	15

The CTL command may only be used on freestanding eQuate controls. If an attempt is made to set a property that does not exist for the control, the setting is ignored. An error is generated if any property name other than those allowed is specified.

CTL command Examples:

Set the CHK_1 check box to enabled with a caption of "I'm Checked":

```
CTL<CR>CHK_1,E,1,c,"I'm Checked"
```

Disable (gray) the button named BTN_EDIT:

```
CTL<CR>BTN_EDIT,E,0
```

DTAFLD

DTAFLD is used to send a single field of data from the host program to the eQuate Session Manager. DTAFLD may be abbreviated DF.

Format:

```
<TAB>DTAFLD<CR>field-name[:n],data-string
```

The *field-name* parameter must be defined in the current form definition.

If embedded spaces or commas (,) are a part of the *data-string*, the entire *data-string* must be enclosed in matching enclosure characters. eQuate will discard the enclosure characters. If the field-name is a repeating field, *n* is the occurrence number of the field and must be a numeric value greater than zero.

Examples:

```
EQ$<TAB>DTAFLD<CR>ACCOUNT,103446<TAB>DTAFLD<CR>COMPANY,"THE BEAR FAX, INC."<TAB>\\
```

DTAREC

DTAREC is used to send a data record from the host program to the eQuate Session Manager. DTAREC may be abbreviated DR. Data record data is in the form of a continuous string of ASCII text. Any ASCII character is allowed except a <CR> or <TAB>. Data field positions are defined in the eQuate form and may be set by clicking the Data Fields button on the eQuate Form Manager dialog.

Format:

<TAB>DTAREC<CR> *data-string*

Example:

EQ\$<TAB>DTAREC<CR>103446THE BEAR FAX, INC. 10YYN1-00034500EV 409<TAB>\\

ERRLST

ERRLST delivers a list of error messages to eQuate for display in the eQuate Error List window. ERRLST may be abbreviated EL. The error list may contain complete error statements, or standard error message references that may include substitution strings. Each message statement must be enclosed between matching enclosure characters (e.g., {message statement}) and will appear on a separate line in the eQuate Error List window. Standard error references must begin with a 1- to 5-digit, standard error number followed by a colon (:). Substitutions within a standard error reference must be separated by spaces and enclosed within matching enclosure characters other than those used for message enclosure. In the example below, substitution strings are enclosed in double quotes, while each message statement is separated by open and closed braces {}.

Format:

<TAB>ERRLST<CR> *message(s)*

The following is an example of an ERRLST message containing both standard error references (with substitution) and error statements:

EQ\$<TAB>ERRLST<CR>{12:"ACCOUNT" "Out of Range"}{This is an independent error statement and will appear as is}{119:"LINE 24"}<TAB>\\

Substitution points in standard error message text are indicated by the sequence "\\n". Where n indicates which word from the input error statement is to be inserted. There may be from 1 to 9 substitutions in a single standard error message. The text of Standard Messages 12 and 119 would appear as follows:

Standard message 12: "Field \\1 is \\2."

Standard message 119: "Value entered in \\1 is out of range."

In the above example, "ACCOUNT" will replace "\\1" and "Out of Range" will replace "\\2" in standard message number 12, and "LINE 24" will replace "\\1" in standard message 119.

See Standard Messages for the procedure for storing messages in a eQuate Database.

FLDCTL

FLDCTL sets highlighting of a field on or off. FLDCTL may be abbreviated FC.

Format:

<TAB>FLDCTL<CR> *field-name[:n],property-name,value[...],property-name,value]*

The *field-name* parameter must be defined in the current form definition. The *property-name* and *value* pairs can be one or more of the following:

<u>property-name</u>	<u>Abbreviation</u>	<u>Description</u>	<u>value</u>
VISIBLE	V	Makes the field visible or invisible.	1 = visible 0 = not visible
ENABLED	E	Makes the field enabled or disabled (grayed).	1 = enabled 0 = not enabled
BACKCOLOR	BC	Sets the background color of the field.	0 through 16, or D for default (see "Color Values", below)
FORECOLOR	FC	Sets the foreground (text color) of the field.	0 through 16, or D for default (see "Color Values", below)

Color Values:

<u>Color</u>	<u>value</u>	<u>Color</u>	<u>value</u>
Black	0	Gray	8
Red	1	Light Red	9
Green	2	Light Green	10
Yellow	3	Light Yellow	11
Blue	4	Light Blue	12
Magenta	5	Light Magenta	13
Cyan	6	Light Cyan	14
White	7	Bright White	15

The FLDCTL command may only be used on data fields.

FLDCTL command Examples:

Set the BACK_ORDERED field to enabled with red text on a white background:

```
EQ$<TAB>FLDCTL<CR>BACK_ORDERED,E,1,FC,4,BC,15<TAB>\\
```

Hide the INCOME field:

```
EQ$<TAB>FLDCTL<CR>INCOME,V,0<TAB>\\
```

LOCKFRM

LOCKFRM disables the currently displayed form until an UNLOCKFRM or DTAREC command is received. LOCKFRM may be abbreviated LF.

Format:

```
<TAB>LOCKFRM<CR>
```

This command allows the host program to prevent users from transmitting data while the host is still busy.

LSTDTA

LSTDTA populates or clears a standard or drop-down list box previously defined for a eQuate Form. LSTDTA may be abbreviated LD.

Format:

```
<TAB>LSTDTA<CR>list-name {CLEAR | {START | APPEND},item-1,item-2...,item-n}
```

CLEAR empties the named list box. START empties the list box then adds all items listed. APPEND adds all items listed without clearing the current contents of the list box.

Examples: Fill a list box:

```
EQ$<TAB>LSTDTA<CR>LIST_1,START,Bob,"Jon Doe"<TAB>\\
```

Add to the above list:

```
EQ$<TAB>LSTDTA<CR>LIST_1,APPEND,{Jane Doe},Marc<TAB>\\
```

Empty the above list:

```
EQ$<TAB>LSTDTA<CR>LIST_1,CLEAR<TAB>\\
```

Note: When embedded spaces are required in an item, the item must be enclosed in matching enclosure characters.

NEWFRM

NEWFRM tells eQuate to load a new form on top of the current one. NEWFRM may be abbreviated NF. The newly displayed form remains disabled until the host program explicitly enables the form with a DTAREC, REFRESH or UNLOCKFRM command.

Format:

```
<TAB>NEWFRM<CR>form-name
```

POSCURS

POSCURS positions the cursor to a specific field in the eQuate Form Window. POSCURS may be abbreviated PC.

Format:

```
<TAB>POSCURS<CR>field-name[:n]
```

The *field-name* parameter must specify an input field within the current form. If the *field-name* is a repeating field, *n* is the occurrence number of the field and must be a numeric value greater than zero.

REFRESH

Reset all field and control properties (changed by CTL or FLDCTL commands) to their designed values. REFRESH may be abbreviated RF.

Format :

<TAB>REFRESH<CR>

UAKCTL

UAKCTL updates the user-mapped action keys. UAKCTL may be abbreviated KC. This command is used to enable and disable user-mapped action keys.

Format:

<TAB>UAKCTL<CR> *action-key-control-name,n*

The *action-key-control-name* is the name of the action key as assigned during eQuate Form Design. Use the Action Script Editor of the eQuate Form Designer to view action key control names assigned to the form. When *n* is set to 1, the action key will be enabled; 0, disabled.

UMNCTL

UMNCTL updates the user menu items. UMNCTL may be abbreviated UC. This command is used to enable or disable menu items and to check or uncheck menu items. Optionally, menu item captions can be altered by this command.

Format:

<TAB>UMNCTL<CR> *menu-item-name,n1,n2[,caption]*

The *menu-item-name* is the name of the menu as assigned during eQuate Form Design. To view menu item names assigned to the form, Select Menu Designer from the Tools menu of the eQuate Form Designer Toolbar.

The *n1* parameter controls whether the named menu item is to be enabled or disabled, where a one (1) enables the menu item; a zero (0), disables it. The *n2* parameter controls whether the menu item is to be checked or unchecked, where a one (1) checks the menu item; a zero (0), removes the check mark. Caption is optional, and is used to replace the specified menu item's caption.

UNLOCKFRM

UNLOCKFRM enables the currently displayed form. UNLOCKFRM may be abbreviated UF.

Format:

<TAB>UNLOCKFRM<CR>

Defining ECM Control Strings in COBOL

Add the following the WORKING-STORAGE Section:

```
01  SPECIAL-CHARACTERS.

    05  SC-VALUES.
        10  SC-N-TAB          PIC 99 COMP VALUE 9.
        10  SC-N-CR          PIC 99 COMP VALUE 13.
        10  FILLER            PIC 9(5) COMP VALUE 0.
        10  FILLER            PIC X(3) VALUE 'EQ$'.
        10  FILLER            PIC 99 COMP VALUE 9.
        10  FILLER            PIC 99 COMP VALUE 9.
        10  FILLER            PIC X(3) VALUE '\\ '.
    05  SC-CHARS REDEFINES SC-VALUES.
        10  TAB               PIC X.
        10  CR                PIC X.
        10  FILLER            PIC XX.
        10  ECM-HEAD          PIC X(4) .
        10  ECM-TRAIL         PIC X(4) .
```

The following is an example of creating an ECM record in a COBOL program:

```
MOVE SPACES TO EQUATE-OUT-BUFF.
MOVE 1 TO ECM-MSG-SIZE.
STRING ECM-HEAD 'NEWFORM' CR 'ORDER-ENTRY'
      TAB 'POSCURS' CR 'ACCOUNT-NO'
      ECM-TRAIL
```

```

        DELIMITED BY SIZE INTO EQUATE-OUT-BUFF
        WITH POINTER ECM-MSG-SIZE.
SUBTRACT 1 FROM ECM-MSG-SIZE.
PERFORM SEND-EQUATE-MESSAGE.
...

```

Constructing ECM Data Records in COBOL

Because of communications handling by various transports, binary data (such as computational data fields) should never be transmitted to or from a remote device. To avoid loss of data during transport and to ease data manipulation within the host COBOL program, the following rules must be followed when creating ECM data:

1. All data must consist of ASCII characters in the range of 32 (space) to 126 (~). Any other characters may cause communications problems or be dropped during transport between the host computer and the eQuate workstation.
2. Unsigned numeric fields must be right zero filled with no embedded decimal point.
3. Signed numeric fields must be defined with SIGN LEADING SEPARATE which will add one additional character in front of the field containing either a + or - symbol.

The following is an example showing how the actual data record is defined and how the same data is defined in an ECM record.

First, the data record:

```

01 ACCOUNT-MASTER-RECORD.
   05 ACCOUNT-NO              PIC 9(9) .
   05 NAME                    PIC X(30) .
   05 CURR-BALANCE            PIC S9(7)V99 COMP.
   05 CREDIT-LIMIT            PIC 9(7)V99 COMP.
   05 ACTIVE-COUNT            PIC 9(5) .

```

The ECM data record:

```

01 ECM-ACCOUNT-RECORD.
   05 EA-ACCOUNT-NO           PIC 9(9) .
   05 EA-NAME                 PIC X(30) .
   05 EA-CURR-BALANCE         PIC S9(7)V99
                               SIGN LEADING SEPARATE.
   05 EA-CREDIT-LIMIT         PIC 9(7)V99.
   05 EA-ACTIVE-COUNT         PIC 9(5) .

```

Remember that the "V" in the picture clause for numeric fields is an implied decimal and the decimal point is not actually present. The EA-CURR-BALANCE field in the ECM record will actually be 10 characters in length. If it contained the value -102.4, it would appear like this:

```
-000010240
```

It is important to correctly define numeric data fields in your eQuate forms to ensure correct decimal alignment with host data. The EA-CURR-BALANCE field should be defined in the form as 10 characters long with 2 decimal positions.

The COBOL code to send the above record to eQuate and position the cursor to the ACCOUNT field would look like this:

```

MOVE ACCOUNT-NO TO EA-ACCOUNT-NO.
MOVE NAME TO EA-NAME.
MOVE CURR-BALANCE TO EA-CURR-BALANCE.
MOVE CREDIT-LIMIT TO EA-CREDIT-LIMIT.
MOVE ACTIVE-COUNT TO EA-ACTIVE-COUNT.
MOVE SPACES TO EQUATE-OUT-BUFF.
MOVE 1 TO ECM-MSG-SIZE.
STRING ECM-HEAD 'DTAREC' CR
      'CST001 '
      ECM-ACCOUNT-RECORD
      TAB 'POSCURS' CR 'ACCOUNT'
      ECM-TRAIL
      DELIMITED BY SIZE INTO EQUATE-OUT-BUFF
      WITH POINTER ECM-MSG-SIZE.
SUBTRACT 1 FROM ECM-MSG-SIZE.
PERFORM SEND-EQUATE-MESSAGE.

```

Note that the data definitions required for eQuate require no additional COBOL code. COBOL will make all necessary conversions automatically.

An eQuate Command Mode Example

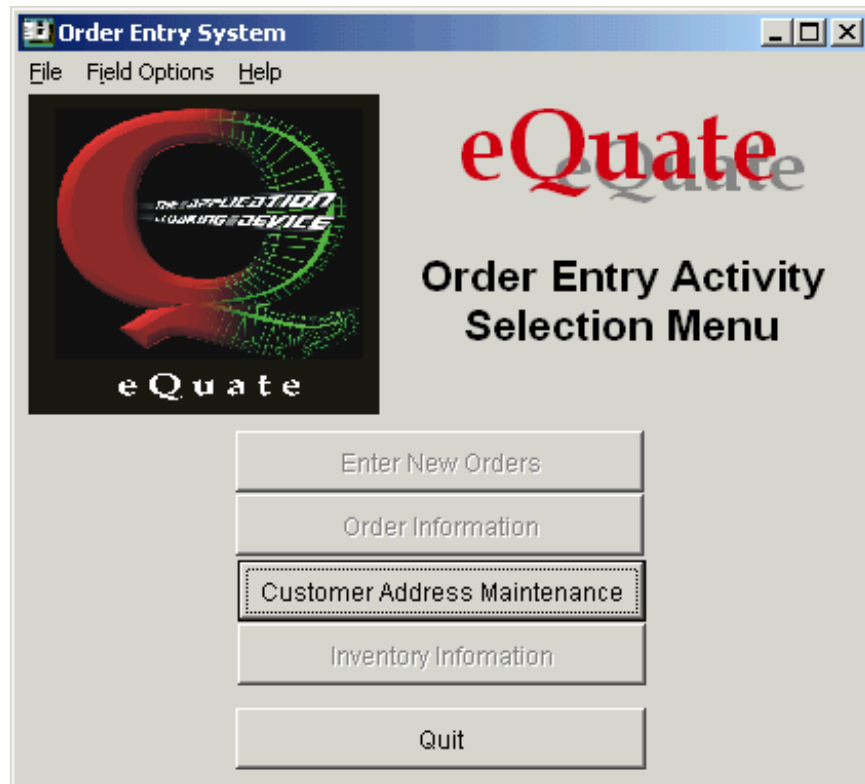
The COBOL example shown at the end of this topic calls for four eQuate forms: ECMSTART, ECMCUST1, ECMCUST2 and ECMCUST3. All the actions for these forms are shown in The Complete Set of ECM Actions and the corresponding COBOL host program is shown in The Complete COBOL Program. Be also sure to see ECM Command Format and Conventions.

