
RFgen Software

All Editions

Release 4.1

A Product Of:
The DataMAX Software Group, Inc.
1101 Investment Blvd., Suite 250
El Dorado Hills, CA 95762
(916) 939-4065

Copyright 2012, The DataMAX Software Group, Inc.,
All Rights Reserved

TABLE OF CONTENTS

Introduction	1
Mobile Development Studio Overview	1
Mobile Enterprise Application Server Overview	3
Mobile Administration Console Overview	3
Windows (Graphical) Clients	4
Before You Begin	4
Loading the Software	5
Basic Implementation Steps.....	7
Support Files	8
Configuring RFgen Software	9
Authorizing the Development Environment	9
Configuring Master Database	10
Encrypting Your Master Database.....	11
Configuring Application Environment Options.....	12
Configuring Desktop Options	16
Configuring a Mobile Device Theme	18
Configuring Security Options	20
Configuring Speech Recognition Options	21
Configuring TCP/IP Services Options	23
Configuring Transaction Management	25
Configuring Visual Basic Options	29
Configuring Visual SourceSafe Integration	30
Configuring ODBC Database Connections	31
Configuring Connection Pooling	34
Configuring Scheduled Down Time	34
Configuring Extended Options.....	35
Configuring ODBC Data Sources.....	36
Configuring a Screen Mapping Connection.....	38
Configuring an ERP Connection	39
Configuring a Web Connection	43
Download Tables/Business Functions	44

View Downloaded Tables/Business Functions	48
Mobile Development Studio Menu Bar	51
Testing	53
Application Testing	53
Transaction Testing	56
SQL Query Testing.....	58
Devices	59
Deploy Mobile Applications	59
Remote Application Explorer	61
Remote Database Explorer	63
Remote SQL Explorer	64
Utilities.....	65
RFgen SourceSafe Browser.....	65
Export/Import Applications from/to RFgen Database	67
Validate Application Scripts.....	71
Find in Application Scripts	71
Replace in Application Scripts	72
Undo Save/Delete Action	73
Reports.....	74
Event Logs	74
Application Statistics.....	75
View Transaction Queues	76
Help.....	78
Mobile Development Studio Tools	79
Design Tab Overview	79
Users List.....	80
Menu List	81
Applications List.....	83
VBA Modules List	117
Transactions List.....	120
Hosts List.....	122
Resources Tab Overview	122
Device Profiles Tree	122
Image Resources.....	137
Speech Resources Tree.....	138
Text Resources.....	141
Vocollect Tasks Tree	142
Screen Mapping	144
How to Make RFgen Screen Mapping Work	144
Theory of Operation	145

Programming Philosophy	145
Design Considerations	146
Screen Mapping Levels	146
Logon Security Considerations.....	147
System Integrity Considerations.....	147
Keyboard/Special Key Configurations	148
Runtime Environment Variables	149
Configuring the Host Connection	149
Scheduled Downtime.....	153
VT220 Key Mapping	154
Hosts Tree	155
Building a Start Menu Macro	156
The Send Keys Selection	158
Start Menu Macro – Logon to the Main Menu	159
Start Menu Macro – Identify the Main Menu.....	161
Start Menu Macro – Recover from Navigation Error	162
Start Menu Macro – Logoff from the Main Menu	163
Start Menu Macro – Test Scripts	164
Building a Host Screen Macro.....	165
Host Screen Macro – Go to the Application Screen.....	166
Host Screen Macro – Identify the Application Screen.....	166
Host Screen Macro – Return to the Main Menu	167
Host Screen Macro – Mark Active Fields and Regions	167
Host Screen Macro – Enter a Sample Transaction	168
Host Screen Macro – Test Scripts	168
Recording Options	170
Building a Screen Mapping Application.....	171
Building a Screen Scraping Application	172
The Mobile Enterprise Application Server.....	173
To Permanently Activate the Server.....	174
Service Configuration.....	175
Client Sessions	177
The Mobile Administration Console.....	178
The Mobile Administration Console Menu Bar	180
The Sessions Tab	180
Connections Tab	183
Mobile Devices	186
Thin Client	186
Mobile Client	186

Client Network Control	187
RFgenCFG	188
RFgenCE	202
Visual Basic Scripts	205
Global User Defined Subroutines and Functions	205
Using External ActiveX files in the VBA Environment	206
VBA Global Variables/Objects.....	206
VBA Declarations	207
VBA Events	207
Click	207
GotFocus	207
Keypress.....	208
Initialize	208
Load.....	208
Lost Focus	208
OnBackup	208
OnConnect.....	209
OnCursor	209
OnDisconnect	209
OnEnter	210
OnEscape	210
OnFkey	210
OnInRange	211
OnLocale	211
OnOutOfRange.....	212
OnReadData.....	212
OnRefresh	212
OnReturn	212
OnScan.....	213
OnSearch.....	213
OnSpeech.....	213
OnTimer.....	214
OnUpdate	214
OnVocollect	214
Terminate.....	215
Unload	215
VBA Language Extensions	217
Prompt Property Extensions.....	217
RFPrompt().AddItem	217
RFPrompt().Autosize	217
RFPrompt().Bitmap.....	218
RFPrompt().Caption	218

RFPrompt().Checked.....	219
RFPrompt().Clear	219
RFPrompt().Defaults.....	219
RFPrompt().Display3D.....	220
RFPrompt().DisplayOnly.....	220
RFPrompt().Edits	220
RFPrompt().ErrMsg	221
RFPrompt().FieldBackColor.....	221
RFPrompt().FieldFontBold.....	222
RFPrompt().FieldFontItalic.....	222
RFPrompt().FieldFontSize	222
RFPrompt().FieldFontUnderline.....	223
RFPrompt().FieldForeColor	223
RFPrompt().FieldId	223
RFPrompt().Format.....	224
RFPrompt().Highlight	224
RFPrompt().ImageGetBitmap	225
RFPrompt().ImagePath	225
RFPrompt().Index	225
RFPrompt().LabelBackColor.....	226
RFPrompt().LabelFontBold.....	226
RFPrompt().LabelFontItalic.....	226
RFPrompt().LabelFontSize	227
RFPrompt().LabelFontUnderline	227
RFPrompt().LabelForeColor	227
RFPrompt().LabelLeft	228
RFPrompt().LabelTop	228
RFPrompt().Length	229
RFPrompt().List	229
RFPrompt().ListCount.....	229
RFPrompt().ListIndex.....	230
RFPrompt().PageDown	230
RFPrompt().PageNo	230
RFPrompt().PageUp	230
RFPrompt().Password	231
RFPrompt().RemoveItem.....	231
RFPrompt().Required	231
RFPrompt().ScrollDown.....	232
RFPrompt().ScrollUp	232
RFPrompt().SelLength	232
RFPrompt().SelStart	233
RFPrompt().Sorted	233
RFPrompt().Stretch.....	233
RFPrompt().Text	234
RFPrompt().TextLeft	234

RFPrompt().TextTop.....	234
RFPrompt().Type	235
RFPrompt().ValField.....	236
RFPrompt().ValTable.....	236
RFPrompt().Visible	237
Application-Based Extensions	237
CallForm	237
CallMacro.....	238
CallMenu.....	238
ChangeLoginForm.....	238
ChangeUserPassword.....	239
ClearValues	239
ClientType.....	239
ConnAvailable.....	240
ErrClear	240
ErrCount	240
ErrDesc.....	240
ErrNative.....	241
ErrNo	241
ExecuteMenuItem.....	241
ExitForm	241
ExitSession	242
GetInput.....	242
GetString.....	242
GetValue.....	243
IpAddress.....	243
Locale	243
LogError.....	244
MakeList	244
MsgBox.....	245
PromptCount.....	246
PromptNo.....	246
SendChar.....	246
SendKey	247
SetDisplay.....	247
SetFocus.....	248
SetMenu	249
SetOption.....	249
SetValue	252
ShowList	252
Sleep.....	252
TimerEnabled	253
TimerInterval.....	253
User	253
UserProperty.....	254

SQLNum	254
Screen Display Extensions	254
Bell.....	255
Clear	255
ClearEOL	255
ClearEOP	255
DrawLine.....	256
Height.....	256
Print.....	256
Refresh	257
ResetCursor.....	257
ReverseOff.....	257
ReverseOn.....	257
Width.....	257
Soft Input Panel Extensions	258
GetCurrentType	258
GetTypes	258
Mode	259
SetType.....	259
Show	260
Server-Based Extensions.....	260
CallMacro.....	260
CommandTimeout	261
Connect.....	261
Disconnect	261
ExecuteSQL.....	262
GetTable	262
IsConnected.....	263
Ping.....	263
QueueMacro	264
SendQueue.....	264
SendTable.....	265
SetHost	265
ShowProgress	265
SyncApps.....	266
WriteFile.....	266
System Level Extensions	267
ConnectionProperty	267
DeleteProperty	269
DisableTimeout.....	269
EnvironmentProperty	270
GetConnection.....	270
GetProperty.....	271
SendMessage.....	271
SetProperty	271

UserList.....	272
ValidateWinUser.....	272
Database Related Extensions	273
BeginTrans	273
CommitTrans	273
Count	273
Execute.....	274
Extract.....	274
MakeList	275
OpenResultset.....	276
RedirectDataSource	277
RollbackTrans.....	277
SaveBitmap	278
UseDataSource	278
Mobile Device Extensions	279
ClickAndSkipPrompts	279
ClickCoordinates.....	279
ForceLocal	280
GetGPSInfo	280
GoOffline.....	281
GoOnline.....	282
IsOffline.....	282
IsOnline.....	283
Platform	283
PlaySound	284
PrinterOff	284
PrinterOn	284
Send	284
SendCommPort	285
SetCameraOption.....	285
SetCommMode.....	285
SetCommPort	286
TakePicture.....	286
WriteFile.....	286
DeviceObject	287
Create	287
Execute.....	288
LastError	289
Name	289
Release.....	289
ReturnValue	289
Transaction Management Extensions	290
AbortTrans	290
GetItems	290
GetItemsEx	291

MacroName	292
MoveQueue	292
QueueMacro	292
QueueName	293
SeqNo	293
Enterprise Resource Planning Extensions	293
BeginTrans.....	294
CommitTrans	294
LogOff	294
LogOn	295
MakeList.....	296
ReadData.....	297
RollbackTrans.....	297
SetHardRelease	298
SetSession.....	298
Printer Extensions	298
Activate	298
Copies.....	299
EndDoc	299
FontBold.....	300
FontItalic	300
FontName	300
FontSize.....	300
FontStrikeThru	301
FontUnderline	301
GetName.....	301
NewPage	301
Orientation	301
PageWidth	302
Print.....	302
PrintQuality	303
PrintRaw	303
Screen Mapping Extensions	304
BeginTrans.....	304
CommitTrans	304
Connected.....	305
CurScreen.....	305
FindText.....	306
GetArea.....	306
GetAttribute.....	307
GetBackColor	307
GetCursor	308
GetForeColor	308
GetPage.....	309
GetText	309

GoToScreen	310
IsScreen.....	310
LogOff	311
LogOn	311
PadInput	312
PingHost	312
ResetConnection	312
SendCTRL	313
SendCTRLAlt.....	313
SendKey	313
SendKeyAlt.....	314
SendText	314
SendTextAlt	315
SessionID	315
SessionPwd	316
SessionUser	316
SetBase	316
SetCursor.....	317
SetDelay	317
SetSession.....	317
SetTimeout	318
WaitForCursor	318
WaitForCursorMove	319
WaitForHost.....	319
WaitForScreen.....	320
WaitForText	320
WaitForWrite.....	321
Text to Speech Extensions.....	321
ActiveGrammer.....	321
AddGrammar	322
AddGrammarFile	322
ClearGrammar	322
ClearProfile	323
ConfidenceLevel	323
DemoMode	323
DisplayGrammar.....	324
Enabled.....	324
GetInput.....	324
LoadProfile.....	325
PauseTime.....	325
ReadMode	325
ReadRate.....	326
RemoveGrammar	326
RemoveGrammarFile	326
Repeat	327

SaveProfile.....	327
SetLanguage	327
Speak.....	327
Spell	328
StopSpeak	328
TestMic	328
TrailingSilence	329
Train.....	329
TrimInput.....	330
Volume.....	330
WaitTime.....	331
Database List Object.....	331
List	331
AddListEntry	331
Clear	332
MaxRows	332
ReturnAllRows	332
SetReportColumn	332
ShowEmptyList.....	333
ShowList	333
SQL.....	333
Embedded Procedure Object.....	334
Embedded Procedures	334
Embedded Procedure Properties and Methods	337
ClassObject.....	337
Clear	337
ColumnCount.....	338
ColumnName	338
DataSource	338
DebugLog	339
DisableParam	339
Execute	339
ExecuteMethod	339
LogMode	340
Name	340
Param	340
ParamCount.....	341
ParamEx	341
ParamName	342
Queue	342
QueueName	342
QueueOffline.....	343
QueueSeqNo	343
RowCount	343
DataSet Object.....	344

AddNew	344
Clear	344
IsEOF	344
MoveFirst	344
MoveLast	344
MoveNext	345
MovePrevious	345
MoveTo	345
Param	345
ParamCount	346
ParamName	346
RowCount	346
Schemalid	347
Dynamic Array Extensions	347
DCount	348
Del	348
Ext	350
FixLeft	351
FixRight	352
Ins	352
LField	354
Locate	354
LocateAdd	355
LocateDel	356
Rep	357
RField	358
Socket Object	358
BytesReceived	358
LocalHostName	359
LocalIP	359
Protocol	359
RemoteHost	360
RemoteHostIP	360
RemotePort	360
State	361
sktClose	361
sktConnect	361
sktGetData	362
sktPeekData	363
sktSendData	364
OnClose	364
OnConnect	364
OnDataArrival	365
OnError	365
OnSendComplete	365

OnSendProgress	365
Web Object	366
ConnectTimeout	366
Data	366
DataSource.....	367
HeaderValue.....	367
ReceiveTimeout.....	367
Reply.....	368
Request.....	368
SendTimeout	368
Execute	369
Login	369
Logout.....	369
Stored Procedure Extensions	369
CallAction (not used with Sybase).....	369
CallProc (must be used with Sybase).....	370
CallSelect (not used with Sybase).....	371
Database Stored Procedure Object	372
CommandText	372
CommandTimeout	372
CommandType	372
CreateParameter	373
Data	373
DataSource.....	374
Dict.....	374
Execute	374
Param	374
Param().Datatype	375
Param().Direction.....	376
Param().NumericScale	376
Param().Precision	377
Param().Size	377
Param().Value	377
ParamCount.....	377
Prepared	378
Results	378
Initialization Files	379
RFgen.ini.....	379
ERPDialogs.ini	381
GPRS.ini	382
SQL Reserved Words	384
Index.....	387

Introduction

The RFgen Mobile Enterprise Application Platform, Mobile Development Studio, Mobile Enterprise Application Server and Mobile Administration Console are Microsoft Windows-based software programs that give numerous 'Data Base Management System' (DBMS) and 'Enterprise Resource Planning' (ERP) system users the ability to program advanced 'Radio Frequency Data Collection' (RFDC) applications via an easy-to-use development environment.

The Mobile Development Studio is a 'point-and-click' remote / wireless data collection applications generator that allows programming results to be achieved in hours, instead of days or weeks. The system is structured to interface with systems using: (1) ODBC/SQL, (2) Host Screen Mapping protocols, (3) XML and (4) select ERP packages. A major advantage of the Mobile Development Studio is that no investment in hardware is required until users actually place their programs into production use.

The Mobile Enterprise Application Server enables applications written with the Mobile Development Studio to work in a multi-user mode with remote (typically wireless) data entry devices. Hard-wired and mobile devices are also supported.

The Mobile Administration Console is the software administrators use to monitor and interact with users as well as managing data connections, transaction processing and the stopping and starting of data collection sessions.

Mobile Development Studio Overview

The Mobile Development Studio is the software used for the design phase of the mobile applications. It is important to note that most corporate databases contain data tables (and procedures) that would be useful in supporting data collection programming efforts. Mobile Development Studio can utilize database table 'fields' (i.e., column names and their associated properties) to determine the makeup of a database, and to allow data collection system developers to automatically interface with it. A preliminary step is to download table fields and properties from a database to a PC-based RFgen development system (see RFgen Basic Implementation Steps). Thereafter, on the PC, a developer may construct appropriate data entry

screens by pointing and clicking on the downloaded items, including dragging and dropping the items onto a simulated display (see The Mobile Development Studio Applications). A similar methodology is available for working with ERP business rules and procedures (BAPIs and business functions) when using the RFgen 'Integration Suites' for SAP, Microsoft Dynamics and JD Edwards / Oracle.

The visual components of an RFgen-based development process are established via the Mobile Development Studio Applications, Menus, and Users development modules. Each module appears as a heading in the Design tree in the Mobile Development Studio main screen display. Many features of an RFgen development system are available as 'properties'. For example, numerous data entry features (including data defaults, data edits/validations, table lookups, etc.) have already been pre-programmed. In practice, only a minimum amount of (or no) actual programming may be required to create a data collection application. The programming language used by RFgen is compatible with Visual Basic for Applications (VBA), the language used by Microsoft in its suite of 'Office' products.

When RFgen-based development efforts are complete, applications run closely coupled with remote devices, under Windows control, with access to corporate databases being 'SQL-driven'. Data processing tasks such as opening database(s) and tables, selecting and updating data records, list box generation, data validations/edits, executing custom SQL statements, VBA program execution, and calling stored procedures are automatically accomplished by RFgen. In a Thin client mode, RFgen-based Application, Menu, and User objects are stored and executed on a Windows-based server, not on remote data collection devices. The benefits associated with this mode of operation are: (1) increased processing power, (2) the availability of higher level programming tools, (3) ease of system deployment, and (4) ease of system maintenance; i.e., program changes are deployed instantly, without having to flash or re-flash remote devices. In a Mobile client mode development items are stored on the CE devices.

Organizationally, data entry Applications are listed on Menus and those menus are made available to data collection device users. All aspects of a development effort may be thoroughly tested in a local (non-network) mode (see the Application Testing option) prior to being placed into service.

Mobile Enterprise Application Server Overview

Once you have constructed and tested your system with the Mobile Development Studio, your programming effort is essentially complete. All that remains to activate your RFgen network is to install the Mobile Enterprise Application Server on your Windows server, configure your hardware, and test your complete system.

The Mobile Enterprise Application Server is a session enabler and manager that allows data collection devices to interact with databases, or (optionally) with legacy/Host screens and ERP packages, in a multi-user/pooled-connection(s) mode, using the programs developed by the Mobile Development Studio. Active data devices may be viewed and managed from the console of an RFgen-based computer system by means of an 'administrator' program included with the Mobile Enterprise Application Server. This program is installed automatically when the Mobile Enterprise Application Server is installed.

Technically, a server-based 'client' task is spawned for each remote device as it logs in. The Mobile Enterprise Application Server works in both character and graphical modes simultaneously to support a multitude of remote devices. Character-based devices use standard telnet-based 'VT220' communications. Graphical clients use the RFgen 'graphical client' software that must be loaded on the remote graphical device. A Mobile Enterprise Application Server program called the Mobile Administration Console is available to view and manage the workings of the Mobile Enterprise Application Server and its clients (active remote devices).

To activate a remote device, ensure that the Mobile Enterprise Application Server service is running and configure the telnet program to point to the IP address of the RFgen server. For more information, see the Mobile Enterprise Application Server section.

Mobile Administration Console Overview

The Mobile Administration Console is an optionally running program that displays all of the connected users to the RFgen server. Additionally, there is the ability to interact with the client's session or simply observe the work being done. Temporarily stopping 1 or many users and permanently shutting down 1 or many users provides complete control over the RFgen network. Managing the queuing capabilities is also done

from the Console application. The queue display allows for transactions that have been queued, completed successfully or have failed to be edited and reposted for another attempt at placing the data into the backend system.

Windows (Graphical) Clients

Note: If you are not currently using Windows desktop PCs or Windows CE-based devices, the information contained in this section may be skipped.

To use Windows-based PCs or mobile devices, the RFgen CD includes both the Windows Desktop Client installation and the Mobile / Wireless Clients installation. These applications take advantage of the Windows operating system and allow special controls like images, signature capture controls or command buttons to be used in an RFgen session. Standard telnet services cannot support these types of prompts. If there is a reason for running multiple copies of the Windows Desktop Client on the same PC, there is a command line option (-IgnoreGUID) that will allow this.

Note: not all PC-based telnet programs support function keys, in particular the telnet program provided by Microsoft with Windows 2000. The Windows Desktop Client supports function keys, and is a great alternative to the use of character-based telnet programs.

Before You Begin

To best use the Mobile Development Studio, it is necessary to understand the basic concepts of your Data Base Management System (DBMS), or for screen mapping applications, the structure and use of your IBM or UNIX-based (legacy) host system. Knowledge of data structures is of particular importance for database applications since it is necessary to understand the basic concepts of 'tables', 'fields/columns', and 'data types' prior to creating applications. Understanding Structured Query Language (SQL) 'syntax' is also helpful with database applications. Experience with the Microsoft Visual Basic/VBA programming language is helpful in the development of advanced data collection applications, for use with both SQL databases and legacy host-based applications.

Loading the Software

The RFgen CD-ROM is set to 'AutoPlay' when the CD is inserted into your CD-ROM drive. If the CD does not AutoPlay, click Start, Run, and open 'CDSetup.exe' on your CD-ROM drive. Selections include:

RFgen Software

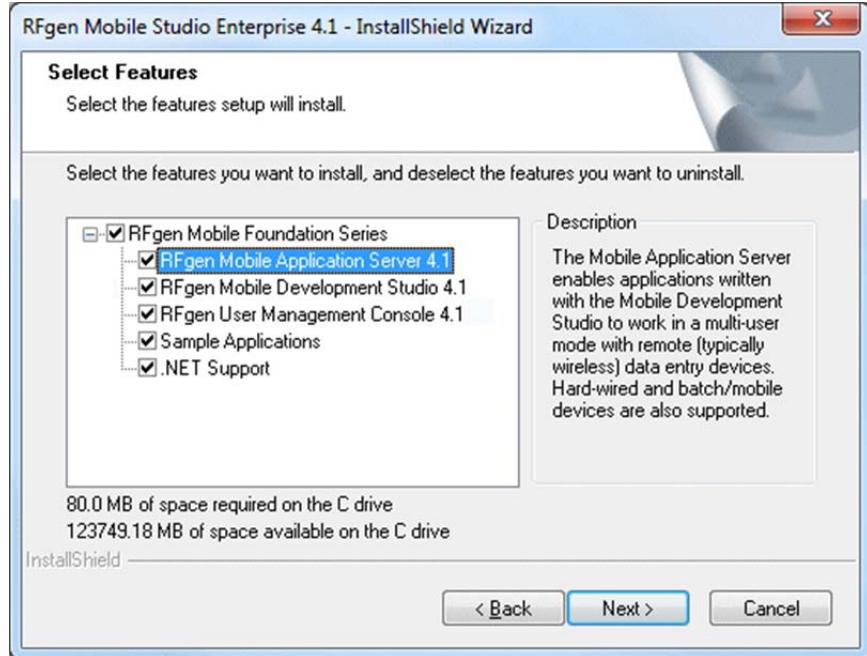
- Mobile Development Studio**
- Mobile Enterprise Application Server**
- Windows Desktop Client**
- Mobile/Wireless Clients**
- Language Packs**
- Support Files**

MDAC v2.8.1

Adobe Acrobat PDF Viewer

The MDAC component is required for the RFgen software. By default, all RFgen Software items are loaded to your \Program Files\RFgen directory. A Microsoft Access database called 'RFgen.mdb' is also loaded and contains a sample application, if it was chosen for installation. This database houses the application programs / objects written with the Mobile Development Studio.

The Mobile Enterprise Application Server install includes all RFgen components except for: the Windows Desktop Client application, the mobile client files and language pack files. By default, each of the components shown here are checked for installation.



The Mobile Foundation Series (for licensed RFgen users only) includes several components. Mobile Enterprise Application Server, Mobile Development Studio, User Management Console, Mobile Administration Console, sample applications, client files and the Service Control Manager.

The Mobile Enterprise Application Server allows multiple remote devices to use RFgen. The Mobile Development Studio is the development environment that is optional for production servers. The User Management Console provides minimal configuration access from the server. It allows the creation of menus and the assignment of users to menus only. The Mobile Administration Console allows administrators to view and manage remote devices. The Client file is started for each connection made by a user. This isolates each user and helps protect the system as a whole. The RFgen Service Control Manager is the graphical user interface to the Mobile Enterprise Application Server program that resides in the Windows System Tray. This program allows the Mobile Enterprise Application Server to be started manually or automatically, as a Windows 'task' or as a Windows 'service' when your system is booted. Other options include starting the Mobile Administration Console and suspending or stopping all Mobile Enterprise Application Server activity.

The **Windows Desktop Client** is a ‘graphical alternative’ to the standard Windows ‘telnet.exe’ character-based interface and currently has its own install program. Objects such as command buttons, signature capture boxes and images are available for the application when using a graphically-capable device.

The **Mobile and Wireless Clients** are files used to create appropriate installs on a CE / Mobile-based device. Based on the processor type and operating system, RFgen will draw from these files to create a working application for each device. They have the same ‘look and feel’ as the Windows Desktop Client for (full) Windows-based systems (above).

The **Language Packs** are files used to create appropriate installs on a CE / Mobile-based device when taking advantage of the text-to-speech and speech-to-text capabilities. Based on the processor type and operating system, RFgen will draw from these files to create a working application for each device.

Basic Implementation Steps

After you have installed and configured the Mobile Development Studio, the Software is normally implemented as follows for ODBC/SQL compliant databases:

1. ‘Transaction tables’ specific to your application are designed via your main database system (i.e., data to be captured is defined in your database). Typically, an individual ‘transaction table’ with appropriate field definitions is established for each data entry process.
2. Database transaction table field definitions are ‘downloaded’ to RFgen (via the Mobile Development Studio menu bar ‘Connections / Download Tables’ selection). These definitions contain all the necessary information concerning the transaction data to be written to your database. No data is downloaded from the database. Table definitions may be partially edited (inside Mobile Development Studio), if required. Similarly, ERP business rules and functions may be accessed and downloaded, if using the RFgen ERP integration suites.
3. Data entry Applications are made for each transaction table. Pre-programmed data properties such as ‘defaults’, ‘validations/edits’, ‘table validations’, etc. are added, as required.
4. Applications are listed on Menus (e.g., menus allow users to select a multitude of Applications).

5. User IDs and passwords are defined; a menu is assigned to each user.
6. System Objects (Applications, Menus, and Users), as constructed, are tested in conjunction with your database. Customized VBA programs and/or SQL statements are added as required.
7. Your data collection hardware (RF and other devices) are configured to support character-based or graphical communications sessions with your Windows-based Mobile Enterprise Application Server.
8. The Mobile Enterprise Application Server software from the RFgen Software CD is installed on your RFgen Server to activate full (multi-user) network usage and the remote management of data collection devices.

The above steps apply to database applications where ODBC/SQL is used to communicate with a database or ERP applications where predefined business rules and functions are used to update the ERP system.

For ‘screen mapping’ applications that interact with legacy host screens, see the RFgen Screen Mapping section.

Support Files

To use RFgen, your system will need an additional set of software drivers called ‘MDAC’ (Microsoft Data Access Components) that are contained on the loading menu ‘Support Files’ section. Newer operating systems usually already have these drivers installed. Also available in this section is the ‘Adobe PDF Reader’ that allows users to read and print RFgen system documentation and memos.

Note: When installing, one must have 'local administrator' privileges to successfully load RFgen.

Configuring RFgen Software

Once the Mobile Development Studio has been loaded, it must be configured. From your Windows desktop, execute the Mobile Development Studio software as follows:

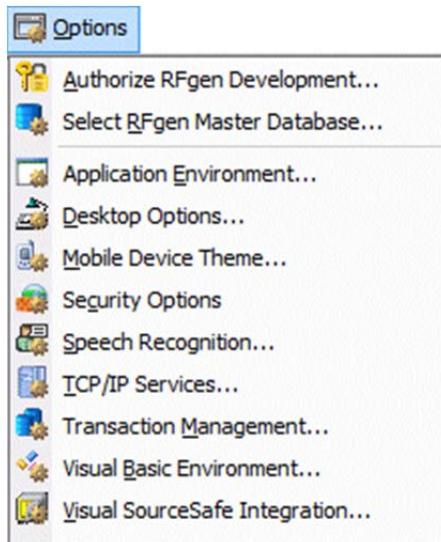
Click on 'Start'

Click on 'Programs', then click on 'RFgen v4'

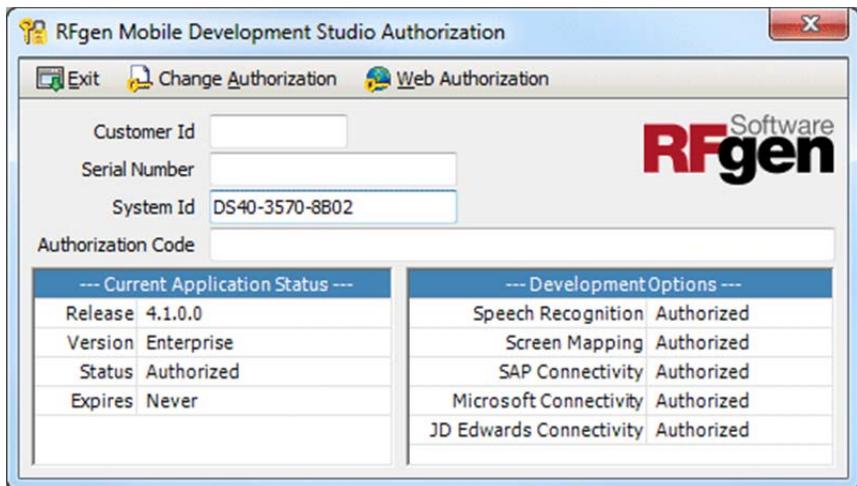
Click on 'Mobile Development Studio'

Authorizing the Development Environment

The Mobile Development Studio begins with a 30 day trial period. Depending on the licensing arrangement, it may be possible to authorize the product immediately. To authorize the development environment, choose the Options menu item and select Authorize RFgen development.



The following window will appear:

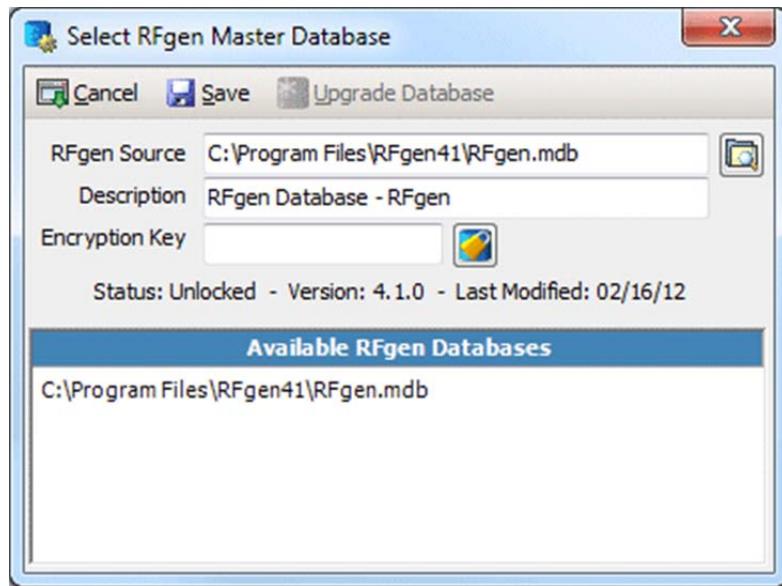


If the RFgen product has been purchased, DataMAX can provide the proper **Customer ID** and **Serial Number** values used to authorize both the development environment and production server. The **System ID** is generated by the software and is unique to each PC. Either enter an **Authorization Code** and choose the Authorize menu option or click the Web Authorization option to automatically obtain a code. Depending on the options that have been purchased, the Development Options will reflect what the software will be capable of. The **Status** field and **Expires** field should change from a trial state to a permanent state.

Configuring Master Database

To configure a Master Database, click the Options menu selection and then click on 'Select RFgen Master Database'.

The following window will appear.



By default, a data source called 'RFgen', defines the location of the 'RFgen Database', as shown in the window above.

This data source was created when the Mobile Development Studio software was loaded and it identifies a Microsoft Access file called '**RFgen.mdb**' (located in the RFgen directory) as the database that contains the programming items (Applications, Menus, Users and VBA code) written with the Mobile Development Studio, including the pre-written example/demo items.

The RFgen Source field contained in this window identifies to the system where this file is located. If the file is relocated, or if multiple projects exist, you may (1) modify the RFgen Data Source (User or System) entries via your Windows Control Panel ODBC Icon, or (2) specify a new Data Source name in this window by pointing to a new file.

Note: If you have multiple RFgen projects with different data source names, you may use this window to switch between them. Any name may be given to the '.mdb' file.

Encrypting Your Master Database

An '**Encryption Key**' entry provides RFgen users the ability to encrypt their RFgen System Database. This capability was previously provided by means of the RFgen 'Resource Compiler'. This feature, a system

option starting with release 3.0, allows the database to be locked so that users may not view or modify RFgen Application objects or VBA scripts. When active, a unique key may be entered in the Master Database selection window to lock, encrypt, unlock, or decrypt the database.

To encrypt the database: enter a key (e.g. 'abcdef') in the Key textbox, and click the Encrypt button with the lock icon.

To lock the database: display the panel again, remove the key and click 'Save'. The database is now locked. RFgen programs will execute but may not be accessed.

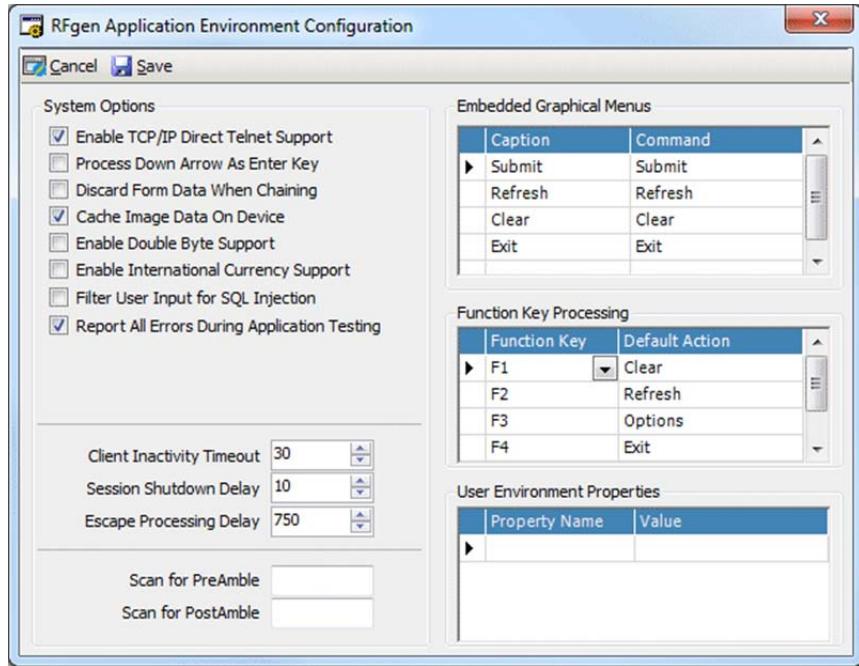
To unlock the database: display the panel and again enter the key, click 'Save'. The database will be unlocked.

To decrypt the database: enter the key, click the Encrypt button and click 'Save'. The database will be decrypted and unlocked.

You must not export encrypted applications to a non-encrypted MDB. RFgen will prompt for the password and decrypt the exported application.

Configuring Application Environment Options

To configure RFgen 'Application Environment' options, click the menu bar 'Options' selection and then click on 'Application Environment'. The following window will appear.



Here, **TCP/IP Direct** is checked, meaning that communications will be established with individual remote devices, rather than by using an intermediate ‘communications controller’ (accordingly if using a network communications controller, uncheck the ‘TCP/IP Direct’ option).

The **Down Arrow as Enter Key** option is not checked indicating that it will not be used as an alternative for the <Enter> key.

Discard Form Data When Chaining sets the current application data to null when another application is called, rather than keeping the original data in memory (the default condition) should the calling application be returned to.

Cache Image Data On Device will default the option for the device profiles to retrieve additional images that were not part of the initial installation when the mobile device updates itself.

The **Enable Double Byte Support** option will allow Chinese, Japanese, and other double byte character sets to display and wrap properly.

Enable International Currency Support enables support for international currency formats, by using the current system regional and language option settings (e.g. \$1,234.56 becomes \$1.234,56).

Filter User Input for SQL Injection – This option will check the SQL statement for specific key words and remove them before sending the SQL request to the ODBC driver. All semi-colons (;) are removed; any single quote (') marks are doubled; any instances of a double dash (--) are replaced with a single dash; and any words specified in a user-created FilterInput.ini file stored in the RFgen installation directory will be removed. For example, if one line in the ini file had the word “select” then a user input of “select * from inventory” would become “ * from inventory”.

Note: this is done on a space separated word basis so a phrase like “selected items” would not be affected as the word “selected” is not “select”.

These are sample words a user might want to filter for:

alter, analyze, associate, audit, backup, call, close, commit, connect, create, dbcc, delete, deny, desc, disassociate, drop, exec, execute, explain, grant, insert, intersect, join, kill, lock, minus, open, purge, recover, rename, restore, revoke, rman, rollback, rpc, select, shutdown, startup, truncate, update, union.

Additional Note: unless you are using a public kiosk where user tampering is of concern, this feature is not recommended.

Report All Errors During Application Testing – This option will bring immediate attention to a developer for incurred “soft” errors. It does so by displaying a message box displaying the error (only in Mobile Development Studio.) For example, each screen mapping command “SM.SendText” could timeout and fail, but the script will continue along (unless they are checking the return value for the function.) Turning this on will visually alert them of this type of “soft” error condition.

A **Client Inactivity Timeout** of 30 minutes is set for network data collection devices (i.e., no activity at the device for 30 minutes will cause the device to be logged off). This setting may be modified as desired.

A **Session Shutdown Delay** of 120 seconds waits an additional 2 minutes after the mobile device sends the “disconnect” command, before ‘releasing’ the session. Sometimes a mobile device will terminate a session and reboot, but the user’s intention is to reconnect and keep working.

An **Escape Processing Delay** of 750 milliseconds tells RFgen to wait for the entire ‘Escape’ character sequence for the specified number of

milliseconds. This feature compensates for some systems that have a delay in sending escape character sequences.

Pre-Angle and Post-Angle filter entries are character strings that are automatically sent from a scanner. They 'surround' the scanned data. They are optional and neither is required.

Common pre-angles include a location number, or perhaps an operator number. Common post-angles include control characters such as a tab or perhaps a carriage return-line feed. See your scanner documentation for information concerning how to establish these entries, or how to disable them.

Pre-angle and post-angle entries entered here are used by RFgen: (1) to identify scanner input, and/or (2) to automatically strip the pre/post entries from the character sequence received from a scanner. They will also cause a VBA Application 'OnScan' event to trigger.

The **Embedded Graphical Menus** settings control the functioning of the menu bar display that appears at the bottom or top of graphical data collection devices. Activation / Deactivation of the graphical menu bar is controlled by the 'Display Menus' setting.

When using this optional on-screen graphical menu bar:

- [Submit] means to enter the data appearing for the current prompt
- [Refresh] refreshes the display screen
- [Clear] clears the data entered for the current prompt
- [Exit] exits the current process (same as F4 normally does).

The Function Key Processing:

- [F1] is used to Clear the current prompt
- [F2] is used to Refresh the current screen
- [F3] proceeds directly from the current prompt to the Options prompt if one is on the screen
- [F4] immediately Exits your current application or menu.

These keys may be reset if desired. For example, changing F4 to F10 would make F10 the exit key for RFgen Applications and menus. When first using RFgen with remote devices, try using the established default settings before making function key changes.

Run your mouse arrow over the function keys to see their function displayed. Screen tips will appear so that you won't have to memorize their usage.

Note that an F4 entered anywhere in the application will return you to your user menu. Selecting Logout or F4 on your menu will return you to your previous menu or the Login screen.