The ECM Commands:

CTL	DTAFLD	DTAREC	ERRLST
FLDCTL	LOCKFRM	LSTDTA	NEWFRM
POSCURS	UAKCTL	UMNCTL	UNLOCKFRM

Also see, Defining ECM Control Strings in COBOL and Constructing ECM Data Records in COBOL.

ECMSTART is a start-up form initially designed in eQuate using the Add button on the Forms tab of the eQuate Application Manager. It contains no data fields — only a set of command buttons used to start a series of different host programs. Notice that all but the Customer Address Maintenance and Quit buttons are disabled. They are disabled by an eQuate Action that is executed when the form is initially displayed, and represent selections that are under development and not ready for use.



The ECMSTART form will be initially displayed when the user selects the corresponding eQuate Application from the eQuate Session Manager; however, the COBOL program will call for this form to be re-displayed whenever the user presses the Exit button on the ECMCUST1 form (see below). There is an eQuate Action that is executed when this button is pressed that causes the host program to call for the re-display of the form:

```
Sub EXIT_BUTTON()
    SetString "TRANS_CODE", "CUSTA"
    SetString "F_CODE", "X"
    XmitFrom "ACCOUNT"
End Sub
```

The above eQuate action sends the "CUSTA" transaction code plus an "X" function code (EB-F-CODE in the COBOL program) to the transaction.

The second form, ECMCUST1, is displayed when the user selects the Customer Address Maintenance from the ECMSTART form. The ECMCUST1 form contains many eQuate controls, that when selected, cause eQuate

actions to be executed. These actions either start a transaction program on the host or interface with other Windows applications. Just a few of these are described below.

The eQuate ECM command that causes the initial display of the second form, ECMCUST1, is the NEWFRM command (shown below). Notice in the figure above that some of the on the form buttons are initially disabled; i.e., they will only be activated once a query has been performed. Unlike the buttons disabled by an eQuate action on the ECMSTART form, these buttons are disabled by the host program sending CTL commands to the eQuate Session Manager (see the 2000-PROCESS-CUST-MAINT section on the COBOL program). Also, notice that when a new form is first displayed, it is in a locked state and must be unlocked with the UNLOCKFRM command prior to setting control properties on the form.

```
STRING ECM-HEAD 'NEWFRM' CR 'ECMCUST1'
  TAB 'UNLOCKFRM'
  TAB 'CTL' CR 'WRITE_FILE_BUTTON,ENABLED,0'
  TAB 'CTL' CR 'LIST_ORDERS_BUTTON,ENABLED,0'
  TAB 'CTL' CR 'DELETE_BUTTON,ENABLED,0'
  TAB 'POSCURS' CR 'ACCOUNT'
ECM-TRAIL
DELIMITED SIZE INTO ECM-BUFF
WITH POINTER ECM-MSG-LEN
```

The fields, TAB and CR, are command and parameter delimiters, respectively. The TAB character is an ASCII 9 and the CR (carriage return) is an ASCII 13. They are defined in the program as follows:

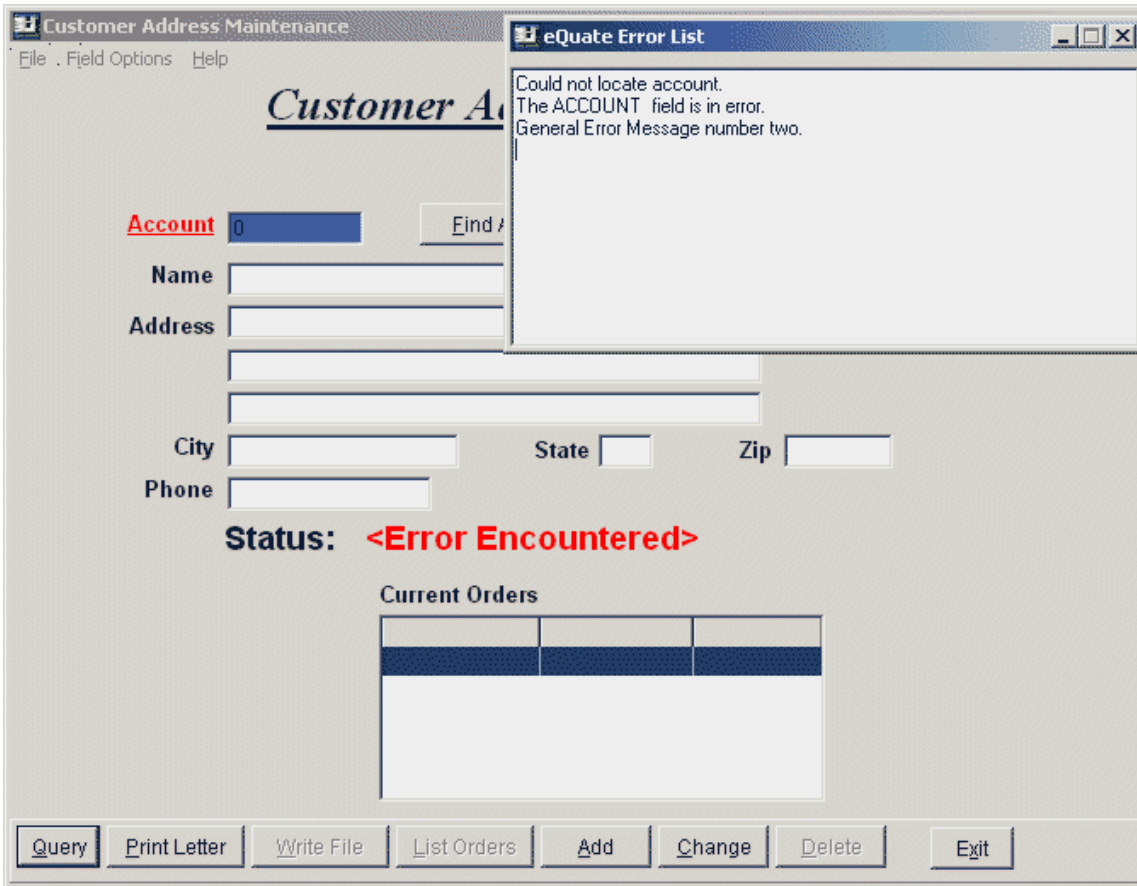
```
01 ECM-VALUES.
  05 FILLER PIC 99 COMP VALUE 9.
  05 FILLER PIC 99 COMP VALUE 13.
  05 FILLER PIC 9(5) COMP.
  05 FILLER PIC XXX VALUE 'EQ$'.
  05 FILLER PIC 99 COMP VALUE 9.
  05 FILLER PIC 99 COMP VALUE 9.
  05 FILLER PIC XXX VALUE '\\ '.
01 ECM-VALUES-R REDEFINES ECM-VALUES.
  05 TAB PIC X.
  05 CR PIC X.
  05 FILLER PIC XX.
```

```

05 ECM-HEAD          PIC X(4) .
05 ECM-TRAIL         PIC X(4) .

```

Another ECM command is used to communicate when an error has occurred within the host program.



In the example above, the host program has been unable to locate a particular account record and returns an ERRLIST command to the eQuate Session Manager causing an independent window titled, "eQuate Error List", to pop up. The host program code used to stage the error with the STRING command is as follows (also see the 4090-Q-INVKEY paragraph in The Complete COBOL Program).

```

IF ECM-ERR-LEN = 0
  MOVE SPACES TO ECM-ERR-MSG
  MOVE 1 TO ECM-ERR-LEN
  STRING ECM-HEAD 'ERRLIST' CR
    DELIMITED BY SIZE INTO ECM-ERR-MSG
    WITH POINTER ECM-ERR-LEN.

STRING '{' ECM-EMSG '}'
  DELIMITED BY '\ ' INTO ECM-ERR-MSG
  WITH POINTER ECM-ERR-LEN.
MOVE 1 TO WW-ERROR-SW.

```

Also, a eQuate Message Action checks for an "E" being returned from the host program, and if true, places the string, "<Error Encountered>", in the caption portion of the text label, "TL_STATUS". Initially, this label was set to "<OK>".

```

If GetString("F_CODE") = "E" then
  SetString "TL_STATUS", "<Error Encountered>"
else
  SetString "TL_STATUS", "<OK>"
End If

```

On a successful Query, the host program returns a DTAREC command that contains all the data fields. The fields are staged as a single "data record" ("CUSTA Q" plus the contents of CM-DATA) and returned to the eQuate Session Manager (also see the 3000-QUERY section in The Complete COBOL Program).

```

01 CM-DATA.
05 CD-ACCOUNT          PIC X(9) .

```

```

05 CD-CUSTNAME          PIC X(30) .
05 CD-ADDR1             PIC X(40) .
05 CD-ADDR2             PIC X(40) .
05 CD-ADDR3             PIC X(40) .
05 CD-CITY              PIC X(18) .
05 CD-STATE             PIC XX .
05 CD-ZIP               PIC X(9) .
05 CD-TELEPHONE         PIC X(10) .
.
.
MOVE SPACES TO ECM-BUFF.
MOVE 1 TO ECM-MSG-LEN.
MOVE SPACES TO CD-ORDER-IDENT.
STRING ECM-HEAD 'NEWFRM' CR 'ECMCUST1'
    TAB 'DTAREC' CR 'CUSTA Q'
    CM-DATA
    TAB 'UNLOCKFRM'
    TAB 'POSCURS' CR 'ACCOUNT'
    TAB 'CTL' CR 'WRITE_FILE_BUTTON,ENABLED,1'
    TAB 'CTL' CR 'LIST_ORDERS_BUTTON,ENABLED,1'
    TAB 'CTL' CR 'DELETE_BUTTON,ENABLED,1'
    DELIMITED BY SIZE INTO ECM-BUFF
    WITH POINTER ECM-MSG-LEN.
*** Add current order to list box in the same form.
PERFORM 4200-LIST-ORDERS.

```

The order of the fields in the record are dependent upon the order specified in the eQuate database when the form was designed. The order is set on the eQuate Form Data Fields window which is selected from the eQuate Form Manager dialog. The order may be changed by selecting a field and moving it vertically with the up and down arrow (↑↓) buttons; however, the order must always match that of the host program.

eQuate Form Data Fields (Command Mode Form: ECMCUST1)

Field Name	Type	Position	Length
TRANS_CODE	X	1	6
F_CODE	X	7	1
ACCOUNT	U	8	9
ORDER_IDENT	X	17	11
CUST_NAME	X	28	30
ADDRESS1	X	58	40
ADDRESS2	X	98	40
ADDRESS3	X	138	40
CITY	X	178	18
STATE	X	196	2
ZIP	X	198	9
PHONE	X	207	10

MEP data field sequence
 ↑ ↓ Gen COBOL Defs

List sort order
☐ Sort list by Name ☒ Sort list by position [Re-Sort](#)

+ Add Field X Delete Field

Data field attributes

Name: CUST_NAME

Location: Length: 30

Repeating Field
☐ Repeating Field Direction: ☒ Down ☐ Right
 Count: 0 Spacing: 1

Data type
☒ Any ☒ Force UPPER Case
☐ Alpha only
☐ Numeric only
☐ Signed ☐ Blank when zero Dec: 0

Other attributes
☒ Tab Stop ☐ Output Only ☐ Video Off

Justified (in Screen)
☒ Left ☐ Right ☐ Center

☐ Do NOT validate to host screen format

Apply Undo

OK Cancel Help

After a successful Query, the ECMCUST1 form will appear as follows. Notice the various buttons that were disabled are now enabled.

Customer Address Maintenance

File Field Options Help

Account 165220003 Find Accounts by Location

Name JERRY & ASSOCIATES IMPROVEMENT

Address 5019 GRIMES DRIVE

City GUNTERSVILLE **State** SC **Zip** 35976

Phone 2055820196

Status: <OK>

Current Orders

Order Ident	Amount	Date
01376660006	271.60	06/16/87
01376660106	256.50	06/16/87
01364160006	286.70	06/09/87
01344920006	2,070.00	05/28/87
01376650006	2,913.95	06/16/87

Query Print Letter Write File List Orders Add Change Delete Exit

The host program code used to fill the list box is similar to that shown for the error display shown earlier. The program accumulates the order entries into a standard COBOL array then uses the STRING command to stage the ECM LSTDTA command (also see the 4200-LIST-ORDERS section in The Complete COBOL Program). Notice that this command requires that the list box be defined/named in eQuate. In this example, the LST_ORD multi-column list box is being filled.

```

*** If first line put start of ECM msg in buffer.
    IF ECM-LST-CNT = 0
        IF SW-FIRST = 1
            *** If first item, indicate START if list, and insert list header.
            *** At this point we are adding to a query response created above.
            STRING TAB 'LSTDTA' CR 'LST_ORD START,'
                '(Order Ident,Amount,Date)'
                DELIMITED BY SIZE INTO ECM-BUFF
                WITH POINTER ECM-MSG-LEN
        ELSE
            *** If not first, start a new msg. and indicate APPENDING to list.
            MOVE SPACES TO ECM-BUFF
            MOVE 1 TO ECM-MSG-LEN
            STRING ECM-HEAD 'LSTDTA' CR 'LST_ORD APPEND,'
                DELIMITED BY SIZE INTO ECM-BUFF
                WITH POINTER ECM-MSG-LEN.

            MOVE 0 TO SW-FIRST.
    *** String this item into ECM message buffer.
    STRING '(' OR-ORDER-IDENT ',' ED-NUM ','
        OR-ENTRY-MO '/' OR-ENTRY-DA '/' OR-ENTRY-YR ')'
        DELIMITED SIZE INTO ECM-BUFF
        WITH POINTER ECM-MSG-LEN.
    ADD 1 TO ECM-LST-CNT.

```

The following figure shows the LST_ORD having been filled and selected (scrolled down).

Customer Address Maintenance

File Field Options Help

Account 165220003 Find Accounts by Location

Name JERRY & ASSOCIATES IMPROVEMENT

Address 5019 GRIMES DRIVE

City GUNTERSVILLE **State** SC **Zip** 35976

Phone 2055820196

Status: <OK>

Current Orders

Order Ident	Amount	Date
01376650006	2,913.95	06/16/87
01376650106	1,123.75	06/16/87
01364150006	9,375.05	06/09/87
01364150106	269.70	06/09/87
01387350006	256.50	06/23/87

Query Print Letter Write File List Orders Add Change Delete Exit

Another way that orders may be displayed is through the List Orders button on the button panel at the bottom of the ECMCUST1 form. This button invokes a eQuate Action that sends an "L" function to the host program in anticipation of the host program issuing an ECM NEWFRM command to load the third form, ECMCUST2.

```
Sub LIST_ORDERS_BUTTON()
    List orders
    SetSessionVar "Order_Total", "0"
    SetString "TRANS_CODE", "CUSTA "
    SetString "F_CODE", "L"
    XmitFrom "ACCOUNT"
End Sub
```

The host program code for the "L" function is similar to that used to fill the list box; however, the program must limit the number of order entries sent to eQuate to 12 or less as the ECMCUST2 form was designed to hold only 12 entries. As with the list box code, the program places the orders in a COBOL array, but limits the size of the array to 12 entries. In addition, instead of issuing an ECM LSTDTA command, a DTAREC is once again used to pass data to the program. The difference here is that the host program is calling for a "new form" in which to load the data (also see the 4000-LIST-ORDERS section in The Complete COBOL Program).

```
IF SUB1 > 12
    MOVE OR-ORDER-IDENT TO OD-NEXT-ORDER
    GO TO 4020-LO-END
ELSE
    MOVE OR-TOT-CHARGES TO OD-ORDER-AMT (SUB1)
    MOVE OR-ORDER-IDENT TO OD-ORDER-IDENT (SUB1) .
    GO TO 4005-READ-NEXT.
4010-AT-END.
    MOVE 'AT END' TO OD-NEXT-ORDER.
4020-LO-END.
    MOVE SPACES TO ECM-BUFF.
    MOVE 1 TO ECM-MSG-LEN.
    STRING ECM-HEAD 'NEWFRM' CR 'ECMCUST2'
        TAB 'DTAREC' CR 'CUSTA L' OR-DATA
        ECM-TRAIL
```

```

      DELIMITED BY SIZE INTO ECM-BUFF
      WITH POINTER ECM-MSG-LEN.
      PERFORM 8000-SEND.

```

Notice the code that is used to determine when the twelfth order has been exceeded (SUB1 > 12). When there are more than 12 orders, the program stores the next order (in this case the 13th) in the OD-NEXT-ORDER field. This field is defined as part of the form, but is not shown on the form; i.e., it is hidden from the user's view. The field is sent by the host program to the eQuate Session Manager along with the visible data fields. Its value will be sent back to the host program if the user selects the "More" button (see below).

List of Orders

File Field Options Help

Current Customer Order Amounts

Account: **165220003**

Orders	Amounts	Orders	Amounts
01376660006	271.60	01364150006	9375.05
01376660106	256.50	01364150106	269.70
01364160006	286.70	01387350006	256.50
01344920006	2070.00	01341830006	530.50
01376650006	2913.95	01387340006	8379.05
01376650106	1123.75	01341820006	1734.70

Accumulated Amount Total: **27468.00**

Return More First

The eQuate Action associated with the "More" button passes this value by transmitting from the NEXT_ORDER field.

```

      Sub MORE_BUTTON()
          SetString "F_CODE", "M"
          XmitFrom "Next_ORDER"
      End Sub

```

The host program contains code that will bypass the orders that have already been displayed (also see the 4100-LIST-ORDERS section in The Complete COBOL Program).

The ECMCUST2 form shown below contains the additional order entries for the customer.

The screenshot shows a window titled "List of Orders" with a menu bar (File, Field Options, Help). The main heading is "Current Customer Order Amounts" in red. Below it, the "Account:" field contains "165220003". There are two tables side-by-side, each with columns "Orders" and "Amounts". The first table has three rows of data: (01378690006, 11.35), (01341810006, 13636.40), and (01319260106, 1077.30). The second table is empty. At the bottom, the "Accumulated Amount Total:" field shows "42193.05". Navigation buttons "Return", "More", and "First" are at the bottom.

Orders	Amounts	Orders	Amounts
01378690006	11.35		
01341810006	13636.40		
01319260106	1077.30		

Accumulated Amount Total: 42193.05

Return More First

Another facet of this form's design is an eQuate Control that executes an Enable Action to provide a running total of all orders (see Accumulated Amount Total, above). The eQuate Host Message Action is executed each time data for this form is received from the host program. The action uses a session variable, "Order_Total," to accumulate the running total. It is only cleared if the eQuate Action associated with the First button is executed.

```
Sub HostMessage()
    Dim x as Integer
    Dim Tot as Double
    Dim Lst as string

    Tot = Val(GetSessionVar("Order_Total"))
    ' Get last value for accumulation between screens

    For x = 1 to 12
        Tot = Tot + Val(GetString("TOTAL_CHARGES:" + Str$(x)))
    next x
    SetSessionVar "Order_Total", format(Tot, "#0.00;\-#0.00;0")
    ' Save for next call

    SetString "LBL_TOT", format(Tot / 100, "#0.00;(#0.00);\Z\e\r\o")
End Sub
```

See also, The Complete Set of ECM Actions and The Complete COBOL Program.

The Complete Set of ECM Actions

ECMSTART Form

```
Option Explicit
Sub BTN_CUSTMAINT()
    Send "CUSTA I"
End Sub
Sub BTN_QUITE()
    CloseApp
End Sub
Sub FormActivate()
    ' Action for FormActivate
    ' Disable buttons that don't don anything now.
    SetState "BTN_ENTER_NEW", tpEnabled, False
    SetState "BTN_INFO", tpEnabled, False
```

```

        SetState "BTN_INVENTORY", tpEnabled, False
        ' Set the focus to the "Addrsss Maint." button.
        SetFocus "BTN_CUSTMAINT"
End Sub

```

ECMCUST1 Form

```
Option Explicit
```

```

Sub BTN_1()
    ' Find Accounts
    SetSessionVar "Find_State", GetString("STATE")
    SetSessionVar "Find_City", GetString("CITY")
    ClearDisplay
    SetString "TRANS_CODE", "CUSTA"
    SetString "F_CODE", "F"
    XmitFrom "CUST_NAME"
End Sub

Sub QUERY_BUTTON()
    ' Query
    SetString "TRANS_CODE", "CUSTA"
    SetString "F_CODE", "Q"
    XmitFrom "ACCOUNT"
End Sub

Sub PRINT_LETTER_BUTTON()
    ' Write letter
    Dim word6 As object
    Set word6 = CreateObject("Word.Basic")
    word6.FileNewDefault
    word6.ViewPage
    word6.InsertPara
    word6.Insert GetString("CUST_NAME") + Chr$(13) + GetString("ADDRESS1") + Chr$(13)
    if Trim$(GetString("ADDRESS2")) <> "" then
        word6.Insert GetString("ADDRESS2") + Chr$(13)
    end if
    If Trim$(GetString("ADDRESS3")) <> "" Then
        word6.Insert GetString("ADDRESS3") + Chr$(13)
    end if
    word6.insert Trim$(GetString("CITY")) + ", " + Trim$(GetString("STATE")) + " " +
Trim$(GetString("ZIP"))
    word6.InsertPara
    word6.InsertPara
    word6.Insert "Dear Valued Customer:"
    word6.InsertPara
    word6.InsertPara
    word6.Insert "The following is your account number and telephone number as stored in our
database:"
    word6.InsertPara
    word6.Bold 1
    word6.Insert "Account:" + Chr$(9) + GetString("ACCOUNT") + Chr$(13)
    word6.insert "Phone number:" + Chr$(9) + GetString("PHONE")
    word6.bold 0
    word6.InsertPara
    word6.Insert "If either of the above is incorrect, please contact us immediately"
    word6.InsertPara
    word6.InsertPara
    word6.Insert "Sincerely,"
    word6.InsertPara
    word6.Insert " John J. Doe"
    word6.Insert " Customer Services Rep."
    word6.InsertPara
    word6.FilePrint
    MsgBox "Your letter has been sent to the printer."
End Sub

Sub WRITE_FILE_BUTTON()
    ' Write file
    Begin Dialog FILE_DIALOG 120, 87, 167, 78, "Address File Name"
        OKBUTTON 112, 50, 38, 11
        TEXTBOX 23, 23, 131, 15, .FNAME
        TEXT 23, 15, 38, 7, "Enter file name"
        CANCELBUTTON 23, 50, 38, 11
    End Dialog
    Dim Dlg as FILE_DIALOG

```

```

    Dim Fn as Integer

    if Not Dialog(Dlg) then
        msgbox "Operation cancelled"
        exit sub
    end if
    if Rtrim$(Dlg.FNAME) = "" then
        MsgBox "File name not entered, operation cancelled."
        exit sub
    End If

    Fn = FreeFile
    Open Dlg.FNAME for Output as Fn
    Print #Fn, GetString("CUST_NAME")
    Print #Fn, GetString("ADDRESS1")
    Print #Fn, GetString("ADDRESS2")
    Print #Fn, GetString("CITY") + ", " + GetString("STATE") + " " + GetString("ZIP")
    Close Fn
    MsgBox "Address Written to file"
End SUB
Sub LIST_ORDERS_BUTTON()
    ' List orders
    SetSessionVar "Order_Total", "0"
    SetString "TRANS_CODE", "CUSTA "
    SetString "F_CODE", "L"
    XmitFrom "ACCOUNT"
End Sub
Sub ADD_BUTTON()
    ' Add
    If EditCust() Then
        SetString "TRANS_CODE", "CUSTA"
        SetString "F_CODE", "A"
        XmitFrom "PHONE"
    End If
End Sub
Sub CHANGE_BUTTON()
    ' Change
    If EditCust() Then
        SetString "TRANS_CODE", "CUSTA"
        SetString "F_CODE", "C"
        XmitFrom "PHONE"
    End If
End Sub
Sub DELETE_BUTTON()
    ' Delete
    If MsgBox("Are you sure you want to delete this account?", 32 + 4) = 6 Then
        SetString "TRANS_CODE", "CUSTA"
        SetString "F_CODE", "D"
        XmitFrom "ACCOUNT"
    Else
        MsgBox "Delete Cancelled."
    End If
End Sub
Sub EXIT_BUTTON()
    SetString "TRANS_CODE", "CUSTA"
    SetString "F_CODE", "X"
    XmitFrom "ACCOUNT"
End Sub
Function EditCust()
    Dim Acc as String
    Dim Zip as String
    Dim i as Integer
    Dim Errors As Integer
    Dim msg As String
    Dim OutMsg as String

    Acc = GetString("ACCOUNT")
    If Left$(Acc, 1) <> "1" And Left$(Acc, 1) <> "2" And Left$(Acc, 1) <> "3" Then
        msg = msg + "Account must start with 1, 2 or 3." + Chr$(13)
        Errors = Errors + 1
    End If
End Function

```

```

End If
If Trim$(GetString("CUST_NAME")) = "" Then
    msg = msg + "You must enter a customer name" + Chr$(13)
    Errors = Errors + 1
End If
If GetString("STATE") = " " Then
    msg = msg + "State abbreviation must be entered." + Chr$(13)
    Errors = Errors + 1
End If
Zip = GetString("ZIP")
For i = 1 To 5
    If Mid$(Zip, i, 1) < "0" Or Mid$(Zip, i, 1) > "9" Then
        msg = msg + "ZIP must start with 5 numbers." + Chr$(13)
        Errors = Errors + 1
        Exit For
    End If
Next i

If Errors > 0 Then
    OutMsg = "The following errors were detected:" + Chr$(13) + Chr$(13) + msg
    MsgBox OutMsg, 48, "Errors Detected"
    EditCust = False
Else
    EditCust = True
End If
End Function
Sub HostMessage()
    SetString "TL_STATUS", "*****"
    If GetString("F_CODE") = "" then
        CloseApp
    Else
        If GetString("F_CODE") = "E" Then
            SetString "TL_STATUS", "<Error Encountered>"
        Else
            SetString "TL_STATUS", "<OK>"
        End If
    End If
End Sub

```

ECMCUST2 Form

```

Sub Return_BUTTON()
    SetString "Trans_code", "CUSTA "
    SetString "F_CODE", "R"
    XmitFrom "ACCOUNT"
End Sub
Sub MORE_BUTTON()
    SetString "F_CODE", "M"
    XmitFrom "Next_ORDER"
End Sub
Sub FIRST_BUTTON()
    SetString "F_CODE", "L"
    XmitFrom "NEXT_ORDER"
    SetSessionVar "Order_Total", "0"
End Sub
Sub HostMessage()
    Dim x as Integer
    Dim Tot as Double
    Dim Lst as string

    ' Get last value for accumulation between screens
    Tot = Val(GetSessionVar("Order_Total"))
    For x = 1 to 12
        Tot = Tot + Val(GetString("TOTAL_CHARGES:" + Str$(x)))
    Next x
    SetSessionVar "Order_Total", format(Tot, "#0.00;\-#0.00;0") ' Save for next call
    SetString "LBL_TOT", format(Tot / 100, "#0.00;(#0.00);\Z\e\r\o")
End Sub

```

ECMCYST3 Form

Option Explicit

```

Sub BTN_1()
    ' Pick customer
    if ListCount("CUST_LIST") > 0 then
        SetString "ACCOUNT", ListGetColText("CUST_LIST",1)
        SetString "F_CODE", "Q"
        XmitFrom "ACCOUNT"
    Else
        MsgBox "No customers in list."
    End If
End SUB
Sub BTN_2()
    ' Cancel
    SetString "TRANS_CODE", "CUSTA"
    SetString "F_CODE", "I"
    SetString "ACCOUNT", "      "
    XmitFrom "ACCOUNT"
End Sub
Sub BTN_3()
    'Find all customers in State/City area
    SetString "TRANS_CODE", "CUSTA"
    SetString "F_CODE", "F"
    SetString "ACCOUNT", "      "
    XmitFrom "CITY"
End Sub
Sub BTN_4()
    SetString "Lbl_Count", Str(ListCount("CUST_LIST"))
End Sub
Sub FormInitial()
    SetString "STATE_CODE", GetSessionVar("Find_State")
    SetString "CITY", GetSessionVar("Find_City")
    SetCursorField "STATE_CODE"
End SUB

```

The Complete COBOL Program

```

IDENTIFICATION DIVISION.
PROGRAM-ID. CUSTA.
*****
*   This is an example of an eQuate Command mode (ECM)
*   transaction program.
*****
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER.      UNIVAC-1100-60.
OBJECT-COMPUTER.      UNIVAC-1100-60 MEMORY SIZE 3 MODULES.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT CUSTFILE ASSIGN TO DISC
    ORGANIZATION IS INDEXED
    ACCESS MODE IS DYNAMIC
    RECORD KEY IS CM-ACCOUNT
        ALTERNATE RECORD KEY IS CM-LOCKEY WITH DUPLICATES.
    SELECT ORDERFILE ASSIGN TO DISC
    ORGANIZATION IS INDEXED
    ACCESS MODE IS DYNAMIC
    RECORD KEY IS OR-ORDER-IDENT
        ALTERNATE RECORD KEY IS OR-CUSTKEY WITH DUPLICATES.
DATA DIVISION.
FILE SECTION.
FD  CUSTFILE
    LABEL RECORDS ARE STANDARD.
01  CUSTOMER-MASTER-REC.
** Primary key **
   05  CM-ACCOUNT                      PIC X(9).
** Secondary key, duplicates allowed **
   05  CM-LOCKEY.
       10  CM-STATE                      PIC XX.