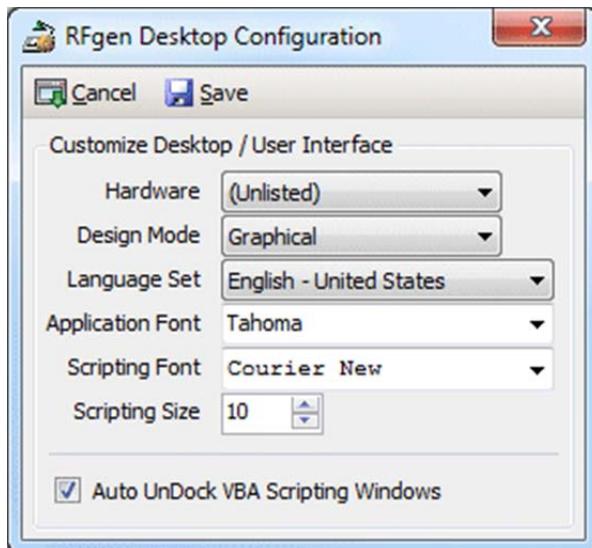
Additionally, you can enter the name of a function or procedure that exists in the application's VBA Win32 or RFgen BAS modules. Choosing a Default Action or Command really intends for you to type in your own subroutine name. Pressing the assigned F key or Windows menu bar button will execute that code.

RFgen graphical mode includes 'sculpted' display screens and allows users to use graphical objects such as 'Images' (pictures), 'Buttons' or Signature Capture prompts as part of their RFgen applications. Graphical mode displays must use Windows-based devices to display properly.

User Environment Properties

Entering a name and value in this box is like creating a global constant with a read-only value. For example, if this installation was designed for multiple warehouses but this particular installation was for a warehouse called 'Main Street' then entering the property name of 'Warehouse' and a value of 'Main' would allow the programmer to identify which warehouse was being used. The command used is System.EnvironmentProperty.

Configuring Desktop Options



The Desktop Configuration allows you to set properties for the **Hardware** type (i.e., the majority of devices) you will be using. Standard hardware types are shown as a drop-down window and include Compaq, Intermec, LXE, Motorola, Psion Teklogix, and Symbol to name a few. If your brand of hardware does not appear, choose 'Unlisted'. Chances are that your hardware will work fine as long as you are using standard telnet communications.

A **Design Mode** option sets the default mode for screen development either to the text-based character style or the Windows Mobile style.

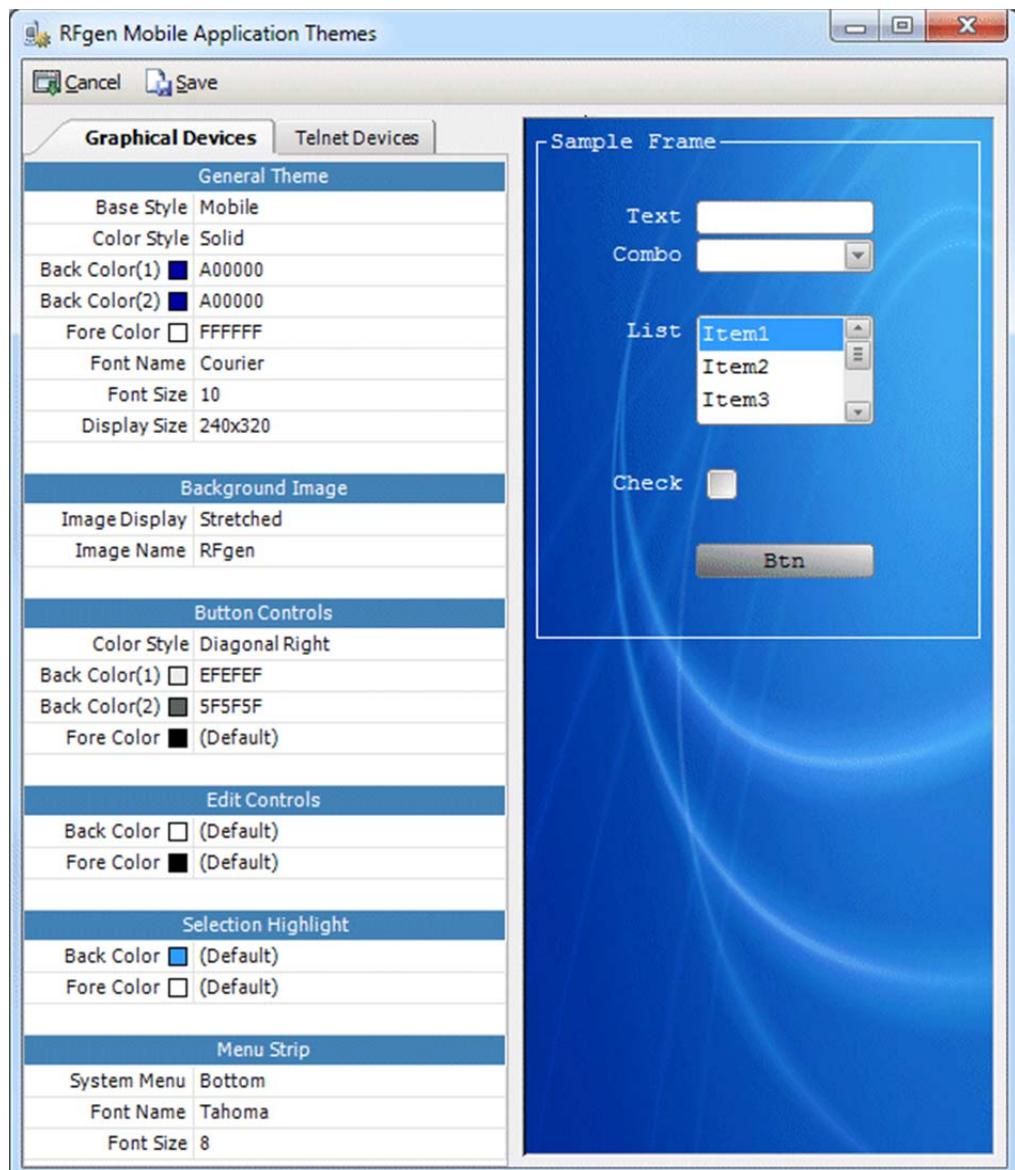
The **Language Set** option changes the expected input language character set.

The **Application Font** selection is used for the Mobile Development Studio displays. (See example VBA Modules window below.)

The **Scripting Font** and **Scripting Size** fields affect the code windows.

Auto UnDock VBA Scripting Windows – When clicking on the Script menu option when designing an application, the script window can either appear as its own window or be docked inside the Mobile Development Studio main window. This option simply sets the default behavior.

Configuring a Mobile Device Theme



General Theme:

The **Base Style** toggles between the more new Mobile look and feel where the objects on the screen are rounded and the Classic option where the objects are square.

The **Color Style** uses the Back Color (1) and Back Color (2) to create an effect of blended colors. This only applies if there is no background image. The options are Solid, Horizontal, Vertical, Diagonal Right and Diagonal Left.

The **Back Color (1 and 2)** are used with the Color Style option to create shading effects.

The **Fore Color** option changes the object's text on the screen.

The **Font Name** and **Font Size** change the font and size for each prompt on the screen.

The **Display Size** changes the default rectangle suggesting the maximum width and height of a mobile device display. This is only a guide and prompts may be placed on top of or outside this guide.

Background Image:

Image Display will stretch, tile or move an image around on the screen for the background of each form. The **Image Name** specified retrieves the image from the Image Resources which must contain the image first.

Button Controls:

The **Color Style** is the same as the Color Style listed in the General Theme section but applies to command button prompts only. **Back Color (1 and 2)** performs the same shading effect and the **Fore Color** is the color of the text on the button.

Edit Controls:

The **Back Color** of the data fields can be controlled here and the **Fore Color** is the color of the text inside the data field portion of the prompt.

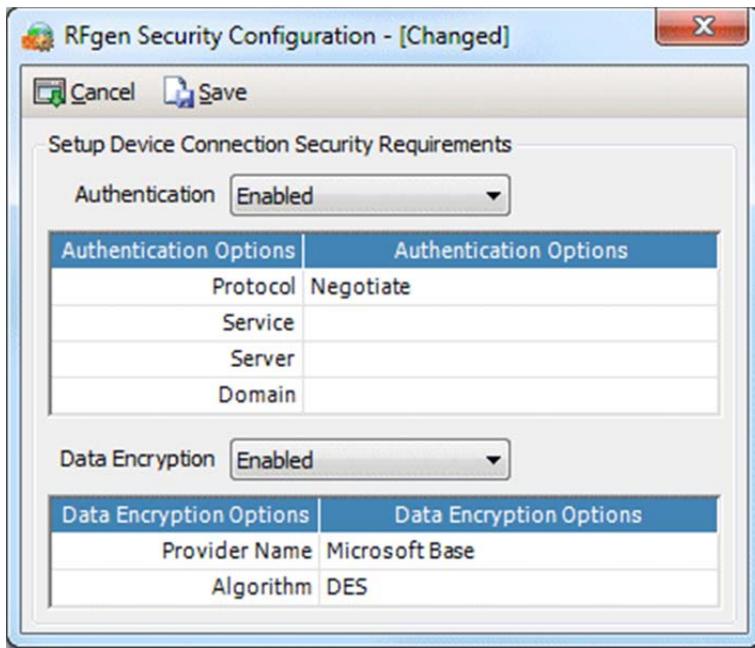
Selection Highlight:

The **Back Color** of the highlighted row in a combo box or list box is set here. The **Fore Color** is the color of the text inside the highlighted row. These should be contrasting colors for readability.

Menu Strip:

The **System Menu** bar can be displayed either at the top or bottom of the screen. It may also be disabled where it will hide the buttons on the menu bar. The **Font Name** property sets the font used on the menu bar and the **Font Size** adjusts the size of the text on the buttons.

Configuring Security Options

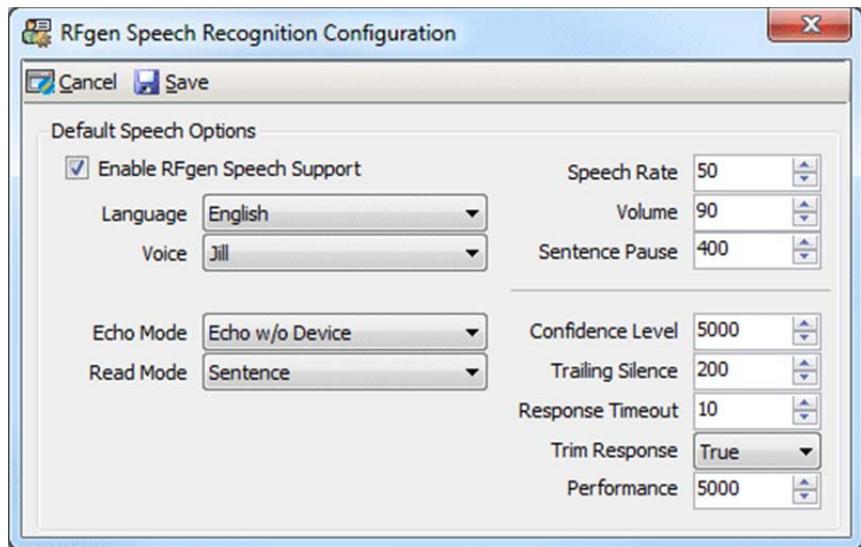


Security is broken into two categories, device authentication and data encryption settings.

Authentication is used to verify the user credentials beyond the RFgen login process. Authentication protocols currently supported are "Negotiate, NTLM, and Kerberos". If authentication is enabled then when an RFgen client first tries to connect, it will pop up a dialog box to capture user information (user id, password, and domain.) An encrypted package of this information will be sent to the configured protocol. A core Windows service on the RFgen server will attempt to authenticate the login request and accept or reject the connection. If Kerberos is used, then you must also specify the Kerberos security service name, the PC name where it is running, and the domain where the PC exists.

Data encryption is used to secure the data being transmitted in the wireless environment. Microsoft provides several cryptographic choices and algorithms that are taken from what the operating system is capable of doing. The client must be configured exactly the same way as the server or it will not connect.

Configuring Speech Recognition Options



Speech configuration requires that the Mobile Development Studio be authorized for speech development.

The **Language** is simply the purchased dictionary the speech engine uses to validate spoken words and the phonetic basis for speaking to the user.

The **Voice** option is the name of the person being emulated.

The **Echo Mode** contains 3 options, Always, Never, and Echo without the device. These settings are only applicable to the Mobile Development Studio since the Application Server assumes proper voice hardware.

Always – the PC hosting the Mobile Development Studio will play through the speakers what is being spoken to the user through the headset attached to the mobile device.

Never – the Mobile Development Studio only sends the spoken text to the mobile device.

Echo w/o Device – if no device is being used the Mobile Development Studio will play spoken text through the PC speakers. Otherwise, only the device will hear what is spoken.

Read Mode changes how the text is spoken. There are 4 options: Character, Line, Sentence and Word. The differences are the pausing used between characters, words and lines.

Character – the system will read character by character.

Line – the system will read line by line.

Sentence – the system will read the text sentence by sentence. In order to divide text into sentences, the system applies heuristics based upon the punctuation marks and capitalization.

Word – the system will read the text word by word.

Speech Rate sets the speed at which the text is played back to the user. The allowed number range is 1 – 100 with the default set to 50. The higher the number the faster the speech will be.

Volume is how loud RFgen will speak the text. The range 0 – 100 and is recommended that it be set at the maximum or loudest level.

This volume is still subject to the volume setting of the operating system and / or hardware device. Since it is usually much easier to access the volume controls of the device (buttons on the side of the device or a speaker icon on the desktop) volume can be more easily adjusted for comfort there.

Sentence Pause is the number of milliseconds between 0 and 1,800 that the system will pause between sentences. Sentences must be in proper sentence case for this to work meaning proper letter case and punctuation.

Confidence Level is a number between 0 and 10,000 that represents the threshold for how well the spoken word matches to the defined allowable answers. The default is 5,000. Setting the number lower means the system is allowed to do more guesswork to interpret what the user said.

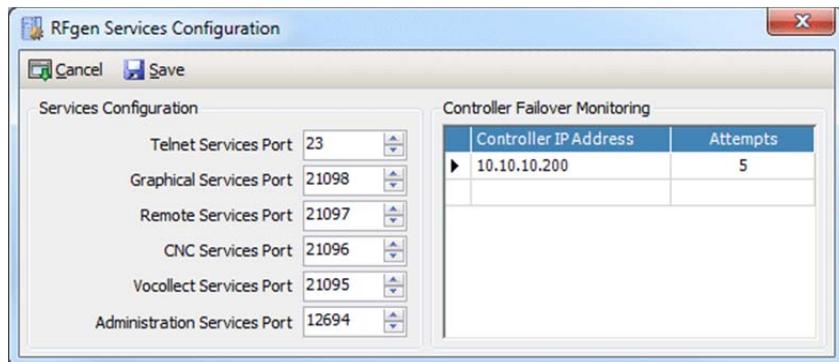
Trailing Silence is a number between 0 and 2,000 milliseconds that is the pause made by the user that the system waits for before accepting what was spoken as the completed response. The default is 200. Speaking too slowly will result in the system thinking it has received the complete response and only capturing a portion of the complete response. If this happens increase the value.

Response Timeout is a number between 0 and 32,000 seconds that the user has to respond while the speech engine is listening or when using the TTS.GetInput command without specifying the optional timeout value.

Trim Response will remove all spaces throughout the captured speech text. For example, if the part number 1234 is spoken it would normally be returned as 1 2 3 4, but with this setting to True it will be returned with no spaces for easier validation.

Performance Level allows the administrator to trade accuracy for speed of recognition. The allowed range is between 100 and 10,000. The lower the number, the faster the speech engine will return but with possibly less accurate results.

Configuring TCP/IP Services Options



Telnet Services are set to port 23 (the standard default setting for telnet servers such as RFgen). If you have another telnet server (i.e., another telnet-based application) running on your RFgen system which is listening on port 23, then your remote devices must be configured to some port other than the 23 default. That port number will have to be entered here and on the mobile device's telnet client.

The **Graphical Services** port is the address that RFgen communicates with remote Windows CE devices that are configured with the RFgen Graphical Client. The default setting is port 21098.

The **Remote Services** port is the address used for interfacing external / 3rd party data with RFgen applications (e.g. XML). The default setting is port 21097.

The **CnC Services** port is used by the RFgen Client Network Control process that deploys updated files, transactions or data to the device.

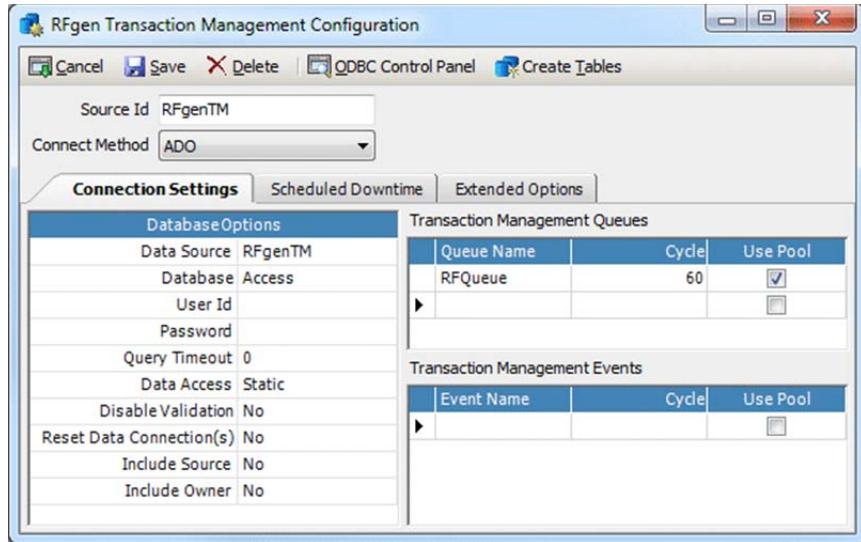
The **Vocollect Services** port is the address that the Vocollect product uses to communicate with RFgen. Vocollect is a hardware solution that replaces barcode scanners for a speech-processing device that accepts the spoken word as data input.

The **Administration Services** port is the address that the Server uses to communicate with the Console. The default setting is port 12694. Please leave as is, unless instructed to change it by DataMAX Software technical support.

Controller Failover Monitoring

When one or more controllers are being used to handle the traffic from the connected devices, (because the devices do not have their own IP address) there is the possibility that connectivity between the RFgen server and the controller may be lost during failover. This box is where you would enter the IP address of the controller and the number of ping attempts to execute before RFgen shuts down the connection. If a backup controller is configured to take control when the primary controller goes offline, RFgen will terminate and free up the licenses currently being used by the devices. They can then log back on (through the second controller) and continue working. To a user, it would appear as if their handheld device rebooted and reconnected.

Configuring Transaction Management



Name the connection using the **Source Id** field and select the appropriate **Connect Method**. For modern databases, ADO is preferred. Choose a **data source** from the drop-down box which points to your preferred ODBC database to be used to store queue information. If you do not plan to use the queuing capability, but only the Processing Events, you do not need to specify a Data Source value. Change the **database** manually, if it does not change when selecting the ODBC entry. This lets RFgen know which provider should be used internally.

If necessary, the **user id** and **password** possibly already entered in the ODBC setup may be required for RFgen to make a connection.

The **Query Timeout** specifies how long RFgen should wait before giving up on the ODBC driver to come back with a response.

Data Access sets the cursor to either Static or Dynamic when retrieving data from the database. Usually, Static is best because it is fast and safe. However, if you have a database like Pervasive that will actually make a copy of the data from the database system to the RFgen system when using a static cursor, you can change this option to Dynamic, so performance will not suffer. Internally, this sets the cursor option to either adoOpenStatic or adoOpenDynamic.

Disable Validation set to Yes will prevent RFgen from internally making sure the database is actively connected.

Reset Data Connection(s) set to Yes this property will reset the data connections if the transaction macro fails (is moved to the Failed Queue) or if the transaction macro executes a TM.AbortTrans command. If the transaction macro is linked to a specific data connector then only that connector is reset. If it is not linked then all data connectors will be reset. Setting this property to No will disable all reset functionality.

Clicking on the ODBC Control Panel menu option allows for each selection of the queue database.

Transaction Management Queues

This table of queues allows for several queue processes to take place at the same time. The name RFQueue is the default name for the first thread that will process transactions. The **Processing Cycle** number is the number of seconds that will pass before RFgen checks this queue for transactions and if 1 or more are found the entire queue is processed. The **Use Pool** checkbox tells RFgen to use pooled handles to connect to the Transaction Management database. All the different queues could all either have their own connection or look to the pool to use 1 handle.

Note: each queue process uses 1 user license because it is, in essence, an automated user performing tasks against RFgen and the backend systems. For example, a 10 user licensed RFgen system with 3 separate queue processes will only allow up to 7 concurrent devices.

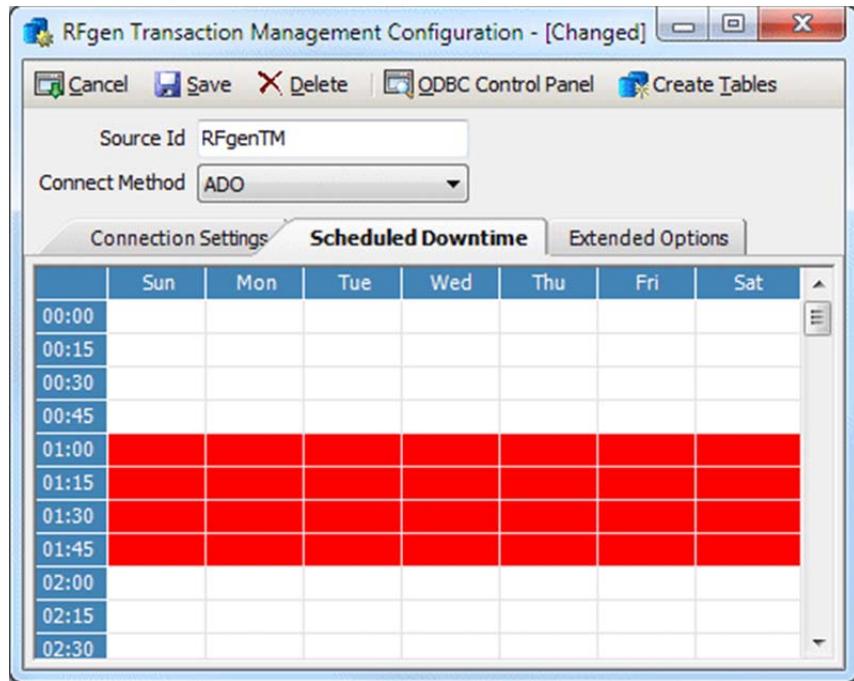
Clicking on the Create Tables menu option will create tables within the database to store queued, completed, and failed transactions.

Transaction Management Events

This table of events allows for several separate processes to take place at the same time. **Event Name** is the name of the Timed Event macro created under the Transactions heading in the tree. The **Processing Cycle** number is the number of seconds that will pass before RFgen executes this subroutine again. The **Use Pool** checkbox tells RFgen to take advantage of the pooled handles configured in the data connection configuration. Any Event entry with a Processing Cycle greater than 0 will also take a license away from the total allowed concurrent user pool.

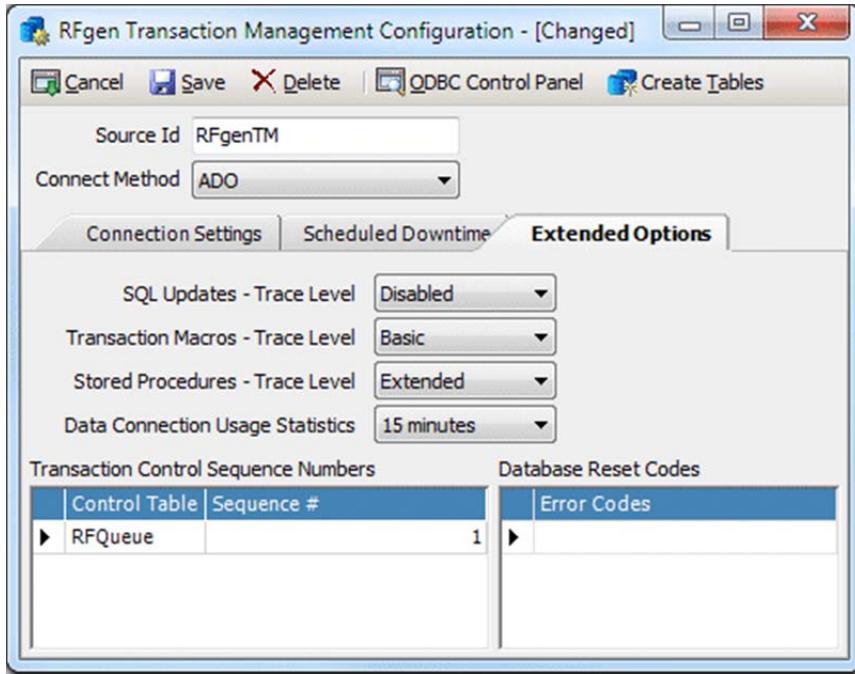
Scheduled Downtime

This option allows users to schedule down time for the Transaction Management process, during which RFgen will disable the queues. The following panel will appear.



As shown, the queues are disabled between 1:00am through 2:00am, every day of the week. Alternatively, you can select smaller amounts of time or individual days. Select the time and day with the mouse and use the spacebar to toggle the time frame on or off. A red square means that RFgen will disable the queues at that time.

Extended Options



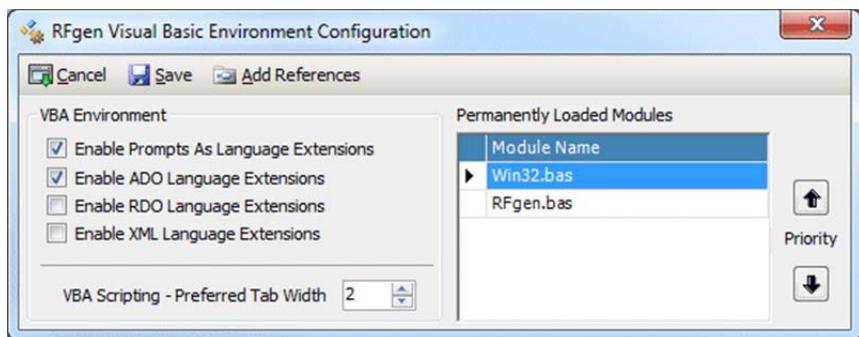
SQL updates, transaction macros and **stored procedures** are all methods for updating the backend system and therefore can be logged if strict compliance to regulatory law is required. There are 3 modes, Disabled, Basic and Extended. This simply refers to the level of detail provided in the log. The **Data Connection Usage Statistics** will enable the Application Statistics under the Reports menu option to generate graphs of data connection information. A table is created in the TM database to store this information, so changing this configuration from Disabled to n-minutes requires clicking on the Create Tables menu option.

Transaction Control Sequence Numbers display and allow the user to change the sequence number used when queuing or making entries in the logs.

Database Reset Codes is a place where you can add additional error codes that come back from the ODBC driver that should cause RFgen to re-establish a connection to the queue database.

When the transaction database entry is saved, a connection icon will appear at the bottom of the screen. This connection does not count as 1 of the 5 allowed data connections in the Enterprise version.

Configuring Visual Basic Options



Enable Prompts as Language Extensions allows the use of abbreviated syntax when referencing the prompts and their properties. For example RFPrompt("txtPart").Text can now be done txtPart.Text.

Enable ADO or Enable RDO Language Extensions allow you to access database(s) directly in VB rather than just through the pre-built RFgen programming extensions available for database access. If you are planning to write your own database access code, you will need to check either ADO or RDO. Support for the specified method will automatically be loaded as required.

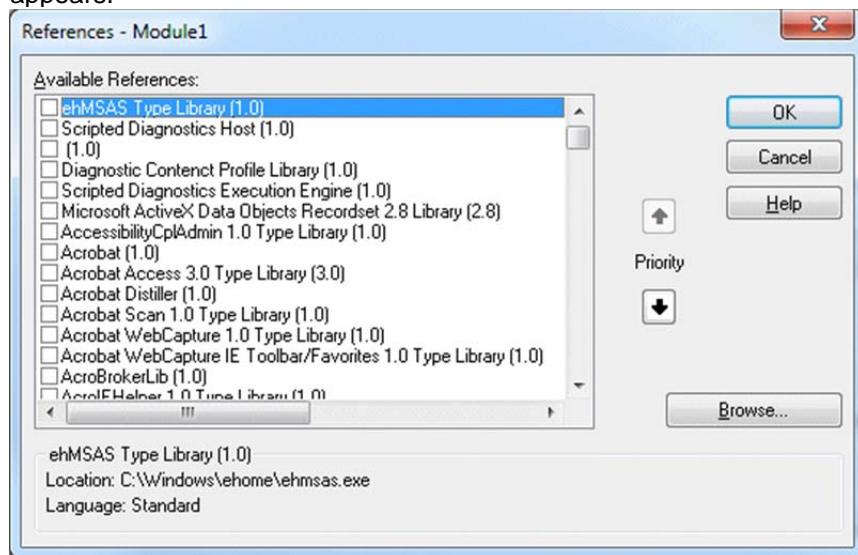
Enable XML Language Extensions provides additional parameters for manually specifying XML communication settings.

The **VBA Scripting – Preferred Tab Width** setting (default is 2) is the number of spaces indented by use of the Tab key when developing VBA programs.

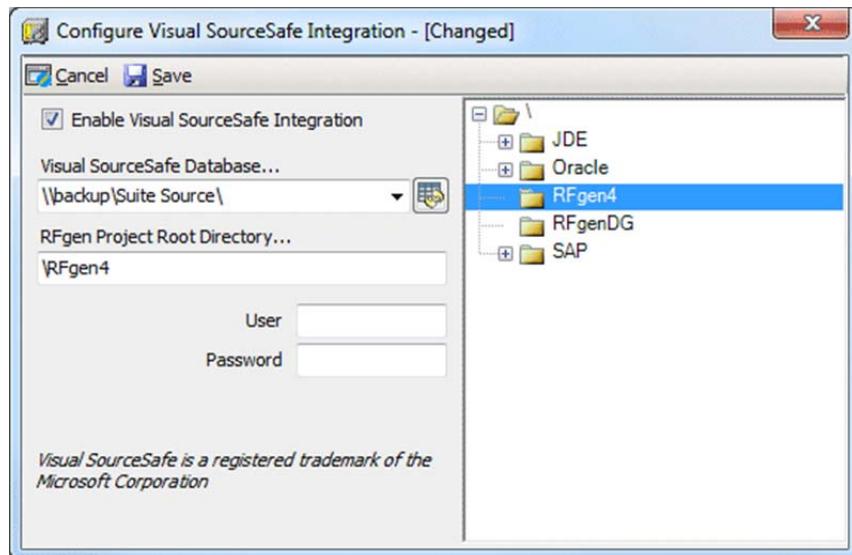
Permanently Loaded Modules

The Win32 and RFgen BAS files are always loaded for each transaction. If another BAS file is created and the programmer does not want to place the code into either of these pre-loaded modules then it may be added to this list. The Win32.bas is typically used to store global variables. The RFgen.bas is typically used to contain functions and procedures that need to be accessible from any transaction. If a BAS file needs to be referenced for only a few or 1 transaction, the VBA Scripts / References menu option should be used.

The Add References menu option will globally add Global Assembly Cache (GAC) classes to the RFgen solution. This is the window that appears.



Configuring Visual SourceSafe Integration



The first step in using Microsoft's Visual SourceSafe is to **Enable Visual SourceSafe Integration** from this window. This makes available the remaining fields.

Visual SourceSafe Database is the already configured location of the SourceSafe srccsafe.ini file. This is the only required field.

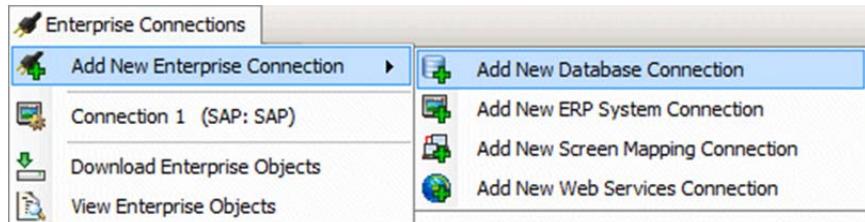
If the **RFgen Project Root Directory** is not already created then it can be created using the SourceSafe interface or the RFgen SourceSafe Browser located under the Utilities menu.

The **User** and **Password** fields are only necessary if the SourceSafe database requires authentication. If RFgen is running as a service and that account has access to the SourceSafe database, these fields may not be necessary.

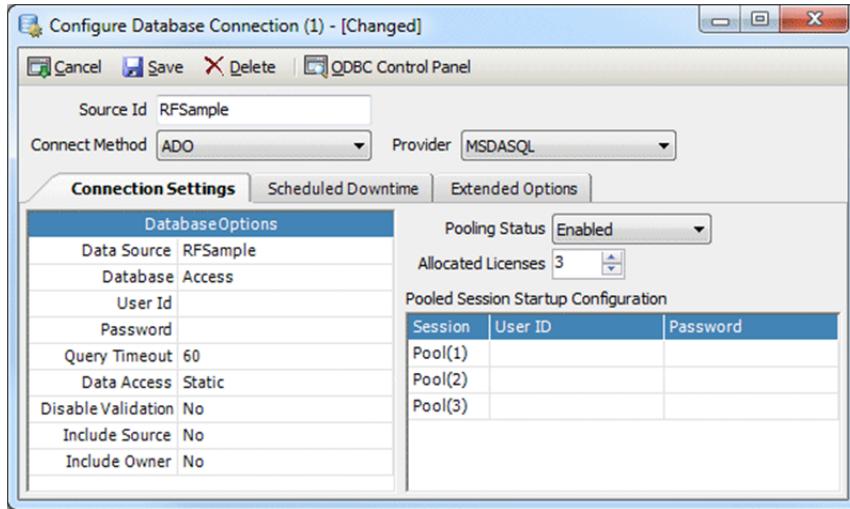
Click the Save menu option when finished.

Configuring ODBC Database Connections

From the Mobile Development Studio, click on the Enterprise Connections menu bar selection.



Click on Add New Database Connection (or on the existing Connection 1, as shown below) to establish (or modify) communications settings for your main application database.



The Mobile Development Studio Configure Database Connection Window will appear. The example that appears is for the RFgen sample applications database RFSample.mdb.

Upon installation, if you have chosen to load the RFgen sample files, the system database is will be configured to a Microsoft Access data source called RFSample (as shown).

At some point, you will want to configure Mobile Development Studio to your own application database. In theory, any SQL compliant database is usable with RFgen.

The **Source Id** is the name of data connection. Spaces and extended characters are not recommended for this field.

The **Connection Method** is the preferred method of connecting to the database. Most databases should be configured to use the ADO command set rather than the older RDO commands. If the database you are connecting to is very old or is unstable using ADO then RDO may be the safe approach. ADO (as shown) is the default used by RFgen. ADO is required starting with Microsoft Access 2000.

The **Provider** gives options for standard Microsoft SQL syntax, Oracle or IBM syntax. MSDASQL is the provider for ODBC, so in most cases, MSDASQL will work for all Microsoft provided ODBC data sources. Other providers are for specific databases and may be required in some cases. In a special case for OraOLEDB.Oracle, the Data Source name needs to match the entry in the TNS file and it does not use ODBC.

The Connection Settings are as follows:

Your database **Data Source** name (if you are not familiar with ODBC data sources, see that topic in this manual) is the ODBC name pre-configured in Control Panel. You may select from the drop-down box or manually type in an entry. Supported/tested database types are displayed as a drop-down menu for the **Database** entry; select the one that corresponds to your database.

For the **User Id** field, enter a user name (if required), and a **Password** (if required). Be sure that the user name entered has read/write privileges to the database. The user name and password are used as a default for all connections unless configured otherwise (see Connection Pooling below). Save your configuration settings by clicking Save. When RFgen connects to a database, it will display a connection indicator at the bottom of the RFgen window. If a red circle  appears in the indicator, a valid connection has not been made. To troubleshoot an invalid database connection, click on the Mobile Development Studio Reports menu bar selection, then Application Logs to see if a message has been generated. Most likely, a problem was encountered with your Data Source entry.

The **Query Timeout** value is how long (in seconds) RFgen will wait for the ODBC driver to come back with any type of response before reporting an error in the RFgen error log.

Data Access sets the cursor to either Static or Dynamic when retrieving data from the database. Usually Static is best because it is fast and safe. However if you have a database like Pervasive that will actually make a copy of the data from the database system to the RFgen system when using a static cursor, you can change this option to Dynamic, so performance will not suffer. Internally, this sets the cursor option to either adoOpenStatic or adoOpenDynamic.

The **Disable Validation** option will disable RFgen's ping to the database (using the statement "Select count(*) from RFGen_ConnCheck") determining if there is a good connection. If this statement fails, but RFgen thinks the connection is good, it will create the table so that it never fails again. It is not recommended that you turn this off, unless it is specifically causing a problem.

Including the Source or Owner information when downloading the tables are options to more uniquely qualify the tables or database

structure in case other connected databases have tables that are named the same.

The RFgen Enterprise Edition has the capability of having up to 5 data connections opened concurrently.

For example, your main database may be SQL Server or Oracle, but you have a need to connect to tables contained in other databases or entirely different ERP or legacy systems. Databases may be different in type, as long as they are SQL compliant.

Configuring Connection Pooling

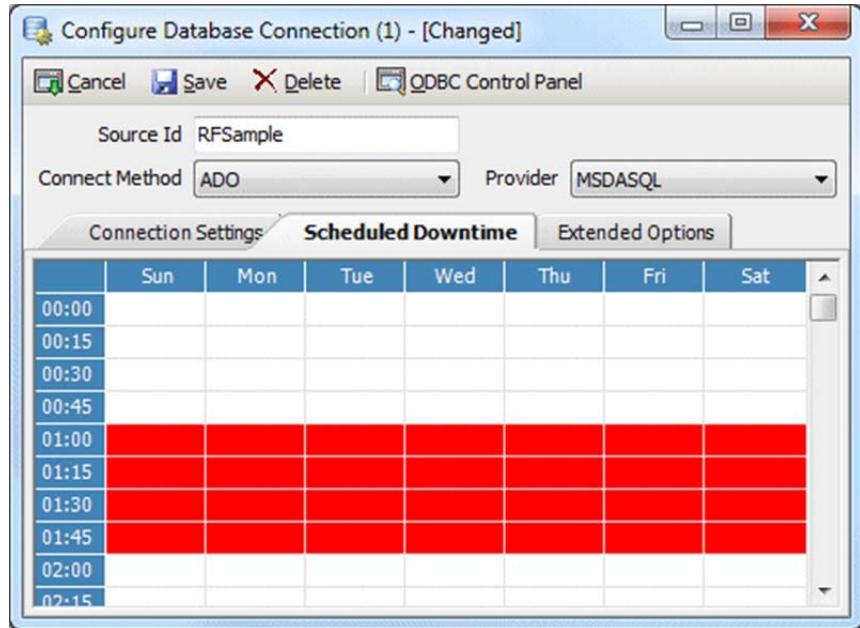
The Pooling option enables RFgen (Enterprise edition only) to pool multiple remote devices to share database licenses. For example, 10 (or more) devices may typically share 1 database connection when pooling is enabled.

As **Allocated Licenses** are incremented, a Pool(n) session appears for each. Enter a User ID and Password for each in the corresponding boxes (if different from the defaults).

The connection pooling User ID and Password fields contained in this window are for allowing users to log in under non-default settings. As each session is taken from the pool (when simultaneous access is required) the next pool's settings will be used. For example, if your system only allowed 2 connections with a particular User ID, Pool(3)'s User ID and Password could be specified and the first 2 will be taken from the default information.

Configuring Scheduled Down Time

The Scheduled Downtime option allows users to schedule down time for a database connection, during which RFgen will disable the connection, so that the database may be used without data collection interfering. The following panel will appear.