```

```

      10 CM-CITY PIC X(18).
05 CM-CUSTNAME PIC X(33).
05 CM-ADDR1 PIC X(30).
05 CM-ADDR2 PIC X(30).
05 CM-ADDR3 PIC X(30).
05 CM-ZIP PIC X(9).
05 CM-TELEPHONE.
      10 CM-AREACODE PIC XXX.
      10 CM-EXCHANGE PIC XXX.
      10 CM-TELNUM PIC XXXX.
05 FILLER PIC X(29).
FD ORDERFILE
  LABEL RECORDS ARE STANDARD.
01 ORDER-RECORD.
** Secondary key, duplicates allowed **
      05 OR-CUSTKEY.
          10 OR-CUSTDIV PIC 9.
          10 OR-CUSTNUM PIC X(5).
          10 OR-CUSTSHIPTO PIC 999.
** Primary key **
      05 OR-ORDER-IDENT.
          10 OR-ORDER-LOC PIC 99.
          10 OR-ORDER-KEY PIC X(7).
          10 OR-SHIP-LOC PIC 99.
05 OR-ORDER-TYPE-CODE PIC X.
05 OR-PRODLIN-CODE PIC X.
05 OR-ENTRY-DATE.
          10 OR-ENTRY-MO PIC XX.
          10 OR-ENTRY-DA PIC XX.
          10 OR-ENTRY-YR PIC XX.
05 OR-BUYER.
          10 OR-BYPASS PIC X.
          10 FILLER PIC X(10).
05 OR-PURCHASE-ORD PIC X(8).
05 OR-REQ-SHIPDATE PIC X(6).
05 OR-SHIP-VIA PIC X(11).
05 OR-CREDIT-HOLD PIC X.
05 OR-HOLD-CODE PIC X.
05 OR-INVOICE-CODE PIC X.
05 OR-BOL-PRT-CODE PIC X.
05 OR-PAY-METHOD-CODE PIC XXX.
05 OR-WORKORD-CODE PIC X.
05 OR-SPECIAL-TERMS PIC X(20).
05 OR-TERMS.
          10 OR-TERM-CODE PIC X.
          10 OR-TERM-PER PIC V9(4) COMP.
          10 OR-TERM-DATE-DAYS PIC 9(6).
05 OR-DELETE-FLAG PIC X.
05 OR-INPROCESS-HOLD PIC X.
05 FILLER PIC XX.
05 OR-ACTUAL-SHIP-DATE.
          10 OR-SHIP-YR PIC XX.
          10 OR-SHIP-MO PIC XX.
          10 OR-SHIP-DA PIC XX.
05 OR-PIECES PIC 9(5) COMP.
05 FILLER PIC XX.
05 OR-WEIGHT PIC 9(7) COMP.
05 OR-SHIP-FEE PIC 9(5)V99 COMP.
05 OR-TOT-CHARGES PIC 9(6)V99 COMP.
05 OR-TOT-CLC PIC 9(6)V99 COMP.
05 OR-DISCOUNT PIC 9(5)V99 COMP.
05 OR-CREDIT-REL-DATE.
          10 OR-CREL-YR PIC XX.
          10 OR-CREL-MO PIC XX.

```

```

    10 OR-CREL-DA          PIC XX.
05 OR-WORKORD-PRT-DATE.
    10 OR-WKORD-PRT-YR     PIC XX.
    10 OR-WKORD-PRT-MO     PIC XX.
    10 OR-WKORD-PRT-DA     PIC XX.
05 OR-STATE-TAX           PIC S9(4)V99 COMP.
05 OR-CITY-TAX            PIC S9(4)V99 COMP.
05 OR-COUNTY-TAX          PIC S9(4)V99 COMP.
05 OR-CREDIT-USERID       PIC X(8).
05 OR-NBR-PALLETS         PIC S9(2) COMP.
05 OR-PALLET-CHG          PIC S9(3)V99 COMP.
05 OR-TOT-PALLET-COST     PIC S9(5)V99 COMP.
05 OR-AUTHDLR-CODE        PIC X.
05 OR-LINE-COUNT          PIC 9(10) COMP.
05 OR-ORDER-LINE-DATA OCCURS 1 TO 50
    DEPENDING ON OR-LINE-COUNT.
*** Order line item data ***
    10 OR-PRODUCT         PIC X(6).
    10 OR-TYPE-ORD-CODE    PIC X.
    10 OR-QUANTITY         PIC S9(5) COMP.
    10 OR-UNIT-PRICE       PIC 9(5)V99 COMP.
    10 OR-DESC             PIC X(25).
    10 OR-WEIGHT           PIC 9(5)V99 COMP.
    10 OR-PACKAGE          PIC X(8).
    10 OR-PRICE-CODE       PIC XX.
    10 OR-BOL-KEY          PIC 999 COMP.
    10 OR-TAX-CODE         PIC X.
    10 OR-REG-CODE         PIC X.
    10 OR-SHIP-QTY         PIC 9(5) COMP.
    10 OR-BILL-ONLY-CODE   PIC X.
    10 OR-PRICE-CHANGE     PIC X.
    10 OR-LAST-DATE        PIC X(6).
    10 OR-PRIORITY         PIC X.
    10 OR-EXCEPTION-SW     PIC X.
    10 OR-SUB-ITEM        PIC X(6).

WORKING-STORAGE SECTION.
01 SUB0                   PIC 9(10) COMP.
01 SUB1                   PIC 9(10) COMP.
01 CSF-IMAGE              PIC X(80) VALUE SPACES.
01 CSF-STATUS             PIC S9(10) COMP.
01 WW-KEY-ERROR           PIC 9 VALUE 0.
01 WW-ERROR-SW            PIC 9 VALUE 0.
01 WW-FILES-OPEN          PIC 9 VALUE 0.
01 WW-BUFF-ADR            PIC 9(10) COMP.
01 SW-MORE                PIC 9(10) COMP.
01 SW-FIRST               PIC 9(10) COMP.
01 ED-NUM                 PIC ZZZ,ZZZ.99.
*** Definitions of eQuate ECM record special characters.
01 ECM-VALUES.
    05 FILLER              PIC 99 COMP VALUE 9.
    05 FILLER              PIC 99 COMP VALUE 13.
    05 FILLER              PIC 9(5) COMP.
    05 FILLER              PIC XXX VALUE 'EQ$'.
    05 FILLER              PIC 99 COMP VALUE 9.
    05 FILLER              PIC 99 COMP VALUE 9.
    05 FILLER              PIC XXX VALUE '\\ '.
01 ECM-VALUES-R REDEFINES ECM-VALUES.
    05 TAB                 PIC X.
    05 CR                 PIC X.
    05 FILLER              PIC XX.
    05 ECM-HEAD            PIC X(4).
    05 ECM-TRAIL           PIC X(4).
*** eQuate ECM Message buffer for building ERRLST output.

```

```

01 ECM-MSG PIC X(80).
01 ECM-ERR-LEN PIC 9(10) COMP.
01 ECM-ERR-MSG PIC X(4000).
01 ECM-LST-CNT PIC 9(10) COMP.
*** eQuate ECM record buffer.
01 ECM-MSG-LEN PIC 9(10) COMP.
01 ECM-BUFF PIC X(4000).
01 ECM-BUFF-R1 REDEFINES ECM-BUFF.
    05 ECM-C OCCURS 4000 PIC X.
01 ECM-BUFF-R2 REDEFINES ECM-BUFF.
    05 EB-TX.
        10 EB-TRANS PIC X(6).
        10 EB-F-CODE PIC X.
    05 EB-IN-DATA PIC X(3993).
    05 EB-IN-DATA-R1 REDEFINES EB-IN-DATA.
        10 EB-ACCOUNT PIC X(9).
        10 FILLER PIC X(3984).
    05 EB-IN-DATA-R2 REDEFINES EB-IN-DATA.
        10 EB-ORDER-IDENT PIC X(11).
        10 FILLER PIC X(3982).
    05 EB-IN-DATA-R3 REDEFINES EB-IN-DATA.
        10 EB-ACCOUNTX PIC X(9).
        10 EB-STATE PIC XX.
        10 EB-CITY PIC X(18).
        10 FILLER PIC X(3964).
*** Customer Master eQuate data record layout.
*** This is the format in which customer data is
*** passed between this program and eQuate.
*** NOTE that field sizes and sequence differ from the
*** data record and match the layout specified in the
*** eQuate form.
01 CM-DATA.
    05 CD-ACCOUNT PIC X(9).
    05 CD-ORDER-IDENT PIC X(11).
    05 CD-CUSTNAME PIC X(30).
    05 CD-ADDR1 PIC X(40).
    05 CD-ADDR2 PIC X(40).
    05 CD-ADDR3 PIC X(40).
    05 CD-CITY PIC X(18).
    05 CD-STATE PIC XX.
    05 CD-ZIP PIC X(9).
    05 CD-TELEPHONE PIC X(10).
*** Order list display data record layout.
*** This is the format in which lists of
*** order amounts are sent to eQuate.
01 OR-DATA.
    05 OD-ACCOUNT PIC X(9).
    05 OD-NEXT-ORDER PIC X(11).
    05 OD-ORDER-IDENT OCCURS 12 PIC X(11).
    05 OD-ORDER-AMT OCCURS 12 PIC S9(7)V99
        SIGN LEADING SEPARATE.
*** Item list display data record layout.
*** This is the format in which lists of
*** item amounts are sent to eQuate.
01 OR-ITEM-DATA.
    05 OI-ACCOUNT PIC X(9).
    05 OI-ORDER-IDENT PIC X(11).
    05 OI-NEXT-PRODUCT PIC X(6).
    05 OI-PRODUCT OCCURS 17 PIC X(6).
    05 OI-DESC OCCURS 17 PIC X(25).
    05 OI-QUANTITY OCCURS 17 PIC S9(5)
        SIGN LEADING SEPARATE.
    05 OI-UNIT-PRICE OCCURS 17 PIC S9(5)V99
        SIGN LEADING SEPARATE.

```

```

05 OI-SHIP-QTY      OCCURS 17 PIC S9(5)
                                SIGN LEADING SEPARATE.

01 HOLD-LOCKEY.
05 HL-STATE          PIC XX.
05 HL-CITY           PIC X(18).
*** MCB Packet Definition.
    COPY MCBDEF-COB.

PROCEDURE DIVISION.
0100-MAIN SECTION.
0110-MLS.
    MOVE 0 TO WW-ERROR-SW.
    PERFORM 1000-INITIALIZE.
    IF WW-ERROR-SW > 0
        GO TO 0180-TERM.
    PERFORM 2000-PROCESS-CUST-MAINT.
0180-TERM.
    PERFORM 1200-CLEAN-UP.
0199-EXIT.
    STOP RUN.
*****
** Setup for current transaction.
*****
1000-INITIALIZE SECTION.
1001-I.
*** Initialeze MCB packet interface.
    MOVE LOW-VALUES TO P-MCB-PACKET.
    MOVE 0 TO P-MCB-STATUS.
    MOVE P-TRINIT TO P-FUNC.
*** Put address of ECM-BUFF into MCB packet (P-BUFF).
    CALL 'CLOCTE' USING ECM-BUFF WW-BUFF-ADR.
    MOVE WW-BUFF-ADR TO P-BUFF.
    MOVE 4000 TO P-LENGTH.
    MOVE 0 TO P-BIT1.
    MOVE 1 TO P-VER.
    MOVE 1 TO P-LVL.
    MOVE 1 TO P-BIT2.
    MOVE 63 TO P-AUX.
    CALL 'CMCB' USING P-MCB-PACKET P-MCB-STATUS.
    IF P-STATBIT NOT = 0
        PERFORM 9900-MCB-ERROR.
*** Assign PCIOS files used in this program.
    MOVE '@ASG,A KMS*CUSTFILE. . ' TO CSF-IMAGE.
    CALL 'ERACSF$' USING CSF-IMAGE CSF-STATUS.
    IF CSF-STATUS < 0
        MOVE 'Can not assign CUSTFILE\' TO ECM-EMSG
        PERFORM 9000-EMSG
        PERFORM 9100-SEND-ERR
        GO TO 1099-EXIT.
    MOVE '@ASG,A KMS*ORDERFILE. . ' TO CSF-IMAGE.
    CALL 'ERACSF$' USING CSF-IMAGE CSF-STATUS.
    IF CSF-STATUS < 0
        MOVE 'Can not assign CUSTFILE\' TO ECM-EMSG
        PERFORM 9000-EMSG
        PERFORM 9100-SEND-ERR
        GO TO 1099-EXIT.
    OPEN I-O CUSTFILE.
    OPEN INPUT ORDERFILE.
    MOVE 1 TO WW-FILES-OPEN.
1099-EXIT.
    EXIT.
*****
** Clean up before ending transation.
*****

```

```

1200-CLEAN-UP SECTION.
1201-CU.
    IF WW-FILES-OPEN = 1
        CLOSE CUSTFILE ORDERFILE
        MOVE 0 TO WW-FILES-OPEN.
*** ALWAYS FREE PCIOS FILES.
    MOVE '@FREE IQ$DEMO*CUSTFILE. . . ' TO CSF-IMAGE.
    CALL 'ERACSF$' USING CSF-IMAGE CSF-STATUS.
    MOVE '@FREE KMS*ORDERFILE. . . ' TO CSF-IMAGE.
    CALL 'ERACSF$' USING CSF-IMAGE CSF-STATUS.
*** TERMINATE MCB PACKET INTERFACE.
    MOVE LOW-VALUES TO P-MCB-PACKET.
    MOVE 0 TO P-MCB-STATUS.
    MOVE P-TERM TO P-FUNC.
    CALL 'CMCB' USING P-MCB-PACKET P-MCB-STATUS.
1299-EXIT.
    EXIT.
*****
*** Determine transaction function and perform required operation.
*****
2000-PROCESS-CUST-MAINT SECTION.
2010-PCM.
    MOVE 0 TO WW-KEY-ERROR.
*** Initial transaction to display empty eQuate form.
*** This is not normally necessary in eQuate ECM mode.
    IF EB-F-CODE = 'I'
        MOVE SPACES TO ECM-BUFF
        MOVE 1 TO ECM-MSG-LEN
        STRING ECM-HEAD 'NEWFRM' CR 'ECMCUST1'
            TAB 'UNLOCKFRM'
            TAB 'CTL' CR 'WRITE_FILE_BUTTON,ENABLED,0'
            TAB 'CTL' CR 'LIST_ORDERS_BUTTON,ENABLED,0'
            TAB 'CTL' CR 'DELETE_BUTTON,ENABLED,0'
            TAB 'POSCURS' CR 'ACCOUNT'
        ECM-TRAIL
        DELIMITED SIZE INTO ECM-BUFF
        WITH POINTER ECM-MSG-LEN
        PERFORM 8000-SEND
        GO TO 2099-EXIT.
*** Terminate transaction
    IF EB-F-CODE = 'X'
        MOVE SPACES TO ECM-BUFF
        MOVE 1 TO ECM-MSG-LEN
        STRING ECM-HEAD 'NEWFRM' CR 'ECMSTART'
            TAB 'UNLOCKFRM'
        ECM-TRAIL
        DELIMITED BY SIZE INTO ECM-BUFF
        WITH POINTER ECM-MSG-LEN
        PERFORM 8000-SEND
        GO TO 2099-EXIT.
*** Query function.
    IF EB-F-CODE = 'Q' OR 'Z'
        PERFORM 3000-QUERY
        GO TO 2099-EXIT.
*** Query function returning from orders window.
    IF EB-F-CODE = 'R'
        PERFORM 3100-QUERY
        GO TO 2099-EXIT.
*** Add customer record function.
    IF EB-F-CODE = 'A'
        PERFORM 3200-ADD
        GO TO 2099-EXIT.
*** Change (replace) customer record function.
    IF EB-F-CODE = 'C'