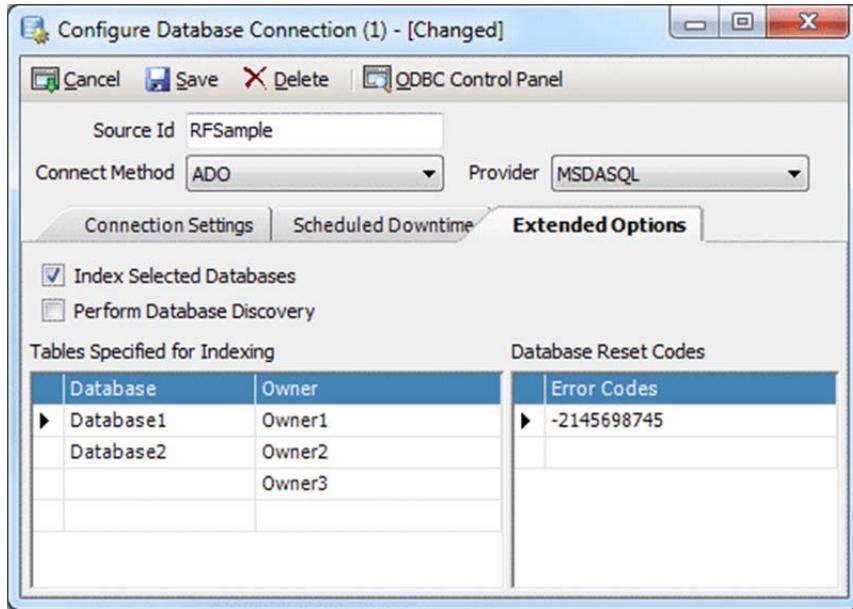


As shown, the connection is disabled between 1:00am through 2:00am every day of the week. Alternatively, you can select smaller amounts of time or individual days. Select the time and day with the mouse and use the spacebar to toggle the data connection on or off. A red square means that RFgen will not connect to the data source at that time.

Configuring Extended Options

This option contains the option RFgen will use to perform the indexing when the user saves a connection with the 'Index Selected Databases' option checked. RFgen will index tables within those additional databases so they can be referenced by name only. For example, in JD Edwards, F0005 is a control table. Using this indexing it may be accessed simply as F0005 and RFgen will qualify it with database.owner.F0005 internally before SQL execution.

The Indexing grid allows the user to restrict which tables are indexed for a specific data connection. In this grid, specify the list databases and / or a list of owners of the tables that are necessary for the data collection application. In the example below, only the tables from Database1 and Database2 with any of the 3 specified owners are indexed. Clicking the 'Index Database' menu option will save all changes and start the indexing process and then the tables can be referenced by name, assuming no duplicate table names.



The Perform Database Discovery option is a JDE specific option that will query certain JDE tables to determine how the system is actively configured for the selected environment.

Database Reset Codes is a place where you can add additional error codes that come back from the ODBC driver that should cause RFgen to re-establish a connection.

To delete an established data connection, simply click on the 'Delete' menu option.

Configuring ODBC Data Sources

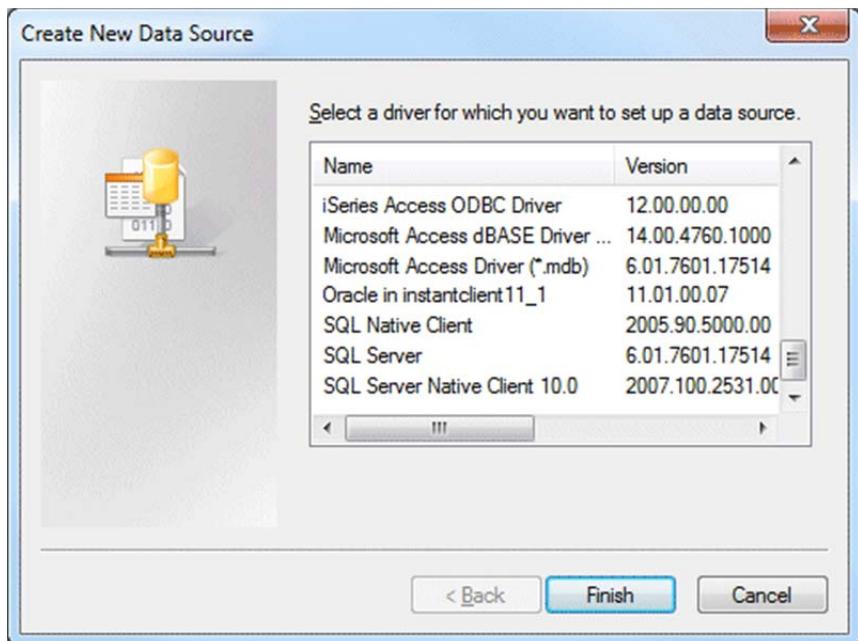
In the database connection window we noted that a 'Data Source', and an optional 'User', and 'Password' needed to be entered. This is true for each database listed.

Data Sources are normally established prior to configuring the Mobile Development Studio by means of the 'ODBC Data Sources' icon in your Control Panel 'Administrative Tools' window (i.e., click on Start, then Settings, then Control Panel, then double-click on Administrative Tools); then double-click on Data Sources (ODBC). Select the 'System DSN' tab.

If nothing appears in this window, click on the ‘Drivers’ tab. If nothing appears in the Drivers tab window, use the RFgen CD to load the Microsoft Data Access Components (MDAC) drivers. The RFgen Release 4 ships with MDAC 2.8. **If your existing MDAC is less than 2.8, please install 2.8.** If a special ODBC driver is needed for your application database and it does not appear, load it using the CD provided by the database manufacturer.

The DSN tabs provide you with a list of established User data sources. Clicking on the ‘User DSN’, ‘System DSN’ or ‘File DSN’ tabs may show more data sources. If a data source name does not appear for your application database, you should add a data source at this time.

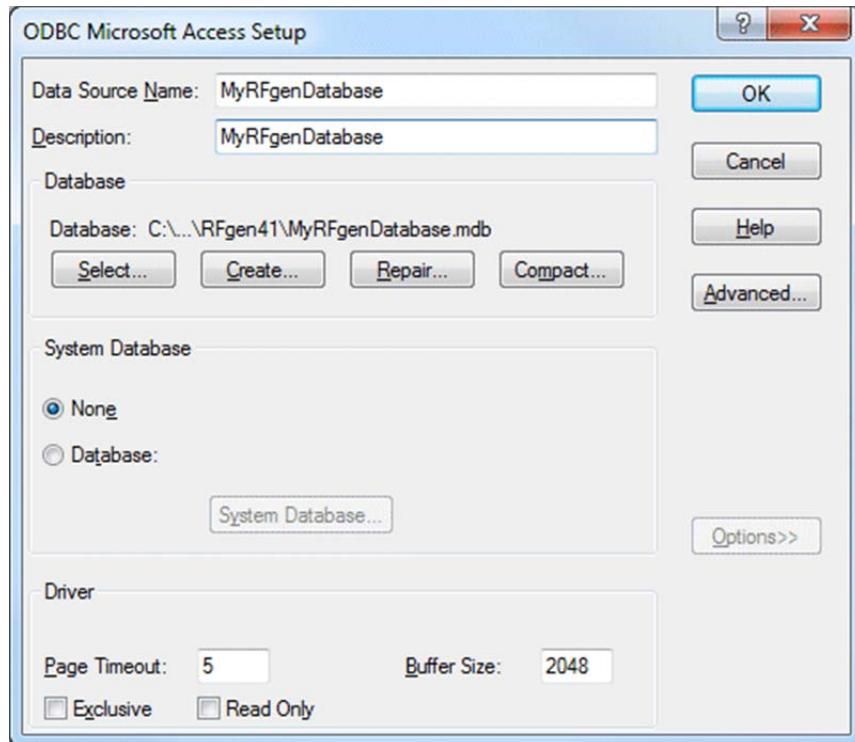
When you click ‘Add’ the ‘driver selection’ window will appear.



Here, select the driver used with your application database and click ‘Finish.’ If an ODBC driver for your database does not appear, you will have to load it from the CD provided by your database manufacturer. Select the ODBC driver to use and click ‘Finish’.

The data source window that appears will depend on the ODBC driver being used. Shown below is a Microsoft Access database data source named ‘MyRFgenDatabase’ located in the ‘C:\My Documents’ folder.

Here, we simply filled in the Name and Description boxes and clicked on the 'Select' box to browse for the Access database file 'MyRFgenDatabase.mdb'. You can see the path to the database directly above the 'Select' box.



Click OK to save your data source.

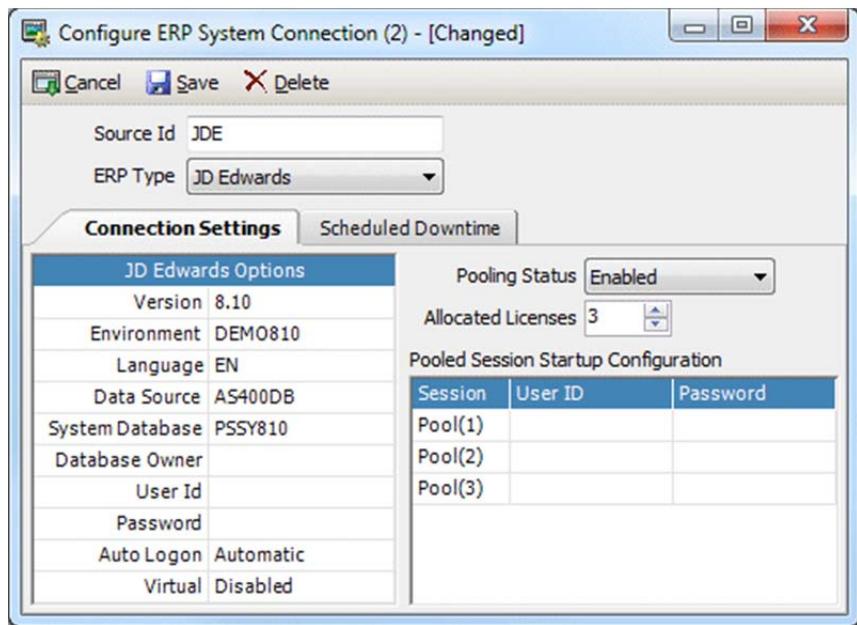
The sample RFgen data connection originally configured (Data Source Name: 'RFSSample') contains the tables used to illustrate Mobile Development Studio and is similar to the above (except that the RFSSample.mdb Access database is located in the RFgen folder, which was created when you loaded the Mobile Development Studio Software).

Configuring a Screen Mapping Connection

For screen mapping related information, please see the RFgen Screen Mapping section.

Configuring an ERP Connection

To configure an ERP connection, select “Add New ERP System Connection”, the following configuration window will appear.



First, give the connection a **Name or ID** and then, choose the **Type** of ERP package you are using.

In the case of JD Edwards, first choose the **Version** of the ERP system. Next, specify the JDE **Environment**. The default **Language** is English, but it can be changed if necessary. The next field is the ODBC **Data Source** entry for the database JD Edwards uses. RFgen accesses this data connection for all the read-only access requests because it is much faster to extract data using SQL statements, than going through the business function layer.

Since this ODBC connection may have many environment databases, use the **System Database** field to specify the proper database. Use the **Database Owner** option to specify the owner of the tables in this database. A **User Id** and **Password** are typically required and are entered next.

The **Auto Logon** mode has 2 settings: Automatic and Manual. The default is Automatic and when RFgen starts or when a client connects,

the default parameters specified are used to make a connection to JD Edwards immediately. Choosing Manual means that the code in the application will log the user on and off the system.

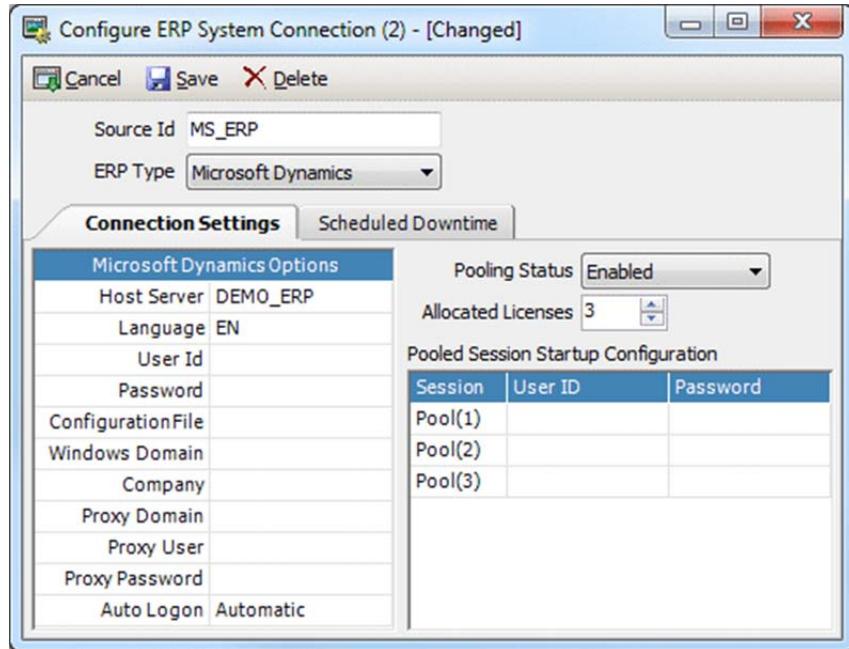
The **Virtual** client option makes each JD Edwards client a separate process from the RFgen process so that if the client should fail it will not bring down the RFgen service as a whole. DataMAX recommends this option be turned on.

The ‘Pooling’ option enables RFgen (Enterprise edition only) to pool multiple remote devices to share ERP licenses.

As ‘Allocated Licenses’ are incremented, a ‘Pool(n)’ session appears for each. Enter a User, Password and Startup Mode for each in the corresponding boxes (if required). ‘User Id’ and ‘Password’ defaults are provided by the entries made with the ‘Connection’ option.

The connection-pooling User ID and Password fields are for allowing users to log into the ERP system under non-default settings. As each session is taken from the pool (when simultaneous access is required), the next pool’s settings will be used. For example, if an ERP system only allowed 2 connections with a particular User ID, Pool(3)’s User ID and Password could be specified and the first 2 will be taken from the default information.

The Scheduled Downtime option is identical to the database connection. Save the settings when all the fields are filled in and RFgen will make the connection.



For Microsoft, change the **ERP Type** to the desired setting (Axapta or Dynamics). This setting does change the list of option slightly.

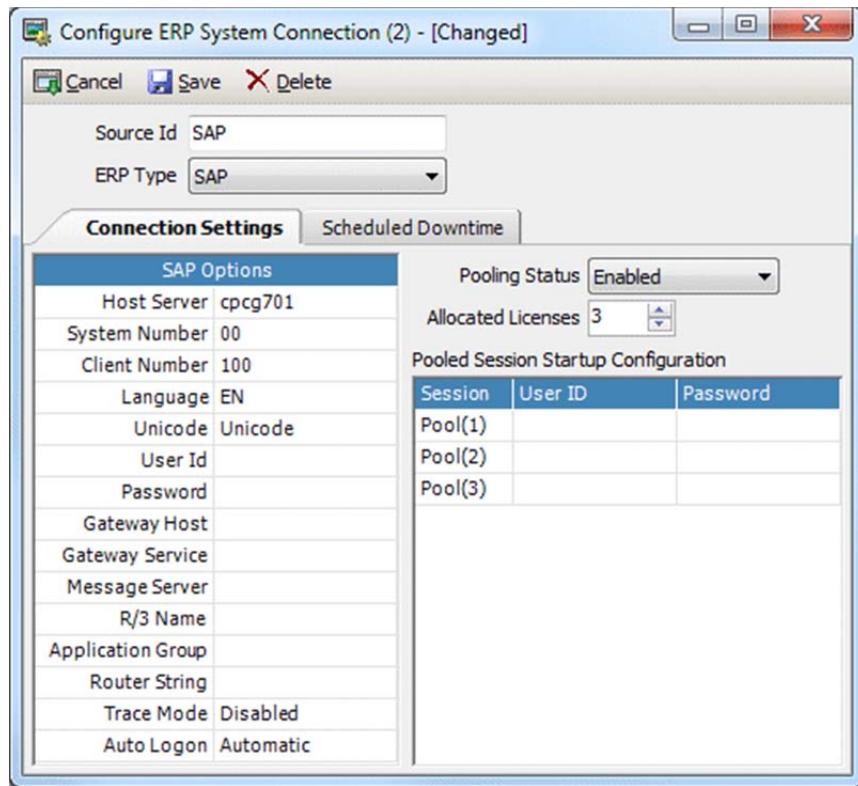
Next, set the **Host Server** and **Language** for the Microsoft ERP. **User Id** and **Password** are for logging in to the ERP system. Microsoft also provides a **Configuration File** that RFgen reads for additional settings. This file is where the Axapta client is installed on the local PC.

Next, enter the **Windows Domain** where the ERP system resides.

In the case of Axapta, fill in the **Windows User Id** and **Windows Password** for connecting to the domain. These are the Windows domain and credentials for DCOM security. For Dynamics, you must specify the Dynamics **Company** name and the Dynamics proxy for the **Domain, User, and Password**.

The **Auto Logon** mode has 2 settings: Automatic and Manual. The default is Automatic and when RFgen starts or when a client connects the default parameters specified are used to make a connection to the system immediately. Choosing Manual means that the code in the application will log the user on and off the system.

The Pooling and Scheduled Downtime options work exactly the same as described above.



For SAP, the **Host Server** is the application server. Enter the **System Number**, **Client Number**, **Language**, **User Id** and **Password** with the same data as stored in the SAP GUI Logon Pad application.

The **Unicode** option tells RFgen how to interact with the system, either by using Unicode formatted communication or non-Unicode (ASCII) formatted data.

The **Gateway Host**, **Gateway Service**, **Message Server**, **R/3 Name**, **Application Group** and **Router String** are optional parameters. If your Logon Pad requires these settings, then add them here for RFgen.

For SAP load balancing configure the following fields: System Number, Client Number, User ID, Password, Message Server, R/3 Name and Application Group. Leave the Host Server blank since the Application Group setting will distribute requests to multiple host servers.

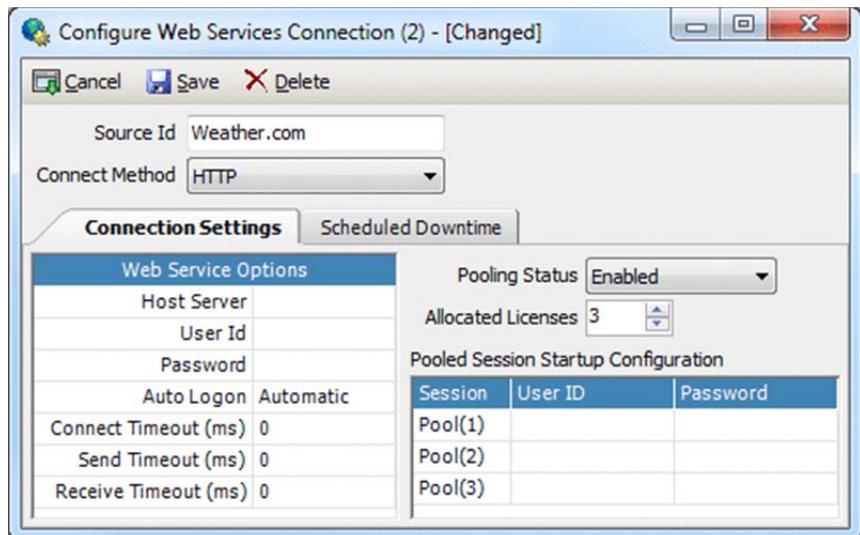
The **Trace Mode** turns on or off the logging of all business function traffic between SAP and RFgen.

The **Auto Logon** mode has 2 settings: Automatic and Manual. The default is Automatic and when RFgen starts or when a client connects the default parameters specified are used to make a connection to SAP immediately. Choosing Manual means that the code in the application will log the user on and off the system.

The Pooling and Scheduled Downtime options work exactly the same as described above.

Configuring a Web Connection

To configure a Web connection, select “Add New Web Services Connection.



Enter a **Source ID** which will be the name to reference when Web object's DataSource property.

Choose a **Connect Method** of HTTP or HTTPS.

The **Host Server** is the IP address or DNS name of the server being used to process requests.

A **User ID** and **Password** can be entered if required by the server.

The **Auto Logon** is either set to Automatic or Manual. In automatic mode, as soon as the RFgen client makes a connection to the RFgen server, this web connection will be started and logged in to if credentials are also entered. In manual mode, the Web object can be used to specify a user and password just before the execution of the HTTP request.

The **Connect Timeout** is a number in milliseconds that will terminate a request for connection if this value is exceeded.

The **Send Timeout** is a number in milliseconds that will terminate a request from the client sent to the server if it has not received the HTTP request from the client.

The **Receive Timeout** is a number in milliseconds that will terminate a response from the server to the client if this value is exceeded.

The **Ignore Certificates** option is a True or False value (only for HTTPS connections) indicating that the data connector will ignore certificate errors from the server. If this is set to False, and there is an error, it will be logged in the RFgen error log and the Web object's Execute method will return a False value.

The 'Pooling' option enables RFgen (Enterprise edition only) to pool multiple remote devices to share Web service connection.

This option is identical to the database, ERP and screen mapping connectors.

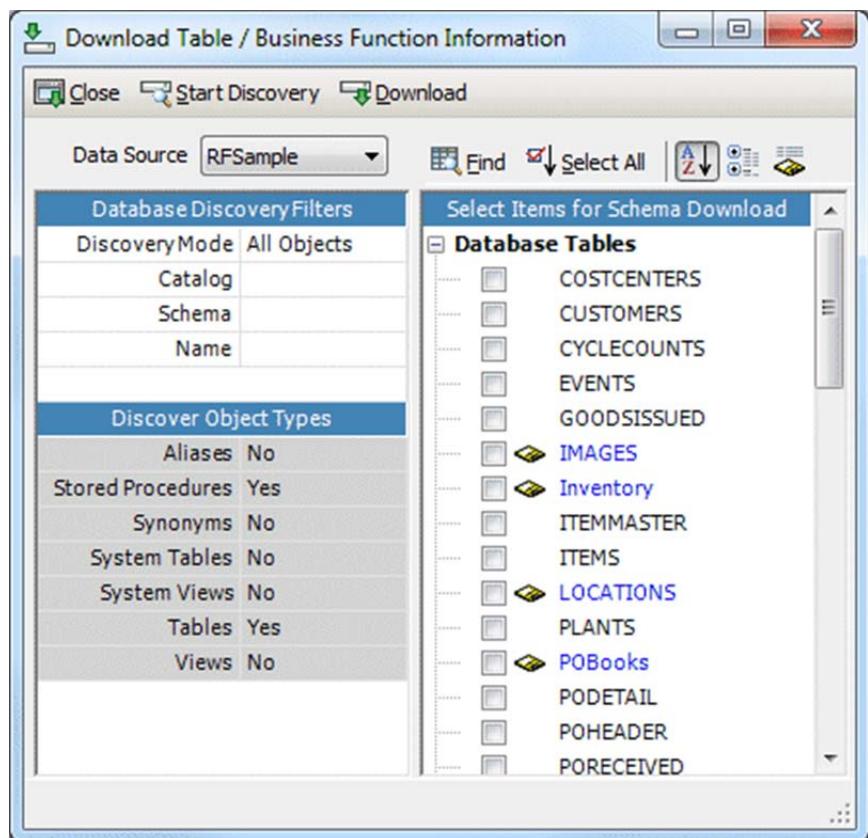
The Scheduled Down Time option is also identical to the other connections. Save the settings when all the fields are filled in and RFgen will make the connection.

Download Tables/Business Functions

To use table fields directly on an application screen, RFgen needs to know which data connection contains the database, which tables are required and what the table structure looks like in order to internally generate the proper SQL and perform the reads and writes without the programmer using the scripts. To do so for ODBC connections:

1. Click on the Connections menu item
2. Click on Download Tables / Business Functions

3. Select the database from the Data Source drop-down
4. Click the Start Discovery menu item
5. Click the rows for tables to be downloaded (or click Select All), and then click Download.



The Discovery Mode option allows for all objects to be discovered or just selected items that may be selected.

Blue items have been previously downloaded. The checked tables will be transferred into RFgen, which uses information from downloaded items when creating data entry / display applications.

 If there is a long list of items, the find option can narrow the focus. Using the “Display On-File Items Only” option will limit the list to previously downloaded tables.



This button selects the alphabetical list view and enables the Filter feature.



This button selects a tree view of the list of items.



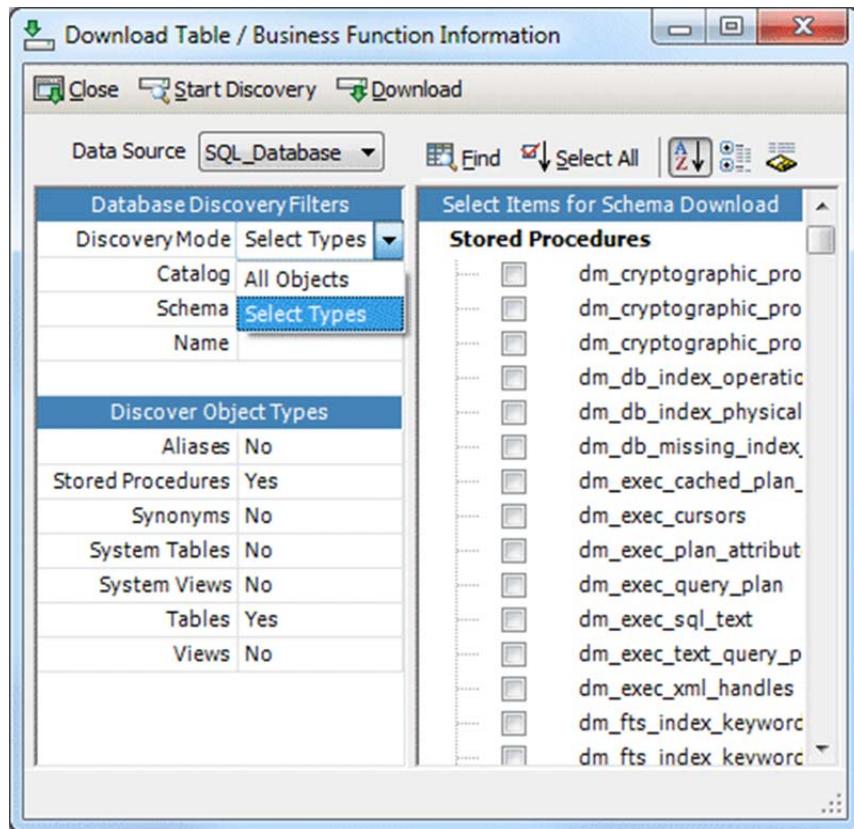
This button displays only the previously downloaded items and enables the Find feature.

Since RFgen is SQL compliant, it is important to note that database table and field names should not use any of the reserved words listed in the section describing the VBA commands.

Downloading Stored Procedures

To work with stored procedures, users must first transfer (download) the desired stored procedure from the database into RFgen. To do so:

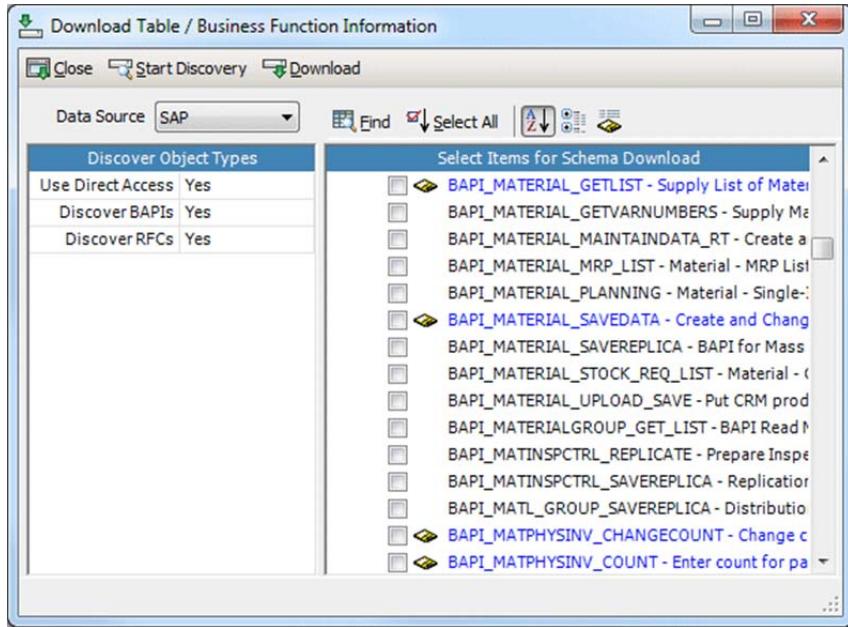
1. Click on the Connections menu item.
2. Click on Download Tables / Business Functions.
3. Select the database from the Data Source drop-down.
4. Click the Start Discovery menu option
5. Click the rows for stored procedures to be downloaded (or click Select All), and then click Download.



Downloading ERP Business Functions

To work with business functions from an ERP system, users must first transfer (download) the desired business functions from the ERP system into RFgen. To do so:

1. Click on the Connections menu item.
2. Click on Download Tables / Business Functions.
3. Select the ERP connection from the Data Source drop-down and select "Start Discovery" from the menu.
4. Click the rows of functions to be downloaded (or click Select All), and then click Download. Selecting all business functions from an ERP system will save an extremely large set of data in the RFgen master database and could take a very long time. Only download the business functions that are required by the applications.



Blue entries have been previously downloaded. The SAP Discovery Filters allow you to select, if just BAPIs are downloaded or if RFCs are as well.

View Downloaded Tables/Business Functions

Table data definitions may be viewed using the Mobile Development Studio by clicking on the Connections – View Downloaded Tables/Business Functions selection. A view window will appear.

##	Parameter Name	Data Type	Size	PriKey	Update	Null	Reqd	Description
1	PartNo	WChar		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Our Part Number
2	Description	WChar		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Description
3	OnHand	Integer		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Qty On/Hand
4	AvgCost	Numeric		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Average Cost
5	LastRcvd	DBTimeStamp		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Last Receipt Date
6	LastSold	DBTimeStamp		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Last Sale Date
7	LastCost	Numeric		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Last Purchase Cost
8	Price	Numeric		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Current Sale Price
9	Rack	WChar		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
10	Status	WChar		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
11	ProdImage	Binary		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Product Image

Select a name and click the “View Schema” menu option or simply double-click the table name to view its parameters.

Shown above are the field definition items for the chosen table in our sample/demo Microsoft Access database (RFSample.mdb) ‘Inventory’ table. Each transaction table must have at least 1 primary key ('PartNo' as indicated above). RFgen identifies database keys simply by determining which database items are 'indexed'. If more than 1 item is indexed, the first item encountered will be marked as the primary key.

In general, use of Numeric, Text/String, Date, and Currency ‘Data Types’ in your database is suggested, as more esoteric data types may cause problems when trying to update your database table(s).

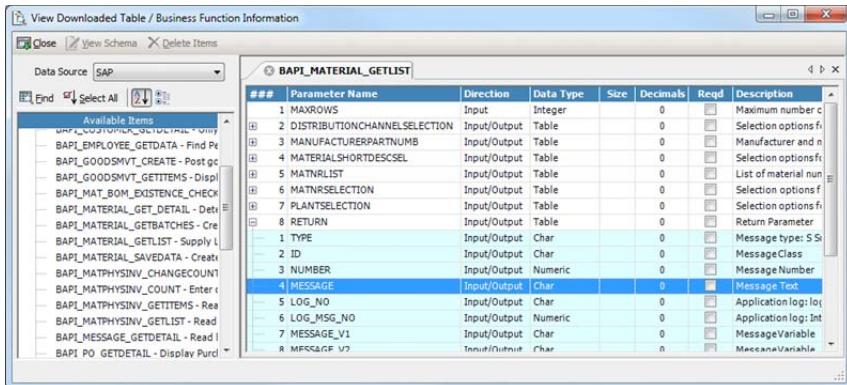
Only fields defined as updateable will be sent to the database when a transaction is completed. This is only true if RFgen is generating the SQL statements internally as opposed to user created SQL statements. Fields not defined as 'null allowed', will send a space or a 0 (zero) if no data is collected for them. Fields marked as 'primary key' are used to access the table data and may be used to retrieve selected data.

When viewing the SAP specific function properties after a download, please note that for “packed” or compressed numeric data elements RFgen displays the byte length and the allowed number of decimals instead of the actual number of characters allowed in the field.

Selecting a table entry and choosing Delete from the menu only removes the stored structure of that table from the RFgen configuration. This delete has no impact on the actual database.

Viewing ERP Business Functions

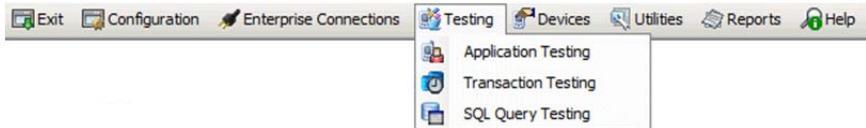
Business Function definitions may be viewed using the Mobile Development Studio by clicking on the Connections – View Downloaded Tables/Business Functions selection. A view window will appear.



A list of downloaded business functions is displayed. If a description is stored in the database and was retrieved by RFgen, it can be displayed after viewing the parameters. Select a name and click the “View Schema” menu item or simply double-click the business function name to view its parameters.

Mobile Development Studio Menu Bar

The Mobile Development Studio 'Menu Bar' displays below the 'RFgen: Mobile Development Studio' banner, as shown here.



Here click on:

Exit: To exit/close the Mobile Development Studio

Configuration: (1) Mobile Application Database configuration and encryption options
(2) Desktop Preferences for colors, fonts, default design mode and alternate languages
(3) Mobile Device Themes changes interface color schemes
(4) Mobile Security Options provide wireless authentication and encryption configurations
(5) Speech Recognition Options and speech authorization
(6) System Environment settings including system options, timeout values, Pre/Post-amble scanner defaults, embedded graphical menu options, function key options and system properties
(7) TCP/IP Service Options include port configurations
(8) Transaction Management connection setup, queue setup, timed event setup, scheduled downtime and debug tracing options
(9) VBA Scripting Environment settings like ADO, RDO, XML extensions and globally loaded BAS files

Enterprise Connections:

(1) Specify the types and locations of additional SQL compliant database connections, ERP connections, screen mapping hosts systems and Web service hosts

- (2-6) The list of configured data connections
 - (7) Download Enterprise Objects like tables, business functions or stored procedures
 - (8) View Enterprise objects like tables, business functions or stored procedures
- Testing:**
- (1) Application Testing to debug applications
 - (2) Transaction Testing to debug transaction macros
 - (3) SQL Query Testing to open an SQL window, to view local database tables and data
- Devices:**
- (1) Deploy Mobile Applications can place a small client (CNC) on the device so RFgen can make changes and updates automatically and full profiles
 - (2) Remote Application Explorer will show the users, menus, applications, etc. installed on the device
 - (3) Remote Database Explorer shows the data on the mobile device's RFgen database
 - (4) Remote SQL Explorer will let the user read and write directly to the user database on the device
- Utilities:**
- (1) Export Mobile Applications exports RFgen objects (applications, menus, users, VBA code; see the next section)
 - (2) Import Mobile Applications will import RFgen objects
 - (3) Validate Application Scripts will syntax check all application scripts, VBA modules and transaction macros and display which ones have errors
 - (4) Find in Application Scripts creates a list of all applications, modules and macros that contain what was searched for
 - (5) Replace in Application Scripts will perform the find and replace in all applications, modules and macros for the criteria given
 - (6) Undo Save/Delete Action
- Reports:**
- (1) Event Logs
 - (2) Application Statistics
 - (3) View Transaction Queues
- Help:**
- (1) Mobile Development Manual displays the RFgen manual
 - (2) VBA Scripting COM Syntax displays VBA COM language options

- (3) VBA Scripting .NET Syntax displays VBA .Net language options
- (4) Obtaining Technical Support has links and e-mail addresses
- (5) Authorize RFgen Voice Development
- (6) About information includes installed options and the version

Testing

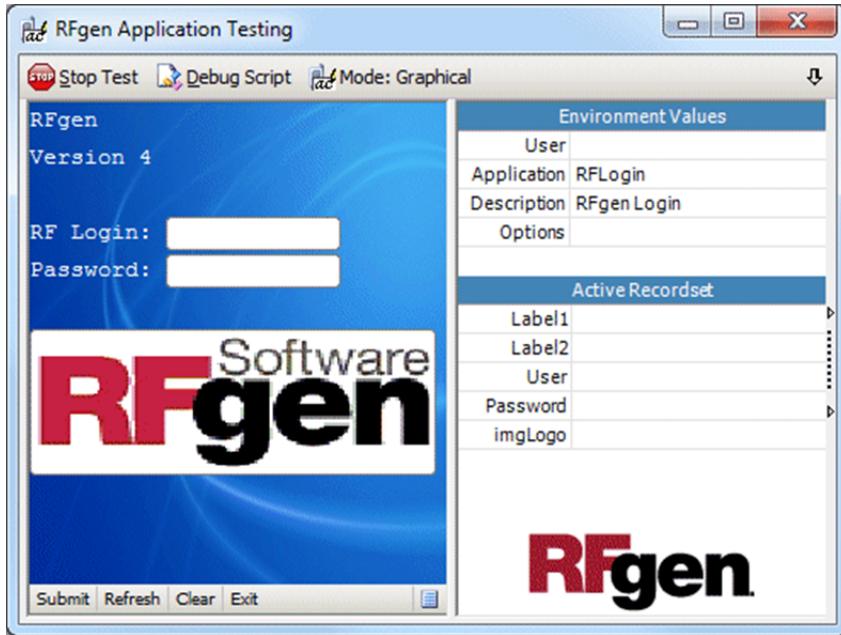
The testing menu option contains the different places developed objects can be exercised. The testing of applications, menus, users, global programming code, data connectors, mobile devices and some other items are all tested using the Application Testing option.

For testing transaction macros, Timed Events and queuing choose the Transaction Testing option.

To test specific SQL statements, interact with table data or test the data connectors themselves choose the SQL Query Testing option.

Application Testing

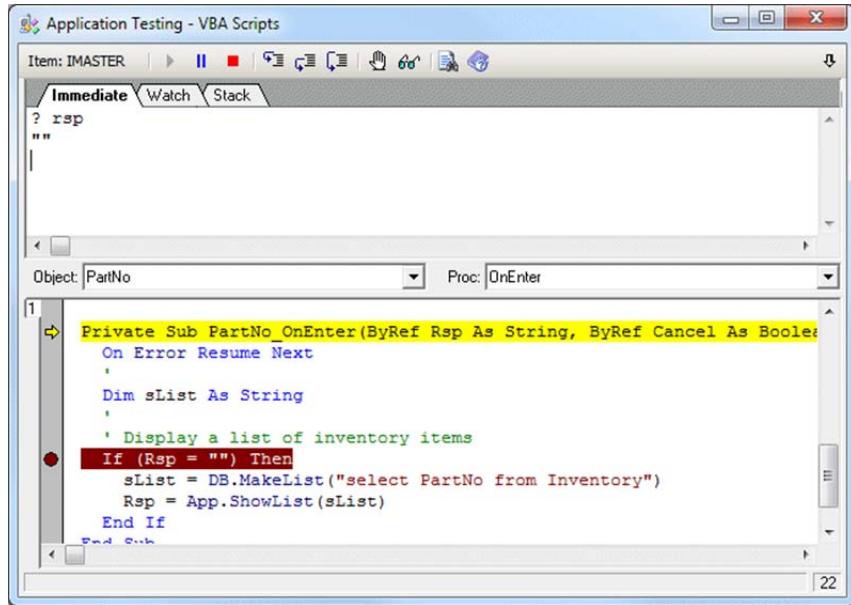
The Application Testing menu option is used to test RFgen applications in a local mode. There are 5 different modes of testing. Character mode, graphical mode, mobile mode, service mode and Vocollect testing may be pre-selected for the simulated device display screen. The Mobile option is for debugging code running on the mobile device. Service mode displays the underlying data stream from such processes as XML or Winsock connections. The Vocollect mode displays the data stream from the Vocollect server as it goes through RFgen to the mobile device.



When the testing window first appears, the display screen will be blank. To start the testing process, click on the 'Start Test' menu selection. This will allow you to Login via one of the user accounts that have been established, e.g. for the sample applications the user is 'SAM', with no password. The Environment Values display (when the pushpin icon is toggled) the user logged in, the application or menu currently presented and its description and any options that will be passed in to an application from a menu that contains passing parameters. The Debug Script menu option displays a code window used to debug applications.

The Mode menu option is a drop-down option. Select the proper mode and then click the Start Test menu option to begin testing.