```

```

        PERFORM 3300-CHANGE
        GO TO 2099-EXIT.
*** Delete customer record funciton.
    IF EB-F-CODE = 'D'
        PERFORM 3400-DELETE
        GO TO 2099-EXIT.
*** List FIRST order totals for customer.
    IF EB-F-CODE = 'L'
        PERFORM 4000-LIST-ORDERS
        GO TO 2099-EXIT.
*** List MORE order totals for customer.
    IF EB-F-CODE = 'M'
        PERFORM 4100-LIST-ORDERS
        GO TO 2099-EXIT.
*** List FIRST item totals for an order.
    IF EB-F-CODE = 'P'
        PERFORM 5000-LIST-ITEMS
        GO TO 2099-EXIT.
*** List MORE item totals for an order.
    IF EB-F-CODE = 'N'
        PERFORM 5100-LIST-ITEMS
        GO TO 2099-EXIT.
*** List customers by state and city (optional)
    IF EB-F-CODE = 'F'
        PERFORM 5500-LIST-CUST
        GO TO 2099-EXIT.
*** Invalid function code received.
    MOVE 'Invalid function code received.\' TO ECM-MSG.
    PERFORM 9000-MSG.
    PERFORM 9100-SEND-ERR.
2099-EXIT.
EXIT.
*****
*** The following sections handle all customer address file
*** maintenance functions.
*****
*** Query for Customer from Maint (EB-F-CODE=Q).
*****
3000-QUERY SECTION.
3010-Q.
    MOVE EB-ACCOUNT TO CM-ACCOUNT.
    READ CUSTFILE RECORD
        INVALID KEY GO TO 3090-Q-INVKEY.
    PERFORM 6000-CUST-TO-EQUATE.
    MOVE SPACES TO ECM-BUFF.
    MOVE 1 TO ECM-MSG-LEN.
    MOVE SPACES TO CD-ORDER-IDENT.
    STRING ECM-HEAD 'NEWFRM' CR 'ECMCUST1'
        TAB 'DTAREC' CR '      Q'
        CM-DATA
        TAB 'UNLOCKFRM'
        TAB 'POSCURS' CR 'ACCOUNT'
        TAB 'CTL' CR 'WRITE_FILE_BUTTON,ENABLED,1'
        TAB 'CTL' CR 'LIST_ORDERS_BUTTON,ENABLED,1'
        TAB 'CTL' CR 'DELETE_BUTTON,ENABLED,1'
        DELIMITED BY SIZE INTO ECM-BUFF
        WITH POINTER ECM-MSG-LEN.
*** Add current order to list box in the same form.
    PERFORM 4200-LIST-ORDERS.
    GO TO 3099-EXIT.
3090-Q-INVKEY.
    MOVE 'Could not locate account.\' TO ECM-MSG.
    PERFORM 9000-MSG.
    MOVE '1:ACCOUNT \' TO ECM-MSG.

```

```

PERFORM 9000-EMSG.
MOVE '2:\' TO ECM-EMSG.
PERFORM 9000-EMSG.
PERFORM 9100-SEND-ERR.
MOVE SPACES TO ECM-BUFF.
MOVE 1 TO ECM-MSG-LEN.
STRING ECM-HEAD 'NEWFRM' CR 'ECMCUST1'
    TAB 'UNLOCKFRM'
    TAB 'DTAFLD' CR 'TRANS_CODE,"CUSTA "'
    TAB 'DTAFLD' CR 'F_CODE,"E"'
    TAB 'FC' CR 'ACCOUNT,BC,12'
    TAB 'POSCURS' CR 'ACCOUNT'
    TAB 'CTL' CR 'WRITE_FILE_BUTTON,ENABLED,0'
    TAB 'CTL' CR 'LIST_ORDERS_BUTTON,ENABLED,0'
    TAB 'CTL' CR 'DELETE_BUTTON,ENABLED,0'
    TAB 'LSTDTA' CR 'LST_ORD CLEAR'
ECM-TRAIL
DELIMITED BY SIZE INTO ECM-BUFF
WITH POINTER ECM-MSG-LEN.
PERFORM 8000-SEND.
3099-EXIT.
EXIT.
*****
*** Query for Customer on return from Orders/Items (EB-F-CODE=R).
*****
3100-QUERY SECTION.
3110-Q.
    MOVE EB-ACCOUNT TO CM-ACCOUNT.
*** If returning from list-orders, redisplay maint window.
    MOVE SPACES TO ECM-BUFF.
    MOVE 1 TO ECM-MSG-LEN.
    STRING ECM-HEAD 'NEWFRM' CR 'ECMCUST1'
        TAB 'POSCURS' CR 'ACCOUNT'
        TAB 'UNLOCKFORM'
    ECM-TRAIL
    DELIMITED BY SIZE INTO ECM-BUFF
    WITH POINTER ECM-MSG-LEN
    PERFORM 8000-SEND.
    READ CUSTFILE RECORD
        INVALID KEY GO TO 3190-Q-INVKEY.
    PERFORM 6000-CUST-TO-EQUATE.
    MOVE SPACES TO ECM-BUFF.
    MOVE 1 TO ECM-MSG-LEN.
    STRING ECM-HEAD 'DTAREC' CR 'CUSTA Q'
        CM-DATA
        TAB 'UNLOCKFRM'
        TAB 'POSCURS' CR 'ACCOUNT'
        TAB 'CTL' CR 'WRITE_FILE_BUTTON,ENABLED,1'
        TAB 'CTL' CR 'LIST_ORDERS_BUTTON,ENABLED,1'
        TAB 'CTL' CR 'DELETE_BUTTON,ENABLED,1'
        DELIMITED BY SIZE INTO ECM-BUFF
        WITH POINTER ECM-MSG-LEN.
    PERFORM 4200-LIST-ORDERS.
    GO TO 3199-EXIT.
3190-Q-INVKEY.
    MOVE 'Could not locate that account.\' TO ECM-EMSG.
    PERFORM 9000-EMSG.
    PERFORM 9100-SEND-ERR.
    MOVE SPACES TO ECM-BUFF.
    MOVE 1 TO ECM-MSG-LEN.
    STRING ECM-HEAD 'DTAFLD' CR 'TRANS_CODE,"CUSTA "'
        TAB 'FC' CR 'ACCOUNT,BC,12'
        TAB 'DTAFLD' CR 'F_CODE,"E"'
        TAB 'POSCURS' CR 'ACCOUNT'

```

```

        TAB 'CTL' CR 'WRITE_FILE_BUTTON,ENABLED,0'
        TAB 'CTL' CR 'LIST_ORDERS_BUTTON,ENABLED,0'
        TAB 'CTL' CR 'DELETE_BUTTON,ENABLED,0'
        TAB 'LSTDTA' CR 'LST_ORD CLEAR'
    ECM-TRAIL
    DELIMITED BY SIZE INTO ECM-BUFF
    WITH POINTER ECM-MSG-LEN.
    PERFORM 8000-SEND.
3199-EXIT.
    EXIT.
*****
*** Add a new Customer (EB-F-CODE=A).
*****
3200-ADD SECTION.
3210-A.
    MOVE EB-IN-DATA TO CM-DATA.
    PERFORM 6100-EQUATE-TO-CUST.
    WRITE CUSTOMER-MASTER-REC
        INVALID GO TO 3290-A-INVKEY.
    MOVE SPACES TO ECM-BUFF.
    MOVE 1 TO ECM-MSG-LEN.
    STRING ECM-HEAD 'DTAFLD' CR 'TRANS_CODE,"CUSTA "'
        TAB 'DTAFLD' CR 'F_CODE,"Q"'
        TAB 'POSCURS' CR 'ACCOUNT'
    ECM-TRAIL
    DELIMITED BY SIZE INTO ECM-BUFF
    WITH POINTER ECM-MSG-LEN.
    PERFORM 8000-SEND.
    GO TO 3299-EXIT.
3290-A-INVKEY.
    MOVE 'Record already exists.\' TO ECM-MSG.
    PERFORM 9000-MSG.
    PERFORM 9100-SEND-ERR.
    MOVE SPACES TO ECM-BUFF.
    MOVE 1 TO ECM-MSG-LEN.
    STRING ECM-HEAD 'DTAFLD' CR 'TRANS_CODE,"CUSTA "'
        TAB 'DTAFLD' CR 'F_CODE,"E"'
        TAB 'CTL' CR 'WRITE_FILE_BUTTON,ENABLED,0'
        TAB 'CTL' CR 'LIST_ORDERS_BUTTON,ENABLED,0'
        TAB 'CTL' CR 'DELETE_BUTTON,ENABLED,0'
    ECM-TRAIL
    TAB 'POSCURS' CR 'ACCOUNT'
    ECM-TRAIL
    DELIMITED BY SIZE INTO ECM-BUFF
    WITH POINTER ECM-MSG-LEN.
    PERFORM 8000-SEND.
3299-EXIT.
    EXIT.
*****
*** Change an Existing Customer (EB-F-CODE=C).
*****
3300-CHANGE SECTION.
3310-C.
    MOVE EB-ACCOUNT TO CM-ACCOUNT.
    READ CUSTFILE RECORD
        INVALID KEY GO TO 3390-C-INVKEY.
    MOVE EB-IN-DATA TO CM-DATA.
    PERFORM 6100-EQUATE-TO-CUST.
    REWRITE CUSTOMER-MASTER-REC
        INVALID GO TO 3390-C-INVKEY.
    MOVE SPACES TO ECM-BUFF.
    MOVE 1 TO ECM-MSG-LEN.
    STRING ECM-HEAD 'DTAFLD' CR 'TRANS_CODE,"CUSTA "'
        TAB 'DTAFLD' CR 'F_CODE,"Q"'

```

```

        TAB 'POSCURS' CR 'ACCOUNT'
        ECM-TRAIL
        DELIMITED BY SIZE INTO ECM-BUFF
        WITH POINTER ECM-MSG-LEN.
    PERFORM 8000-SEND.
    GO TO 3399-EXIT.
3390-C-INVKEY.
    MOVE 'Record does not exist.\' TO ECM-EMSG.
    PERFORM 9000-EMSG.
    PERFORM 9100-SEND-ERR.
    MOVE SPACES TO ECM-BUFF.
    MOVE 1 TO ECM-MSG-LEN.
    STRING ECM-HEAD 'DTAFLD' CR 'TRANS_CODE,"CUSTA "'
        TAB 'DTAFLD' CR 'F_CODE,"E"'
        TAB 'CTL' CR 'WRITE_FILE_BUTTON,ENABLED,0'
        TAB 'CTL' CR 'LIST_ORDERS_BUTTON,ENABLED,0'
        TAB 'CTL' CR 'DELETE_BUTTON,ENABLED,0'
        TAB 'POSCURS' CR 'ACCOUNT'
        ECM-TRAIL
        DELIMITED BY SIZE INTO ECM-BUFF
        WITH POINTER ECM-MSG-LEN.
    PERFORM 8000-SEND.
3399-EXIT.
    EXIT.
*****
*** Delete an Existing Customer (EB-F-CODE=D).
*****
3400-DELETE SECTION.
3410-D.
    MOVE EB-ACCOUNT TO CM-ACCOUNT.
    READ CUSTFILE RECORD
        INVALID KEY GO TO 3490-D-INVKEY.
    DELETE CUSTFILE RECORD
        INVALID KEY GO TO 3490-D-INVKEY.
    MOVE SPACES TO ECM-BUFF.
    MOVE 1 TO ECM-MSG-LEN.
    STRING ECM-HEAD 'DTAREC' CR 'CUSTA Q999999999'
        TAB 'CTL' CR 'WRITE_FILE_BUTTON,ENABLED,0'
        TAB 'CTL' CR 'LIST_ORDERS_BUTTON,ENABLED,0'
        TAB 'CTL' CR 'DELETE_BUTTON,ENABLED,0'
        TAB 'POSCURS' CR 'ACCOUNT'
        ECM-TRAIL
        DELIMITED BY SIZE INTO ECM-BUFF
        WITH POINTER ECM-MSG-LEN.
    PERFORM 8000-SEND.
    GO TO 3499-EXIT.
3490-D-INVKEY.
    MOVE 'Record does not exist.\' TO ECM-EMSG.
    PERFORM 9000-EMSG.
    PERFORM 9100-SEND-ERR.
    MOVE SPACES TO ECM-BUFF.
    MOVE 1 TO ECM-MSG-LEN.
    STRING ECM-HEAD 'DTAFLD' CR 'TRANS_CODE,"CUSTA "'
        TAB 'DTAFLD' CR 'F_CODE,"E"'
        TAB 'CTL' CR 'WRITE_FILE_BUTTON,ENABLED,1'
        TAB 'CTL' CR 'LIST_ORDERS_BUTTON,ENABLED,1'
        TAB 'CTL' CR 'DELETE_BUTTON,ENABLED,1'
        TAB 'POSCURS' CR 'ACCOUNT'
        ECM-TRAIL
        DELIMITED BY SIZE INTO ECM-BUFF
        WITH POINTER ECM-MSG-LEN.
    PERFORM 8000-SEND.
3499-EXIT.
    EXIT.

```

```

*****
*** The following sections handle all order file
*** retrieval functions.
*****
*** List order amounts on Order form (EB-F-CODE=L).
*****
4000-LIST-ORDERS SECTION.
4001-LO.
    MOVE EB-IN-DATA TO OR-DATA.
    MOVE OD-ACCOUNT TO OR-CUSTKEY.
*** Clear order list.
    MOVE 1 TO SUB1.
4002-LO.
    MOVE SPACES TO OD-ORDER-IDENT (SUB1).
    MOVE ZERO TO OD-ORDER-AMT (SUB1).
    ADD 1 TO SUB1
    IF SUB1 NOT > 12
        GO TO 4002-LO.
*** Find first order for this customer.
    START ORDERFILE KEY IS EQUAL TO OR-CUSTKEY
    INVALID KEY GO TO 4090-L-INVKEY.
    MOVE 0 TO SUB1.
4005-READ-NEXT.
    READ ORDERFILE NEXT RECORD
    AT END GO TO 4010-AT-END.
*** If not same account, we're done.
    IF OD-ACCOUNT NOT = OR-CUSTKEY
        MOVE 'ACCT NOT = ' TO OD-NEXT-ORDER
        GO TO 4020-LO-END.
    ADD 1 TO SUB1.
    IF SUB1 > 12
        MOVE OR-ORDER-IDENT TO OD-NEXT-ORDER
        GO TO 4020-LO-END
    ELSE
        MOVE OR-TOT-CHARGES TO OD-ORDER-AMT (SUB1)
        MOVE OR-ORDER-IDENT TO OD-ORDER-IDENT (SUB1).
        GO TO 4005-READ-NEXT.
4010-AT-END.
    MOVE 'AT END      ' TO OD-NEXT-ORDER.
4020-LO-END.
    MOVE SPACES TO ECM-BUFF.
    MOVE 1 TO ECM-MSG-LEN.
    STRING ECM-HEAD 'NEWFRM' CR 'ECMCUST2'
        TAB 'DTAREC' CR 'CUSTA L' OR-DATA
        ECM-TRAIL
        DELIMITED BY SIZE INTO ECM-BUFF
        WITH POINTER ECM-MSG-LEN.
    PERFORM 8000-SEND.
    GO TO 4099-EXIT.
4090-L-INVKEY.
    MOVE 'No order records found.\' TO ECM-EMSG.
    PERFORM 9000-EMSG.
    PERFORM 9100-SEND-ERR.
4099-EXIT.
    EXIT.
*****
*** List MORE order amounts on Order form (EB-F-CODE=M).
*****
4100-LIST-ORDERS SECTION.
4101-LO.
    MOVE EB-IN-DATA TO OR-DATA.
    MOVE OD-ACCOUNT TO OR-CUSTKEY.
*** Clear order list.
    MOVE 1 TO SUB1, SW-MORE.