The Debug Script menu option displays a window containing the Immediate, Watch, Stack and code windows. These windows are similar to the programming environment of Microsoft VB and only the differences will be discussed here.



The most effective way to debug code is to click on the blue pause button and then trigger an event. Code execution will be paused on the first line of that event. To debug code in the Form Load event, add the line 'Stop' so that execution is halted on that line.

After you have entered your user ID and password, the menu assigned for the user will display. Use your keyboard/keypad arrow keys to select an application and press <Enter>. In a graphical mode you may use the mouse to select and execute menu items. In test mode, you may enter data exactly as if you were using a remote device.

The 'Active Recordset' grid shows what your data item (record) looks like as data is added.

Click on 'Stop Test' to stop the testing process.

Network Application Testing

For an actual RF network test of your RFgen applications, you'll first need to configure your remote data entry devices with either:

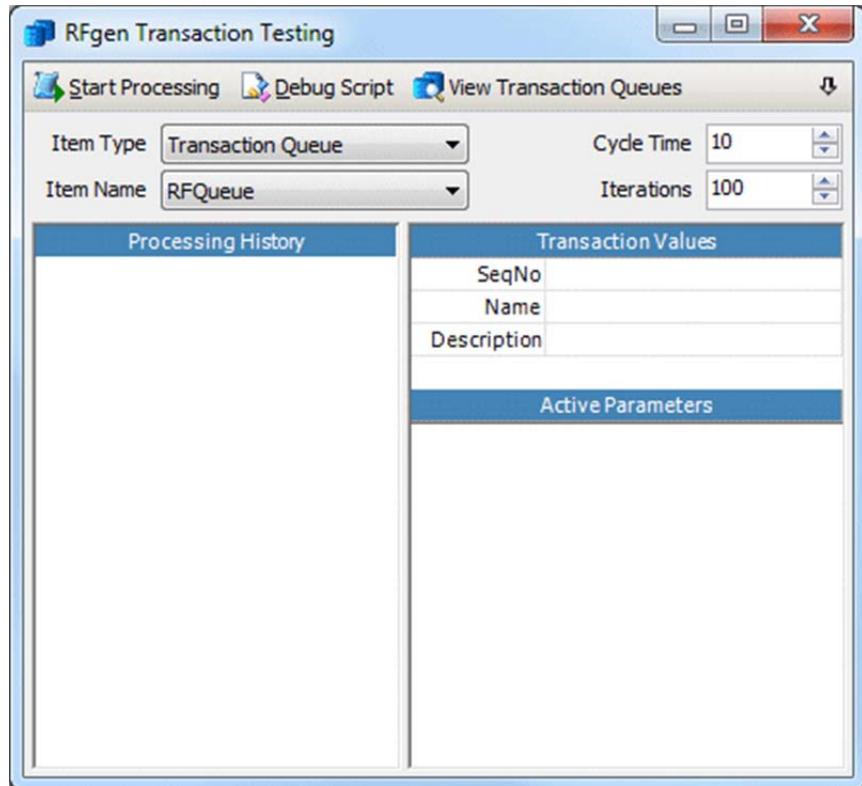
- (1) FOR CHARACTER-BASED DEVICES ...TCP/IP Telnet communications software using VT220 emulation, or
- (2) FOR GRAPHICAL DEVICES (including WinCE-based, PocketPC or Mobile) ... the RFgen 'desktop' or 'graphical client', available on the RFgen Software CD.

Hard-wired network devices running Windows may also connect as character-based applications by using the Windows ‘Telnet.exe’ program but the graphical telnet is preferred.

Mobile Development Studio, by itself, allows one usable network device to log in, for test purposes (for 30 days), without requiring an RFgen Software license. Multiple device tests require using the Server that can allow up to 254 concurrent users for a period of two hours until the service is stopped and restarted. An RFgen Software license is required to permanently activate your RFgen network (by means of the RFgen ‘Service Control Manager’ (SCM) icon that then appears on your Windows system tray).

Transaction Testing

This option will let the user test Timed Event macros and queue processing.

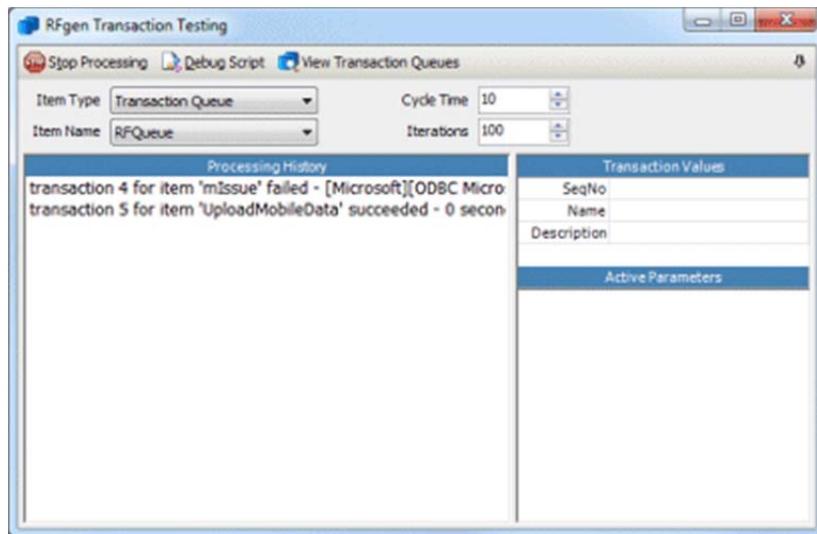


The **Item Type** option switches between queue processing and timed Events. If Transaction Queue is selected, the **Item Name** specifies which queue is to be processed. If Transaction Event is selected, then the Item Name list is populated with the configured events under the Transaction Management / Processing Events option.

The **Cycle Time** is an interval in seconds that is how often the queue will be checked for new transactions. For events this is how often RFgen waits between each execution of the event.

RFgen will continue testing for a total number of times specified in the **Iterations** box.

Select the item to be tested from the drop-down options. In the case of Transaction queues the user should have already queued what they need tested. The window pane on the right will display the parameters of the item being tested.



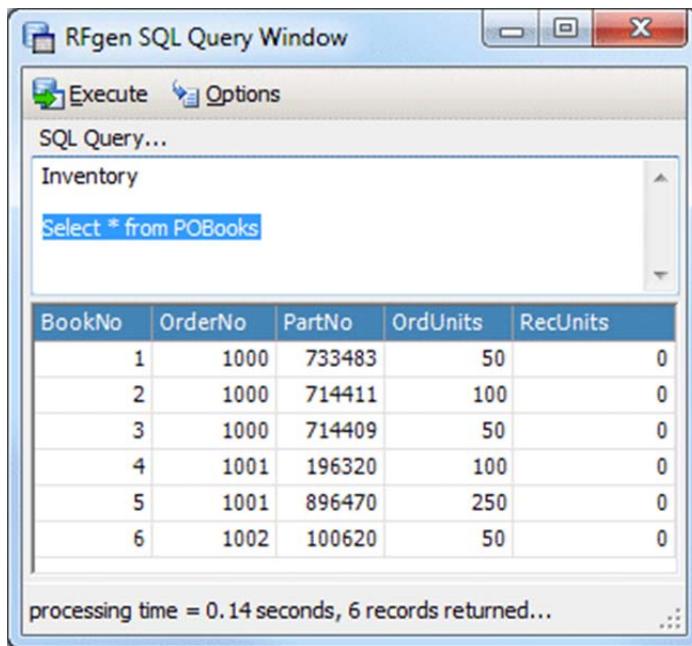
As testing is taking place the window will show a status of the processing. If a transaction fails, the error message is displayed. If it is successful, the amount of time required to process that transaction is displayed with a success message.

The Debug Script menu option allows the user to stop, debug and test the scripting of the Transaction or Timed Event macros as they are executed.

SQL Query Testing

RFgen provides a utility that lets the user test SQL statements to see the results before executing them in the code. This utility can also be used to undo updates, check results, delete values or even adding or dropping tables. Any SQL command entered here is submitted to the ODBC driver for execution. There are no limitations by RFgen as to what can be submitted.

If the intent is “select * from TableName”, then only specifying the table name will default to the “select *” when executed.



The multi-line text box allows the entry of several SQL statements. In this case, highlight the intended SQL statement and click Execute from the menu.

Multiple SQL statements must be separated by semi-colons “;”
Select * from inventory;
Select * from pobooks;

These will be considered 2 different SQL statements. If no text is highlighted, then it will look at the current insertion point to determine which SQL statement to execute based upon semi-colon delimiters. Further, if you click on Options menu / Display Query History – then

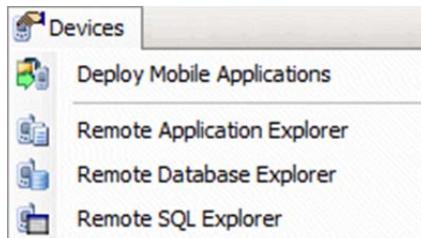
double-click on an item, it will append it to the SQL window instead of replacing the existing contents.

The SQL Query Window gives a current snapshot of the data in your database. As transactions are applied against your data, you will need to re-execute your SQL statement.

The Options menu will allow the user to select from any of the previously executed SQL commands and to also limit the output to a maximum number of records. The default is 1,000 records.

Devices

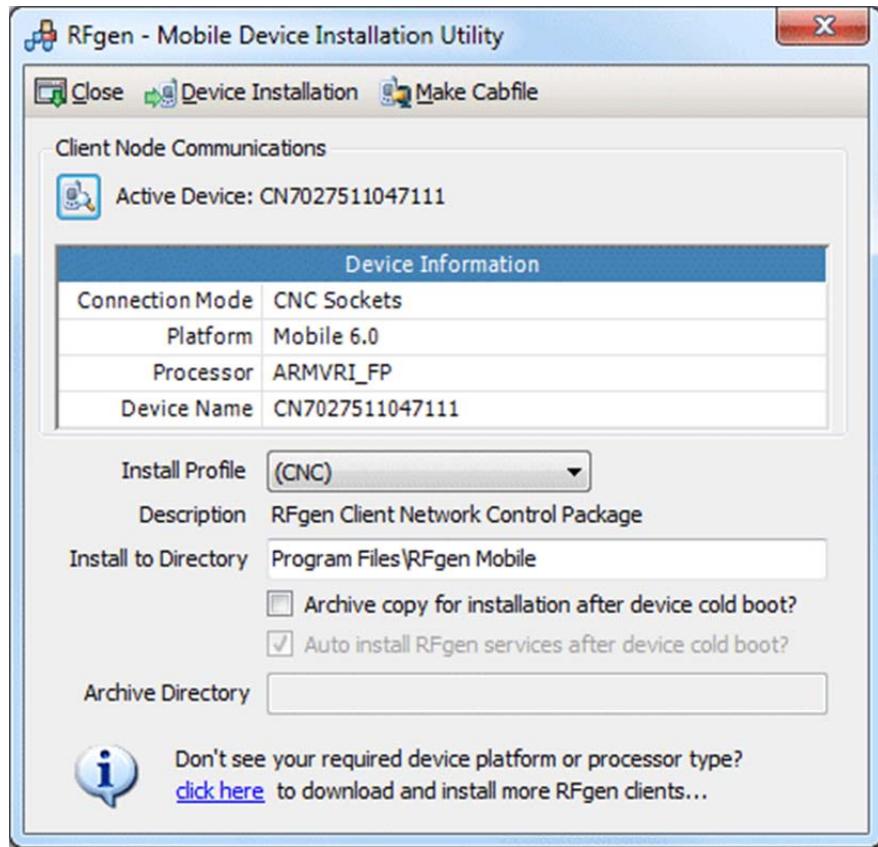
The Devices menu contains the applications to control and deploy files to mobile devices. The mobile device must contain the RFgen “listener” application called the Client Network Control. All communication and interaction with the mobile device go through this application.



Deploy Mobile Applications

The Client Network Control application is used by RFgen to provide access to the file system of the device. RFgen can then update Applications, Menus, Users, etc as part of the data collection need and also update the underlying RFgen client files in cases where the server is upgraded and new client files need to be distributed. Full profiles can also be sent to the device.

From the menu choose Devices / Install Applications on Device.



For this initial install, the mobile device needs to be connected to the Mobile Development Studio through Microsoft ActiveSync and the Mobile Client files must be installed on the machine running the Mobile Development Studio.

If RFgen can detect the settings, it will pre-populate the options automatically.

The **Active Device** is informational only.

The mandatory parameters are the **Platform** and the **Processor**. These combine to tell RFgen which files must be installed on the device. Usually in the Control Panel, on the mobile device, there is an About program that will show these values. If the processor says some variation of "X-scale", it is referring to the ARM processor type. It is recommended to use the latest ARM driver in the list first and only choose others if there are compatibility issues.

The **Device Name** is equivalent to the PC name as seen by the network.

The **Install Profile** option allows the user to select between the CNC deployment or any saved device profiles.

The **Installation Directory** is where all RFgen files will be placed.

The **Archive Copy** option places a CAB file on the storage card that can be run again as needed to install the RFgen software.

The **Auto Install** option places a CAB file in a specific folder that is used by the operating system to automatically install any CAB files placed in that folder.

The **Archive Directory** usually is a place on the storage card where a backup of the CAB file resides.

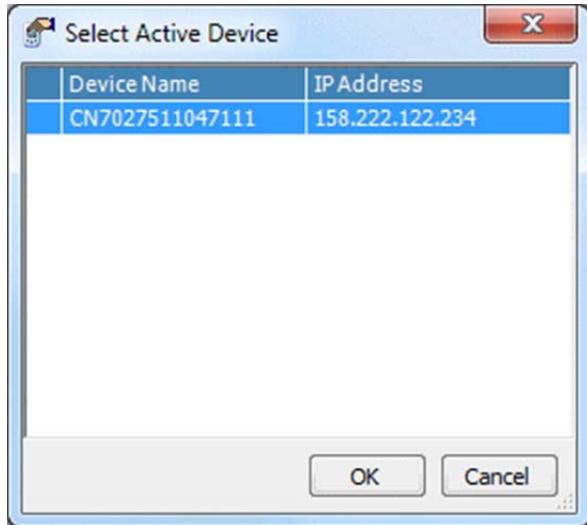
Click the **Device Installation** option to download and install the Client Network Control application or profile. Once this is done, the mobile device can be disconnected from the server. Click the **Make Cabfile** option to create an installation CAB file that will be manually deployed to the mobile device at a later date.

The link at the bottom of the window will take the user to a web page where additional files may be downloaded, if various files for devices were not installed.

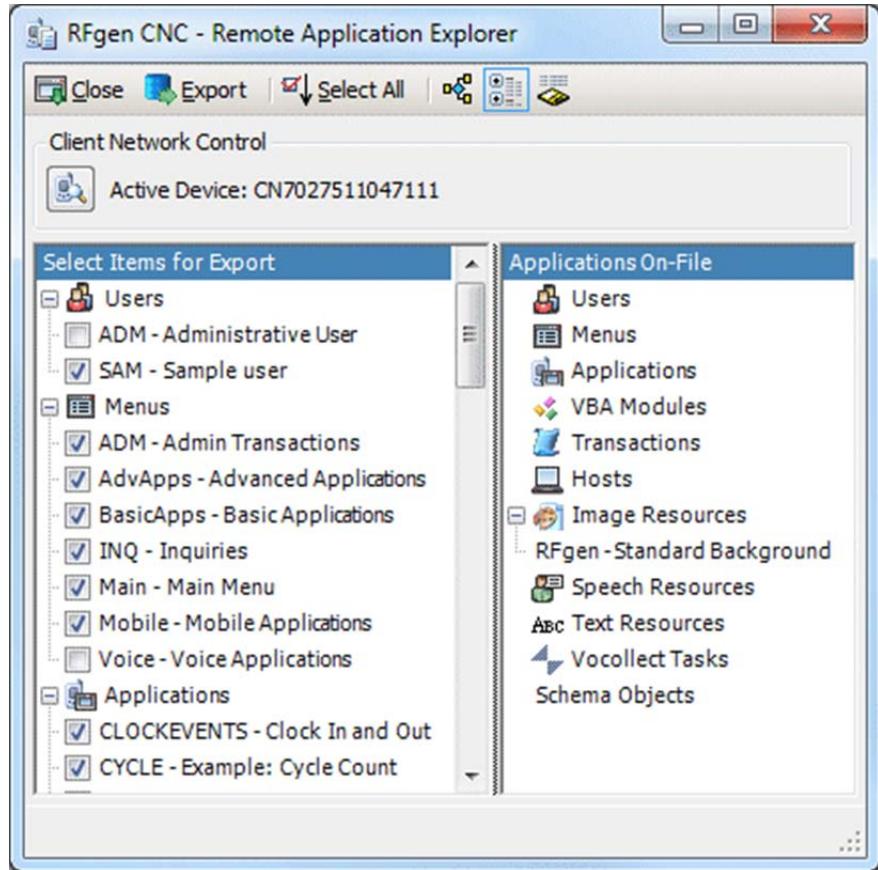
Remote Application Explorer

This screen is used to send updated or additional components to a mobile installation on the device. Thin client solutions do not have users, menu, applications, etc.

Click on the Remote Application Explorer menu option and then select the active device that requires an update. (See the Server.SyncApps command to update multiple devices.) Click the button in the Client Network Control group to bring up this window.



Double-click the proper entry or use the Select Device menu option.
This will take the user back to the previous screen.



This list on the left is the available objects that may be selected for export to the mobile device. Using the check boxes or the Select All option and toolbar options, select the required items and click Export from the menu. The mobile device now has an updated copy of the selected objects.



The toolbar options allow the user to simply select all objects, select automatically associated objects and display the objects either by name or selection. Most importantly, selecting the automatic association will select all the menus, applications, macros etc. related to the chosen user for example.

Remote Database Explorer

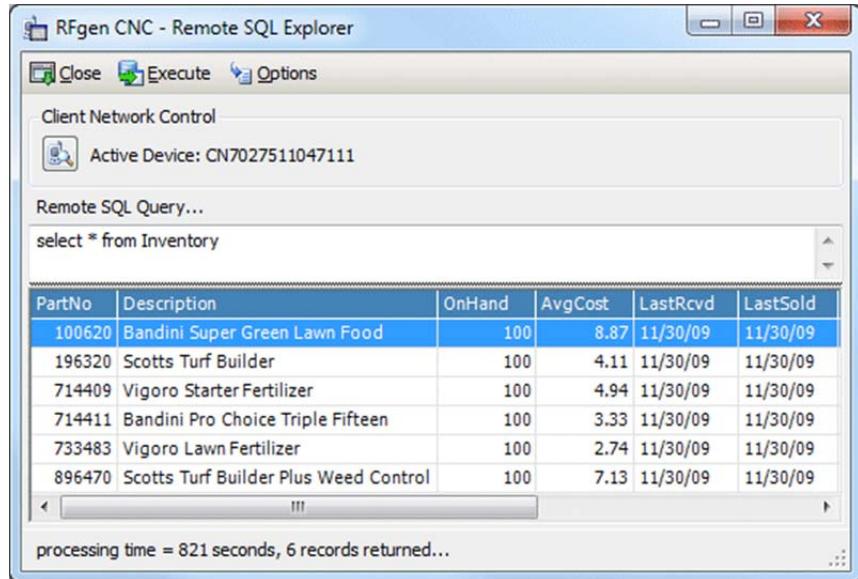
This screen is used to see the schema of the tables stored on the mobile device when the device is setup as a mobile client. To see the data in these tables use the Remote SQL Explorer.

The screenshot shows the 'RFgen CNC - Remote Database Explorer' window. At the top, there are 'Close' and 'Execute' buttons. Below that, a 'Client Network Control' section displays 'Active Device: CN7027511047111'. The main area is a table titled 'Table' with columns 'Table', 'Type', and 'Size'. The table lists the schema of the '\Flash File Store\ UserData.xdb' database:

Table	Type	Size
\Flash File Store\ UserData.xdb		
ITEMMASTER		
IMAGES		
Inventory		
PartNo	VarWChar	
Description	VarWChar	
OnHand	Integer	
AvgCost	Double	
LastRcvd	Date	
LastSold	Date	
LastCost	Double	
Price	Double	
Rack	VarWChar	
Status	VarWChar	
ProdImage	LongVarBinary	
LOCATIONS		
LOCATIONID	VarWChar	
NAME	VarWChar	

Remote SQL Explorer

This screen is used to inquire or update data stored in the mobile device's database. Thin client solutions do not have users, menu, applications, etc. and therefore do not contain databases.



Click on the Remote SQL Explorer menu option and then select the active device. Enter any valid SQL statement, or multiple SQL statements separated by a semi-colon in the Remote SQL Query window and click the Execute menu option. In the case of multiple SQL statements, the statement that contains the blinking I-beam cursor or has been highlighted will be executed.

The Options menu item allows for a restricted number of rows to be returned in case the statements will bring back more data than desired. It will also switch the grid display into a list of previously executed SQL statements for easily repeating previous statements.

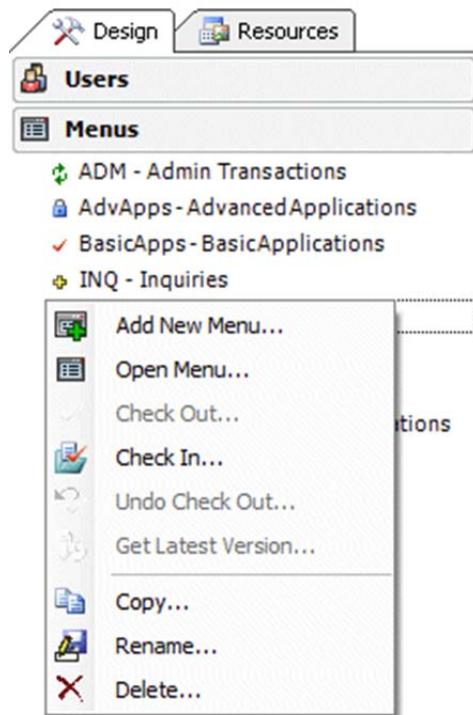
Utilities

The Utilities menu option provides tools to manage RFgen objects. RFgen objects such as users, menus, applications, VBA modules and others can be imported and exported as well as managed through Microsoft SourceSafe, if it is installed. The ability to syntax check all code at the same time and undo otherwise permanent save or delete actions on the RFgen database can also be found here.

RFgen SourceSafe Browser

The browser provides a very similar view of the SourceSafe database that the standard Microsoft SourceSafe GUI would show.

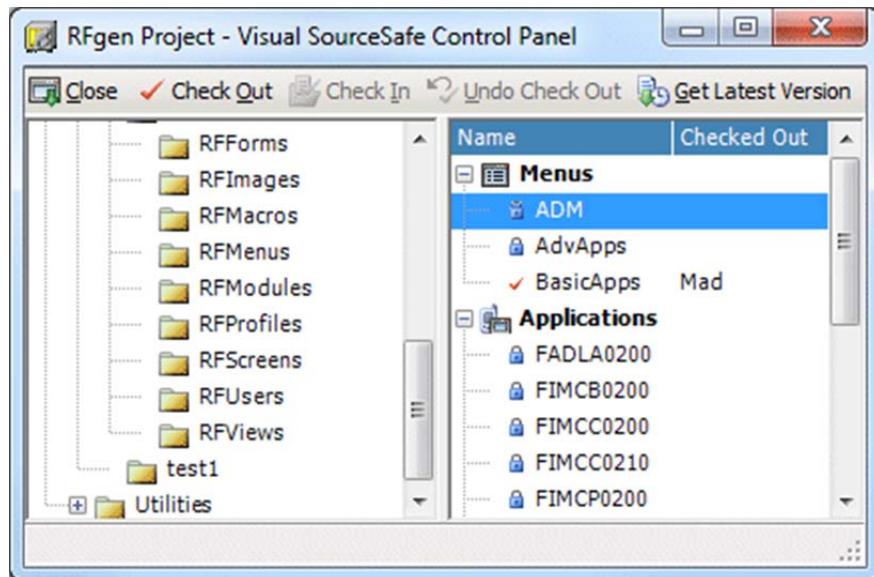
Looking at the list of items within an RFgen project with SourceSafe enabled, there are 3 different icons, a plus, check mark and a lock.



The recycle icon means that RFgen is out of sync with source control. In this case choose “Get Latest Version” to update RFgen. The lock icon represents that the item is in the SourceSafe database and has not been checked out for use. The check mark means that the item has been checked out and can be changed freely. The plus icon means that the item does not exist in the SourceSafe database and should eventually be added for version control purposes. A person icon means that it is checked out by someone other than the current user.

Right-clicking on any one item or an entire section title a menu with SourceSafe options will appear. Choosing one of the SourceSafe menu options while selecting a section title will apply the menu option to all items within that section.

With this example the RFgen SourceSafe Browser will display the following.



The left pane shows the same tree that was configured under the Options / Visual SourceSafe Integration menu option when the database was selected. The right pane shows all items known to the database. In this case, only the menus were checked in and one item is checked out for changes by the name specified in the Checked Out column.

The **Check Out** will make an RFgen item available for changes and reflect that check mark icon in the main tree. The **Check In** saves the latest copy of that RFgen item in the database and makes the item read-only or unchangeable. **Undo Check Out** changes the status of the item to “checked in”, without saving the local copy to the SourceSafe database. It has no impact on the local item, which means all changes that were made still exist. If edits were made that are not desired, choose both the Undo Check Out and Get Latest Version to reset the local copy. **Get Latest Version** will take the last copy from the database and overwrite the local copy losing all changes made and take that item back to the beginning of the edits.

Export/Import Applications from/to RFgen Database

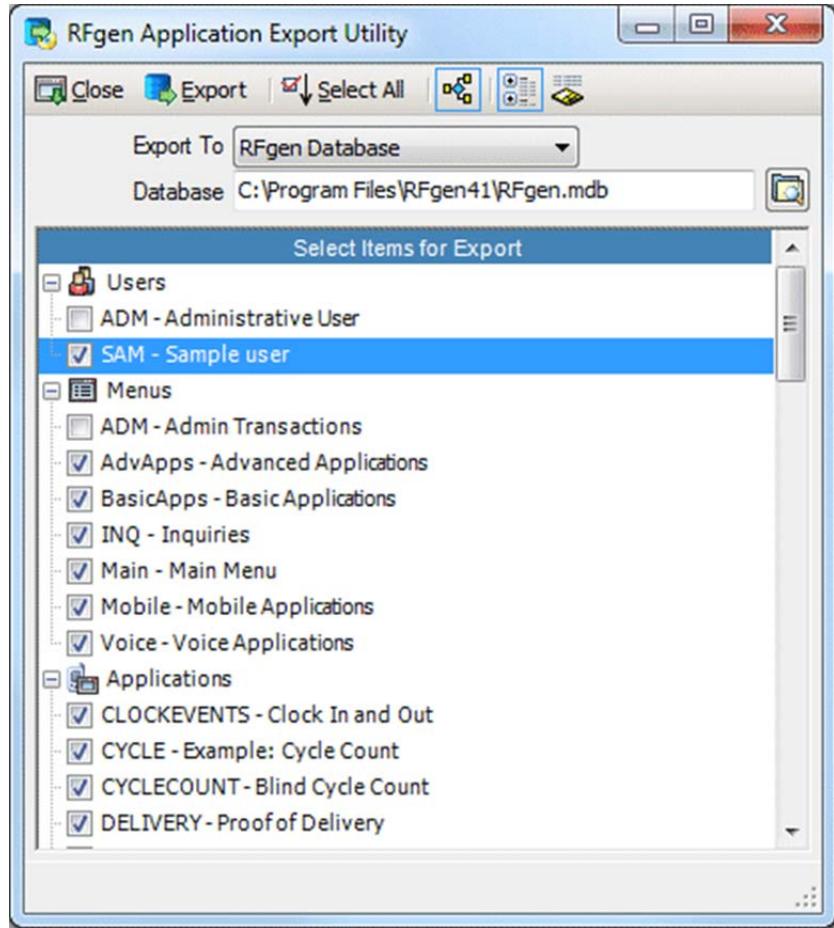
If your RFgen data collection applications have already been developed and you merely wish to use them, you'll need to load the Mobile Development Studio objects (Applications, Menus, Users and VBA code and macros) into your newly installed system.

The 'Utilities' Menu Bar selection allows you to transfer specific (or all) RFgen objects into your RFgen system. RFgen objects are freely transferable from/to other RFgen Databases, or other external files, for the following purposes: (1) production usage, (2) ongoing development, and (3) backup/ retrieval.

Internal to the Mobile Development Studio, these objects are, by default, stored in a Microsoft Access database called 'RFgen.mdb'. Most RFgen installations find it convenient to have the production version of this file resident on a dedicated RFgen-based system, serving the active data collection process, and a development version on a separate system. Since the Mobile Development Studio may be loaded onto a separate development system, this methodology allows off-line programming enhancements to be tested without affecting production applications. The Mobile Development Studio Import/Export utilities support this methodology.

Note: to transfer an entire set of RFgen (same release) applications, and to overwrite the current set, simply copy the Mobile Development Studio RFgen.mdb file from the development system to the production system while the production system is not in use.

The Mobile Development Studio export window (with sample data entry RFgen Applications shown) appears below.



Here we are exporting to an RFgen database, chosen in the 'Export To' drop-down option. A standard Windows file is an alternate choice.

Enter the path to the remote file, or select the icon to browse, and click 'Export' after selecting the desired objects.



This menu option will toggle whether or not dependent items are also automatically selected.



If this menu option is highlighted, all possible items will be available for selection.



If this menu option is highlighted, only the already selected items will be visible.

By choosing only the SAM – RFgen user, the Export utility selected all related items from all other categories, in this case, the applications, menus and the BAS files utilized by the user. You can always remove those items that are not wanted.

The available groups for Import and Export are:

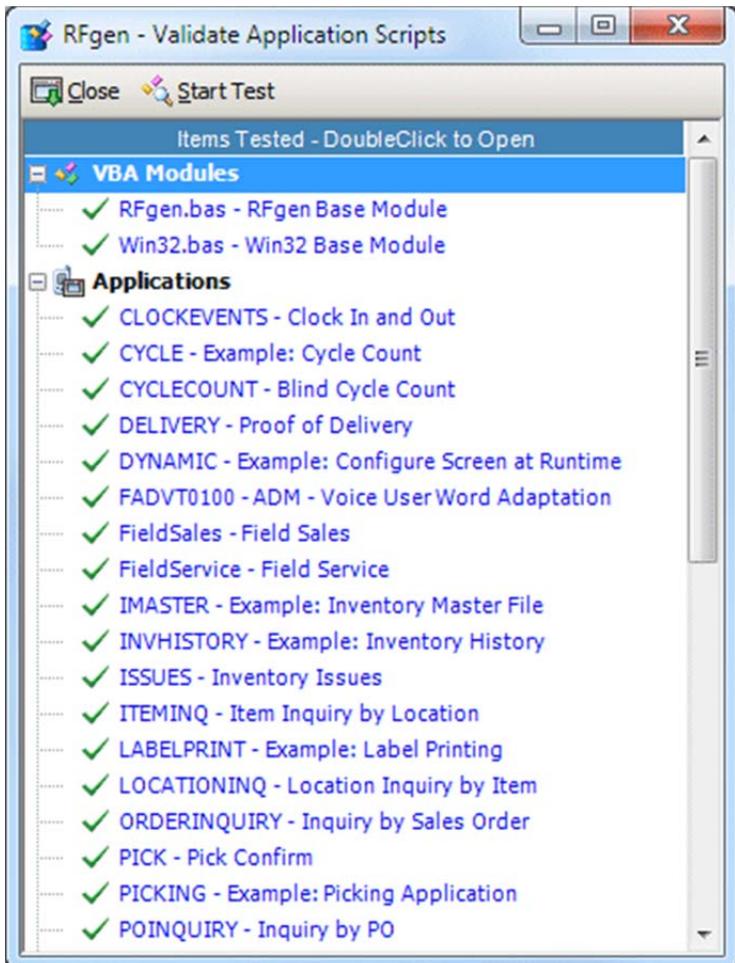
Users:	This includes the user, assigned applications and associated properties.
Menus:	The menu and assigned applications/sub-menus
Applications:	The application layout, VBA script, and properties
VBA Modules:	VBA script modules
Transactions:	VBA script for database or screen mapping transactions
Hosts:	The screen picture and VBA navigation script
Device Profiles:	The profile created to deploy a complete package to a mobile device
Speech Resources:	The grammar to be recognized on a global scale for speech applications
Vocollect Tasks:	The Vocollect tasks used in the application
Schema Objects:	Downloaded tables, stored procedures and business function definitions
System Files:	Data connection configurations, global properties and other system level attributes can be moved between master databases.

Importing objects works similarly, except that you will be overwriting items in your local RFgen application from a remote file.

When importing or exporting RFgen items, the RFgen release number (e.g. 3.0, 4.0, etc.) used to create the items must be the same as the release number for the items being overwritten. You will be stopped if they are not the same.

Validate Application Scripts

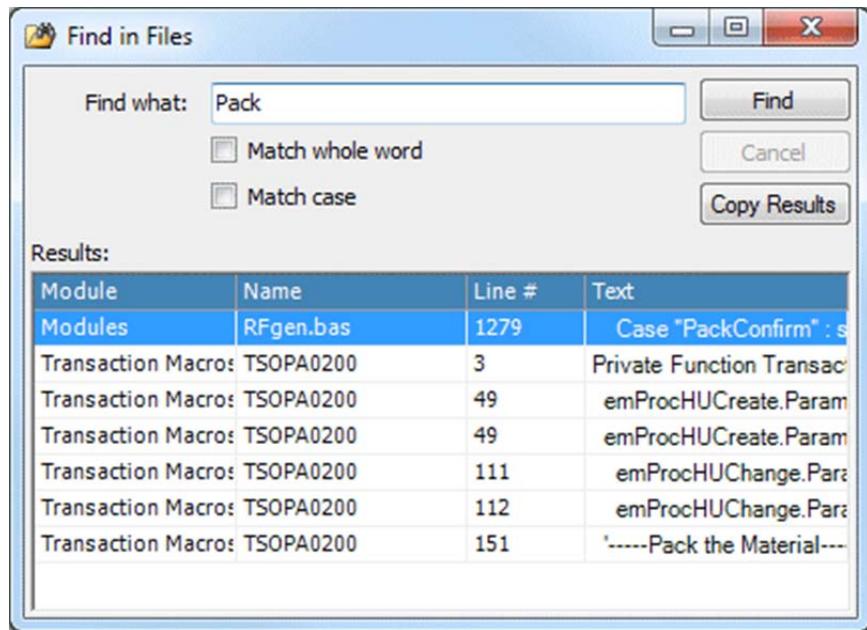
This utility will perform the VBA syntax check for all coded objects. Any application or macro, etc. that has a syntactical error will display the yellow triangle-warning icon. Double-clicking on any line will load and display that code page for convenience.



Find in Application Scripts

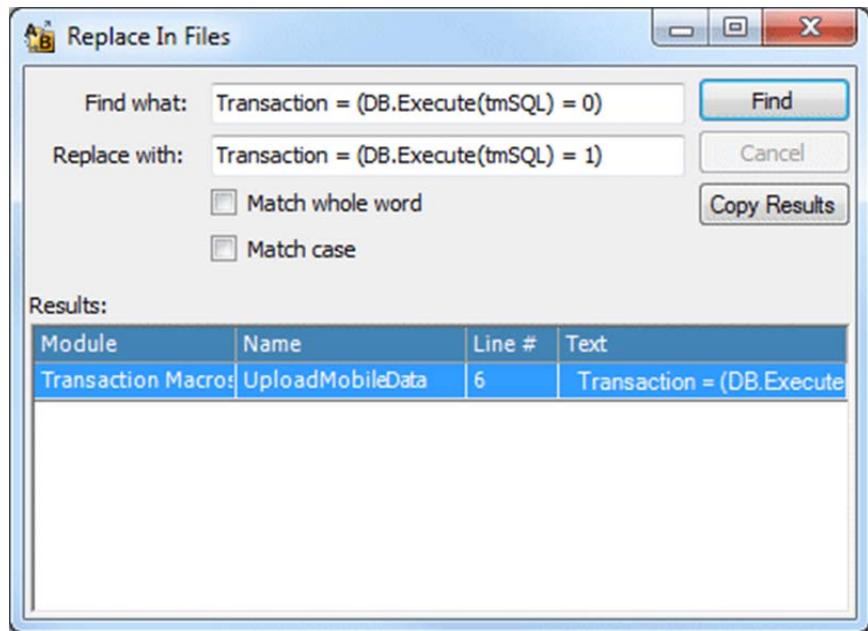
The Find in Application Scripts will generate a list of applications, modules and macros that contain in their scripts the text to be found. Double-clicking on a resulting application, module or macro will open the code window for that item and position the cursor on the correct line.

The **Copy Results** button creates a comma-delimited list of the columns displayed in the grid.



Replace in Application Scripts

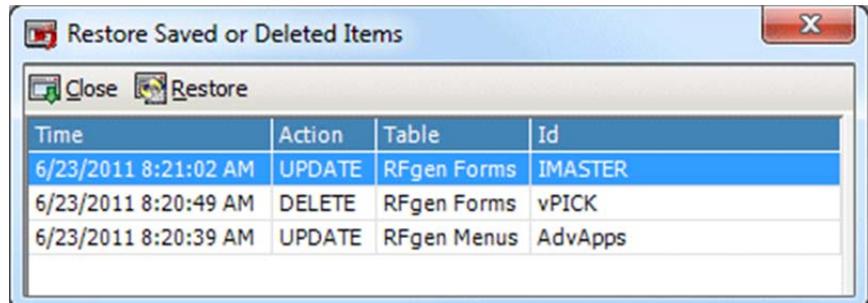
The Replace in Application Scripts finds and replaces specified text in all applications, modules and macros at one time.



The **Copy Results** button creates a comma-delimited list of the columns displayed in the grid.

Undo Save/Delete Action

During the time the Mobile Development Studio is open, all saves and deletes of applications, menus, users, or other objects are listed here and can be undone. When the Mobile Development Studio is closed, this Undo list is lost permanently.

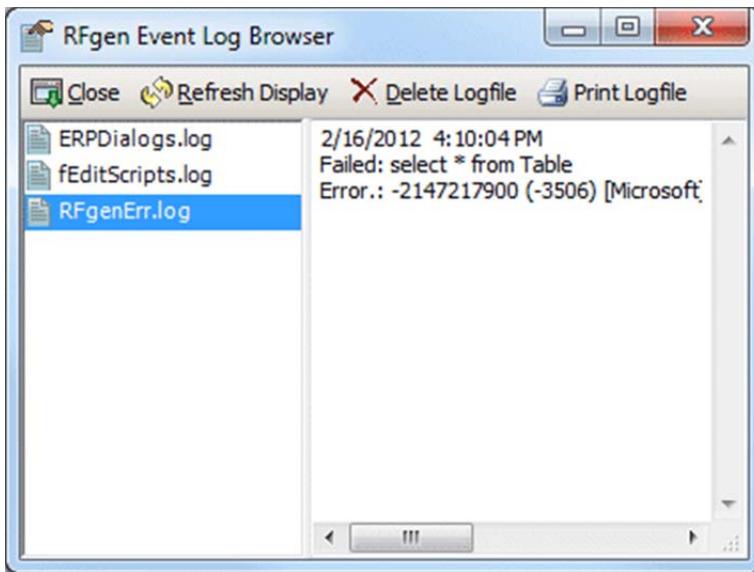


Reports

There are a number of available logs that can be seen from the Reports menu option. Remote function calls, standard database errors, ERP dialogs, IP addressing between the mobile device and server, socket connections and speech recognition tracing are all available here based on the configuration. Any file with the extension 'log' or 'trc' in the RFgen directory will display in the following screen.