```

```

4102-LO.
    MOVE SPACES TO OD-ORDER-IDENT (SUB1).
    MOVE ZERO TO OD-ORDER-AMT (SUB1).
    ADD 1 TO SUB1
    IF SUB1 NOT > 12
        GO TO 4102-LO.
*** Find first order for this customer.
    START ORDERFILE KEY IS EQUAL TO OR-CUSTKEY
    INVALID KEY GO TO 4190-L-INVKEY.
    MOVE 0 TO SUB1.
4105-READ-NEXT.
    READ ORDERFILE NEXT RECORD
    AT END GO TO 4110-AT-END.
*** If not same account, we're done.
    IF OD-ACCOUNT NOT = OR-CUSTKEY
        MOVE 'ACCT NOT = ' TO OD-NEXT-ORDER
        GO TO 4120-LO-END.
*** If next order specified, skip order records until next
*** order record is found.
    IF SW-MORE = 1
        IF OD-NEXT-ORDER = OR-ORDER-IDENT
            MOVE 'FOUND NEXT ' TO OD-NEXT-ORDER
            MOVE 0 TO SW-MORE
        ELSE
            GO TO 4105-READ-NEXT.
    ADD 1 TO SUB1.
    IF SUB1 > 12
        MOVE OR-ORDER-IDENT TO OD-NEXT-ORDER
        GO TO 4120-LO-END
    ELSE
        MOVE OR-TOT-CHARGES TO OD-ORDER-AMT (SUB1)
        MOVE OR-ORDER-IDENT TO OD-ORDER-IDENT (SUB1).
        GO TO 4105-READ-NEXT.
4110-AT-END.
    MOVE 'AT END      ' TO OD-NEXT-ORDER.
4120-LO-END.
    MOVE SPACES TO ECM-BUFF.
    MOVE 1 TO ECM-MSG-LEN.
    STRING ECM-HEAD 'NEWFRM' CR 'ECMCUST2'
        TAB 'DTAREC' CR 'CUSTA L' OR-DATA
        ECM-TRAIL
        DELIMITED BY SIZE INTO ECM-BUFF
        WITH POINTER ECM-MSG-LEN.
    PERFORM 8000-SEND.
    GO TO 4199-EXIT.
4190-L-INVKEY.
    MOVE 'No order records found.\' TO ECM-MSG.
    PERFORM 9000-MSG.
    PERFORM 9100-SEND-ERR.
4199-EXIT.
    EXIT.
*****
*** This section handles listing of order amounts in list box.
*****
4200-LIST-ORDERS SECTION.
4201-LO.
    MOVE CM-ACCOUNT TO OR-CUSTKEY.
*** Find first order for this customer.
    START ORDERFILE KEY IS EQUAL TO OR-CUSTKEY
    INVALID KEY GO TO 4210-LO-INVKEY.
    MOVE 1 TO SW-FIRST.
    MOVE 0 TO ECM-LST-CNT.
4205-READ-NEXT.
    READ ORDERFILE NEXT RECORD

```

```

        AT END GO TO 4220-LO-END.
*** If not same account, we're done.
        IF CM-ACCOUNT NOT = OR-CUSTKEY
            GO TO 4220-LO-END.
        MOVE OR-TOT-CHARGES TO ED-NUM.
*** If buffer full, add ECM trailer and send buffer.
        IF ECM-MSG-LEN > 3800
            STRING ECM-TRAIL DELIMITED BY SIZE
                INTO ECM-BUFF
                WITH POINTER ECM-MSG-LEN
            PERFORM 8000-SEND
            MOVE 0 TO ECM-LST-CNT.
*** If first line put start of ECM msg in buffer.
        IF ECM-LST-CNT = 0
            IF SW-FIRST = 1
*** If first item, indicate START if list, and insert list header.
*** At this point we are adding to a query response created above.
                STRING TAB 'LSTDTA' CR 'LST_ORD START,'
                    '(Order Ident,Amount,Date)'
                DELIMITED BY SIZE INTO ECM-BUFF
                WITH POINTER ECM-MSG-LEN
            ELSE
*** If not first, start a new msg. and indicate APPENDING to list.
                MOVE SPACES TO ECM-BUFF
                MOVE 1 TO ECM-MSG-LEN
                STRING ECM-HEAD 'LSTDTA' CR 'LST_ORD APPEND,'
                    DELIMITED BY SIZE INTO ECM-BUFF
                WITH POINTER ECM-MSG-LEN.
            MOVE 0 TO SW-FIRST.
*** String this item into ECM message buffer.
                STRING '(' OR-ORDER-IDENT ',' ED-NUM ','
                    OR-ENTRY-MO '/' OR-ENTRY-DA '/' OR-ENTRY-YR ')'
                DELIMITED SIZE INTO ECM-BUFF
                WITH POINTER ECM-MSG-LEN.
            ADD 1 TO ECM-LST-CNT.
            GO TO 4205-READ-NEXT.
4210-LO-INVKEY.
*** Come here if nothing found.
        MOVE SPACES TO ECM-BUFF.
        MOVE 1 TO ECM-MSG-LEN.
        STRING ECM-HEAD
            'ERRRLST' CR
            '{No orders for this customer:' CM-ACCOUNT '}'
            ECM-TRAIL
            DELIMITED SIZE INTO ECM-BUFF
            WITH POINTER ECM-MSG-LEN.
        PERFORM 8000-SEND.
        GO TO 4299-EXIT.
4220-LO-END.
**** If anything in list buffer, send it.
        IF ECM-LST-CNT > 0
            STRING ECM-TRAIL DELIMITED BY SIZE
                INTO ECM-BUFF
                WITH POINTER ECM-MSG-LEN
            PERFORM 8000-SEND.
4299-EXIT.
        EXIT.
*****
*** The following sections handle all order item listing
*** functions.
*****
* List FIRST items for an order (EB-F-CODE=P).
*****
5000-LIST-ITEMS SECTION.

```

```

5001-LI.
    MOVE EB-IN-DATA TO OR-ITEM-DATA.
    MOVE OI-ORDER-IDENT TO OR-ORDER-IDENT.
*** Clear item list.
    MOVE 1 TO SUB1.
5002-LO.
    MOVE SPACES TO OI-PRODUCT (SUB1), OI-DESC (SUB1).
    MOVE ZERO TO OI-QUANTITY (SUB1), OI-UNIT-PRICE (SUB1),
        OI-SHIP-QTY (SUB1).
    ADD 1 TO SUB1
    IF SUB1 NOT > 17
        GO TO 5002-LO.
    START ORDERFILE
        INVALID KEY GO TO 5090-L-INVKEY.
    READ ORDERFILE NEXT RECORD
        AT END GO TO 5010-AT-END.
    MOVE 1 TO SUB0, SUB1.
5005-NEXT-ITEM.
    IF SUB0 > 50
        GO TO 5010-AT-END
    ELSE
        IF OR-PRODUCT (SUB0) = SPACES
            OR OR-QUANTITY (SUB0) = ZERO
            OR OR-UNIT-PRICE (SUB0) = ZERO
            ADD 1 TO SUB0
            GO TO 5005-NEXT-ITEM
        ELSE
            IF SUB1 > 17
                MOVE OR-PRODUCT (SUB0) TO OI-NEXT-PRODUCT
                GO TO 5020-LO-END
            ELSE
                MOVE OR-PRODUCT (SUB0) TO OI-PRODUCT (SUB1)
                MOVE OR-DESC (SUB0) TO OI-DESC (SUB1)
                MOVE OR-UNIT-PRICE (SUB0) TO OI-UNIT-PRICE (SUB1)
                MOVE OR-QUANTITY (SUB0) TO OI-QUANTITY (SUB1)
                MOVE OR-SHIP-QTY (SUB0) TO OI-SHIP-QTY (SUB1)
                ADD 1 TO SUB0
                ADD 1 TO SUB1.
            GO TO 5005-NEXT-ITEM.
5010-AT-END.
    MOVE 'AT END      ' TO OI-NEXT-PRODUCT.
5020-LO-END.
    MOVE SPACES TO ECM-BUFF.
    MOVE 1 TO ECM-MSG-LEN.
    STRING ECM-HEAD 'NEWFRM' CR 'ECMCUST2'
        TAB 'DTAREC' CR 'CUSTA P' OR-ITEM-DATA
        ECM-TRAIL
        DELIMITED BY SIZE INTO ECM-BUFF
        WITH POINTER ECM-MSG-LEN.
    PERFORM 8000-SEND.
    GO TO 5099-EXIT.
5090-L-INVKEY.
    MOVE 'No items found.\' TO ECM-EMSG.
    PERFORM 9000-EMSG.
    PERFORM 9100-SEND-ERR.
5099-EXIT.
    EXIT.
*****
* List MORE items for an order (EBF-CODE=N)
*****
5100-LIST-ITEMS SECTION.
5101-LI.
    MOVE EB-IN-DATA TO OR-ITEM-DATA.
    MOVE OI-ORDER-IDENT TO OR-ORDER-IDENT.

```

```

*** Clear item list.
    MOVE 1 TO SUB1, SW-MORE.
5102-LO.
    MOVE SPACES TO OI-PRODUCT (SUB1), OI-DESC (SUB1).
    MOVE ZERO TO OI-QUANTITY (SUB1), OI-UNIT-PRICE (SUB1),
        OI-SHIP-QTY (SUB1).
    ADD 1 TO SUB1
    IF SUB1 NOT > 17
        GO TO 5102-LO.
    START ORDERFILE
        INVALID KEY GO TO 5190-L-INVKEY.
    READ ORDERFILE NEXT RECORD
        AT END GO TO 5110-AT-END.
    MOVE 1 TO SUB0, SUB1.
5105-NEXT-ITEM.
*** If next item specified, skip items until next
*** item is found.
    IF SW-MORE = 1
        IF OI-NEXT-PRODUCT = OR-PRODUCT (SUB0)
            MOVE 'FOUND NEXT ' TO OI-NEXT-PRODUCT
            MOVE 0 TO SW-MORE
        ELSE
            GO TO 5105-NEXT-ITEM.
    IF SUB0 > 50
        GO TO 5110-AT-END
    ELSE
        IF OR-PRODUCT (SUB0) = SPACES
        OR OR-QUANTITY (SUB0) = ZERO
        OR OR-UNIT-PRICE (SUB0) = ZERO
            ADD 1 TO SUB0
            GO TO 5105-NEXT-ITEM
        ELSE
            IF SUB1 > 17
                MOVE OR-PRODUCT (SUB0) TO OI-NEXT-PRODUCT
                GO TO 5120-LO-END
            ELSE
                MOVE OR-PRODUCT (SUB0) TO OI-PRODUCT (SUB1)
                MOVE OR-DESC (SUB0) TO OI-DESC (SUB1)
                MOVE OR-UNIT-PRICE (SUB0) TO OI-UNIT-PRICE (SUB1)
                MOVE OR-QUANTITY (SUB0) TO OI-QUANTITY (SUB1)
                MOVE OR-SHIP-QTY (SUB0) TO OI-SHIP-QTY (SUB1)
                ADD 1 TO SUB0
                ADD 1 TO SUB1.
            GO TO 5105-NEXT-ITEM.
5110-AT-END.
    MOVE 'AT END      ' TO OI-NEXT-PRODUCT.
5120-LO-END.
    MOVE SPACES TO ECM-BUFF.
    MOVE 1 TO ECM-MSG-LEN.
    STRING ECM-HEAD 'NEWFRM' CR 'ECMCUST2'
        TAB 'DTAREC' CR 'CUSTA N' OR-ITEM-DATA
        ECM-TRAIL
        DELIMITED BY SIZE INTO ECM-BUFF
        WITH POINTER ECM-MSG-LEN.
    PERFORM 8000-SEND.
    GO TO 5199-EXIT.
5190-L-INVKEY.
    MOVE 'No items found.\' TO ECM-EMSG.
    PERFORM 9000-EMSG.
    PERFORM 9100-SEND-ERR.
5199-EXIT.
    EXIT.
*****
** Find customers by LOC-KEY(State/city).

```

```

** Show list in ECMCUST3 form.
*****
5500-LIST-CUST SECTION.
5501-LC.
*** Show the customer finder form.
    IF EB-STATE = SPACES
        MOVE SPACES TO ECM-BUFF
        MOVE 1 TO ECM-MSG-LEN
        STRING ECM-HEAD 'NEWFRM' CR 'ECMCUST3'
            TAB 'UNLOCKFRM'
            ECM-TRAIL
            DELIMITED BY SIZE INTO ECM-BUFF
            WITH POINTER ECM-MSG-LEN
        PERFORM 8000-SEND
        GO TO 5599-EXIT.
*** Get list of customer for this state and city.
    MOVE EB-STATE TO HL-STATE.
    MOVE EB-CITY TO HL-CITY.
    MOVE HOLD-LOCKEY TO CM-LOCKEY.
*** Find first customer for this location.
    START CUSTFILE KEY IS NOT LESS THAN CM-LOCKEY
        INVALID KEY GO TO 5510-LC-INVKEY.
    MOVE 0 TO ECM-LST-CNT.
    MOVE 1 TO SW-FIRST.
5505-READ-NEXT.
    READ CUSTFILE NEXT RECORD
        AT END GO TO 5520-LC-END.
*** If not same location, we're done.
    IF CM-STATE NOT = HL-STATE
        IF SW-FIRST = 1
            GO TO 5510-LC-INVKEY
        ELSE
            GO TO 5520-LC-END.
*** If buffer full, add ECM trailer and send buffer.
    IF ECM-MSG-LEN > 3800
        STRING ECM-TRAIL DELIMITED BY SIZE
            INTO ECM-BUFF
            WITH POINTER ECM-MSG-LEN
        PERFORM 8000-SEND
        MOVE 0 TO ECM-LST-CNT.
*** If first line put start of ECM msg in buffer.
    IF ECM-LST-CNT = 0
        MOVE SPACES TO ECM-BUFF
        MOVE 1 TO ECM-MSG-LEN
        IF SW-FIRST = 1
*** If first item, indicate START if list and insert list header.
        STRING ECM-HEAD 'LOCKFRM'
            TAB 'LSTDTA' CR 'CUST_LIST START,'
            '(Customer Name,Account,St.,City)'
            DELIMITED BY SIZE INTO ECM-BUFF
            WITH POINTER ECM-MSG-LEN
        ELSE
*** If not first, indicate APPENDING to list.
        STRING ECM-HEAD 'LSTDTA' CR 'CUST_LIST APPEND,'
            DELIMITED BY SIZE INTO ECM-BUFF
            WITH POINTER ECM-MSG-LEN.
        MOVE 0 TO SW-FIRST.
*** String this item into ECM message buffer.
    STRING '(' CM-CUSTNAME ',' CM-ACCOUNT ','
        CM-STATE ',' CM-CITY ')'
        DELIMITED BY SIZE INTO ECM-BUFF
        WITH POINTER ECM-MSG-LEN.
    ADD 1 TO ECM-LST-CNT.
    GO TO 5505-READ-NEXT.

```

```

5510-LC-INVKEY.
*** String this item into ECM message buffer.
    MOVE SPACES TO ECM-BUFF.
    MOVE 1 TO ECM-MSG-LEN.
    STRING ECM-HEAD
        'ERRLST' CR
        '{No customers for this location:' HOLD-LOCKEY '}'
    ECM-TRAIL
    DELIMITED SIZE INTO ECM-BUFF
    WITH POINTER ECM-MSG-LEN.
    PERFORM 8000-SEND.
    GO TO 5599-EXIT.
5520-LC-END.
**** Unlock the form and send anything left in buffer
    STRING TAB 'UNLOCKFRM'
        ECM-TRAIL DELIMITED BY SIZE
        INTO ECM-BUFF
        WITH POINTER ECM-MSG-LEN
    PERFORM 8000-SEND.
5599-EXIT.
    EXIT.
*****
** These sections move data between eQuate record layout and
** data file record layout.
*****
6000-CUST-TO-EQUATE SECTION.
6001-CTM.
    MOVE CM-ACCOUNT TO CD-ACCOUNT.
    MOVE CM-CUSTNAME TO CD-CUSTNAME.
    MOVE CM-ADDR1 TO CD-ADDR1.
    MOVE CM-ADDR2 TO CD-ADDR2.
    MOVE CM-ADDR3 TO CD-ADDR3.
    MOVE CM-CITY TO CD-CITY
    MOVE CM-STATE TO CD-STATE
    MOVE CM-ZIP TO CD-ZIP
    MOVE CM-TELEPHONE TO CD-TELEPHONE.
6099-EXIT.
    EXIT.
6100-EQUATE-TO-CUST SECTION.
6101-MTC.
    MOVE CD-ACCOUNT TO CM-ACCOUNT.
    MOVE CD-CUSTNAME TO CM-CUSTNAME.
    MOVE CD-ADDR1 TO CM-ADDR1.
    MOVE CD-ADDR2 TO CM-ADDR2.
    MOVE CD-ADDR3 TO CM-ADDR3.
    MOVE CD-CITY TO CM-CITY.
    MOVE CD-STATE TO CM-STATE.
    MOVE CD-ZIP TO CM-ZIP.
    MOVE CD-TELEPHONE TO CM-TELEPHONE.
6199-EXIT.
    EXIT.
*****
** General send outptut routine.
** Uses MCB packet interface.
*****
8000-SEND SECTION.
8001-S.
*** SEND eQuate ECM RECORD
    SUBTRACT 1 FROM ECM-MSG-LEN.
    MOVE ECM-MSG-LEN TO P-LENGTH.
    MOVE P-SENDD TO P-FUNC.
    MOVE LOW-VALUES TO P-FLAGS.
    MOVE 1 TO P-BIT0.
    MOVE 1 TO P-START.

```

```

      MOVE 0 TO P-OFF.
      MOVE 0 TO P-ID.
      MOVE LOW-VALUES TO P-OUTQ.
      CALL 'CMCB' USING P-MCB-PACKET P-MCB-STATUS.
      IF P-STATBIT NOT = 0
          DISPLAY 'MCB SEND ERROR'
          PERFORM 9900-MCB-ERROR.
8099-EXIT.
      EXIT.
*****
** Accumulate error messages into eQuate ECM record.
** The message is NOT sent here.
*****
9000-EMSG SECTION.
9001-E.
** Move in header if this is first call.
      IF ECM-ERR-LEN = 0
          MOVE SPACES TO ECM-ERR-MSG
          MOVE 1 TO ECM-ERR-LEN
          STRING ECM-HEAD 'ERRLIST' CR
              DELIMITED BY SIZE INTO ECM-ERR-MSG
              WITH POINTER ECM-ERR-LEN.
          STRING '(' ECM-EMSG ')'
              DELIMITED BY '\' INTO ECM-ERR-MSG
              WITH POINTER ECM-ERR-LEN.
          MOVE 1 TO WW-ERROR-SW.
9099-EXIT.
      EXIT.
*****
** Send eQuate error mssage. Trailer is appended first.
*****
9100-SEND-ERR SECTION.
9101-SE.
      STRING ECM-TRAIL DELIMITED BY SIZE
          INTO ECM-ERR-MSG
          WITH POINTER ECM-ERR-LEN.
      MOVE ECM-ERR-MSG TO ECM-BUFF.
      MOVE ECM-ERR-LEN TO ECM-MSG-LEN.
      PERFORM 8000-SEND.
9199-EXIT.
      EXIT.
*****
** FATAL MCB error occured.
*****
9900-MCB-ERROR SECTION.
9901-ME.
      DISPLAY 'FATAL MCB ERROR:' P-STATBIT ':' P-ERRCODE.
9999-EXIT.
      EXIT.

```


How to ...

Initialize eQuate User Databases

The first step after installing the eQuate Developer's Edition, or whenever you wish to change the location of the eQuate databases (USER.DBF and USERAPP.DBF), is to run the InitReg.exe program. This program will update the Windows registry with the location of the user databases and initialize the user databases in the location that you select. To execute this program, use the following procedure:

1. Using the Windows Explorer or the Windows Task Bar (Start | Run), browse to the eQuate installation directory (the default directory is: C:\Program Files\KMSYS Worldwide\eQuate\3.0).
2. Double-click on the InitReg.exe program. Click OK if using the Windows Task Bar.
3. To change the location of the user databases, click the Change Location button.
4. On the Select Directory dialog, you may change the drive (Drives drop-down list box), select from an existing directory (Directories list box) or create a new directory by typing into the Directory Name list box.
5. To initialize a user database, click the Initialize Databases button.

Warning: If you click the Initialize Databases button for an existing user database, all user registration (user ids, passwords and user application associations) will be lost.

6. Close the InitReg.exe program once the location has been set and/or initialization has been completed.

Assign User Privilege

Registering a user with eQuate serves two purposes: 1) extending privileges to the users that will be responsible for administration and development of eQuate applications and 2) assigning developed eQuate applications to a user's profile. Initially, you would assign those users that would develop, maintain and administer eQuate, and only after an application had been completed, would you revisit the registration process and attach the application to the user profiles. Both tasks are accomplished with the eQuate Administration program.

To register a user, use the following procedure:

1. From the Windows Task Bar, select Start | Programs | eQuate | Administration.
2. Click the Add New button in the Users group box.
3. Double-click "NEW_USER_1" in the User Id edit box.
4. Type the desired user name.
5. From the User Type drop-down list box, select the user type according to the privileges that the user will be allowed.
6. Click the Close button.

Note: An Administrator can perform all eQuate administrative, developmental and runtime functions. A Developer can perform all functions except administrative functions (i.e., an administrator may not run the eQuate Administration program or the InitReg.exe program). An End User can only run the eQuate Session manager. A User Profile can be established to facilitate assigning large groups of users to multiple applications. For information on linking users to developed applications, see Assign Applications to Users.

Create an Application Database

In addition to user registration, the eQuate Administration program is user to create application databases. Before you can begin the forms design process, an application database must be created and initialized. The application database consists of the binary form files (.DBF) containing runtime information regarding the application, its forms, etc.

To create and initialize an application database, do the following:

1. From the Windows Task Bar, select Start | Programs | eQuate | Administration.
2. Select the Database Management tab.
3. Select Application Databases as the Database Type.
4. Click the New Database button in the Database Utility Functions group box.
5. Enter the application database name in the Enter New Database Directory Name edit box and click the OK button.

Note: The application database name is the same as the database directory name. The application database's .DBFs will reside in that directory under the user database directory created by the eQuate Application program.

6. Click the Close button.

Configure a Host Connection

Before you attempt to use the eQuate Capture program or Session Manager to connect to a host, you first must run the QPort T27 Config or QPort UTS Config program found in the eQuate program folder.

The two programs are similar in appearance and functionality. Only the 2200 information required to make a connection is more extensive in that CMS and/or Telcon values must be supplied.

Use the following procedure to configure the connection for ClearPath Plus MCP Servers or Local (PCA) machines:

1. On the Visual Configuration window (working from right to left), click the Virtual Dest Actions button and select Add.
2. Select any name you prefer for the Virtual Destination Id.
3. Set the IP Address of the host. This value can be entered in dot notation or as a symbolic destination machine name (see IP Address).
4. The IP Port Id must be 23.
5. Select the Connection Type.
6. Click the OK button.
7. Click the Route Actions button and select Add.
8. Select any name you prefer for the Route Name and Station Name. The Route name is used to link the Route to the Virtual Destination. Note the route name as you will need to select it in the eQuate Capture program (see Route Name).
9. From the Select Virt. Destination drop-down list box, select the newly configured destination.
10. Choose the Terminal Type and the number of Rows, Columns and Pages that you are use to seeing in your host application.
11. Click the OK button.
12. Click the Save button.

For ClearPath Plus OS 2200 Server connections, use the following procedure:

1. On the Visual Configuration window (working from right to left), click the Virtual Dest Actions button and select Add.
2. Select any name you prefer for the Virtual Destination Id.
3. Set the IP Address of the host. This value can be entered in dot notation or as a symbolic destination machine name. Note: If you are going through a DCP and the DCP is not an IP router, this is the IP address of the DCP; otherwise, it is the IP address of the host (see IP Address).
4. The Port Id must be 102.
5. Select the Connection Type.
6. Click the OK button.
7. Click the Open Id Actions button and select Add.
8. Enter an Open Id. The Open Id is the name used on the \$\$OPEN statement (see Open Id).
9. Select from the list of Available CMS Processes or Telcon XEUs. When routing directly to the 2200 (no DCP), this name is usually RSDCSU (for DEMAND) or TIPCSU (for TIP).
10. If your previous entry was either TIPCSU, enter the App. Name. The App. Name is the name on an APPLICATION statement in CMS1100.
11. From the Select Virtual Destination drop-down list box, select the newly configured destination.
12. Click the OK button.
13. Click the Route Actions button and select Add.
14. Select any name you prefer for the Route Name. The Route name is used to link the Route to the Open Id. Note the route name as you will need to select it in the eQuate Capture program.
15. Enter a unique Station Name (i.e., there must not be another session active with this same name or an error stating, "The specified network name is no longer available," will occur upon attempt to open the session). For TIP access, this name must also appear as an EU-NAME on a PID statement in the CMS1100 configuration (see Station Name).
16. From the Select Open Id drop-down list box, select the newly configured open id.
17. Choose the Terminal Type and the number of Rows and Columns that you are use to seeing in your host application.
18. Check the Auto Open Session checkbox.
19. Click the OK button.
20. Click the Save button.

Now you should be able to test the connection with the help of the eQuate Capture program found in the eQuate program folder:

1. Click the Open Terminal button.
2. Select a route to test the connection and click the OK button.

If you see the expected response from the host, you are connected.

Capture a Screen

eQuate provides a utility strictly for the purpose of capturing screens displayed by host applications. The eQuate Capture program will connect you to the host where you sign on and issue the necessary key sequences to display the desired screen. Once the screen is visible, a single click of a button will capture the screen in GUI format, ready to be transported to the eQuate Form Designer. You may do some editing with the capture program, but most editing and arranging will be accomplished with the eQuate Form Designer.

Note: If your host is a 2200 and your screen definitions are stored in a DPS form file, you may skip the capture step and import the form definitions from downloaded files created by DPS's Form Language Manipulation Utility (FLMU). These files may be imported directly into the eQuate Application Manager.

Capture

To capture a screen, use the following procedure:

1. Set the type of connection required; T27 for A Series or UTS for 2200.
2. Click Open Terminal button.
3. Select a route (previously configured with a QPort Config program) from the Available Routes list box and click the OK button.
4. Use your usual procedure to sign on to the host and display an application screen.
5. Return to the eQuate Form Capture dialog and click the Capture Screen button.

Delete Unwanted Fields

Use the following procedure to delete any unwanted/pseudo fields:

1. Select a field with the mouse. The selected field will appear with a gray background.
2. Click Ctrl+D. The deleted field will appear with a red border indicating that no form definition will be generated for it.

Note: You may delete multiple fields at the same time by selecting them with the mouse while holding down the Shift key. Another way to select multiple fields is to move the mouse over a teal (dark greenish blue) portion of the screen without a field, and while holding down the left mouse button, drag the cursor across a group of fields to be deleted. Any field the cursor touches will be selected.

Field Naming and Attributes

The Form Capture program names data fields in a generic fashion: `FIELD_row_startingcolumn`. You may change a field name by selecting it with the mouse and make the desired change in the Selected Field Name edit box at the bottom of the window. You may also change the field case, justification and data type in this same area.

Selecting a Form Id

When developing an eQuate application that will operate in "Screen Mode" (screen scraping), an important part of the process is to choose a "Form Id String." This Form Id String can be any text (hidden or not) on the screen that uniquely identifies the screen. The Form Id String will be used to signal the eQuate Session Manager to load the form at execution time. To select the string, use the following procedure:

1. Press Ctrl+I or select Mark Form Id String from the Edit menu. The mouse cursor will change to crosshairs.
2. Move the crosshairs over the chosen string, and while holding down the left mouse button, drag the mouse over the string.

Setting Repeating Rows

If a row repeats (or multiple rows repeat) multiple times on a screen, the capture program needs to have that row identified as a repeating row since there is nothing on the captured screen that identifies it as a repeating row. Use the following procedure to mark repeating rows:

1. Select a field in the first row of repeating rows with the mouse.
2. Press Ctrl+R or select Set Repeating Rows from the Edit menu.
3. Enter the total number of repetitions in the Repeat Count edit box.
4. In the Rows in Repeat edit box, enter the number of rows that make up one occurrence. For example, if a field appears on every other line, this value would be 2; then again, if the field appears on every line, the value should be 1.

5. Click the OK button.

Note: Once the repeating rows have been marked, it is no longer possible to change the name or attributes of any field except for those in the first occurrence.

Generating the Form Definition

The final step on the Edit Capture dialog is to generate the form definition code that will be imported into the eQuate Form Designer. From the File menu, select Generate Form Definition or press Ctrl+G.

The generated code will appear in the Code Generation dialog allowing you to view the definition, save it to a file or copy it to the Windows clipboard. Next, you may capture other forms or proceed to the eQuate Application Manager and begin the forms design process.

Create an Equate Application

Once the application database has been created by the eQuate Administration program, you are ready to begin adding applications and forms to the database with the eQuate Application Manager program. This program is used for ninety percent of the design process; from initial input from the form capture process and the placement of controls on the form, to the assignment of control properties and event actions to be performed when controls are activated by users. Users

To add an application use the following procedure:

1. On the Database tab select the application database you created with the eQuate Administration program.
2. Select the Applications tab.
3. Click the Add button.
4. On the eQuate Application dialog, enter a name in the Application Name edit box.

Note: If you are just beginning to use eQuate, you may temporarily skip the remaining steps, click the OK button and proceed to the forms design process. You may return and edit the application later to set the terminal type, method of starting the application, etc.