Event Logs

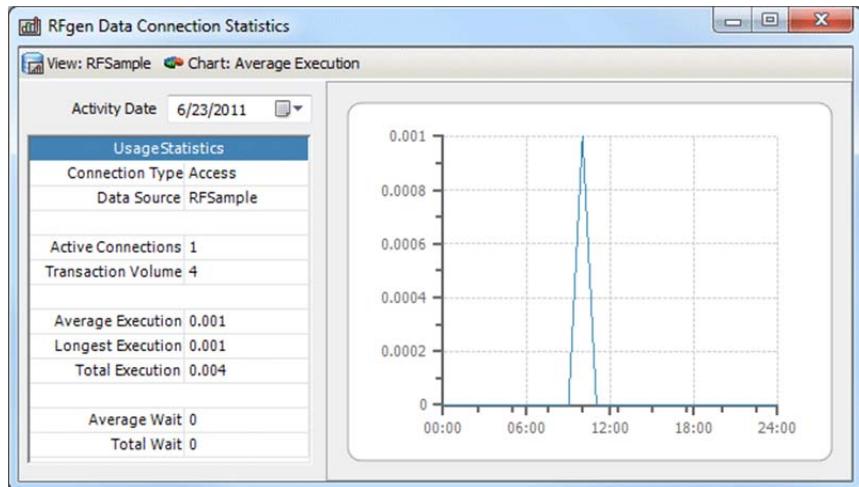
Clicking on the 'Reports' menu selection and selecting 'Event Logs' displays a window showing system error messages, including errors encountered when RFgen attempts to access your configured database(s). Here, an entry in the 'Error Log' is shown.



Note that RFgen uses standard SQL 'SELECT', 'INSERT', and 'UPDATE' statements to process device data in conjunction with your database. If, for example, you use SQL reserved words (see SQL Reserved Words section), or include spaces in your transaction table or column field names, your data will not be processed. The specifics of your error(s) will be written in this General Error Log window. Items in the window may be viewed, printed, refreshed as necessary and cleared as desired. You may also copy and paste the log entries using <CTRL> C and <CTRL> V after highlighting the section, row, or column with the mouse.

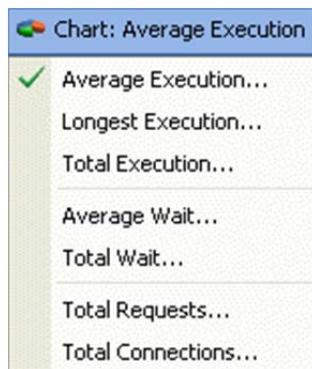
Application Statistics

Clicking on the ‘Reports’ menu selection, the ‘Application Statistics’ will display statistics regarding database and ERP transactions.



The View menu option selects between each of the configured data connections and the Transaction Management database.

The Chart menu option will show the performance of a given statistic over the course of the day. The options are:



Average Execution is the typical time it takes to execute 1 call to the specified data connector. The graph shows this average across the whole day. The lower the number, the better. Typical values should be well under 1 second.

Longest Execution is the longest time RFgen had to wait for 1 call to the specified data connector. The lower the number, the better. Typical values should be well under 1 second.

Total Execution is an accumulated amount of time that RFgen has spent waiting for all executed calls to the specified data connector.

Average Wait refers to how long on average a user must wait for RFgen to provide them a connection to the specified data connector using the Connection Pooling process. Typical values should be less than 1 second. If the user must wait longer, then the connection pool should be increased.

Total Wait is an accumulated amount of time that users have spent waiting for RFgen to assign a data connection handle from the pool.

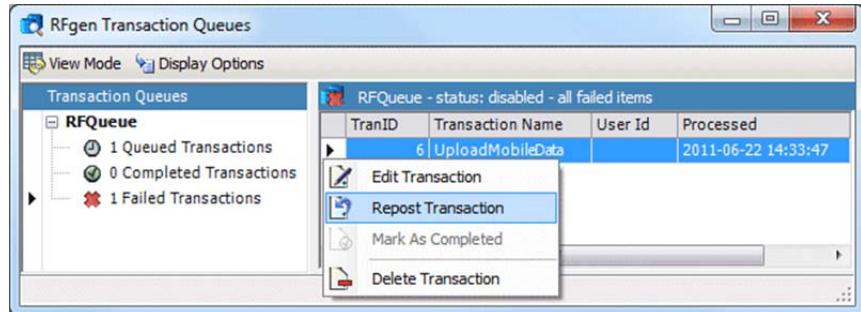
Total requests is the total number of times RFgen access the specified data connector for any reason.

Total Connections is the number of currently open connections to this data connector. Without Connection Pooling, each logged in user will have their own connection. With Connection Pooling enabled, the maximum should be the limit placed on the pool in the configuration and the minimum should be 1.

To enable the Statistics, choose the Options / Transaction Management menu option, select the Extended Options button and change the Data Connection Usage Statistics value from Disabled to some increment for refreshing the data. Since the data is stored in the Transaction Management database, Transaction Management itself must be enabled even if the solution does not require queuing macros for functions.

View Transaction Queues

Clicking on the ‘Testing’ menu selection, then ‘View Transaction Queues’ displays a window that contains transactions.

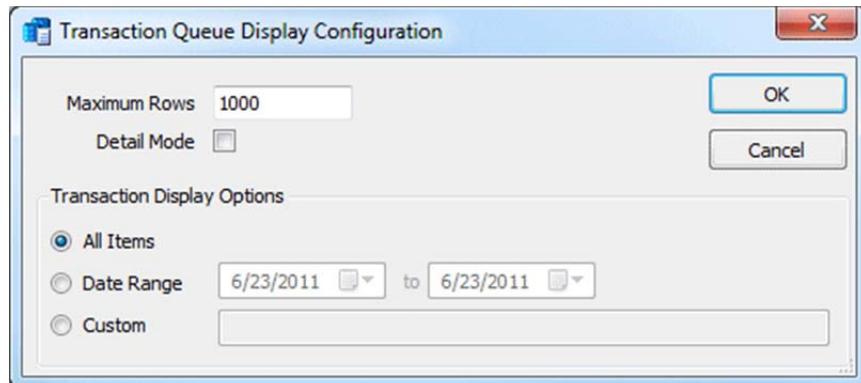


3 types of logs are available: ‘Queued’ transactions are data collection entries waiting to be posted to your host application (typically because your host has been offline or not available); ‘Completed’ transactions and ‘Failed’ transactions may also be displayed. Transactions may be edited, reposted, marked as completed or deleted by means of the right-click menu options as shown above.

Note: Queued transactions will not automatically post when using the Mobile Development Studio. This allows for the testing of posting transactions. When in production using the Server, all queued transactions are posted automatically within 60 seconds (or less depending on configuration) of the host becoming available.

The Refresh menu option simply updates the display if you believe it is necessary.

The Display Options are used to narrow down the records being displayed in this window.



Over time the list of completed transactions can become very large.

Maximum Rows will limit the display to the first configured number of entries. To see the most recent entries, use the data range option and set the Maximum Rows to a high value.

Detail Mode will show the data passed in to the macro.

Transaction Display Options – **All Items** shows an unrestricted list of entries and **Date Range** will limit the entries to a date-based on their created date.

The **Custom** option is an ability to specify your own Where clause for the lookup. The actual names of the fields in the Queue database must be known as well as the type of field. An example would be:
where SeqNo = 1

(See *TM.GetItemsEx* for examples of table fields and types.)

Help

Clicking on ‘Help’ will display these options.

Mobile Development Manual – displays the RFgen manual for the Studio product

VBA Scripting COM Syntax – displays the COM specifics of the VBA language

VBA Scripting .Net Syntax – displays the .Net specifics of the VBA language

Obtaining Technical Support – provides links and e-mail addresses for sales and technical support

Authorize RFgen Voice Development – provides the authorization screen to authorize RFgen voice development

About Mobile Development Studio – About information including what is installed and the version

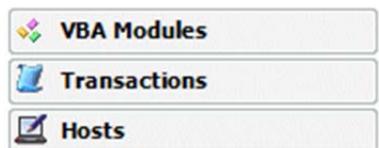
Mobile Development Studio Tools

The Mobile Development Studio development tabs are displayed beneath the menu bar on the left. The Design tab has the most commonly used aspects of development, all items necessary to create a solution. The Resources tab contains deployment and global configuration data.

Design Tab Overview



RFLogin - RFgen Login
ADM
Administrative
Adv_Human Resources
Adv_Inventory Management
Adv_Purchase Order
Adv_Sales Order
Basic
Mobile
Voice
Web



Here:

The **Users** tree is used to provide user names, passwords, and a primary menu for each user.

The **Menus** tree is used to organize Applications for selection purposes.

The **Applications** tree is used to create the visual aspects (i.e., device displays) and basic validation for the data collection transaction.

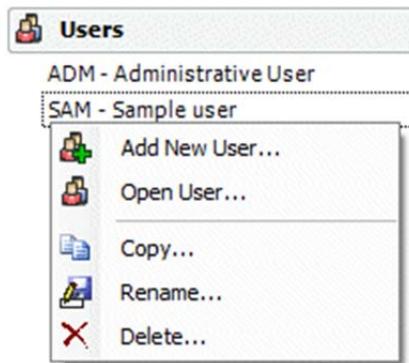
The **VBA Modules** tree is used to manage global scripting that can be used by any application, timed event, transaction macro, etc.

The **Transactions** tree (used with Screen Mapping and non-screen mapping applications) is used to manage host (data entry) transaction macros. See the RFgen Screen Mapping documentation or the Transactions tree section below for more information.

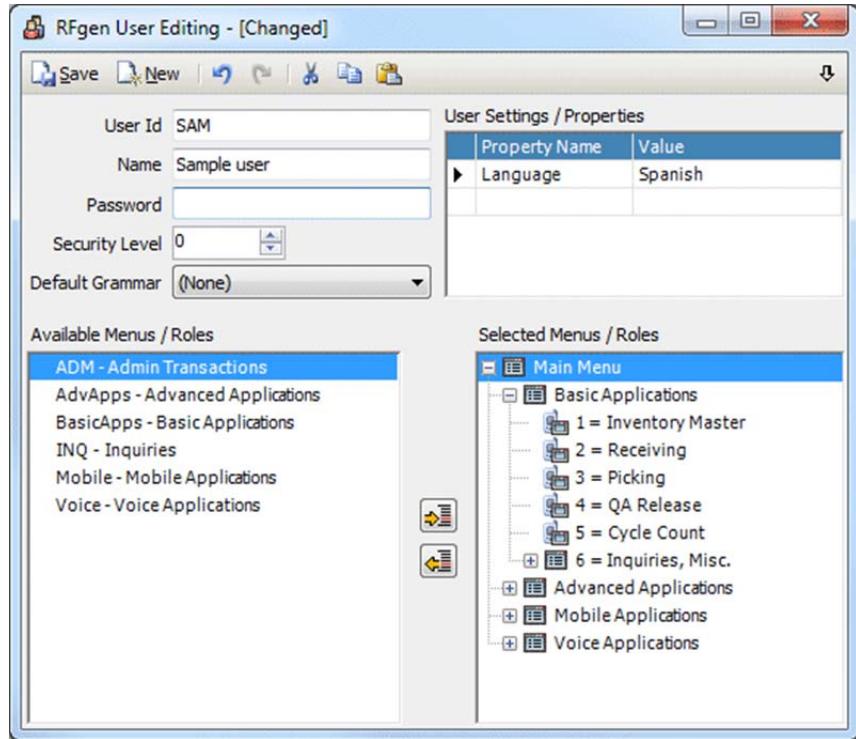
The **Hosts** tree (used with Screen Mapping only) is used to manage host screen (navigation) macros. See the RFgen Screen Mapping documentation for more information.

Users List

Right-click on an existing user (or in the blank space) to add a new user. Double-click or right-click on a specific user to make additional changes.



Double-clicking on an existing user opens the User Editing window.



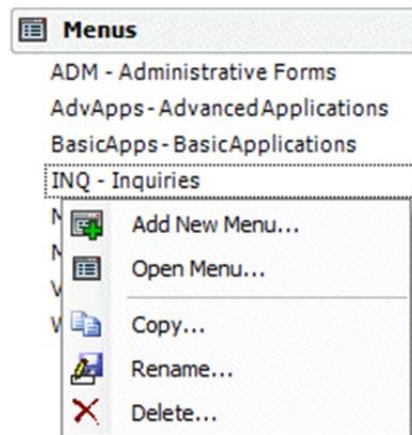
Here a user code of 'SAM' is illustrated. A password is not required for a user account. SAM's startup menu is 'Main Menu'. Security Level is a numeric value between 0 – 100 that will be compared to the menu's required security level before allowing that user access to the following menus or forms. If RFgen's voice capability is being used and if grammar profiles have been created, each user has the option of choosing which grammar will be their default. (To learn more about grammar, see the Speech Resources tree option under the Resources tab.) Other menus are embedded in the Main Menu for SAM's use (created using the Menus tree).

To enter the menu name, select the existing menu from the drop-down list or type in the name manually. The Optional User Settings / Properties section allows the administrator to include any property and value for the selected user for the purposes of retrieving that data at run time. This has no effect on the user logging in. This data can be retrieved using the VBA extension command 'App.UserProperty'.

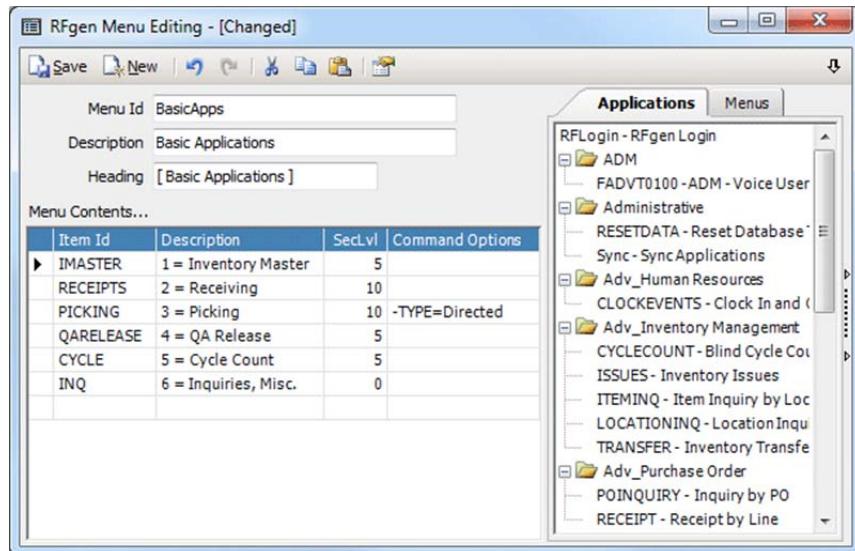
When done, click the Save menu item.

Menu List

Right-click on an existing menu (or in the blank space) to add a new menu. Double-click or right-click on a specific menu to make additional changes.



Double-clicking on an existing menu opens the Menu Editing window.



Menus are groups of existing applications and other menus that will be made available to RFgen Users who are assigned available menus.

The **Menu Id** is the name of the menu that is used when assigning them to users. The **Description** is required and only shows in the Menu list. The **Heading** value will be placed at the top of the menu and is optional.

To select an item to appear on your menu, right-click on the row you want to add or insert into. The options are to insert an application, existing menu, label and deleting the entry on the current row. Alternatively, just press the delete keyboard key to delete the selected row. The show/hide toolbar button will display the list of available applications and menus. When done with your selections, click on the Save menu item.

In the case where one form is used for multiple purposes, variables can be defined in the Menu grid itself. Then those variables can be referenced when the application is loaded to determine how to use the application.

In a case where you want the user to select from two variations of the same application, add the application to the menu twice, give them different descriptions and then use two different Command Options. Using the format –VARNAME=VALUE will create variables with values that can be referenced at runtime.

In the Form_Load event, set a global variable = App.GetValue (TYPE) and based on the result, show or hide prompts or change the logic of the transaction.

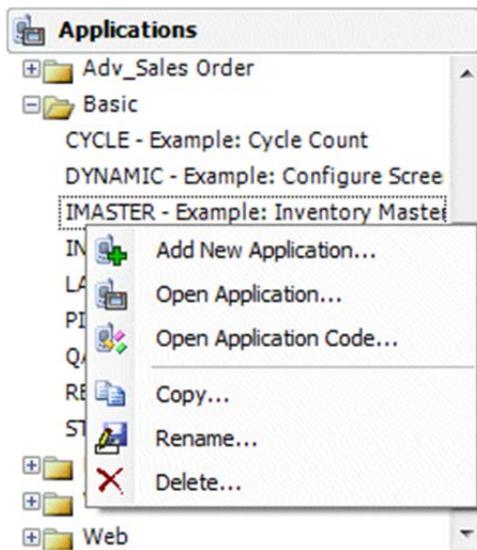
Making the application name a dash, supports comment lines within a menu. Right-clicking on the left-most column for a row and inserting a label also achieves this.

Menu Contents...				
	Item Id	Description	SecLvl	Command Options
▶	CLOCKEVENTS	1 = Clock In and Out	0	
-			0	
-		[Inventory Management]	0	
	CYCLOCOUNT	2 = Blind Cycle Count	0	
	TRANSFER	3 = Inventory Transfers	0	

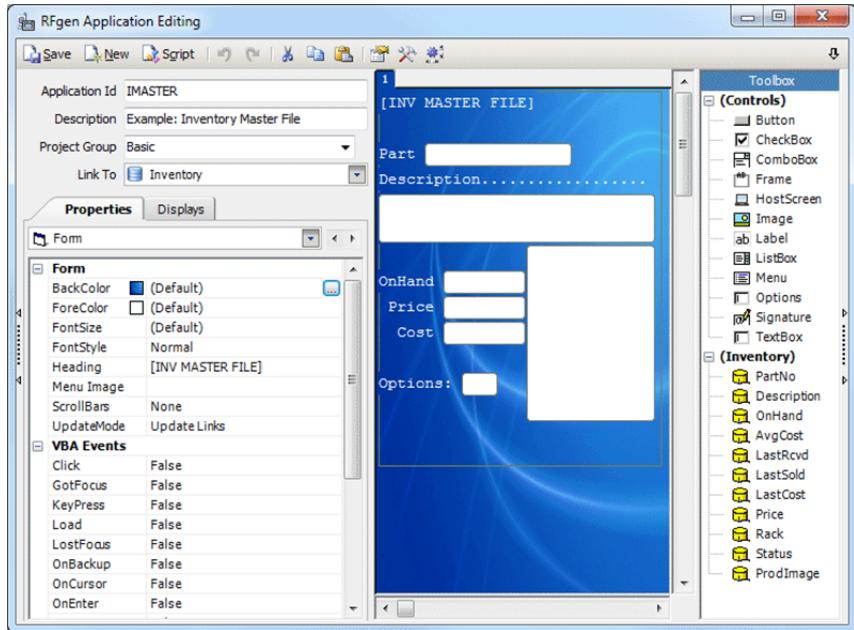
In this case, what is displayed on the menu is a blank line and then a comment line indicating that the following list of items are their own group. The user has no ability to select label entries on the menu. The highlight bar will skip over these entries.

Applications List

Right-click on an existing application (or in the blank space) to add a new application. Double-click or right-click on a specific application to make additional changes.

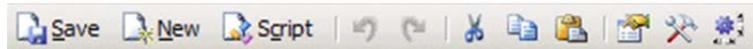


Double-clicking on an existing application opens the Application Editing window.



The Application Editing Menu Bar

The following selections appear across the top of the Application window.



The **Save** option saves the current script to the RFgen database. The **New** option clears the screen so a new application may be created. The **Script** option displays the VBA script for the application. Also provided are **Undo**, **Redo**, **Cut**, **Copy** and **Paste**. The **Properties** button hides or unhides the left half of the screen containing the properties grids and header textboxes. The **Toolbox** button hides or unhides the toolbox containing the prompts that can be added to the application. The **Display Options** button provides scroll bars in the design environment for larger application screens. The last **Dock** icon, (the down arrow) displays on every un-docked window, allowing the user to re-dock back to the Mobile Development Studio main window. Double-clicking the top-most tab will undock the window.

The **Application ID** and **Description** fields are required. RFgen and the menus use the Application ID to record and display the application and the Description field is the default menu description for the user.

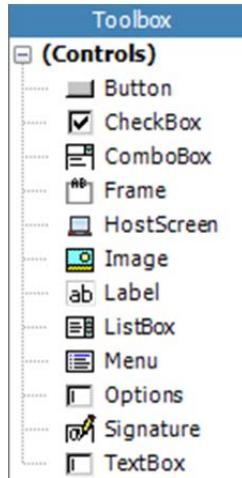
The **Project Group** property allows the forms to be grouped together into categories such as “Inquiry Transactions” or any other named grouping. This has no impact on production. It is just an option for making the management of the development database easier.

The **Link To** drop-down property establishes if there will be a connection between the application and any pre-established back-end connections or macros. The options are:

- No Links: the application does not link to any external connections or macros
- Database Table: the application will use linked textboxes and potentially automatically update a database
- Macros: the application will use a screen mapping Application Screen macro or a Data Transaction macro
- System Functions: Login Request: if the application has a prompt called ‘User’ and a prompt called ‘Password’ then the user will be validated against RFgen’s list of users and the assigned menu will be called.

Menu Request: if the application has a prompt called ‘Menu’ then any selection will internally be called with an App.CallForm. You must populate the menu prompt either through the List property or scripting.
- Vocollect Tasks: This provides a list of Vocollect resources that the application may use. The application is acting as the data collection, validation and back-end update script only.

Additional objects besides linked fields are available in the Controls Toolbox. They are: Button (graphical only), CheckBox, ComboBox, Frame, HostScreen, Image (graphical only), Label, ListBox, Menu, Options, Signature Box (graphical only) and TextBox. The following window is the toolbox.



The Button Object (Graphical Item)

The Button object is used to allow easy access to a function on the application such as Save or Exit. This object can only be seen in the graphical version of RFgen.

To use a Button, place it on the application and give it the appropriate caption. In text mode, the GotFocus, OnEnter and LostFocus events are executed when the image is next in the tab order. In the graphical mode, the Click event will also execute.

Note: this prompt will never hold the focus. When focus comes to this prompt it immediately processes the events and focus then moves to the next prompt.

The CheckBox Object

The CheckBox object is an application prompt that allows the user to select a True or False option based on the object's label.

To use a CheckBox, simply place it on the screen and change the Checked property to True or False in the prompt's properties tab to set the default value.

The ComboBox Object

The ComboBox object is an application prompt that allows multiple items to be displayed in an area of the application, one of which may be selected. Items may be sorted and/or selected as required by the needs of the application.

To use a ComboBox, VBA script is used to populate and manage items displayed in the box or the ComboBox prompt can contain the values itself by entering them in the List property.

The Frame Object

The Frame object is used to put a box around other prompts to give a grouping effect. The Frame prompt must come before the prompts that will be inside the frame. Otherwise, the frame will cover the prompts on the inside.

The frame can be stretched to be either a single horizontal line or a vertical line and its caption is optional.

Note: this prompt will never hold the focus. When focus comes to this prompt it immediately processes the GotFocus, OnEnter and LostFocus events and focus then moves to the next prompt.

The HostScreen Object (Graphical Item)

The HostScreen object provides a pass-through telnet emulation window on an RFgen display. You can map an RFgen form to an alternate console application or Telnet service. Placing this control on a form would allow the user to use RFgen forms for some data collection and this form to execute SAP Console transactions, Telnet to a legacy host system or simply start any executable with console output. It is configured as a Screen Mapping connection with an emulation type of 'Console'. All the keystrokes pass into the prompt's KeyPress event allowing the user to monitor the keyboard and trigger actions like exiting the application based on custom keystrokes. Only 1 of these controls is allowed per application.

The Image Object (Graphical Item)

The Image object is used to display a picture on the application. This object supports a large variety of image formats in a Thin Client environment. They are BMP, DIB, GIF, JPG, WMF, EMF, ICO, and CUR. This object can only be seen in the graphical version of RFgen. When using the Windows CE / Mobile environment, this control only supports the BMP format.

To use the Image object browse for the desired image using the Image property and select it from the Image Resources. The GotFocus, OnEnter and LostFocus events are executed when the image is next in the tab order or when the user clicks on the image. In this case, the Click event will also execute. The GotFocus, OnEnter and LostFocus events will execute in character mode.

At runtime check the Defaults property to get the name of the image resource currently loaded into the prompt. If it is blank the image may have come from the file system using the RFPrompt().ImagePath property.

Note: this prompt will never hold the focus. When focus comes to this prompt it immediately processes the events and focus then moves to the next prompt. If it is simply clicked on, the focus will stay where it was before the click.

The Label Object

This object is used to add additional text to the screen not necessarily associated with one prompt. All of the default objects come with their own label. For example, if the customer desired the text "F4=Exit" at the bottom of the screen, a label would be used. If a prompt's label required 2 lines, you could use the prompt's initial label for the first line and add a Label object to fill in the second line.

Note: this prompt will never hold the focus. When focus comes to this prompt, it immediately processes the GotFocus, OnEnter and LostFocus events and focus then moves to the next prompt.

The ListBox Object

The Listbox object is an application prompt that allows items to be displayed in an area of the application. Items may be sorted and/or selected as required by the needs of the application.

To use a ListBox, VBA script is used to populate and manage items displayed in the box, or the ListBox prompt can contain the values itself by entering them in the 'List' property. A ListBox is different from an RFgen list. Clearing the application prompts and displaying items in a list in full screen mode is an RFgen list.

The Menu Object

The Menu object is used in the same way as a ListBox, but the look is that of the menu layout. There is no difference in the way they both can be used.

The Options Object

The Options object (when used) is typically the last prompt in a data entry or display application and is used to write data to the linked database table or view. This prompt has no effect if the prompts are not linked fields. The object is a prompt that provides a pause in the data entry process so that the device user may select which of 4 options should be taken next.



4 choices are available with this object:

'V' = verify (cycle through) the current data

'F' = file the data (write to the appropriate table)

'N' = next, i.e., don't write this data

'Q' = quit, i.e., don't write, then quit the process.

When the Options Object appears, all processing is suspended until one of the choices above is entered. 'F' is the default, so that if just the device <Enter> key is pressed, the data collected in the application is filed (i.e., written) to the associated database table(s).

This control can be replaced with a standard textbox control that accepts any values of your choosing and then processes the user response in the OnEnter event, possibly even performing SQL statements to write data to the database.

The Signature Object (Graphical Item)

The Signature object is used to capture any hand-written text that will then be saved as a 2-tone bitmap image. This image can be placed in the Text property of the signature prompt to be re-displayed. There is no built in pattern matching to compare recorded signatures. Since the bitmap is saved in black and white, changing the background color to red, for example, will make the box appear solid black since red is a dark color and converted to black.

The Text property of this prompt will contain a text representation of the signature which can easily be stored in a character type field in a database. The Bitmap property contains a byte stream representing the signature which can be written to a BMP file if desired or stored in a binary type field in a database.

The TextBox Object

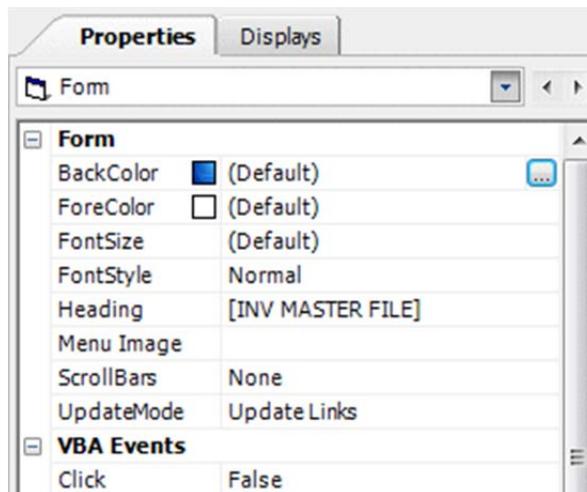
The TextBox object is a text box that can be used for many purposes. Dragging or double-clicking on the TextBox control places a textbox on the screen. Once added, you may set the properties for the object including the data to appear in it. For example, the data display area may be used to print (translate) data from your database using the '@T'

default Property (see the Data Entry Default Properties section). Translate data is typically data from your database; i.e., 'CA' is entered at a prompt, 'California' is retrieved from your database and displayed in the unlinked textbox field. Defaults other than 'translate' may also be used.

Properties Tab

The Properties tab contains the properties specific to the form or prompt depending on what is selected. Colors, font sizes and styles, links to database tables or macros, the application's heading and all the events that can execute at the application level are all displayed on this tab.

Form Group



The application's **BackColor** and **ForeColor** properties change the colors of the background of the application and the colors of the heading of the application. The **FontSize** and **FontStyle** properties change the size and style of the application's heading.

The **Heading** property will place text at the top of the form and persist for each additional page should there be more than one.

The **Menu Image** property specifies an image from the Image Resources stored with the solution that will display on the menu if the menu is set to Graphical List or Desktop Icons mode.

The **ScrollBars** property determines if and how scroll bars are used to display this form at runtime. It is equivalent to the commands:

```
App.SetOption(ShowHorizontalScrollBar, True)  
App.SetOption(ShowVerticalScrollBar, True)
```

Update Mode has 3 options.

No Update – which will not try and update linked tables with collected data.

Update Links – which will update the linked table and for macros the application will try and call the linked macro if it has not been called already. If you set one or more macro fields as “key fields”, then once all those have received values, RFgen calls the macro. If no key fields are defined, the macro executes at the end of the application’s data entry. This field has no meaning for Vocollect links.

Exit Form will take the user back to the calling menu without updating linked tables.

Form VBA Events Group

Visual Basic for Applications (VBA) compatible programming scripts may be used to achieve results not otherwise available within the Mobile Development Studio. VBA scripts allow technical personnel to use application, field and other events to provide additional functionality (e.g., field defaulting, error checking, network printing, etc.) to standard RFgen applications. Special pre-built VBA ‘language extensions’ dealing with numerous aspects of the data collection process have been added to simplify programming efforts.

To enter the VBA programming environment from the Application Properties Window, click either the Script menu option or on a ‘VBA Events’ property. The specifics about the VBA events will be discussed later.

Fields Control Group

(Control)	
FieldId	User
Object	TextBox
Page Number	1
Prompt Number	3

The **FieldId** is the internal name RFgen uses to identify the prompt in the script. Standard programming practices suggest using the Hungarian notation where textboxes are named ‘txtPart’ and list boxes are named ‘lstParts’ as examples. This way, when referring to them in

the script, there is an inherent understanding of what types of data will be used for the prompt.

The **Object** property is a read-only property that displays which type of prompt is currently selected. In some cases, prompts may be invisible or on other pages and, short of good naming practices for the Field ID, this is the fastest way to identify the type of prompt.

The **Page Number** property shows which display page the prompt is on and it can be changed to move prompts to other pages. If a prompt is moved from page 1 to page 2, all following prompts in the prompt sequence will also be changed to page 2. If you change the page number from 2 back to 1, any prompts that come before the changed prompt will also have their property changed to page 1. The best approach is to make sure all the prompts are in the proper order and then break them up into multiple pages.

The **Prompt Number** property shows the position of the selected prompt in the sequence of all prompts in the application. This property can be changed to re-order the selected prompt to a different position. If the new position is between 2 other prompts that are both on a different page, the Page Number property will also change otherwise it will not.

Fields Label Group

Label	
Caption	TextBox
BackColor	<input checked="" type="color"/> (Default)
ForeColor	<input type="color"/> (Default)
FontSize	(Default)
FontStyle	Normal
Left	0
Top	12

The **Caption** property contains the text portion of the label of the prompt.

The **BackColor** property (graphical mode only) allows the user to select from a color pallet or enter a 6 character hex value (for 16 million colors) to set the back color of the label caption of the prompt.

The **ForeColor** property (graphical mode only) allows the user to select from a color pallet or enter a 6 character hex value (for 16 million colors) to set the fore color of the label caption of the prompt.

The **Font Size** property (graphical mode only) sets the prompt's caption to a particular font size. The default is 10.

The **Font Style** property (graphical mode only) sets the prompt's caption to a particular style. The default is Normal. Other options are combinations of Bold, Italic and Underline.

The **Left** property, for the caption portion of the prompt, sets the left-most edge of the caption to the specified column.

The **Top** property, for the caption portion of the prompt, sets the top edge of the caption to the specified row.

Prompt Field Group

Not all properties appear for all control types. Image controls will have image specific properties and check box controls will also have unique properties.

TextBox	
Border	True
BackColor	<input type="checkbox"/> (Default)
ForeColor	
■ (Default)	
Defaults	
Display Format	
Display Only	False
Edits	
Entry Required	False
Error Message	
FontSize	(Default)
FontStyle	Normal
Key Field	False
Password	False
Left	8
Top	12
Height	1
Width	10
Validation Table	
Validation Field	
Visible	True
Voice Input	False

The **Autosize** property set to True for a Menu or Host Screen control, will expand the object to the lowest and right-most portion of the screen. The upper left corner is static. An Image control has the options of Standard (where no resizing of the image or control takes place), Size To Control and Size To Image.

The **Border** property (graphical mode only) set to False, removes the outline of the prompt leaving just the background color.

The **Caption** property (graphical mode only for a Button prompt) contains the text portion of the label of the prompt and only exists under the Field group for a command button.

The **Checked** property sets the status of a checkbox object.

The **BackColor** property (graphical mode only) allows the user to select from a color pallet or enter a 6 character hex value (for 16 million colors) to set the back color of the data field.

The **ForeColor** property (graphical mode only) allows the user to select from a color pallet or enter a 6 character hex value (for 16 million colors) to set the fore color of the label caption of the prompt.

The **Image** property (graphical mode only) is for image controls only and retrieves a graphic file from the Image Resources and displays it in an image prompt. When using thin client mode, the supported image types are BMP, DIB, GIF, JPG, WMF, EMF, ICO, and CUR. When running in mobile mode on a CE device, only BMP is supported.

The **Defaults** property sets any number of built-in values or custom values as the initial value of the prompt. See the *Default Property Details at the end of this section* for a complete list of options.

The **Display Format** property is an extension of the VBA Format command and pre-formats the entered data to the mask entered here. See the VBA Format command for examples. The double quotes are not necessary as they are in the VBA Format command.

The **Display Only** property, set to True, makes a prompt into a display only field, while setting it to False, allows users to access and change data at the prompt.

The **Edits** property sets any number of built-in values as the requirement for the entered data. See the *Edit Property Details at the end of this section* for a complete list of options.

The **Entry Required** property, set to True, forces users to enter data into the prompt, while setting it to False, allows users to skip the field. If the prompt never gets the focus, RFgen will not have a chance to use this property.

The **Error Message** property is text displayed as an App.MessageBox when the data entered fails to meet the criteria in the Edits property.

The **Font Size** property (graphical mode only) sets the prompt's data field display to a particular font size. The default is 10.

The **Font Style** property (graphical mode only) sets the prompt's data field display to a particular style. The default is Normal. Other options are combinations of Bold, Italic and Underline.

The **Integral Height** property (graphical mode only) is for list boxes only and will change the size of the object based on the font being used. When set to True, a different calculation is used to draw the size of the object. This property is only available at design time and therefore has no RFPrompt method.

The **Key Field** property is for linked textboxes only and designates which prompts will be used as key fields when RFgen attempts to perform an internal SQL Update statement when the linked application is complete. This property is automatically filled in when the user downloads a table or view structure and links the application to that structure.

The **List** property is for list boxes, combo boxes and menus only and contains a collection of values to be assigned to the prompt when the application loads.

The **List Height** property is for combo boxes only and sets the number of rows the control will use when displaying a list of possible values.

The **Left** property, for the data field portion of the prompt, sets the left-most edge of the data field to the specified column.

The **Password** property, for the data field portion of the prompt, sets the display of the text equal to asterisks (*) instead of clear text.

The **Top** property, for the data field portion of the prompt, sets the top edge of the data field to the specified row.

The **Height** property is the number of rows the data portion of the prompt takes up.

The **Width** property, for the data field portion of the prompt, sets the width of the data field or the maximum number of characters to display on one line.

The **Sorted** property is for list boxes, combo boxes and menus only and keeps the contents of the list sorted.

The **Validation Table** property presents a list of downloaded tables that can be used to verify that the data entered already exists in this table and the Validation Field. The 2 properties must be used together.

The **Validation Field** property presents a list of table fields specified by the Validation Table property. This is the field RFgen will reference to determine if the data entered in the prompt already exists. If it does not, the Error Message property will be used to warn the user.

The **Visible** property, set to True, makes a prompt visible, while setting it to False, makes it invisible. Even though the prompt may be invisible, the GotFocus, OnEnter and Lost Focus events will still be executed for this prompt if the focus automatically shifts from a prompt before this prompt to one after this prompt.

The **Voice Input** property, set to True, makes a prompt begin to listen for speech input. This assumes that the Language Packs have been installed and configured. After speech has been received for this prompt, RFgen will stop listening.

Fields VBA Events Group

VBA Events

Click	False
GotFocus	True
KeyPress	False
LostFocus	False
OnBackup	False
OnCursor	False
OnEnter	False
OnEscape	False
OnFkey	False
OnScan	False
OnSearch	False
OnSpeech	False
OnVocollect	False

To enter the VBA programming environment from the Field Properties Window, click either the Script menu option or on a VBA Events property. The specifics about these events will be discussed further down.

Default Property Details

A default property allows a data value to be generated automatically. A default property is available for each application entry prompt and defaults are entered in the Properties window. Placing a ';'O' (semicolon and the letter O, not zero) after the default parameter is optional, and allows the default to be overwritten. A blank value means that there is no default. Note: if using the VBA scripts, the Got_Focus event for a prompt contains a variable called 'RSP' which may be set to the value of the data default and a variable called 'AllowChange' (for overriding the value) which represents the value of the ';'O' expression; i.e., AllowChange = True, is the same as placing a ';'O' after the specified default property.

Text Default

This will default the string value as entered. A string value can be any number, combination of letters, or special characters. Format is:

String value(;O)

1000;O means default equals '1000', allow 'O'verwrite. Values contained in () are optional.

System Date Default

This will default the current Windows system date in the format specified by 'exp1'. If no format is specified then the date will display using the format specified by the data item used for the prompt. Format is:

@DATE(;O)

@DATE;O means default equals current date displayed using MM/DD/YY format

System Time Default

This will default the current Windows system time using the format specified by the data item used for the prompt. Format is:

@TIME(;O)

@TIME(;O) means default equals current time

Translate Default - Displaying Data From Other Tables

This will extract data from another table; i.e., 'translate' the data. Format is:

@T,table,exp1,exp2,exp3,(exp4)(;O)

e.g.: @T,STATES,ID,3,NAM

Where:

Table is the name of a database 'translate table'.

exp1 is the column/field name for the translate table where indexed data is found.

exp2 is the current prompt number or column/field name where the key for the indexed table data (i.e., data to match on) can be found; value must be a number lower than the current category number.

exp3 is the column/field name for the translate table which contains the data to be retrieved.

exp4 an optional expression used only if exp2 is an 'unlinked' textbox field (i.e., not a database field for which RFgen can determine a

data type); leave blank if exp2 data is numeric; let exp4 be a single quote " " if exp2 data is a string.

@T,STATES,ID,3,NAM retrieves a state name from a table called STATES. Here ' ID' is an indexed column/field name for the STATES table, and 'NAM' is the column/field name that contains the state name. Data from the current application, in prompt '3', provides the value of ID to use for the indexed search of the STATES table

Concatenation Default

This will concatenate any items specified, and default the resulting string. Valid items are entry categories or items enclosed by quotes. Format is:

@C,exp1,exp2,...,expN(;O)

@C,4,5,6,'***' means default equals prompt '4' value, joined with the prompt '5' value, joined with the prompt '6' value, joined with the text string '***'. Since no ';' is present, the data will be entered and the application will continue at the next prompt

Character String Extraction Default

This will allow the default to be a portion of a previously entered input. Format is:

@SE,exp1,exp2,exp3(;O)

Where:

exp1 = entry category for the extraction.

exp2 = starting position of character string.

exp3 = length of string.

@SE,3,4,5;O means default equals data entered at prompt '3', starting at the '4'th character, and continuing for '5' characters

Calculation Defaults

This will allow the default to be the result of an on-line calculation. Calculations may be performed using previously entered entry

categories. Calculations are performed from left to right, and only arithmetic data is allowed. Format is:

`@=calculation(;O)(;precision 'n') n=0 to 4`

Where:

`+` = addition

`-` = subtraction

`*` = multiplication

`/` = division

`@=6+7**"1.5";;2` means default equals the result of adding data from prompt '6' and '7', then multiplying the result by '1.5', and displayed using '2' decimal points.

All calculations are rounded to the precision selected after each calculation. If no precision is specified, then '4' decimal precision is used

Image Name Retrieval

In the case image controls, the Defaults property is filled in with the name of the image resource that has been loaded into the prompt.

Last/Prior Entry Default

This will default the last/prior data value, entered in the previous use of this prompt. If the default value is empty then the AllowChange parameter will be ignored. Thus a default of "@Last" will stop and allow input on the first pass and skip the field on all subsequent passes. The intended behavior of "@Last" is to maintain the last entered value until the application is exited.

Format is:

`@LAST(;O)`

Lists (Listing Choices) Default

This default is used to list multiple specific choices, from which 1 item may be selected. Items selected appear in a 'List' on the device. Format is:

@list,heading,item,item,item,...

Where 'heading' is the column heading name to print.

@list,Colors,RED,WHITE,BLUE allows the user to select 1 of the 3 colors

Skip the Current Entry Default

This default is used to conditionally skip the current prompt. Format is:

@SKIP,exp1,exp2,exp3

Conditional Operators (exp1,exp2,exp3) are:

exp1 = the prompt number

exp2 = a conditional operator from 1 of the following:

- = for an equal to comparison
- # for a not equal to comparison
- > for greater than
- < for less than
- M for a matches comparison
- nM for a not matches test
- I: exp4 exists in exp2 (same as NC)
- C: exp2 exists in exp4

NC: exp4 exists in exp2

exp3 = the prompt number or string value (in quotes) to match on.

@SKIP,2,=,'N' means to skip the current prompt if the data entered at prompt '2' equals 'N'.

Note in this example that exp1="2", exp2="=", and exp3="N"

Set Default

This default is used to validate data before inserting the specified default value. Format is:

@Set,exp1,exp2,exp3,exp4

Conditional Operators (exp1,exp2,exp3,exp4) are:

exp1 = the value that will become the default

exp2 = the parameter to be compared to exp4. This can be a literal, a prompt number, or the RSP variable.

exp3 = a conditional operator from 1 of the following:

- = for an equal to comparison
- # for a not equal to comparison
- > for greater than
- < for less than
- M for a matches comparison
- nM for a not matches test
- I: exp4 exists in exp2 (same as NC)
- C: exp2 exists in exp4

NC: exp4 exists in exp2

exp4 = the second parameter to be compared to exp2. This parameter can be a literal or an edit such as NUM and ALPHA.

@Set,'Hello',2,=,'100260'

This sets the default to 'Hello' if prompt 2 is equal to the string value 100620.

@Set,'Hello','XYZ',#,ALPHA

This sets the default to 'Hello' if the string literal XYZ is not an ALPHA string. In this case, the default would never be used.

@Set,'Hello','3',C,'12345'

This sets the default to 'Hello' if the second parameter (3) is contained within the forth parameter (12345).

SQL Statement (List) Default

This default is used to develop multiple selection items from your database so that 1 may be selected. Format is:

@sql, statement

(where statement is a valid SQL select statement)

The heading that appears is the database column name.

@sql, select PONum from OpenPO would select the purchase orders numbers column, from open purchase orders table 'OpenPO'.

A List, when used, clears the existing screen and lists selected data items (vertically) on the display screen. The RFgen 'List' display feature is a 'full screen display'. The user then selects 1 of the items in the list. There are 3 examples of list creation which follow.

1. For predefined/known list box items use an '@list' default:

```
@list,heading,item.1,item.2,item.3,...,item.n
```

Here: '@list,' indicates to RFgen that the statement is a 'default' parameter 'heading' is a name for the list that prints at the top of the box when it appears on a device 'item.1, item.2,...' are a listing of selection items to appear in the list.

e.g. - @list,Colors:,RED,WHITE,BLUE means to display a list with a heading of 'Colors:', and 3 selection items RED, WHITE, and BLUE to appear in the list.

2. For selection items which come from a database table, use an '@sql' default:

An SQL statement as an application 'default property' (for the prompt that will use the list) may be used to select the necessary data from your database, as illustrated here:

```
@sql,select fieldname1 from tablename where fieldname2 ='%n'
```

Here: '@sql' indicates to RFgen that the statement is a 'default' parameter

'fieldname1' is the name of the table field/column to be displayed

'tablename' is the name of a database table which contains the fields/columns

'where fieldname2=' allows selection of certain data only

'%n' indicates data entered at prompt 'n' will be used for selection purposes.

`@sql.select OrderNO from PObooks where PartNO ='%1'` means to create a list box of order numbers from table 'PObooks' where the part number was entered at prompt number 1.

3. Alternatively, for database items, use a VBA script, e.g.:

```
Public Sub Fieldname_GotFocus(Rsp As String,  
AllowChange As Boolean)  
    Dim SQL As String  
    SQL = "Select fieldname1 from tablename where  
          fieldname2 = '%n'"  
    Rsp = DB.MakeList(SQL)  
    AllowChange = True  
End Sub
```

Here, the RFgen VBA extension 'DB.MakeList' is used in a GotFocus event to make a list from the results of the specified SQL statement. The results are passed to a prompt default variable named 'Rsp'. A variable called AllowChange is set to True, meaning override allowed.

User Default

This default will put in the login name of the user. Format is:
`@User`

Edit Property Details

Validations/edit checks are available to test if data meets a certain criteria (e.g., the data entered is numeric). A validation property is available for each prompt and validations are entered in the Properties window. Multiple validations/edits are allowed. Select the edit property, and select the 'Ellipse' button. A multiple line input box will display. Enter one edit per line. *Note that as soon as an entry passes an edit, any subsequent edits are ignored.*

If an entry has no validation criteria, leave its edit property blank.

Text (Character String) Validation

This will test for an exact match between the data entered and the text edits required. A text edit may be entered either with or without quotes. Format is:

'string' (in single quotes)

'CA' means the user must enter CA to pass this edit test

Pattern Match Validation

This will test the data entered to see if it matches the pattern specified. Format is:

xA or xN

For example, 2A means that the data entered must have the format 'AA' (i.e., match 2 alphabetic characters, for example: 'OR'). 4N means the data entered must be in the form 'NNNN' (i.e., 4 numbers; for example: '1234').

4N means the user must enter a valid 4-digit number.

Pattern Not Allowed Validation

This will test the data entered to assure that it does not match the pattern specified. Format is:

#PATTERN

#4N means the user may enter anything but 4 numbers (i.e., not 4 numeric characters)

#Hello means the user may not enter Hello in the prompt.

Alpha Only Validation

This will test the data entered to assure that it only contains alpha characters (A-Z and a-z). Note: no spaces are allowed. Format is:

ALPHA

Numeric Only Validation

This will test the data entered to assure that it only contains numeric characters (0-9), and the special characters '.' (period), '+' (plus), and '-' (dash). Note: no spaces are allowed. Format is:

NUM

Integer Only Validation

This will test the data entered to assure that it only contains an integer number (no decimal). Note: no spaces are allowed. Format is:

INT

Greater Than Validation

This will test the data entered to see if it is greater than a specified value. Format is:

>number or >=number

>99 means that the user must enter a number greater than 99.

Less Than Validation

This will test the data entered to see if it is less than a specified value. Format is:

<number or <=number

<99 means that the user must enter a number less than 99

Not Equal To Validation

This will test the data entered to assure that it does not equal a specified value. This is a string comparison only. Format is:

#'stringvalue'

#'99' means that the user must enter a string that does not equal '99'. Since prompts accept data as strings by default this validation would work with numbers while treating them like strings.

Data Range Validation

This will test the data entered to see if it is within a specified range. This is a numeric comparison only. Format is: RG/exp1,exp2

Where:

exp1 = starting number for range.

exp2 = ending number for range.

RG/1,100 means that the user must enter a number between 1 and 100, inclusive

Date Validation

This will test the data entered to see if it is a proper date. Format is:

DATE

Entering MMDDYY, MMDDYYYY, MM-DD-YY, MM/DD/YY among others are all valid entries.

Index Validation

This will test the data entered to see if it can be found within the edit string. Format is:

I:string

I:YN means data must be either 'Y' or 'N', or 'YN'.

Contains String Validation

This will test the data entered to see if it contains the specified edit string. Format is:

C:string

C:abc means data entered must contain the characters 'abc' somewhere within it

Not Condition Validation

This will test the data entered to see if it does not match a given condition. Format is:

NC,exp1,exp2,exp3

Conditional Operators (exp1,exp2,exp3) are:

exp1 = the parameter to be compared to exp3. This can be a literal (in quotes, even if numeric, eg '3'), a prompt number, or the RSP variable.

`exp2` = a conditional operator from 1 of the following:

- = for an equal to comparison
- # for a not equal to comparison
- > for greater than
- < for less than
- M for a matches comparison
- nM for a not matches test
- I: exp3 exists in exp1 (same as NC)
- C: exp1 exists in exp3

NC: exp3 exists in exp1

`exp3` = the second parameter to be compared to `exp1`. This parameter can be a literal or an edit such as NUM and ALPHA.

`NC,2,=,'100260'`

This checks prompt 2 to see if it is equal to the string value 100620. If it is NOT, the edit is satisfied.

`NC,'XYZ',#,ALPHA`

This compares if the string literal XYZ is not an ALPHA string. Because XYZ IS an alpha, this edit is satisfied.

`NC,'3',C,'12345'`

This edit is NOT satisfied because the first parameter (3) is contained within the third parameter (12345).

Branch/Goto Validation

Technically, this is not a validation. Allows the display to Goto a 'downstream' prompt, based upon the data 'RSP' entered. Format is:

`@GOTO,prompt#,conditions)`

`@GOTO,5,RSP,=,"99"` means to go to prompt 5 if the response at the current prompt equals "99". See the @SKIP default parameters for examples of 'conditions'

Strip (Data Entry) Validations

Validates that 'str' is there, then strips data 'str' from the entry. Format is:

@STRIP,P,str to strip data prefix 'str' if it occurs

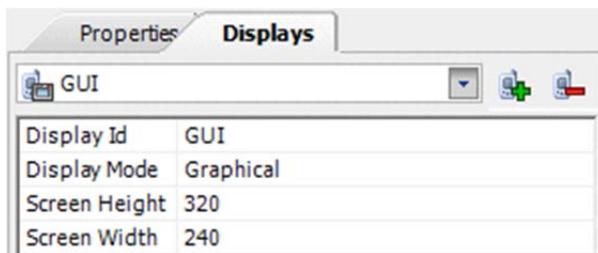
@STRIP,S,str to strip data suffix 'str' if it occurs

@STRIP,C,str to strip the first occurrence of the data 'str' if it is contained in the data.

@Strip,P,NBR: means to strip the character string 'NBR:' if it prefixes the data.

Displays

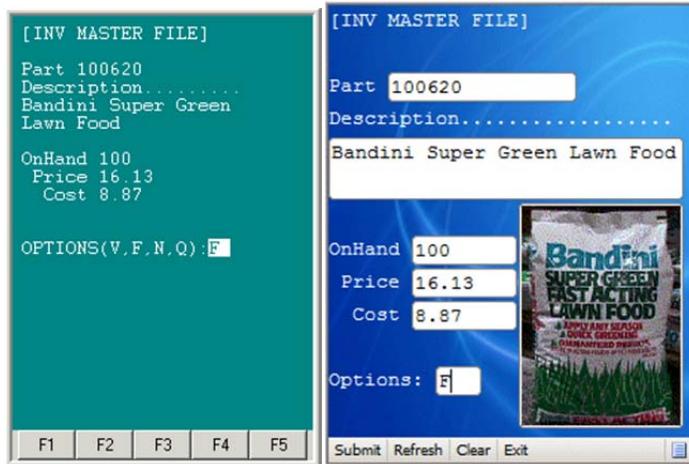
The Displays tab allows the user to create variations of the display for the same transaction.



The drop-down box lists all the displays already created. To add an additional display or remove the current display, click on the plus and minus icons.

The **Display ID** is the name of the display that can be referenced in the script to call up the custom display based on any given criteria like user profile properties or IP address. `App.SetDisplay` is the script command.

The **Display Mode** property is only for the design time creation of the application. If at run time the user is using a graphical device and calls an application designed in character mode, it will still render as a graphical application. RFgen supports both 'character/text' and 'graphical' data entry applications. Consider the following display differences for the IMASTER application:



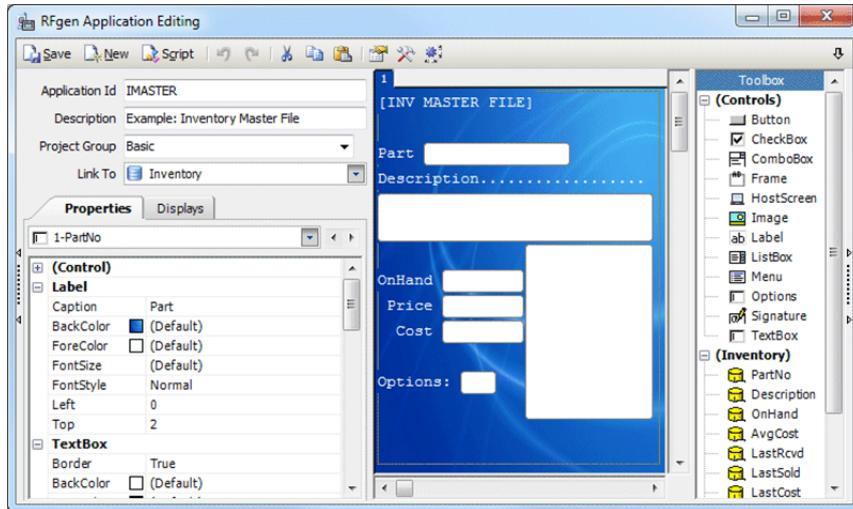
Character Mode

Graphical Mode

To alter the transaction perimeter outline change the **Screen Height** and **Screen Width** properties. When operating in Character Mode, remote devices use telnet to communicate with the Server and the size is dictated by columns and rows; in Graphical Mode, remote devices use the RFgen 'windows/graphical client' to communicate with the Server and the size is referenced in pixels. These properties place a rectangle on the design window to suggest what might not be displayed if the prompt fields should be placed outside the design area or are stretched too far. Under the menu option Options / TCP/IP Services, in the Controller Failover Monitoring grid, you can specify specific IP addresses and their preferred display screen size. For example, if a fork-lift mounted device prefers a 40x8 sized screen, RFgen can automatically load the created display of that size by default if one exists for any application requested by the user from the menu.

The Application Construction Area

A multi-page display area appears on the right-hand side of the window. This is where the application is constructed. Multiple pages can be constructed if an application is to have more than one page to complete a transaction. This is only the case if there are more prompts than will fit within the confines of the mobile device's screen.



To build a data collection application, drag-and-drop field objects into the display area (or double-click on the objects and they will be automatically transferred to the display), in the order in which prompts will appear. The critical properties for prompts are:

1. The Field ID
2. The Left, Top and Caption for the prompt
3. The Left, Top and Width of the data field

Since your first attempt will probably not appear exactly as desired, these items may be changed. To accomplish this:

Click on the Left, Top, Caption or Width properties and modify the existing properties, or select and drag the prompts with your mouse. Note that a data field and its associated Caption (prompt) are addressable separately.

The upper left-hand corner of the prompt display area is 0,0; the next line down is 0,1 (Column 0, Row 1), and so forth.

To drag a displayed item, position your mouse on the desired prompt control and select it with the left mouse button. Drag the item to the desired position, and then release the mouse button. Holding down <ctrl> while clicking in multiple prompts will select all of them and make moving several prompts easier.

Notice the double arrows located on the Properties tab.  They may be used to select each prompt in the order that they were placed. Click on the left or right arrow to select a different prompt. The combo box provides the same feature in a list format.

To insert a prompt, add the prompt as usual and then change the order on the Properties tab using the Prompt Number property. To delete a prompt, right-click on the prompt and click on Delete from the popup menu. The prompt will disappear.

To reorder prompts, change the 'Prompt Number' property on the Properties tab for each prompt.

Scripts – Menu Bar

When the script window is opened, the menu bar at the top of the window provides some standard and some custom features.



The **Item** label displays which application is being edited. The **Save** option saves the current script to the RFgen database. The **Test** option performs a syntax check of the script. Also provided are **Undo**, **Redo**, **Cut**, **Copy** and **Paste**. The **Comment** and **Un-Comment** options allow quick removal or addition of code blocks.

The **Add Module** specific dependencies button allows for non-globally loaded VBA Modules to be associated to the application. This window displays existing VBA modules. (See the VBA Modules tree) Check those modules you wish to include with the current application. Modules designated Win32.bas and RFgen.bas are automatically included for each application.

The **Find** icon allows the user to find text or replace text within the current application or all applications and macros at the same time.

The **Print** button will send the script to the default printer.

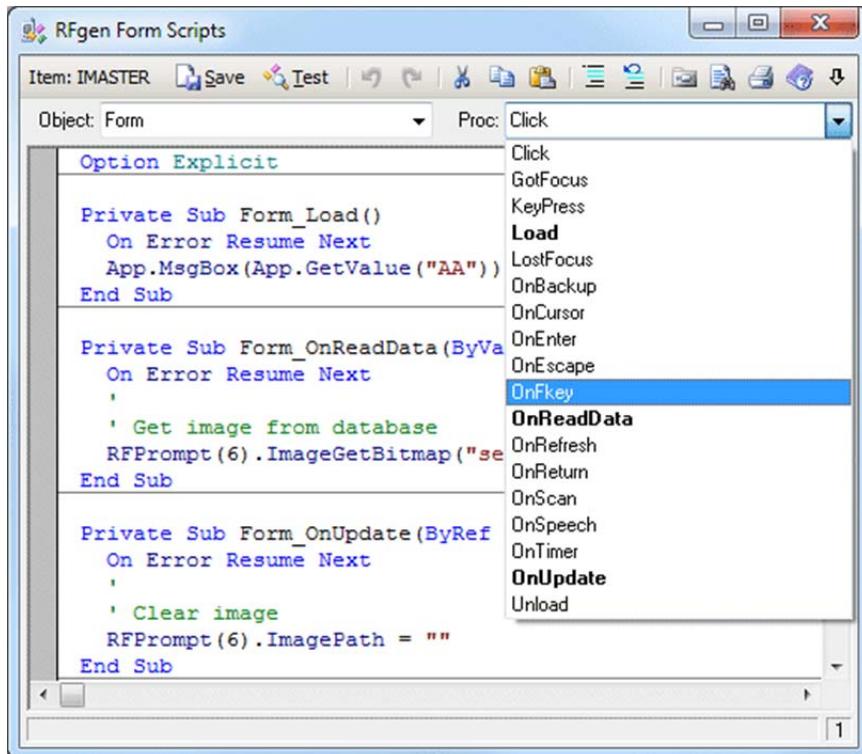
The **Help** icon is for opening the RFgen manual.

The **Dock** icon, (the down arrow) displays on every un-docked window, allowing the user to re-dock back to the Mobile Development Studio main window. Double-clicking the top-most tab will undock the window.

Scripting windows also provide some keyboard shortcuts that may make moving around in the environment easier. Ctrl + A will highlight all the text in the window. Ctrl + G will request a line number and then move the focus to that location. Ctrl + Spacebar will pop up a search window for language extensions, embedded procedure parameters and other similar uses.

Scripts – Global, Application and Prompt Events

When the user clicks on the Script menu option a scripting window similar to the example shown below will appear.



When using VBA, note that each prompt on the application, and the application itself, may have associated scripting for the numerous associated events. All possible VBA events are:

Click occurs when the user clicks on a prompt or button with a mouse or pen

GotFocus	occurs whenever the user enters a prompt. First, the application level executes and then, the prompt level executes
Keypress	occurs when the user presses and releases a key on the keyboard
Initialize	occurs when an RFgen client is initially loaded. It will occur only once, and is typically used to initialize variables, open additional database connections or create links to ActiveX objects. (RFgen.bas only)
InRange	occurs when a Mobile Client notices that there is an IP address available on the network adapter. This event does not execute when using the Mobile Client in a Roaming state because the OnConnect will execute anyway. (RFgen.bas only)
Load	occurs once when the application is initially loaded
LostFocus	occurs whenever the user leaves a prompt. The prompt level executes first and then the application level executes
OnBackup	occurs when the up arrow was pressed to move to the previous prompt
OnConnect	occurs when the Mobile Client in the Roaming state automatically discovers it has connectivity to a network and makes a connection to the RFgen server. The event only executes after 10 seconds of continuous connectivity to the RFgen server. To maintain data integrity, it may be a good idea to include the Server.SyncApps and Server.SendQueue in this event. (RFgen.bas only)
OnCursor	occurs whenever one of the arrow / cursor keys are pressed
OnDisconnect	occurs when the Mobile Client in a Roaming State is disconnected from the RFgen server because the client moved out range. This event does not execute when the Server.Disconnect command is issued. (RFgen.bas only)

OnEnter	occurs whenever the user exits a prompt by the Enter key, or the AllowChange variable in the GotFocus event is set to False
OnEscape	occurs whenever the Escape key is pressed
OnFkey	occurs whenever a function key is pressed
OnLocale	occurs after OnConnect (only when a client application makes a connection) and passes in the locale number based on the client device's location (RFgen.bas only)
OnReadData	occurs whenever data is successfully retrieved from a database by an RFgen application that is linked to a table (application level only)
OnRefresh	occurs when the screen is completely repainted (as when the user presses the F2 key). It is typically used to redisplay information to the user that was displayed via Screen.Print statements (application level only)
OnReturn	occurs when the user returns from a called application that had its return flag set to True. It is typically used to process data from the called application (application level only)
OnScan	occurs when RFgen identifies a pre-amble or post-amble string in the data stream coming from a client device. This event is typically used to validate that data was scanned, or to parse the data (PDF, RFID, etc.) into a more usable format.
OnSearch	occurs when a user clicks on the search icon created at the end of a textbox. There is an icon to click only if there is code in this event. (field level only)
OnSpeech	Using the language extension TTS.Listen, RFgen will detect that speech has occurred and execute this event. (Also exists in RFgen.bas which executes first, then the form level and finally the prompt level)

OnTimer	occurs at a user-defined interval when it is enabled. This event is used to perform some action based upon defined time interval (application level only)
OnUpdate	occurs after the last prompt has been entered and just prior to RFgen updating the database (application level only)
OnVocollect	a specialty event that is used with Vocollect tasks embedded on an application that communicates to and from Vocollect hardware and RFgen (field level only)
OutOfRange	occurs when a Mobile Client notices that the IP address is no longer available on the network adapter. This event does not execute when using the Mobile Client in a Roaming state because the OnDisconnect will execute anyway. (RFgen.bas only)
Terminate	occurs when an RFgen client is terminating. It will occur only once, and is typically used to close manually opened database connections or release links to ActiveX objects. (RFgen.bas only)
UnLoad	occurs once when the application is exited (application level only)

Application events take precedence over prompt events; e.g., events linked to the application will occur prior to the event firing for a prompt.

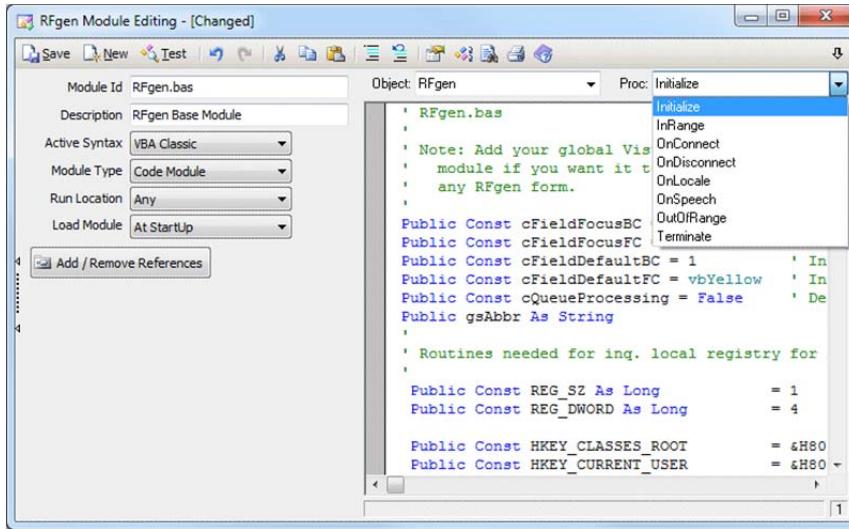
Note: VBA code is very sensitive to variable typing. Most errors result from using a String variable when an Integer or Long data type is required.

For complete information on VBA scripting, see Visual Basic Scripts, and the RFgen documentation accessible via the Start \ Programs \ RFgen v4.1 \ Documentation menu.

VBA Modules List

The VBA Modules contain global definitions of variables, functions, and procedures that can be referenced by any application or macro script. The default installation provides two “Code” BAS modules: Win32 and RFgen.

The Win32.bas module provides a place for global variables and procedures that pertain to system level requirements. The RFgen.bas module provides a place for global variables and procedures that pertain to operation of the RFgen transactions. Both are always available for all applications and macros and are functionally identical. The RFgen.bas module does contain an RFgen object with its own events as described above.



VBA Module Editing Menu Bar

The **Save** option saves the current script to the RFgen database. The **New** option clears all fields and provides a blank window for creating a new bas module. The **Test** option performs a syntax check of the script. Also provided are **Undo**, **Redo**, **Cut**, **Copy** and **Paste**. The **Comment** and **Un-Comment** options allow quick removal or addition of code blocks. The **Properties** icon hides or un-hides the configuration options panel. The **Module** icon opens the Visual Basic Environment Configuration screen and lets the user prioritize the load sequence of the BAS files. This is important if some BAS files rely on other BAS files for declared variables or functions. The **Find** icon allows the user to find text or replace text within the current application or all applications and macros at the same time. The **Printer** icon will print the modules code to the printer and add line numbers to the left edge for easier review. The next **Help** icon is for opening the RFgen manual.

The **Dock** icon, (the down arrow) displays on every un-docked window allowing the user to re-dock back to the Mobile Development Studio main window. Double-clicking the top-most tab will undock the window.

VBA Module Editing Window Fields

The **Module ID** is the RFgen identifier for the bas module. Typing in an existing name and exiting the field will check if that name is already used and load that module, if it is found. The **Description** is listed in the Design tree for clarification of the modules purpose.

Active Syntax specifies what kind of module is being created. A VBA Classic BAS file is treated as a standard referenceable module. A .Net module allows for Global Assembly Cache (GAC) references to classes created outside of RFgen. This option is not compatible with a Mobile installation because the device will not have access to these classes.

The **Module Type** option will create the BAS file to be either a Code, Class or Object module. A Code module is the standard referenceable module as before. The Class module with public and private functions can be created and then referenced in the application's code. For example a Class module called Math has one public function called Add and one private function called Subtract then in the application script:

```
Dim MyClass As New Math  
MyClass.Add()
```

The Subtract function is not available because it is private. Because it is a class you may create multiple instances of the class as needed. The Object module type is similar but RFgen will create the first instance automatically. Using the same example the Dim statement would not be required. Instead all that is needed is:

```
Math.Add()
```

In this case the initial object instance is used. If more are required then you would use the same example as the Class.

The **Run Location** refers to the mode of the client. The module will either be available on both the server and the device (in the case of a Mobile implementation), just the server (in case there is code that will not be compatible in the CE / Android / iOS environment) or just the device (in case the code is specific to the mobile environment. For example if there was a need to have the same global function work in two different ways depending on if the device was in a mobile mode or

thin client mode, two differentBAS files could be created with identical function names, one set to Server and the other set to Device. Then if the device was online the Server version would be used and if the device went offline the Device version would be used and a global syntax check of the solution would not see a conflict.

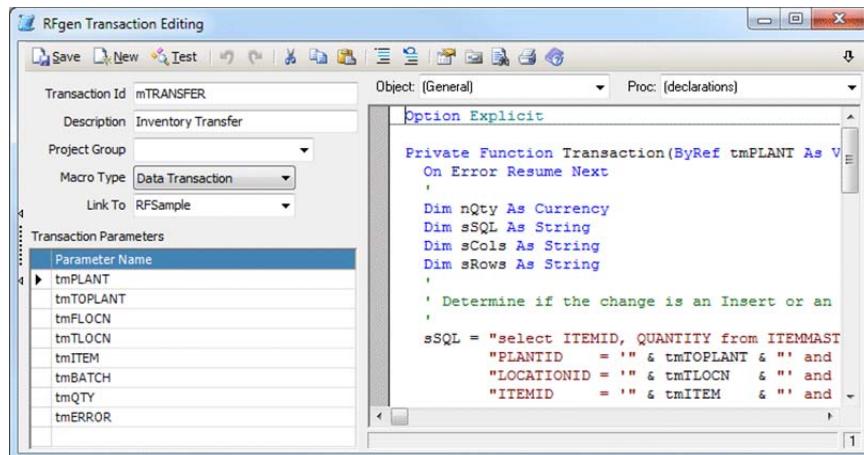
The **Load Module** option makes the BAS file either globally loaded and available all the time or just another module that must be referenced from each application that needs to use its functions.

Add / Remove References allows the programmer to add Global Assembly Cache (GAC) references to the VBA module such as a custom class created outside of RFgen. This is very useful for .Net module types.

In the code environment the Object drop-down box will contain the RFgen object for the RFgen.bas module only. For all other modules, it will remain empty. The Procedure drop-down box will contain any user-created function or procedure names. In the case of the RFgen object (as shown above) the user will have access to the internal events.

Transactions List

The transaction design window has 3 purposes: to create and edit Timed Event macros, Host Transaction and Data Transaction macros.



A Timed Event macro is a macro that runs on a timer configured under the Options / Transaction Management / Processing Events dialog. If you want some script to run without an RFgen user being present,

create a Timed Event macro and enable the Transaction Management capabilities. There are no passing parameters for Timed Event macros.

A Data Transaction macro is a script that can accept parameters passed in and out and can execute in a queued or non-queued manner. You would create a Data Transaction macro: if multiple applications could take advantage of the same process, you need a history of the data being processed kept by the Transaction Manager, applications are run in a Mobile environment and later uploaded to the RFgen server for processing or queuing is implemented for all applications.

To create a Data Transaction macro, follow these steps:

1. Enter a Transaction ID, Description, set the Macro Type to Data Transaction and set the Link To option to the appropriate data connection. The Project Group lets the user group like items together in the navigation tree.
2. Add Field IDs for each value being passed in to the macro. The Location and Length columns do not apply to Data Transaction macros. A maximum of 31 can be used due to the integration with the VBA environment. To work around this you may concatenate multiple values separated by a unique string like “ | “ and only use 1 parameter.
3. Select the Transaction function from the Procedure drop-down and a shell function will be created for you.

A Host Transaction macro can be created and linked to a Host Screen macro. However, the Host Screen macro is capable of containing the script for performing the data entry on the host system. This macro type is here only for backward compatibility. Older versions of RFgen that use screen mapping will have their Host Transaction macros placed under the Transactions list.

Transaction Module Editing Menu Bar



The **Save** option saves the current script to the RFgen database. The **New** option clears all fields and provides a blank window for creating a new transaction macro. The **Test** option performs a syntax check of the script. Also provided are **Undo**, **Redo**, **Cut**, **Copy** and **Paste**. The **Comment** and **Un-Comment** options allow quick removal or addition of code blocks. The **Properties** button hides or unhides the left half of the screen containing the parameters grids and header textboxes. To allow references to global scripts that are in a VBA module but are not

globally available select the **References** menu item. This window displays existing VBA modules. (See the VBA Modules tree) Check those modules you wish to include with the current macro. Modules designated Win32.bas and RFgen.bas are automatically included for each macro.

The **Find** icon allows the user to find text or replace text within the current application.

The **Printer** icon will print the modules code to the printer and add line numbers to the left edge for easier review.

The **Help** icon is for opening the RFgen manual.

The **Dock** icon, (the down arrow) displays on every un-docked window allowing the user to re-dock back to the Mobile Development Studio main window. Double-clicking the top-most tab will undock the window.

Hosts List

The Hosts List displays all screen mapping related macros. Please see the screen mapping section for more information.

Resources Tab Overview

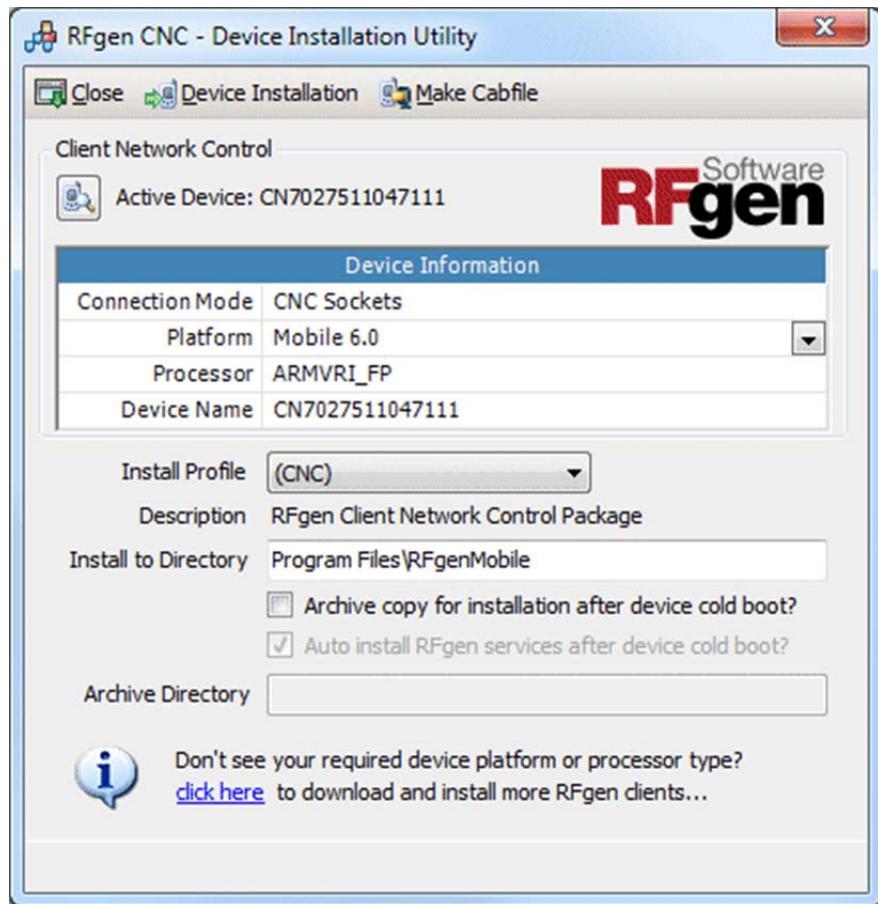
The Resources tab group together the items that make up the non-scripting portion of the solution. In this section, there are mobile device profiles, image resources, voice specific resources with grammar customizations, text resources and Vocollect specific tasks that interface the Vocollect hardware devices to RFgen's data connectors.

Device Profiles Tree

The Device Profile Tree is used to prepare a mobile device for use in the wireless / wired environment or the mobile environment.

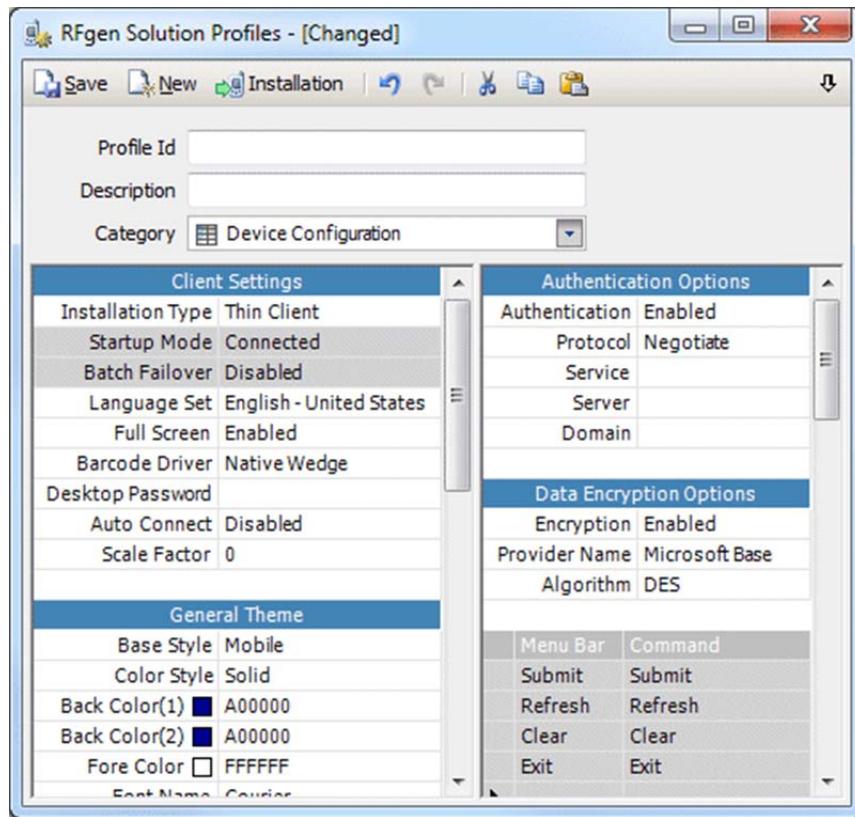


The **Save** option saves the current profile to the RFgen database. The **New** option clears all fields and provides a blank window for creating a new profile. The **Installation** brings up the CNC installation window.



This is the same window as found under the Devices menu option.

Also provided are **Undo**, **Redo**, **Cut**, **Copy** and **Paste**. The **Dock** icon, (the down arrow) displays on every un-docked window allowing the user to re-dock back to the Mobile Development Studio main window. Double-clicking the top-most tab will undock the window.



The **Profile ID** is the RFgen identifier of the configuration and gives the user the ability to open, edit and save a certain configuration to avoid recreating it for each mobile device. The **Description** explains the purpose of the profile and displays in the tree when saved.

Category: Device Configuration

This category contains some basic setup information starting with the Client Settings Group.

For **Installation Type**, choose the type of client that will reside on the handheld. Your choices are Thin Client or Mobile Client.

Thin Client The traditional wireless real-time interface to RFgen where the mobile device is restricted to the RF environment.

Mobile Client This option allows the user to leave the RF environment and manually or automatically switch to and from a

connected state. See the Mobile Devices Section for more detailed information.

The **Startup Mode** refers to which state the mobile client will use when started, connected, disconnected or roaming. The Thin Client option will only have the 'Connected' choice.

The **Batch Failover** refers to the mobile client requesting to switch from thin client state to a disconnected state when the RF signal is lost. The user can deny this request and wait for RF coverage to return.

Language Set The default is English. Any language may be chosen from this menu as long as the language is installed on the mobile device.

Full Screen This option determines if the display on the mobile device is in a window (smaller) or if the RFgen application is maximized for the screen (larger.)

Barcode Driver This is for the future use of a manufacturer that may choose to release a special driver for their device. RFgen uses the hardware's native driver by default.

Desktop Password This option is designed to prevent unauthorized users of the mobile device from leaving the RFgen client and returning to the device desktop.

Auto Connect If multiple RFgen servers are configured and this option is enabled the client will rotate through all configured servers until it finds one that will accept a connection. Servers are configured in the RFgen Servers category.

Scale Factor This option is designed to fix a problem on some devices where they do not report their DPI (dots per inch) correctly. If the high resolution screen forces the RFgen screen into a very small space, change this value from 1 to 2 to double the rendering of the RFgen screen.

The General Theme settings are optional and are derived from the Mobile Theme settings from the Options menu.

Base Style	Options include Mobile or Classic themes for a more modern look or the older square shaped prompts.
Color Style	This applies to the color pattern for the background if there is no image chosen in the Background Image section.
Back Color (1), (2)	Depending on the environment, you may choose to change the back color for better visibility. The first background color is used with the second background color to create shading effects based on the Color Style option.
Fore Color	This is the color of the text on the screen. Use this in conjunction with the Back Color to achieve the best visibility.
Font Name	The font used on the mobile device
Font Size	The font size used to display all transaction applications
The right pane is for all the optional settings that may be used by the client.	
Authentication	Authentication options are taken from the configuration settings. It is best to leave these consistent with the server settings.
Data Encryption	Data encryption options are taken from the configuration settings. It is best to leave these consistent with the server settings.

The **Menu Bar** options are taken from the configuration settings. This list can be altered as needed for the mobile applications.

The next section completes the look-and-feel of the mobile device.

Background Image	
Image Display	Stretched
Image Name	RFgen
Button Controls	
Color Style	Diagonal Right
Back Color(1)	<input type="checkbox"/> EFEFEF
Back Color(2)	<input checked="" type="checkbox"/> 5F5F5F
Fore Color	<input checked="" type="checkbox"/> (Default)
Edit Controls	
Back Color	<input type="checkbox"/> (Default)
Fore Color	<input checked="" type="checkbox"/> (Default)
Selection Highlight	
Back Color	<input checked="" type="checkbox"/> (Default)
Fore Color	<input type="checkbox"/> (Default)
Menu Strip	
System Menu	Bottom
Font Name	Tahoma
Font Size	8

Background Image:

The **Image Display** will stretch, tile or move an image around on the screen for the background of each form. The **Image Name** specified retrieves the image from the Image Resources which must contain the image first.