5. Set the Terminal Type: UTS for 2200 or T27 for MCP.
6. Set the method for starting the application. If you plan to create a "start-up" form that will drive the application and provide multiple choices to the user at runtime, set the Display Start Form option button. If, on the other hand, you plan to execute a single transaction or program call to begin the application, set the Send Transaction option button. Note: The Send Transaction option may be best for those who are evaluating eQuate and wish to see its capabilities quickly. Later you can change the start-up method, as the application grows.
7. Fill in the Form Name or Transaction edit box. If the Display Start Form Option is chosen, this value is the name of the form that will automatically be displayed when the user starts the application with the eQuate Session Manager. If the Send Transaction is set, this value is the initial string to send to the host and activate some process. For a 2200, it might be a TIP transaction code, a MAPPER run call, a DEMAND program or processor call, etc. For A Series it might be a Marc menu selection, a task or program.
8. Check the Start in Command Mode box

Index

A			
Abs Function	123	BlinkForeColor Property	64
Accessing an Object	117	Blinking Property	64
Action Key Assignment	39	BlinkIntervalOff Property	64
Action Key Assignments	77	BlinkIntervalOn Property	64
Action Reloads	227	Border Property	64
Add Application Access	19	BorderStyle Property	64
Adding Controls and Fields	39	BorderWidth Property	64
addressStreetUser Route	228	Browser	60
Administrator	15	Button Group	39
Align Property	63	Button Groups	55
Aligning and Sizing Controls	39	ButtonStyle Property	64
Alignment	39	ByRef and ByVal	101
Alignment Property	63	C	
AllowAllUp Property	63	CalendarDialog Function	125
AppActivate Statement	123	Call Statement	125
Application	228	Calling Procedures in DLLs	102
Application Access	19	Cancel button	109
Application Database	15	Cancel Property	65
Application Manager	23	CancelButton Statement	125
Application Name	19, 20, 35	Caption Property	65
Application Sign-On Script	19, 20	Capture a Screen	275
Applications	23, 35, 227	CBool Function	126
Arithmetic Operators	121	CDate Function	126
Arrays	105	CDBl Function	126
Asc Function	123	Center Property	65
Assign User Privilege	273	Change eQuate Runtime Settings Location	229
Atn Function	123	ChangeCursorStyle Subroutine eQuate	127
AutoCenter Property	63	CharCase Property	65
Automatic Alignment	39	ChDir Statement	127
Automatic Properties Assignment	39	ChDrive Statement	128
Automation	117	Check Box	39
AutoSize Property	63	Check Boxes	49
AutoSnap Property	63	Check Boxes in Dialog	111
Available Backup Files	21	Check Script	91
B		CheckBox Statement	128
BackColor Property	63	Checked Property	65
Backup File Directory	21	Choose Function	129
Beep Statement	124	Chr Function	129
Begin Dialog Statement	124	CInt Function	129
Bevel	39	Class	118
BevelInner Property	63	ClearDisplay Subroutine	129
BevelOuter Property	63	ClearFields Subroutine	129
Bevels	55	ClearScreenChanged Subroutine	130
BevelWidth Property	63	CLng Function	130
Bitmap Property	64	Close Statement	130
BitmapPosition Property	64	CloseApp Subroutine	130
BlinkBackColor Property	64	ClosePage Subroutine	130
BlinkColor Property	64	Color Selection	71
		ColumnHeaders Property	65

Columns Property	65	Declare Statement	135
ComboBox Statement	131	Default Form	35
Command Button	39	Default Property	65
Command Buttons	48	Default Sign-On Script	15
Command Mode	37	Defining ECM Control Strings in COBOL	241
Command Mode Commands	237	Dialog Box Controls	114
Command Mode Example	243	Dialog Designer	91
Comments	95	Dialog Function	114, 136
Compile Script	91	Dialog Support	109
Concatenation	97	Dim	105
Configure a Host Connection	274	Dim Statement	137
Connection Route	35	Dir Function	138
Const Statement	131	DlgEnable Statement	138
Constant Names	95	DlgText Statement	139
Constructing ECM Data Records in COBOL	242	DlgVisible Statement	139
Continuation	95	Do Loops	99
Control Structures	99	Do NOT Validate to Host Screen Format	79
Controls	39	Do...placeLoop Statement	140
Cos Function	132	DoTerminalKey Subroutine	140
Create an Application Database	273	Double	97, 121
Create an Equate Application	276	Down Property	65
Create Runtime Files	23	download	158
Create Runtime Files Now	15	Drop-Down List Box	39
CreateObject Function	132	Drop-down List Boxes	52, 110
CSng Function	133	DropListBox Statement	142
CStr Function	133	DTAFLD	238
CTL	237	DTAREC	238
Ctl3d Property	65	E	
CurDir Function	133	ECM COBOL Example	243
Current Password	13, 15, 228	Edit Application Access	20
CVar Function	134	Edit Box	39
D		Edit Boxes	47
Data Field	79	Edit Capture	225
Data Types	97, 121	Edit Data Field Name	81
Data Types Operators and Precedence	121	Edit eQuate Network User Setup Settings	15
Database	19, 20	Edit Mask	72
Database Backup	21	EditMask Property	66
Database Directory	23	Editor Properties	94
Database Management	15	Enabled Property	66
Database Management tab	15	End Statement	143
Database Type	15	EOF Function	143
Database Utility Functions	15	eQuate 3.5 Language Translations	27
DataSource Property	65	Equate Components	11
Date Function	134	eQuate Global Maintenance Utility	233
Date Time Format Editor	74	eQuate Restore	21
Date/Time Label	39	eQuate Session Manager	227
Date/Time Labels	59	eQuateNavigate Function	144
DateSerial Function	134	Erase Statement	144
DateValue Function	135	ERRLST	239
Day Function	135	Exit Statement	145
Declaration of Variables	98	Exp Function	145

F			
Field	79, 225	GetPages Function	155
Fields	39	GetPassword Function	155
File Input/Output	103	GetReservedString Function	155
FileCopy Function	145	GetScreenAttribute Function	156
FileLen Function	145	GetScreenData Function	157
FileOpenDialog Function	145	GetScreenLine Function	158
FileSaveDialog Function	146	GetSessionVar Function	158
FillFormWhenMaximized Property	66	GetState Function	158
Fix Function	146	GetString Function	159
Flat Property	66	GetStringProp Function	159
FlatColor Property	66	GetTranslation Function	159
FLDCTL	239	GetUser Function	159
Font Property	66	GetUserAppOptions Function	160
For ... Next placeLoop	99	GetUserOptions Function	160
For Each...Next Statement	146	GetUserType Function	160
For...Next Statement	147	Global array	105
ForeColor Property	66	Global Statement	160
Form Activate Action	45	GoTo	99
Form Capture	221	GoTo Statement	161
Form Data Fields	79	Group Box	39
Form Definition Import	81	Group Boxes	56, 113
Form Design	37	GroupBox Statement	161
Form Designer	39	GroupIndex Property	66
Form Designer Toolbar	39	H	
Form Id String	37, 225	Height Property	66
Form Import View and Adjust	81	Help Property	67
Form Initial Action	45	Help Selector	71
Form Manager	37	Help Text	23
Form Merge/Update	82	Help Text Edit	89
Form Mode	37	Hex Function	162
Form Properties	45	Hint Property	67
Format Function	147	Host Error Action	45
Format Property	66	Host Error Detection String Edit	83
Forms	23	Host Error Ids	23
FrameStyle Property	66	Host Message Action	45
FreeFile Function	152	Hour Function	162
Function Naming Conventions	101	I	
Function Statement	152	If and Select Statements	99
G		If...Then...Else Statement	162
Get Statement	153	Image	39
GetColor Function	153	ImageOpenDialog Function	163
GetCurrentLanguage Function	153	Images	58
GetCursorCol Function	154	ImageSaveDialog Function	163
GetCursorField Function	154	Import Form	37
GetCursorFieldRep Function	154	Imported Label Edit	82
GetCursorRow Function	154	Initialize eQuate User Databases	273
GetMemoSelection Function eQuate	154	Input Edit Mask	72
GetNumericProp Function	154	Input Function	164
GetObject Function	154	InputBox Function	164
GetPage Function	155	InStr Function	165
		Int Function	165

Integer	97, 121	Media Players	59
IsArray Function	165	Memo	54
IsDate Function	165	Menu Designer	39, 77
IsEmpty Function	166	Method of Starting Application	35
IsNull Function	166	Methods	118
IsNumeric Function	166	Mid Function	175
IsObject Function	166	MinimizedButton Property	67
ItemIndex Property	67	MinSize Property	67
Items Property	67	Minute Function	175
K		Mkdir Statement	176
Kill Statement	167	MonoChromeButtons Property	67
L		Month Function	176
Language Translations	27	Moving and Sizing Controls	39
LBound Function	167	MsgBox Function	176
LCase Function	168	MsgBox Statement	176
Left Function	168	Multi-Column List Box	39
Left Property	67	Multi-column List Boxes	53
Len Function	169	Multi-Column List Setup	74
Let Statement	170	N	
Line Continuation Character	95	Name Property	67
Line Input # Statement	170	Name Statement	178
Lines Property	67	Naming Conventions	101
List Box	39	NetSet.EXE Program	15
List Boxes	51, 110	Network User Setup Settings	15
ListBox Statement	170	New Password	13, 228
ListClear Subroutine	171	NEWFRM	240
ListColHeader Subroutine	171	No Host	19, 20
ListCount Function	171	Now Function	178
ListGetColText Function	171	Numbers	95
ListGetIndex Function	172	NumBitMaps Property	68
ListGetItem Function	172	NumericSort Property	68
ListItemAdd Subroutine	172	O	
ListItemRemove Subroutine	172	Object	117
ListSetColText Subroutine	173	Oct Function	178
ListSetIndex Subroutine	173	OK button	109
ListSetItem Subroutine	173	OKButton Statement	178
LoadImage Function	173	OLE Automation	117, 118
LoadMMFile Function	173	OLE Automation and Word 6.0 example	119
LoadUserList Function	174	OLE Fundamentals	118
Local array	105	OLE Object	118
LOCKFRM	240	On Error Statement	179
LOF Function	174	Open Statement	181
Log Function	175	Operator Precedence	121
Logical Operators	121	Option Base	105
Long	97, 121	Option Base Statement	182
LSTDTA	240	Option Button	39
LTrim Function	208	Option Buttons	50, 113
M		Option Explicit Statement	182
MaximizedButton Property	67	OptionButton Statement	182
MaxLength Property	67	OptionGroup Statement	183
Media Player	39	Other Data Types	97

Overview	1	ReDim Statement	190
P		REFRESH	241
Panel	39	Relational Operators	121
Panels	57	Rem	95
Parent Controls	71	Rem Statement	190
ParentColor Property	68	Repeat Capture	226
ParentCtl3d Property	68	Repeating Field	79
ParentFont Property	68	Repeating Rows	225
ParentShowHint Property	68	Restore Options	21
Password	13, 15, 228	Right Function	190
PassWordChar Property	68	Rmdir Statement	191
Pick List Edit	87	Rnd Function	191
Pick List Selector	73	Route	35
Pick Values	23	RTrim Function	208
PickList Property	68	Runtime Connections	15
Picture Property	68	Runtime Database	15
placeCityContext StateId.	71	Runtime Directory	227
Player	39	Runtime Distribution	15
Pointer button	39	Runtime File UN-Packager	231
POSCURS	240	Runtime Modes	9
Precedence	121	Runtime Packager tab	15
Predefined Constants	215	Runtime Programs	15
Print # Statement	183	Runtime Publish	15
Print Statement	184	Runtime Scripts	15
PrintBeginDoc Subroutine	185	S	
PrintDlg Function	185	Scope of Variables	98
PrintDraw Subroutine	185	Screen Complete Check Action	45
PrintEndDoc Subroutine	185	Screen Mode	37
PrintMoveTo Subroutine	186	ScreenChanged Function	192
PrintNewPage Subroutine	186	Script Directory	227
PrintPageHeight Function	186	ScrollBars Property	69
PrintPageWidth Function	186	Second Function	192
PrintRect Subroutine	186	Seek Function	192
PrintSetFont Subroutine	187	Seek Statement	192
PrintSetFontSize Subroutine	187	Select Application Database	15
PrintSetFontStyleSubroutine	187	Select Case Statement	193
PrintSetOrientation Subroutine	187	Select Form	36
PrintTextHeight Function	187	Select General Profile	21
PrintTextWidth Function	188	Select New User Registration Database	15
Profile	15	Select Repeating Field Index	71
Profiles	227, 228	Select Screen Data Field Source	71
Properties	118	Select Statements	99
Properties Assignment	39	Send Subroutine	194
Publishing and Runtime Options	5	SendKey Subroutine	194
PushButton Statement	188	SendKeys Statement	194
Put Statement	189	SendMail Subroutine	195
Q		Session Manager	227
Quick Start	5	Set Admin. Script Directory	15
R		Set Statement	196
Randomize Statement	189	SetColor Subroutine	196
ReadOnly Property	68	SetCurrentLanguage Function	198

SetCursor Subroutine	198	TabStop Property	69
SetCursorField Subroutine	198	Tan Function	205
SetExecState Subroutine	198	Terminal Type	35
SetFocus Subroutine	198	Text	112
SetMemoSelection Subroutine eQuate	198	Text Boxes	112
SetNumericProp Subroutine	199	Text Label	39
SetPage Subroutine	199	Text Labels	46
SetScreenData Subroutine	199	Text Property	69
SetSessionVar Subroutine	200	Text Statement	205
SetState Subroutine	200	TextBox Statement	206
SetString Subroutine	200	The Basic Steps	3
SetStringProp Subroutine	201	The Complete COBOL Program	254
Sgn Function	201	The Complete Set of ECM Actions	250
Shape Property	69	Time Function	206
Shell Function	201	Time Statement	206
ShowAccelChar Property	69	Timer Event	207
ShowFormById Subroutine	202	TimeSerial Function	207
ShowFormByName Subroutine	202	TimeValue Function	207
ShowHelp Subroutine	202	Top Property	70
ShowHint Property	69	Translations	27
ShowPickList Function eQuate	202	Transparent Property	70
Sign-On Script	15	TransparentColor Property	70
Sin Function	202	TransparentMode Property	70
Single	97, 121	Trim Function	208
Sizing Controls	39	Type Statement	208
SortColumn Property	69	U	
SortDescending Property	69	UAKCTL	241
Sorted Property	69	UBound Function	209
Space Function	202	UCase Function	210
Speed Button	39	UMNCTL	241
Speed Buttons	48	UNLOCKFRM	241
Splitter	39	upload	170
Splitters	58	URL Link	61
Sqr Function	203	URL Property	70
Standard Messages	23, 85	User Application Options	20
Starting Application	35	User Defined Types	107
Statements	95	User Id.	15
Static array	105	User Options	15
Static Statement	203	User Options Map Editing	15, 20
Stop Statement	203	User or General Profile	19
Str Function	204	User placeStateId.	13, 228
StrComp Function	204	User Registration	15
Strech Property	69	User Registration tab	15
String	97, 121	User Sign On	13, 228
String Function	204	User Type	15
Style Property	69	UserName Function	210
Sub Statement	205	UserType Function	210
Subroutine and Function Naming Conventions	101	UTS	35
T		V	
T27	35	Val Function	211
TabOrder Property	69	Validate to Host	79

Values	101	Width Property	70
Variable and Constant Names	95	Windows Help	71
Variable Types	97	WindowState Property	70
Variables	98, 107	WinHelpFile	45, 71
Variant	97, 121	WinHelpFile Property	70
Variants and Concatenation	97	WinQT27 routename	223
VarType Function	211	WinQUTS32 routename	221
Visible Property	70	With Statement	212
W		WordWrap Property	70
Wait Subroutine	211	Write # - Statement	213
WantsReturns Property	70	X	
Weekday Function	211	XmitCursor Subroutine	214
What is an OLE Object?	118	XmitFrom Subroutine	214
What is OLE Automation?	117	Y	
While placeLoop	99	Year Function	214
While...Wend Statement	212		