Button Controls:

The **Color Style** is the same as the Color Style listed in the General Theme section but applies to command button prompts only. **Back Color (1 and 2)** performs the same shading effect and the **Fore Color** is the color of the text on the button.

Edit Controls:

The **Back Color** of the data fields can be controlled here and the **Fore Color** is the color of the text inside the data field portion of the prompt.

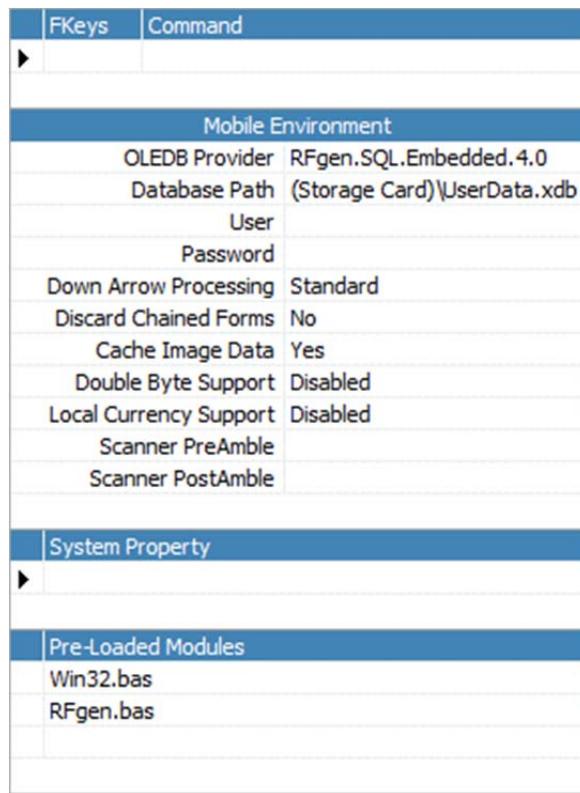
Selection Highlight:

The **Back Color** of the highlighted row in a combo box or list box is set here. The **Fore Color** is the color of the text inside the highlighted row. These should be contrasting colors for readability.

Menu Strip:

The **System Menu** bar can be displayed either at the top or bottom of the screen. It may also be disabled where it will hide the buttons on the menu bar. The **Font Name** property sets the font used on the menu bar and the **Font Size** adjusts the size of the text on the buttons.

The next section continues the mobile device's configuration settings.



The **FKeys** list is taken from the configuration settings. This list can be altered as needed for the mobile applications.

OLEDB Provider Specifies which provider is used on the mobile device. Although you can freely enter

	<p>text in this property, it is pre-populated with the RFgen suggested values.</p>
Database Path	The database path refers to where on the mobile device the file should be created. The file name by itself will be placed in the RFgen directory. Specifying a path such as APPS\RFSSample.xdb will place the file in that location.
User	If a user is required to access this mobile database, enter it here. There is no relation to the authentication required for the server's databases.
Password	If a password is required to access this mobile database, enter it here.
Down Arrow Processing	If this option is set to Standard, then the down arrow will not be used as an alternative for the <Enter> key.
Discard Chained Forms	This sets the current application data to null when another application is called, rather than keeping the original data in memory (the default condition) should the calling application be returned to.
Cache Image Data	This defaults to the setting in the Application Environment options and will place images on the mobile device if they were not pre-loaded during installation and the client determines that an update is necessary.
Double Byte Support	This option will allow Chinese, Japanese, and other double byte character sets to display and wrap properly.
Local Currency Support	This option enables support for international currency formats, by using the mobile's system regional and language option settings (e.g. \$1,234.56 becomes \$1.234,56).

Scanner PreAmpble	An optional character string filter that comes before the scanned data. It is compared to the scanned data and if there is a match RFgen will execute the OnScan event if it exists.
Scanner PostAmpble	An optional character string filter that comes after the scanned data. It is compared to the scanned data and if there is a match RFgen will execute the OnScan event if it exists.

The **System Property** Ids and values are taken from the configuration settings. This list can be altered as needed for the mobile applications.

The **Pre-Loaded Modules** list is taken from the configuration settings. If they are not necessary or if supplemental BAS files are required for the mobile applications, this list can be changed as needed.

Speech Recognition	
Speech Recognition	Enabled
Default Language	(None)
Default Voice	(None)
Read Mode	Sentence
Speech Rate	50
Sound Volume	90
Sentence Pause	400
Confidence Level	5000
Trailing Silence	200
Response Timeout	10
Trim Response	True
Performance Level	5000

Language Pack	Voice
▶ English	Jill

Speech Recognition Set to Enabled, it will deploy the speech files and settings to the mobile device

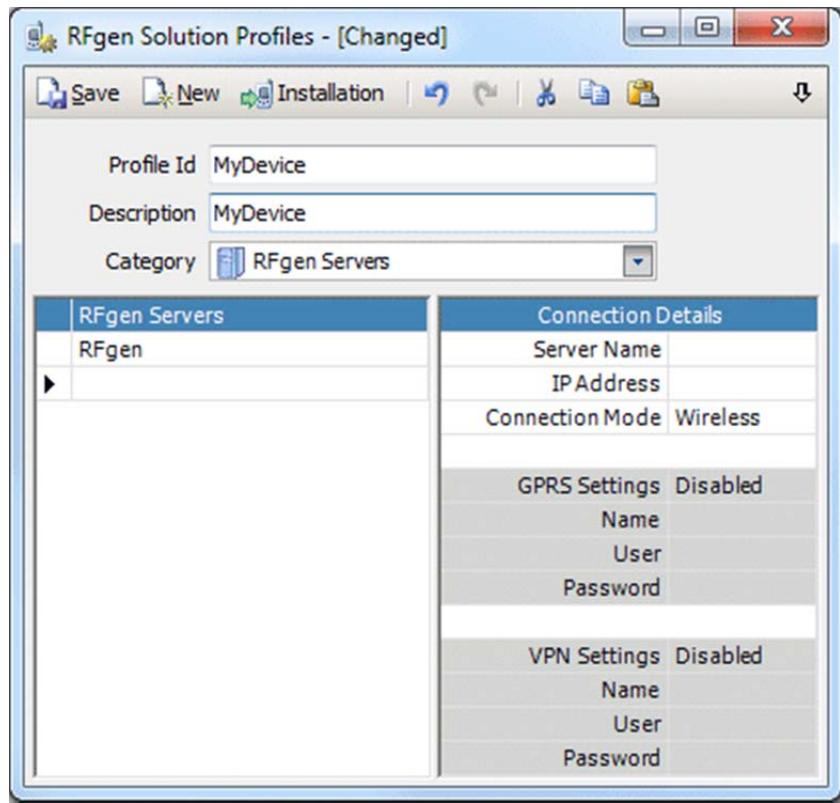
Default Language is simply the default dictionary the speech engine uses to validate spoken words and the phonetic basis for speaking to the user.

Default Voice	is the name given to the rate and pitch of the spoken text for the chosen language.
Read Mode	changes how the text is spoken. There are 4 options: Character, Line, Sentence and Word. The differences are the pausing used between characters, words and lines.
	Character – the system will read character by character
	Line – the system will read line by line
	Sentence – the system will read the text sentence by sentence. In order to divide text into sentences, the system applies heuristics based upon the punctuation marks and capitalization.
	Word – the system will read the text word by word
Speech Rate	sets the speed at which the text is played back to the user. The allowed number range is 1 – 100 with the default set to 50. The higher the number the faster the speech will be.
Sound Volume	is how loud RFgen will speak the text. The range 0 – 100 and is recommended that it be set at the maximum or loudest level. This volume is still subject to the volume setting of the operating system and / or hardware device. Since it is usually much easier to access the volume controls of the device (buttons on the side of the device or a speaker icon on the desktop) volume can be more easily adjusted for comfort there.
Sentence Pause	is the number of milliseconds between 0 and 1,800 that the system will pause between sentences. Sentences must be in proper sentence case for this to work meaning proper letter case and punctuation.

Confidence Level	is a number between 0 and 10,000 that represents the threshold for how well the spoken word matches to the defined allowable answers. The default is 5,000. Setting the number lower means the system is allowed to do more guesswork to interpret what the user said.
Trailing Silence	is a number between 0 and 2,000 milliseconds that is the pause made by the user that the system waits for before accepting what was spoken as the completed response. The default is 200. Speaking too slowly will result in the system thinking it has received the complete response and only capturing a portion of the complete response. If this happens increase the value.
Response Timeout	is a number between 0 and 32,000 seconds that the user has to respond while the speech engine is listening or when using the TTS.GetInput command without specifying the optional timeout value.
Trim Response	will remove all spaces throughout the captured speech text. For example, if the part number 1234 is spoken it would normally be returned as 1 2 3 4 but with this setting to True it will be returned with no spaces for easier validation.
Language Pack	The user can add as many language packs to the installation as desired.
Voice	Shows the name of the voice used for a chosen language.

Category: RFgen Servers

The RFgen Servers category records which RFgen servers the mobile device will have access to and what the connection type will be.



The list of RFgen servers provide the mobile user a choice of which RFgen server they want to connect to. In the left pane simply list the name of each server. The Connection Details on the right will determine exactly how the client makes the connection.

The **Server Name** is reiterated and is changeable in either the left or right pane. The **IP address** is the IP of the RFgen server itself. The **Connection Mode** is either Wireless (which assumes the client is already on the private LAN or WAN) or Cellular. If cellular is chosen, then there are additional options for making a cellular connection.

Depending on the platform of the mobile device, making a cellular connection can vary greatly. For example, if a VPN is configured on the device and it automatically starts the GPRS connection, then RFgen only needs to start the VPN, not both.

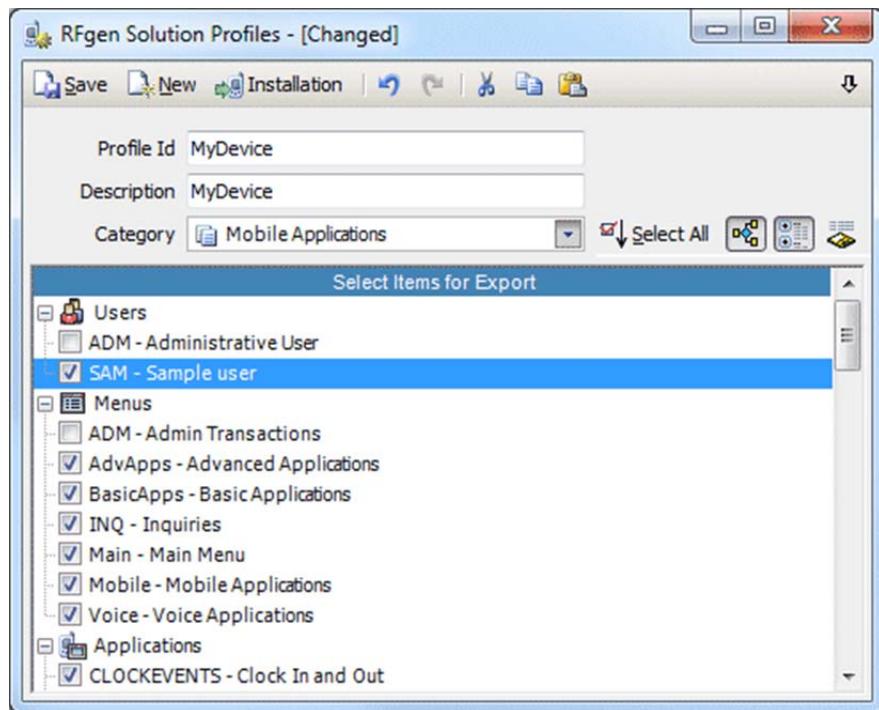
Note: The most effective way to determine what RFgen needs is to perform the connection manually, first, to see what the working device

configuration is and then, tell RFgen to start 1 or both connections by name.

GPRS Settings is either enabled or disabled. If one is configured on the device that RFgen must start itself, then fill in the Name of that connection. A **User** and **Password** may not be necessary. If RFgen must start the **VPN session** then reference it by **Name**. Again a **User** and **Password** may not be necessary.

Category: Mobile Applications

The Mobile Applications category applies only for a Mobile mode client. This window allows the selection of the users, menus, applications, VBA modules, and table, macro and business function schemas to be transferred to the mobile device to be used in a disconnected state only.



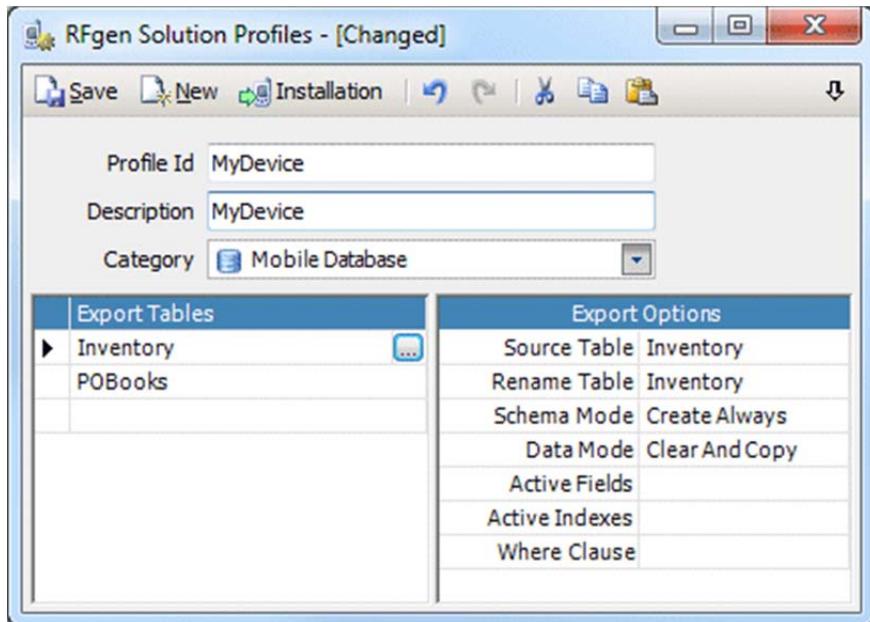
Select the required items or use the toolbar to help in the selection.



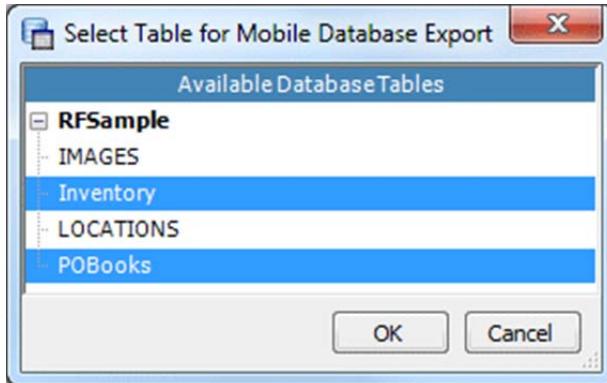
Select All obviously selects any visible item in the list. The next toolbar icon will automatically select other items that are required for the user-selected item to function. For example, selecting a user, all menus, applications, etc. will be selected automatically. The last 2 buttons toggle between displaying the whole list and only the selected items.

Category: Mobile Database

The Mobile Database category applies only for a Mobile mode client. Here, you select which database tables from any source that is necessary for the mobile applications to function.



Select the export tables necessary and then configure each table individually.



You may select multiple tables for faster entry. Click on Select Table to return to the previous screen.

The Export Options allow the user to configure each table.

The **Source Table** is the name of the table in the database referenced through the connector.

The **Rename Table** property allows the user to change the name, as it will exist on the mobile device.

Schema Mode sets how the table will be created. There are 2 options: Create Always (default) and Create on Change. Create Always means the table will be created on the mobile database and Create on Change means that the existing table schema on the mobile database will be compared to the equivalent table schema on the RFgen server. If there is a difference, the existing table schema on the mobile database will be replaced with the updated table schema from the RFgen server.

The **Data Mode** property has 3 options: Clear and Copy (default), Copy Only and Schema Only. Clear and Copy will delete the contents of the table and re-populate it with the new data from the server. Copy Only will place the data from the server on top of the existing data. Where it is the same, there is no change. Where it is different and depending on the keys, either it will overwrite data or insert new records. Schema Only will always send an empty table to the mobile database.

Active Fields are the fields that should be copied from the server database and used in the mobile database. This allows the user to have a smaller table structure, if the server's table contains fields that are not required for the mobile data collection effort. Leaving it blank assumes all the fields are necessary.

Active Indexes specifies the indexes on the mobile database for more efficient data retrieval.

The **Where Clause** specifies a subset of the data from the server's table in case the server contains more data than is necessary on the mobile database.

Category: Additional Files

Additional files allow the user to copy other miscellaneous files to a specified directory and dictate whether they are registered or installed. The Install Action options are to copy the file or to copy and install the file. The Install to Directory drop-down offers a number of places the file can be placed.

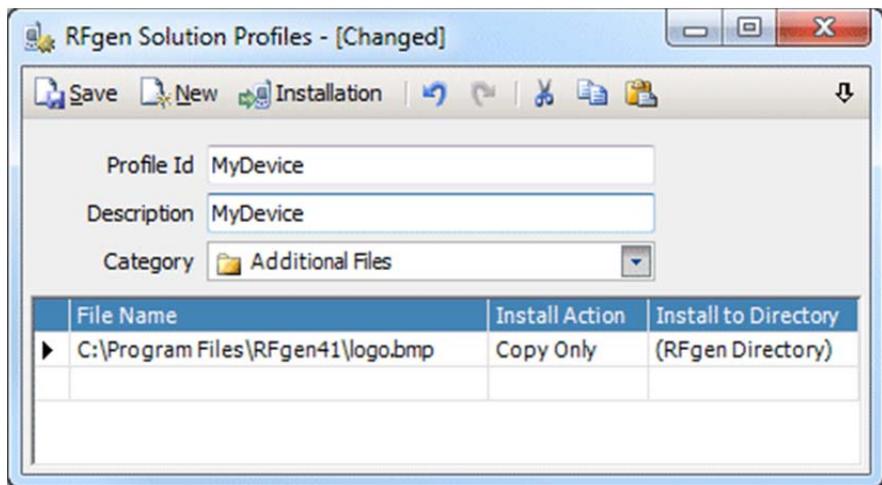
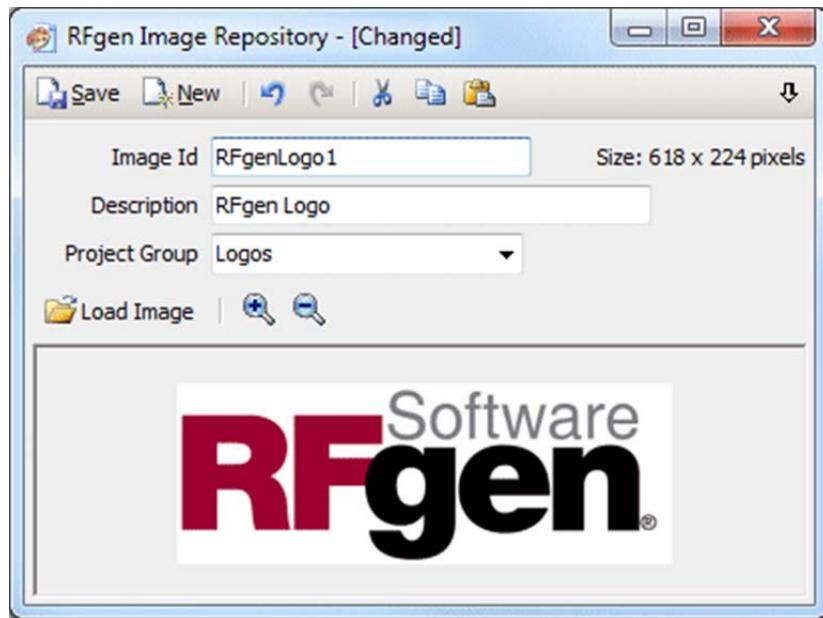


Image Resources

The Image Resource provides for a collection of pictures of different formats to be stored inside the RFgen master database. The images can be referenced by Image prompts at design-time or runtime or used as part of the RFgen configuration for mobile device backgrounds. This window allows the user to drag and drop an image for quick selection.



The **Save** option performs a save of the image to the database. The **New** option clears the picture box and provides a blank window. Also provided are **Undo**, **Redo**, **Cut**, **Copy** and **Paste**. The last icon, (the down arrow) displays on every un-docked window allowing the user to re-dock back to the Mobile Development Studio main window. Double-clicking the top-most tab will undock the window.

The **Image ID** is referenced from the GUI properties or from the code to extract this image from the master database. The **Description** of the image is also required.

The **Load Image** menu option provides a Windows file explorer to select an image.

The **Manual Zoom** begins at 100% and the magnifying glasses increase or decrease the percentage from 20% to any large value. You may also type in the value desired.

Speech Resources Tree

For users of the built-in voice product (not a Vocollect solution) the Speech Resources allow the user to customize what words are understood and how certain words should be spoken to the user.



The **Save** option performs a save of the profile to the database. The **New** option clears all fields and provides a blank window for creating a new language profile. Also provided are **Undo**, **Redo**, **Cut**, **Copy** and **Paste**. The **Dock** icon, (the down arrow) displays on every un-docked window allowing the user to re-dock back to the Mobile Development Studio main window. Double-clicking the top-most tab will undock the window.

RFgen Speech Resources - [Changed]

Save New Undo Redo Cut Copy Paste Find

Resource Id: Generic
Link To: English

Text-to-Speech Dictionary

Word	Pronunciation
▶ headset	hed set

Speech-to-Text Global Word List

Word	Substitution
▶ 0-9	
dash	-
slash	/
dot	.
Alpha	A
Bravo	B
Charlie	C
Delta	D
Echo	E
Foxtrot	F
Golf	G
Hotel	H
India	I
Juliet	J
Kilo	K
Lima	L
Mike	M
November	N
Oscar	O
Papa	P
Quebec	Q
Romeo	R
Sierra	S
Tango	T
Uniform	U
Victor	V
Whiskey	W
X-ray	X
Yankee	Y
Zulu	Z

The **Text-to-Speech Dictionary** is a list of words that are not pronounced properly by default or you want to alter what is spoken from the default. Generally, this list is not necessary. A play button is provided to test if the pronunciation change is correct.

The **Speech-to-Text Global Word List** is a list of the allowed words which also gives the user the ability to substitute what the user has spoken for something else for data input. Using a range is allowed, but RFgen does a simple ASCII conversion internally to determine all the items in the list. In this case, 0 through 9 are allowed and do not require any change. If the user speaks "dash" it will be understood but also translated. If the user speaks something not in this list, it will not be understood. It is recommended that if individual letters need to be spoken, that they be assigned words since B, C, D, E, G, 3 and others can sound very similar. Therefore, if the data being spoken was "Three Alpha Bravo" then the prompt will fill in with "3AB".

Text Resources

The Text Resources are meant to easily create a list of customizable strings that may be referenced from the code. Error messages, warnings or just general text are examples of strings that could be referenced. The key to the table is the ID of the country locale in case additional languages require the same strings but translated.

The screenshot shows a Windows-style dialog box titled "ABC RFgen Text Resources - [Changed]". The window has standard operating system controls at the top right. Below the title bar is a toolbar with icons for Save, New, Undo, Redo, Cut, Copy, Paste, and Delete. The main area contains two text fields: "Language Id" set to "1033" and "Description" set to "Messages". Below these fields is a section titled "Application String Resources" containing a table with two columns: "String Id" and "String Value". The table rows are:

String Id	String Value
ERR_QTY1	Quantity must be a number.
ERR_QTY2	Quantity must be positive.
ERR_PART	Invalid part
WARN_LOGIN1	Your password will soon expire.
WARN_LOGIN2	Password not strong enough.
TEXT_INV1	ITEM
TEXT_INV2	BATCH



The **Save** option performs a save of the text resource to the database. The **New** option clears the grid and provides a blank window for creating a new set of strings. Also provided are **Undo**, **Redo**, **Cut**, **Copy** and **Paste**. The **Dock** icon, (the down arrow) displays on every un-docked window allowing the user to re-dock back to the Mobile Development Studio main window. Double-clicking the top-most tab will undock the window.

The **Language ID** is the key to the table. You may only have one table per language so all strings should be labeled properly to allow for easy reference. The **Description** is also required.

The String Id is the key to the text and should never change between languages. This way the code can remain the same and only the Language ID changes. The String Value can be any character.

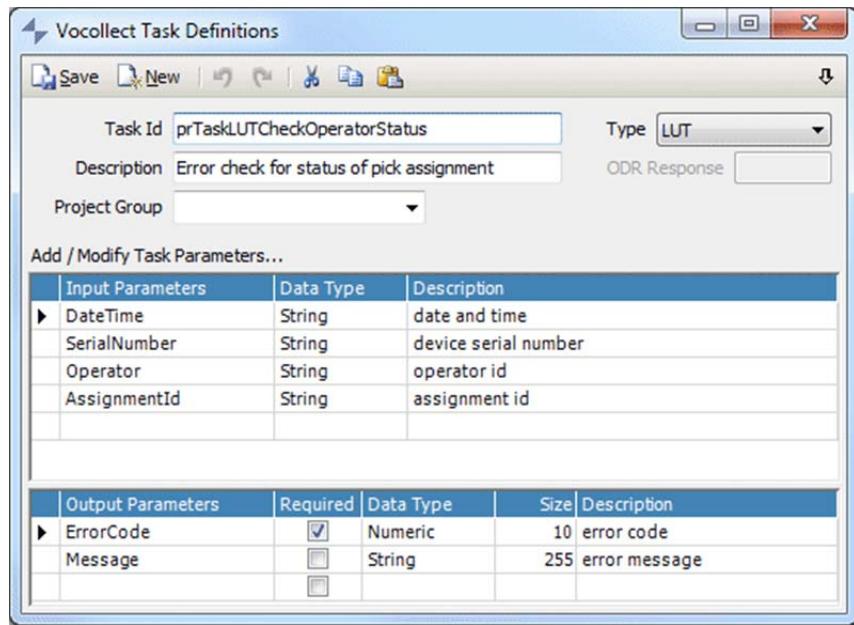
Vocollect Tasks Tree

Vocollect tasks are pre-defined scripts that execute on Vocollect hardware and interact with RFgen to provide the voice solution together with the backend data connection solution.

Either right-click on the tree or double-click an existing task to bring up the design window. Alternatively, you can use the Import utility from the RFgen System Files group and choose predefined Vocollect tasks rather than creating them manually.



The **New** option clears all fields and provides a blank window for creating a new Vocollect task. The **Save** option performs a save of the task to the database. Also provided are **Undo**, **Redo**, **Cut**, **Copy** and **Paste**. The **Dock** icon, (the down arrow) displays on every un-docked window allowing the user to re-dock back to the Mobile Development Studio main window. Double-clicking the top-most tab will undock the window.



The **Task ID** is RFgen's identifier but in the case of the pre-defined tasks Vocollect provided the names. The **Description** clarifies the task and displays in the tree. The **Type** is either LUT or ODR. LUT is two-way communication between the Vocollect device and RFgen and ODR represents one-way communication. For ODR, there are no output parameters. The **ODR Response** field can contain any value and it represents an acknowledgement bit from RFgen to make the Vocollect device stop polling RFgen for a response. Types are input only and are used for tasks such as updating a count or updating a status. The **Project Group** lets the user group like items together in the navigation tree.

Screen Mapping

The RFgen Screen Mapping module enables mobile applications to be mapped against multiple host systems like the AS/400, IBM Mainframe, UNIX systems and other character-based ‘legacy’ applications. In practice, screen mapping applications use keystrokes recorded for a host screen’s navigation and data entry, along with the collected data and play back of the keystrokes, while replacing the recorded data with the newly collected data. Accuracy in staging and applying keystrokes is of the utmost importance. RFgen’s recording capabilities provide this needed level of accuracy. RFgen’s solution provides 3 host protocols: TN5250, TN3270, and VT220 in order to interact with legacy hosts.

RFgen screen mapping applications may be created by means of an automatic recording processes, and point and click, drag and drop development methods. The automatic recording processes, a unique feature of RFgen, create Visual Basic for Applications (VBA) macros (i.e., scripts) that utilize pre-built screen mapping extensions for system navigation and data handling. An intuitive set of VBA extensions have been designed to interact with any character-based legacy application. Users may, of course, modify scripts as desired or create new scripts. **RFgen screen mapping supports transaction queuing so that when a host is offline, data collection may continue uninterrupted.** The system thus allows true 24/7 support for critical data collection operations.

How to Make RFgen Screen Mapping Work

RFgen must be properly loaded onto a Microsoft Windows-based computer system; the Screen Mapping module is included with this base system. When loaded and authorized, an RFgen-based data collection system functions (simultaneously) as both an ODBC database server and a legacy host terminal server.

To properly function with a host AS/400, IBM mainframe, UNIX, or other legacy-based system, an RFgen-based data collection server must be part of a communications network, capable of interacting with a host via TCP/IP networking protocols.

Theory of Operation

Screen Mapping works by identifying a Main Menu in the host system that is used as a base reference point for navigation to and from transaction screens. This is the starting point for transaction processes. With RFgen, the navigation process (i.e., series of keystrokes) required to proceed from a login state to the Main Menu may be automatically recorded by the system. A small visually unique portion of the main menu screen is marked for identification purposes. If required, multiple areas of the main menu may be marked. This allows RFgen to know that it has arrived at the requested destination.

The next step is the navigation from the main menu to the transaction screen. Transaction screens are the displays that users use to input data into the host system. Again, the navigation process (i.e., required keystrokes) to reach a screen and then return to the main menu may be automatically recorded by RFgen.

When a transaction screen has been identified, the next step in the process is to identify the fields on the screen where the data will be entered. Screen fields are marked and each is given a field name. These fields may optionally be used by RFgen in much the same way as ODBC fieldnames are used by the system; i.e., they may be used to create RFgen transactions by dragging the host screen's marked fields on to an RFgen application screen. Doing this, means that the user can scan and enter all the data on the mobile device and RFgen already knows how to log on to the host system, how to navigate to the proper screen and where to place the collected data on the host screen, all without having to program the scripts yourself.

Programming Philosophy

Programming Screen Mapping applications differ from typical data collection applications in that problems, if and when encountered, need to be handled automatically by the application program without the involvement of the remote data collection users. For example, the RFgenSM module contains built-in commands, such as 'SM.GetText', that can be used to search for specified text in a host screen (at a specific 'Col, Row' location, or anywhere on the screen). This, plus other diagnostics, allow programmers to positively identify the correctness of 'happenings' within a host application.

Design Considerations

Before starting a screen mapping project, users should consider certain project design issues related to the following topics: Screen Mapping Level, Logon Security, Data Integrity, Keyboard/Special Key Configuration, and Runtime Environment/Variables.

Screen Mapping Levels

RFgen divides its Screen Mapping interface into 3 levels of usage: **Low**, **Medium** and **High** levels.

Low level use is represented by scripting in the VBA environment all aspects of interaction between RFgen and the host system using the SM object's methods and procedures. These commands allow the developer complete control over the host session. Examples include sending/receiving text, control keys, cursor positioning, "WaitFor" statements, "Find" statements, etc. It is entirely possible for the user to write/program complete solutions using only these low-level commands. These commands are documented in the Screen Mapping Extensions section.

Medium level use is typified by the creation Host Screen macros and / or Data Entry macros using the recording capabilities of RFgen. At any time in the VBA script, one of these macros can be called and the host screen can be made to navigate or transact instantly. This capability simplifies the programming of the navigation requirements within a host system. Calling a transaction macro will place all collected data into the host screen's fields and submit the screen to the host for processing. Transaction macros can have input / output parameters. These parameters are used to send and receive data from the host screen. Because of RFgen's unique design, transaction data can be stored while the host is offline, and send to it for processing later when the connection is re-established.

Medium level usage, next entails the development of a VBA script to call the pre-recorded macros. One Screen Mapping command 'SM.CallMacro' is oftentimes sufficient to update the host.

High level use of RFgen's Screen Mapping capabilities is represented by the automatic recording of the 'Host Screen' and 'Transaction Macros' discussed above. Host transaction fields are then embedded in RFgen applications in much the same manner (e.g., drag and drop) as table fields from ODBC databases. Using embedded methods, data automatically posts to the host once all input fields have been entered

(note: posting was accomplished manually as the last step in Medium level usage, not automatically as with embedded fields).

A final note: All automatically recorded macros are created using base low-level commands. Thus, users have complete access to all VBA scripts, including modifying them as desired. An example of user modifications might include checking for error conditions (such as bad data) and/or warning, error, or informational messages. Copying and pasting the recorded macro script and placing it in the RFgen application directly is a quick way to build a low level solution.

Logon Security Considerations

RFgen supports a variety of ways the programmer can implement security. One important thing to remember with screen mapping is **that the end-user is never on-line with the host system**. The end-user has no way of interacting with the host system that you haven't provided for. With this in mind, the following are a few examples of login security methods:

The developer can create a "Login" Transaction Macro that is linked to the Login Host Screen. If a new user needs to sign on, this can be accomplished through a simple call to the "Login" macro.

The user ID and password specified when the user logged into RFgen could be provided to the script. The login would occur when the user called the first macro.

A generic login could be specified, and the user changed dynamically using a system function such as "sign-on" or "change-to". This command could be executed as part of a single Transaction Macro, or as a separate one called only when the user changes.

System Integrity Considerations

Screen mapping actions that cause a host system to be updated should be acknowledged by the host before another command is sent. RFgen has designed its host interface to automatically accommodate for this as much as possible. The script or macro waits for the host to not be busy before reading from or writing to the host screen. In a host-busy condition (input inhibited), RFgen will wait for the timeout period specified in the settings for the condition to clear. However, RFgen screen mapping VBA extensions should be incorporated to provide ways of acknowledging successful actions. For instance, the following commands may be used to provide programmer control over host sessions:

SM.WaitForText – This function looks for a unique text string on the screen for a specified amount of time. If the text is found, it returns True, otherwise, it times out with a value of False.

SM.WaitForCursor – This function waits until the screen input cursor stops at the desired location for a specified amount of time. If the cursor stops at the desired location, it returns True, otherwise, it times out with a value of False.

SM.WaitForScreen – This function looks for the desired screen identifier for a specified amount of time. If the application is in the correct screen, it returns True, otherwise, it times out with a value of False.

SM.WaitForHost – This function is designed for vt220 connections. As the vt220 protocol does not typically include input inhibit conditions, this function will return True once the host responds to the previous action command, otherwise, it will time out with a value of False.

SM.CallMacro – This function has a “Queue-Offline” argument, which, if enabled, will store the transaction if the host is offline. These “store-and-forward” transactions will be processed automatically once the host connection is re-established.

Note: the ‘**SM.WaitFor...**’ commands are primarily useful (and perhaps required) for VT hosts.

See Screen Mapping Extensions section for more information.

Keyboard/Special Key Configurations

RFgen understands that not all terminal emulations use the same keyboard layout. Accordingly, a developer is provided with 2 options to send special keys to the host:

The first is a pop-up window that is accessed by clicking on the ‘Hot-Key’ icon at the top of the Host Session window. You can then select the desired special key from the list in the window and transmit it to the host.

A second option is to re-map selected keys on your keyboard to transmit the special keys instead. These re-mappings are defined by clicking on the Session menu item at the top of the Host Session window. To re-map a key simply press the key(s) you want to re-map and then select the desired special key to send from the drop-down list.

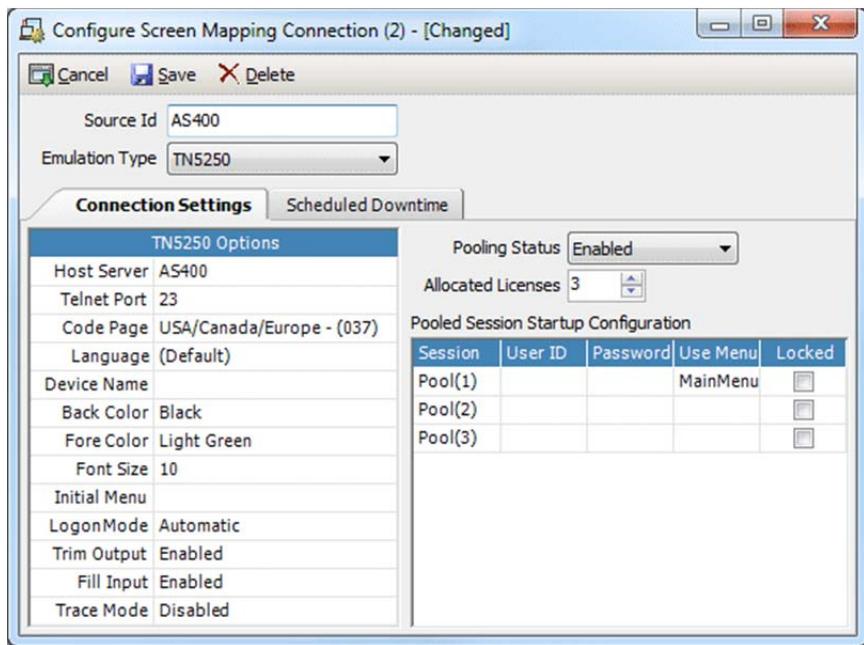
See section 4 for more information on Hot-Keys.

Runtime Environment Variables

A programmer can make use of any of the standard RFgen language extensions while creating or modifying macros. In addition, any global variables specified in the “Win32.bas” or “RFgen.bas” modules are available for use in any macro. However, the user cannot call another macro from within a macro.

Configuring the Host Connection

In the Mobile Development Studio, click the menu option Connections / Add New Data Connection / Add New Screen Mapping Connection. The following window will appear.



The first entry is the **Source Id** used to reference the data connection in RFgen only. This can have any value but spaces and extended characters are not recommended.

Choose the **Emulation Type** (VT220, TN5250 or TN3270); i.e., the protocol used to communicate with your host system. Notice that there is an additional option called Console Application. This type is designed to launch a console application rather than use a telnet server and then pass that display through RFgen to the device using the HostScreen

prompt control. One example would be the SAP console application (SAPCNSL.EXE) running on the server and being displayed and allowing interaction with the user on a mobile device. Simply specify a process or executable name to run and any passing parameters necessary.

Next, type in the **Host Server** name or IP address. The **Telnet Port** is the port that RFgen uses to communicate with your host. The default for a telnet server is port 23.

If TN5250 or TN3270 are selected, you may enter a **Code Page** for specifying the language being used in the protocol and an **IBM Device Name** for the host system. Code pages were selected for loading when you loaded the screen mapping software. These fields are hidden in the VT setup.

The Device Name field is designed to make each connected device appear unique to the host system. Fill this field in with a name and 2 or 3 zeros. When RFgen connects, RFgen will replace the 00 or 000 with a zero filled right-justified number. So pool entry 3 would report a device name of either RFGEN03 or RFGEN003 based upon the format. If there are no trailing zeros, then the device name will be used as it is during the IBM report environment request.

For VT220 the **Data Stream** field can be set to either Legacy or UTF-8 to accommodate the type of packet data coming from the host system. The preferred option is UTF-8 but if a legacy system's output is language specific then the **Language** field should be changed to make the screen render correctly. The Language field can be left as "default" if a code page is specified or if UTF-8 is used.

Preferences for the emulation screen include **Back Color**, **Fore Color** and **Font Size**. These are only for development since the screens are hidden during production.

The **Initial Menu** is the Hosts macro to be used to log the host system in. This is typically the launching point for all navigation to host transactions.

The **Login Mode** can be either Automatic or Manual. Automatic means that the defaults will be used and when the session is started, the default user, password and main menu will be used to log in. Manual means that the session will be started and the script must pass the user, password, and navigation for the main menu.

Trim Output set to enabled will auto trim spaces from the host output fields. If a variable is defined for a section of the host screen (like where error messages are displayed), this feature will trim the text for easier use in message boxes, for example.

The **Fill Input** option when enabled will use spaces to pad any input. A variable defined for a region of the host screen where input will take place also has a length property assigned at the time the field was defined in RFgen. If the data is 3 characters, but is placed in a host screen field designed for 10 maximum characters, RFgen can pad the input data to fill up the host screen input field.

Trace Mode enables a screen mapping diagnostic trace that RFgen technical support can use to debug packet traffic between the host and RFgen.

Connection Pooling can be enabled and the maximum connections allowed in the pool can be selected. This selection will determine how RFgen and its clients will interact with your host system.

The options for the Pooling Status are:

Disabled – Setting connection pooling to disabled will cause RFgen to spawn a connection to the host system for each active mobile device. Each connection will be linked to a particular device on a one-to-one basis, and will be shut down when that device disconnects. Note: there is no limitation on the number of connections allowed.

Enabled – Setting connection pooling to Enabled will cause RFgen to spawn a single connection to your host system. As each device requires access to the host system, they will go to the pool and retrieve 1 of the available connections. When they are finished, the device will release the connection back to the pool. If no connections are available, RFgen will start a new connection (up to the specified maximum) and add it to the pool. After 10 minutes of non-use, an opened pooled connection will be terminated releasing resources on the server and potentially licenses on the host system. Keep in mind that unless the SM.BeginTrans and SM.CommitTrans commands are used, it would be possible for one user to position the screen in one place while another user also uses that pooled connection to perform their tasks causing both users to get failures.

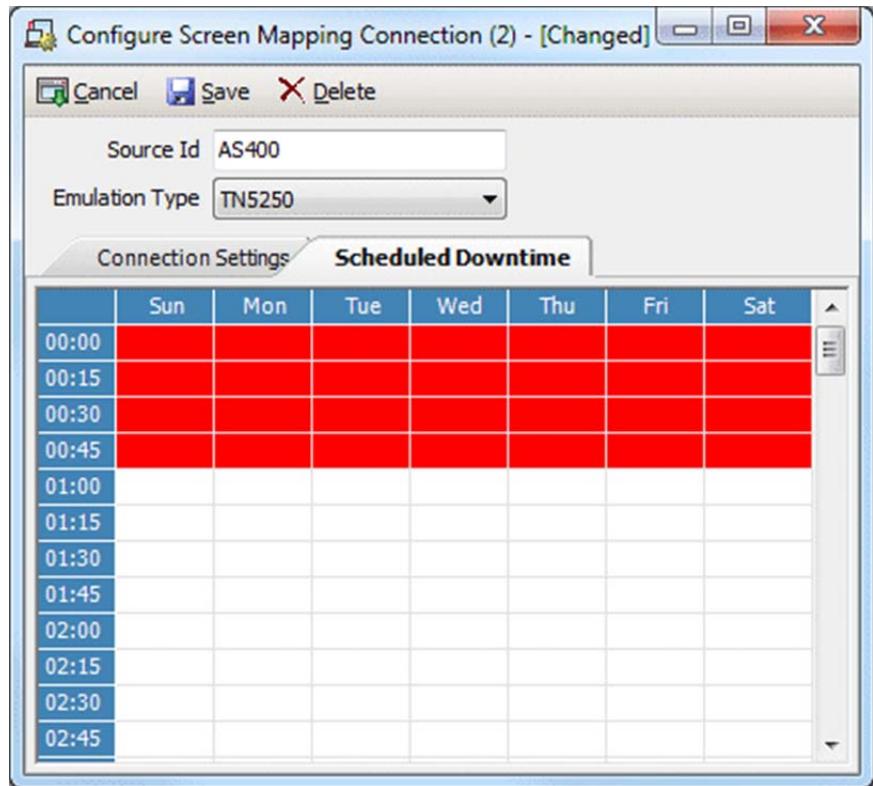
The Pooled Session Startup Configuration options dictate how each host session is started. You may also override the default settings by configuring a specific pooled session separately.

Session	Each of the individual pooled connections are listed separately. This provides for specific settings for each connection.
User Id	If the host system requires that unique names be used or creating multiple logins with the same user is prevented, each pooled connection can have its own user ID. Session, user, and password information can be obtained at runtime with the commands SM.SessionUser, SM.SessionPwd, and SM.SessionID.
Password	This is the corresponding password used for each unique user ID.
Use Menu	Each session can have its own main menu. When a session is requested and no main menu is specifically assigned or the "(Default)" value is used, the next available session will execute the requested main menu based on the scripts and chosen transaction. If a session is requested and the next available session does have a main menu assigned, and it is not the required one, other sessions will be evaluated for a matching main menu. If one is found and available, it will be used.
Locked	<p>The ability to lock a session means that the session can ONLY be used with the specified main menu and will not allow other main menus, even if all other available sessions are in use. For example, there are 10 pooled sessions, 5 locked on main menu A and 5 locked on main menu B. If a session with main menu A is requested and all 5 sessions for main menu A are currently used, RFgen will look to the sessions assigned to main menu B. If they are not locked, RFgen will take 1 of them. Since they are locked into main menu B, in use or otherwise, RFgen will wait for 1 of the first 5 to be released.</p> <p>The purpose of locking a set number of sessions to a specific main menu is to ensure there is always some bandwidth available for certain transactions. Not locking them means that they will be marked with a preference for a type of transaction (the use of a specific main menu), but will switch to another main menu when necessary.</p>

For example, there are 10 pooled sessions available and the first 5 have 1 main menu assigned and the last 5 have a second main menu assigned. When a session with the second main menu is requested, the 6th session handle will be used. This is only significant because of the Locked property.

Scheduled Downtime

The Scheduled Downtime option in the ‘Screen Mapping Connection’ Window allows users to schedule ‘down time’ for a host connection; i.e., **RFgen will disable the connection** during the time that a host is offline. The following panel will appear.



As shown, the connection is disabled ‘Every Day’ from 12:00 AM to 1:00 AM. (Time is shown in a 24-hour notation.)

During the down time, transactions can be ‘queued’ for automatic posting to your host when the connection becomes available.

When you ‘Save’ the screen mapping configuration entries, a session with your host should be available by right-clicking on the screen mapping connector and selecting Show Host Connection and the host name will appear as a ‘connection indicator’ at the bottom of the Mobile Development Studio window. A red circle in the connection indicator...

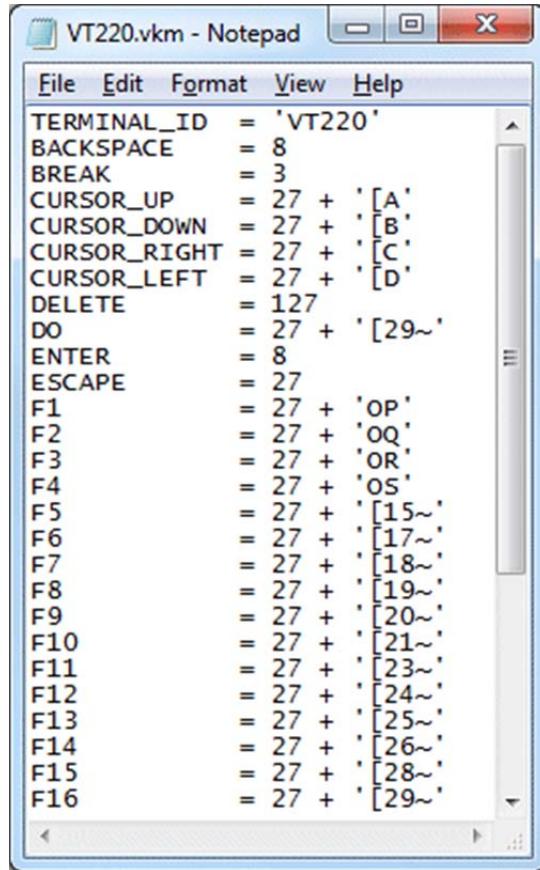


indicates that a connection has not been established with your identified host.

With a host connection established, your screen mapping development project is ready to be started.

VT220 Key Mapping

The default key mapping for VT sessions can be edited if certain keys are not working correctly with the host. Use the Import utility from the menu, choose RFgen System Files, and import the VT220 Key Map Template. A file called VT220.VKM will be placed in the RFgen directory and can be edited with a text editor.

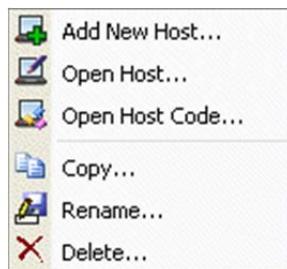


The screenshot shows a Windows Notepad window titled "VT220.vkm - Notepad". The window contains a list of macro definitions for a VT220 terminal. The macros are listed in pairs, where each macro name is followed by its definition. The definitions involve the character code '27' followed by various control codes and sequences. The scroll bar on the right indicates there are more lines of text.

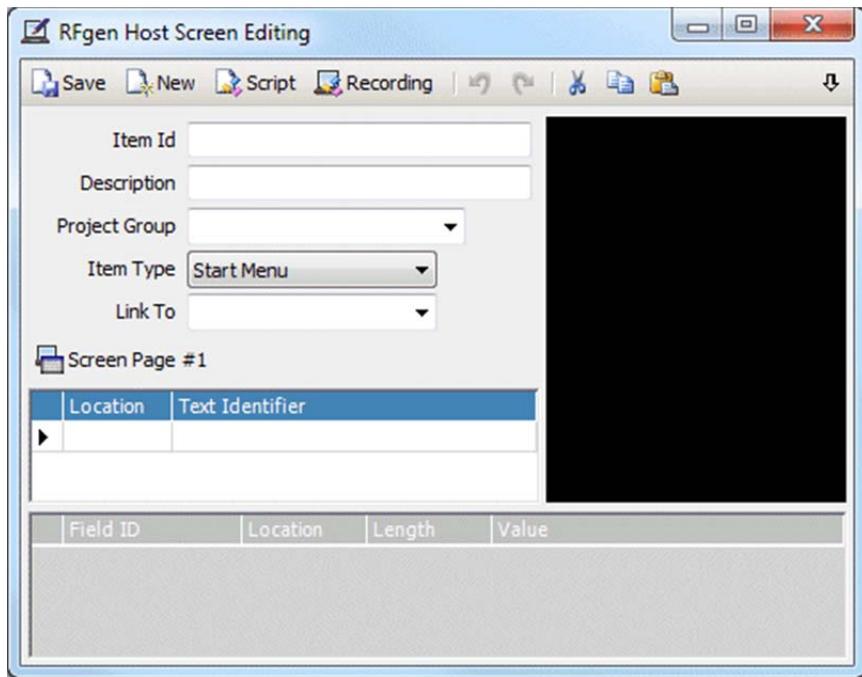
Macro	Definition
TERMINAL_ID	= 'VT220'
BACKSPACE	= 8
BREAK	= 3
CURSOR_UP	= 27 + '[A'
CURSOR_DOWN	= 27 + '[B'
CURSOR_RIGHT	= 27 + '[C'
CURSOR_LEFT	= 27 + '[D'
DELETE	= 127
DO	= 27 + '[29~'
ENTER	= 8
ESCAPE	= 27
F1	= 27 + 'OP'
F2	= 27 + 'OQ'
F3	= 27 + 'OR'
F4	= 27 + 'OS'
F5	= 27 + '[15~'
F6	= 27 + '[17~'
F7	= 27 + '[18~'
F8	= 27 + '[19~'
F9	= 27 + '[20~'
F10	= 27 + '[21~'
F11	= 27 + '[23~'
F12	= 27 + '[24~'
F13	= 27 + '[25~'
F14	= 27 + '[26~'
F15	= 27 + '[28~'
F16	= 27 + '[29~'

Hosts Tree

Double-click on the Hosts heading to create a new screen mapping macro or right-click on the title of an existing macro to make additional changes.



Double-clicking on an existing macro opens the Host Screen Editing window.



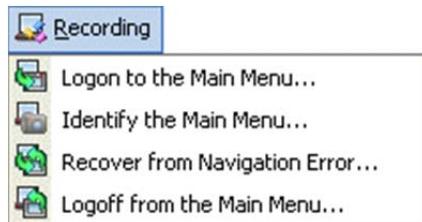
There are 2 types of macros that can be created here. **Start Menu** macros take the data connection as it is when first connected and logs in and navigates to a main menu used as a generic starting point for all screen mapping transactions. **Host Screen** macros are used when a specific transaction is chosen by the mobile user to navigate the host system to the proper screen meant to accept specific data (ex.: Cycle Count screen). Additionally, this macro can be used to play back the keystrokes of a user entering the collected data into the screen itself. This macro stores the x,y coordinates of the fields on this host screen and places the collected data in the proper places before sending additional keystrokes to submit the data (ex.: F8). At that time the host screen processes the data just as if the user entered the data directly to the host screen.

Building a Start Menu Macro

Begin by entering the ItemID of the Start Menu macro. This will be the name given to the primary main menu used to log the host system in.

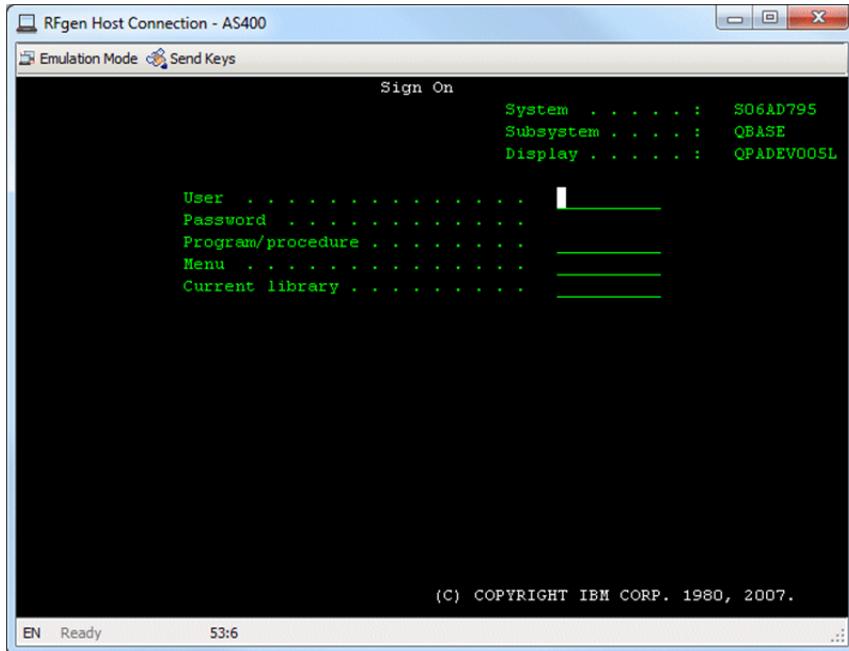
Fill in the Description field, select Start Menu for the Item Type and Link To the name of the screen mapping data connection.

Next select the Recording menu option.



There are 4 different scripts that this macro can contain. **Logon to the Main Menu** records the keystrokes to successfully log in to the host system and navigate to the main menu. **Identify the Main Menu** records the x,y coordinates of some text on the host screen, so when RFgen attempts to reach this page it will compare the host screen to what is known to be the proper screen. **Recover from Navigation Error** records the keystrokes to do whatever the user must to get the host back to the main menu. Usually the safest solution is to back out as far as will ever be needed and then log in again. **Logoff from the Main Menu** contains the keystrokes recorded for exiting the main menu and going back to the login screen.

When recording these macros the Host Session window will appear. In the case of this host system, it is an IBM AS/400 connected to RFgen using a TN5250 telnet protocol.



The first toolbar icon represents the mode the window is in. Examples are Emulation Mode, Recording Mode and Identifying Mode.

The Send Keys Selection

Selecting a 'Hot Key' from the available list is an alternative to re-mapping your keyboard. The following list of keys will appear.



Clicking on a menu item will send the selected key to your host. If you are currently recording a macro, the selected key will become part of that macro.

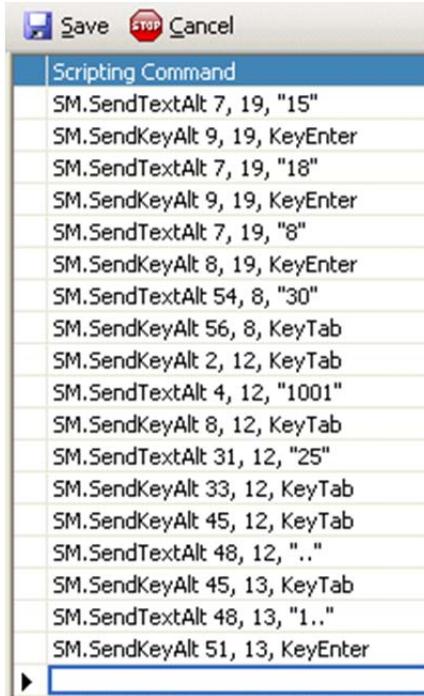
Start Menu Macro – Logon to the Main Menu

Choose the Recording menu option and select Logon to the Main Menu. The RFgen Host Session window will display with a column on the right for recording and editing all keystrokes.

Enter the keystrokes necessary to display the main menu. During the recording phase the user may right-click on the host system to bring up a menu of additional commands that may be inserted into the script. See the Recording Options section below for a complete description of these commands.

Add FindText Statement
Add GetArea Statement
Add GetCursor Statement
Add GetText Statement
Add WaitForCursor Statement
Add WaitForCursorMove Statement
Add WaitForHost Statement
Add WaitForText Statement
Add WaitForWrite Statement
Select Current Field

The Scripting Commands column on the right can also be edited in case a mistake is made, variables need changing or timeout values need to be adjusted. If a step is forgotten, such as waiting for text to appear on the screen before performing the next keystroke, simply position the insert arrow on the row to be preceded by the missing command and perform that command. In this case, highlight a section of text, right-click and select the Add WaitForText Statement option. If a keystroke was pressed accidentally, simply delete it from the list. Be sure to put the insert arrow back at the bottom before continuing.



The screenshot shows a window titled "Scripting Command" with a toolbar at the top featuring "Save" and "Cancel" buttons. The main area contains a list of recorded macro steps, each consisting of a command name followed by parameters. The steps include various key presses and text sends, such as "SM.SendTextAlt 7, 19, "15"" and "SM.SendKeyAlt 9, 19, KeyEnter". The list ends with "SM.SendKeyAlt 51, 13, KeyEnter".

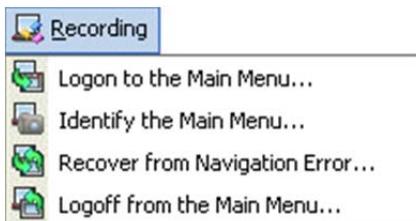
```
Scripting Command
SM.SendTextAlt 7, 19, "15"
SM.SendKeyAlt 9, 19, KeyEnter
SM.SendTextAlt 7, 19, "18"
SM.SendKeyAlt 9, 19, KeyEnter
SM.SendTextAlt 7, 19, "8"
SM.SendKeyAlt 8, 19, KeyEnter
SM.SendTextAlt 54, 8, "30"
SM.SendKeyAlt 56, 8, KeyTab
SM.SendKeyAlt 2, 12, KeyTab
SM.SendTextAlt 4, 12, "1001"
SM.SendKeyAlt 8, 12, KeyTab
SM.SendTextAlt 31, 12, "25"
SM.SendKeyAlt 33, 12, KeyTab
SM.SendKeyAlt 45, 12, KeyTab
SM.SendTextAlt 48, 12, ".."
SM.SendKeyAlt 45, 13, KeyTab
SM.SendTextAlt 48, 13, "1.."
SM.SendKeyAlt 51, 13, KeyEnter
```

When completed, click Save. Cancel may be clicked at any time to cancel the recording session. An internal macro called **LogOnToMainMenu** is recorded by this step.

Clicking on the Script menu option will show the code generated and allow the user to test and change it.

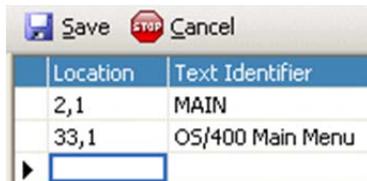
Start Menu Macro – Identify the Main Menu

In order for RFgen to positively identify the main menu, a portion of the screen needs to be 'marked'. From the Recording menu option choose Identify the Main Menu.



Next, select unique text for this screen on the host system by left-clicking and dragging across the text and then select the Mark Field menu option. If necessary, multiple selections can be made.

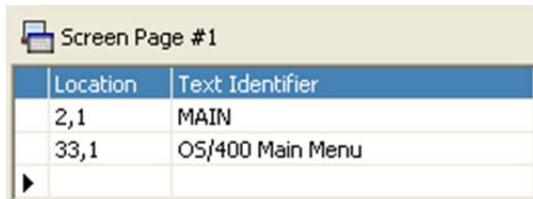
The marked region and its coordinates are placed in a grid on the right side of the host screen window where the whole list is captured and can be edited.



A screenshot of the RFgen software interface. At the top, there is a toolbar with icons for Save (floppy disk), Stop (red octagon), and Cancel. Below the toolbar is a table titled "Location" and "Text Identifier". The table contains two rows of data:

Location	Text Identifier
2,1	MAIN
33,1	OS/400 Main Menu

When complete, click 'Save' to save all marked areas. These identifiers will appear in the Host Screen Editing window.



A screenshot of the Host Screen Editing window. At the top, there is a toolbar with icons for Save (floppy disk) and Screen Page #1. Below the toolbar is a table titled "Location" and "Text Identifier". The table contains two rows of data:

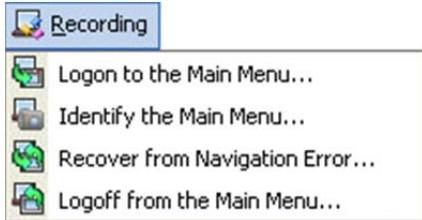
Location	Text Identifier
2,1	MAIN
33,1	OS/400 Main Menu

The Screen Page #1 menu button allows the user to record additional pages if RFgen needs to locate fields that do not appear on the first page. This concept however does not apply to Start Menu macros and will be discussed later.

Start Menu Macro – Recover from Navigation Error

This step records the keystrokes that will navigate the host system out of any possible screen or menu back to the known main menu. If the host system pops up additional screens like system messages that the RFgen scripts do not take into account, then RFgen would notice that none of the recorded identifiers match were the host is and run this script.

From the Recording menu option choose Recover from Navigation Error.



Enter the keystrokes necessary to get back to the main menu. Possibly, an easier solution would be to type in the SM.ResetConnection command as shown.



When the system resets, it automatically re-runs the LogOnToMainMenu macro taking the host to the main menu. In this case, the complete **AbortNavigation** macro would look like this:

```
Private Function AbortNavigation() As Boolean
    On Error Resume Next
    '
    SM.ResetConnection
    '
    AbortNavigation = SM.WaitForScreen("Base", 10)
End Function
```

Be sure the host system is not adversely impacted by using the SM.ResetConnection command.

Start Menu Macro – Logoff from the Main Menu

Choose the Recording menu option and select Logoff from the Main Menu. The RFgen Host Session window will display with a column on the right for recording and editing all keystrokes.

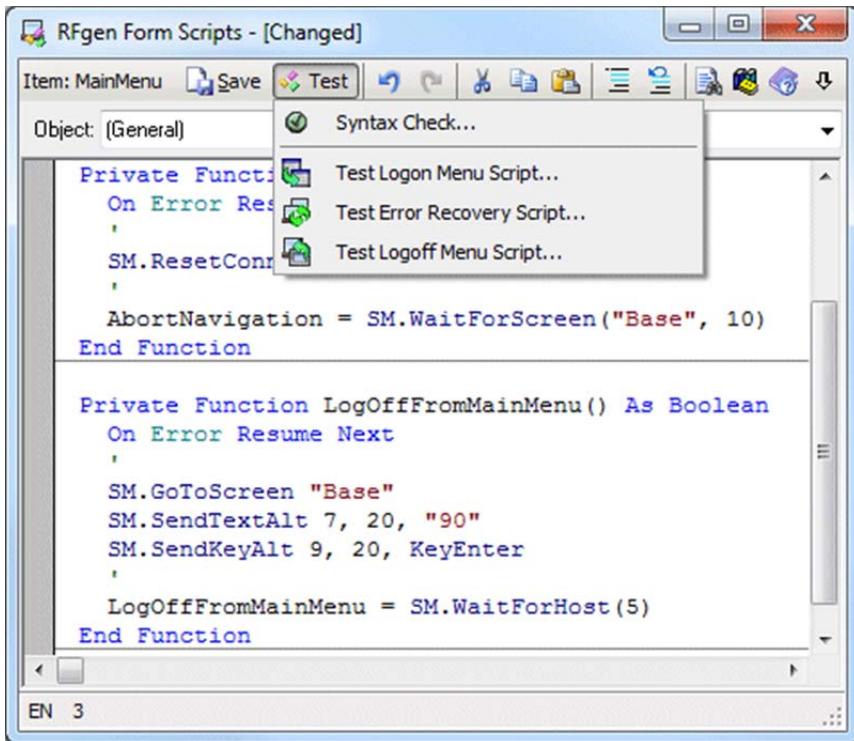
Enter the keystrokes necessary to logoff from the main menu. During the recording phase, the user may right-click on the host system to bring up a menu of additional commands that may be inserted into the script. This was described in the first step.

When completed, click Save. Cancel may be clicked at any time to cancel the recording session. An internal macro called **LogOffFromMainMenu** is recorded by this step.

Clicking on the Script menu option will show the code generated and allow the user to test and change it.

Start Menu Macro – Test Scripts

After the first 4 steps have been completed, the recorded macros should be tested. Click on the Scripts menu option and the script window will appear.



The drop-down menu from the 'Test' button allows you to select the macro to be tested or to syntax check all the scripts. A message box will appear showing the success or failure of the macro.

Important:

1. To test the **LogOnToMainMenu** macro, your host screen should first be positioned at your login screen.
2. To test the **LogOffFromMainMenu** macro, your host screen should be positioned at your Main menu.

3. To test the **AbortNavigation** macro, your host screen should be positioned anywhere in the menu structure but not on an application screen.

Be sure to save all the work that has been done before moving on to the Host Screen macro recording steps.

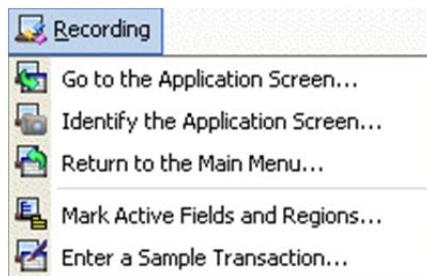
Building a Host Screen Macro

The Host Screen macro has a dual purpose. This macro can contain just the navigation to and from a transaction screen where the Transaction macro is linked to this macro or it can be used for Screen Scraping and Screen Mapping. In the first case, this method was how prior releases of RFgen implemented screen mapping, keeping the navigation separate from the transaction macro that did the data entry. Now the Host Screen macro can contain its own navigation.

Screen Scraping is where the identified fields on the host screen are directly placed on the mobile device form and as the user interacts with the fields on the mobile device, those fields are manipulated also on the host system in real time. This does require a thin client connection for the real time interaction.

Begin by entering the ItemID of the Host Screen macro. Fill in the Description field, select Host Screen for the Item Type and Link To the name of the start menu just defined.

Next select the Recording menu option.



There are 5 different scripts that this macro can contain. **Go to the Application Screen** records the keystrokes to successfully navigate the host system to the particular transaction screen from the already defined main menu. **Identify the Application Screen** records the x,y coordinates of some text on the host screen so when RFgen attempts to

reach this page, it will compare the host screen to what is known to be the proper screen. **Return to the Main Menu** is the keystrokes recorded for exiting the transaction screen and going back to the main menu. These 3 items are grouped because traditionally the Host Screen macros only provided the navigation to and from the transaction screen that is used to submit a transaction. **Mark Active Fields and Regions** is used for identifying the important areas of the host screen that will be used by the screen mapping and by the screen scraping process. The last option to **Enter a Sample Transaction**, is used by the screen mapping process to know how to use and submit the data to the host screen. The screen scraping process does not require this step as the user is interacting with the host screen in a one-to-one basis.

Host Screen Macro – Go to the Application Screen

Choose the Recording menu option and select Go to the Application Screen. The RFgen Host Session window will display with a column on the right for recording and editing all keystrokes.

Before proceeding, please ensure that you are on the host's Main menu (as previously identified).

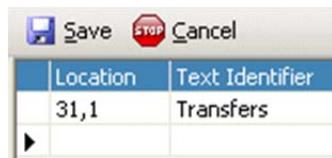
Enter the keystrokes necessary to navigate to the transaction screen. During the recording phase, the user may right-click on the host system to bring up a menu of additional commands that may be inserted into the script. When completed, click 'Save'. 'Cancel' may be clicked at any time to cancel the recording session. An internal macro called **GoToScreen** is recorded for the screen by this step.

Host Screen Macro – Identify the Application Screen

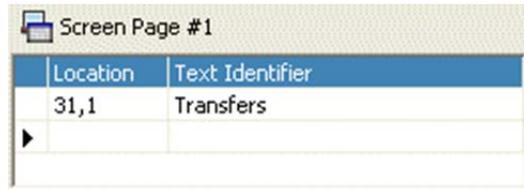
In order for RFgen to positively identify the application screen a portion of the screen needs to be 'marked'. From the Recording menu option, choose Identify the Application Screen.

Next, select unique text for this screen on the host system by left-clicking and dragging across the text and then select the Mark Field menu option. If necessary, multiple selections can be made.

The marked region and its coordinates are placed in the grid on the right side of the host screen window where the whole list is captured and can be edited.



When complete, click ‘Save’ to save all marked areas. These identifiers will appear in the Host Screen Editing window.



Location	Text Identifier
31,1	Transfers

The Screen Page #1 menu button allows the user to record additional pages and their identifiers if RFgen needs to know that the secondary pages are expected.

Host Screen Macro – Return to the Main Menu

From the Recording menu option choose Return to the Main Menu. The RFgen Host Session window will display with a column on the right for recording and editing all keystrokes.

Enter the keystrokes necessary to return to the Main menu. When completed, click ‘Save’. ‘Cancel’ may be clicked at any time to cancel the recording session. An internal macro called **ReturnToMainMenu** is recorded for the application screen by this step.

Host Screen Macro – Mark Active Fields and Regions

For both screen scraping and screen mapping applications, select the fields on the host screen by left-clicking and dragging across each field and then select the Mark Field menu option. Alternatively, simply left-click once on the desired field and then right-click and choose Mark Field. This will highlight the whole field automatically and record its coordinates if the telnet emulation supports it, such as tn5250. Optionally, choose the Mark All menu option which will only mark all input fields automatically. For screen scraping applications, regions of the screen that will display output like error messages can be selected as well. These regions can later be placed on the mobile device screen and the user will see updates on the mobile device at the same time the host updates these regions.

Field Id	Location	Length	Value
FPlant	21,8	12	
TPlant	54,8	12	
Item	4,12	26	
► Field4	31,12	13	
Field5	48,12	20	
Field6	48,17	20	

When complete, click 'Save' to save all marked areas. These fields will appear in the Host Screen Editing window and their names can be changed to reflect the type of data that belongs in each field. The names could also be changed during the recording, if it would be easier to keep track which field represented which data value.

Host Screen Macro – Enter a Sample Transaction

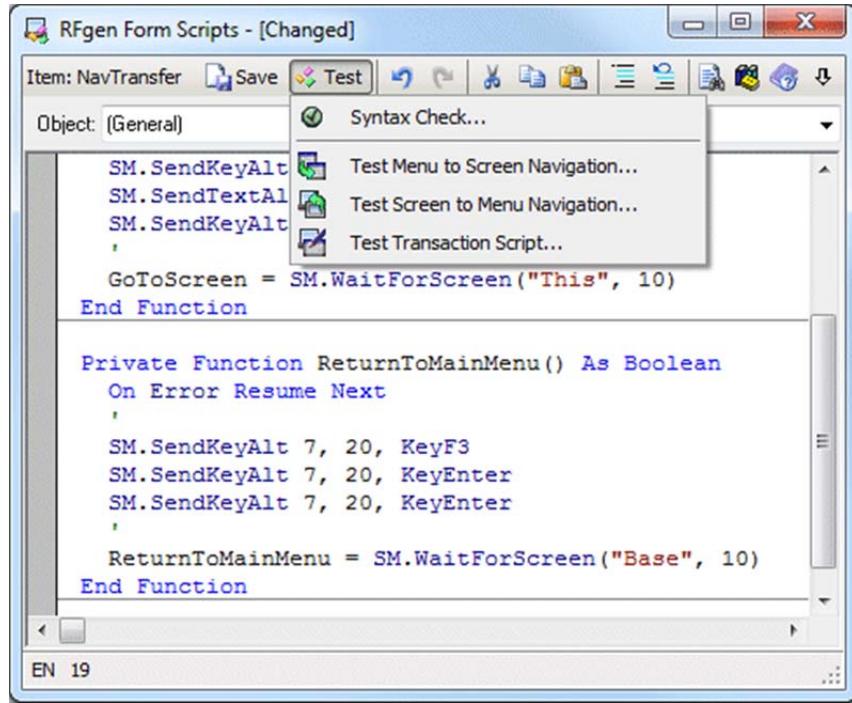
The sample transaction recording is for screen mapping only. Since the user will not be interacting with the host screen directly, a function will be generated with the marked fields as passing parameters and the keystrokes used to submit a sample transaction will tell RFgen how to use the passed in data.

From the Recording menu option choose Enter a Sample Transaction. The RFgen Host Session window will display with a column on the right for recording and editing all keystrokes. Navigate around the screen as necessary entering known good test data. Complete the transaction just as a user would and click Save. An internal function called **Transaction** will be created for this application screen by this step.

Be sure to leave the host screen in the same state as before the keystrokes were recorded. If the macro should be run more than once, then a consistent starting point is desired.

Host Screen Macro – Test Scripts

After each step is completed, or at the end of all steps, the recorded macros should be tested. Click on the Scripts menu option and the script window will appear.



The drop-down menu from the ‘Test’ button allows you to select the macro to be tested or to syntax check all of the scripts. A message box will appear showing the success or failure of the macro.

Important:

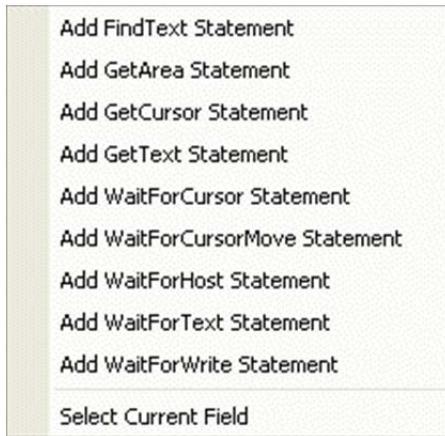
1. To test the **GoToScreen** macro, your host screen should first be positioned at your main menu.
2. To test the **Transaction Script** macro, your host screen should be positioned at your transaction screen. This macro was created when recording the Enter a Sample Transaction option.
3. To test the **ReturnToMainMenu** macro, your host screen should be positioned at your transaction screen.

Many keystrokes may have been used to navigate between screens or around the transaction screen itself that may not be necessary. By default, the text written to the screen uses the coordinates to locate the proper input field so any additional tabs, for example, to move between fields are not necessary and can be deleted. The reserved word “This” and “Base” are internally substituted at runtime depending on what

names were given to the transaction macro and the main menu macro. Be sure to save all the work before exiting.

Recording Options

While recording a host macro, you can highlight a region or field with the mouse and right-click on the highlighted area. A popup menu appears with the following selections. These options are used to add control statements to the current macro during the recording process.



Add FindText. SM.FindText is used to determine if a specified text string is currently displayed on the host screen. It can be used to look for the text in a specific screen location or to search the entire host screen for the text. See Screen Mapping Extensions for details.

Add GetArea. This command gets the text off the screen in any rectangular area. With the option to trim the result, selecting a column of data from the screen, parsing it and using it is much easier than getting all the data with several SM.GetText commands.

Add GetCursor. Used to determine where the cursor is currently located on the host screen. See Screen Mapping Extensions for details.

Add GetText. This selection adds SM.GetText which is used to retrieve text from the host screen at the column/row position established by the area highlighted by the mouse. See Screen Mapping Extensions for details.

Add WaitForCursor. This does the same as SM.WaitForText, only with the cursor; i.e., the script waits until the cursor reaches the specified

location, before executing the next statement. See Screen Mapping Extensions for details.

Add WaitForCursorMove. This function is also used to time your commands to the host session. With this command, you specify only an amount of time in seconds. If the cursor has changed positions within that time, a True result is returned. Otherwise, it will timeout and return False. See Screen Mapping Extensions for details.

Add WaitForHost. SM.WaitForHost is used to time your commands to a vt220 host session. With it, you can delay sending text or keys to the host session, or retrieving data from the host session until the host has responded to the last command sent. See Screen Mapping Extensions for details.

Add WaitForText. Adds an SM.WaitForText statement to the macro, with the column/row position and the length being established by the area highlighted by the mouse, i.e., the script waits until the text appears at the specified location. See Screen Mapping Extensions for details.

Add WaitForWrite. This function waits for a specified number of seconds for data to be entered at a specific location and returns a True or False. If data was written within the wait time, True is returned. If the number of seconds expires first, False is returned. See Screen Mapping Extensions for details.

Select Current Field. Selects the current input field and records its attributes.

Building a Screen Mapping Application

Once the recorded macros are built, there are 2 ways they can be implemented.

Create an application with data fields, collect and validate the data and use the Embedded Procedure object to pass that data to the Host Screen macro. (In the code window there is a right-click option to insert embedded code. Select Transaction Macros as the data source and then pick the appropriate Host Screen macro. See the Embedded Procedure section for more details.) The macro passes back either a True or False (based on setting the function name “Transaction” equal to one value), indicating the success or failure of the macro to complete its script. Alter the Host Screen macro’s script to send back an appropriate Boolean value. This method isolates the code responsible

for interacting with the host system which is ideal for version control and frequent updates to application code unrelated to the host system.

Take the code generated in the macro and place it in the application itself. This gives the application total control and has another advantage. If the login screen is a host screen that must allow different user credentials, then the solution cannot rely on the automatic logging in and navigation to the main menu. An application can be created that collects the user credentials and screen maps the data to the host login screen. Having already recorded the macro, that code can be placed in the application directly and the macro may be deleted to avoid confusion. Taking this path requires that all applications be responsible for navigating and populating screens and fields on the host because the host is not automatically taken to a main menu, a generic starting point for the Host Screen macros. Since the Host Screen macros are linked to a Start Menu macro and they have been replaced by an application that performs that task after a custom login, Host Screen macros will not work. The solution is to create a Transaction macro with the same code, since it does not rely on links to previous macros.

Building a Screen Scraping Application

After creating a Host Screen macro complete with the identified fields, start a new application and in the Link To drop-down select the Host Screen macro. On the Controls tab additional fields will be listed. These are all the marked fields from the macro. Double-click or drag them on to the form designer. If other standard controls are necessary, add them as well. When running this form data entered into the text boxes will be also entered directly on the host screen at the same time.

The concept of scraping a host screen is to place on the mobile device only the important fields from the host screen but to interact with the host as if using it directly.

The Mobile Enterprise Application Server

When started, the Server enables multiple communication sessions between your RFgen-based computer system and your remote devices (up to the number of authorized users for your particular installation). For telnet clients, if other telnet servers are activated on your RFgen-based system, then they must be deactivated, or RFgen must be reconfigured to listen on a port other than the standard telnet port 23. The menu bar 'Options / TCP/IP Services' may be used to specify a different port. The Server services all Telnet and GUI-based devices simultaneously.

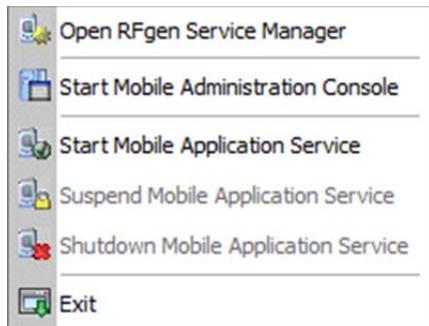
The Service Control Manager (SCM) is the graphical user interface to the service that administers the remote devices. It allows you to:

1. Start and Stop Server programs
2. Authorize permanent usage of the RFgen system
3. Configure the system for running as a Windows task or as a service
4. Start or stop the Console



← Note the SCM icon in your Windows system tray

After installation, each time you boot your system, the Server 'Service Control Manager' icon (shown above) will appear on your windows task bar. Right-click on the SCM icon, the following service control menu will appear.



Here, you may start the Console and Start, Suspend or Shutdown the Server. If you start the Console, and the Server is not active, the system will activate it automatically. If you stop the Server, the Console will automatically be stopped.

In order to start and stop the Server from the command line, navigate to the RFgen directory using the DOS 'dir' and 'cd' commands. Then use the following commands:

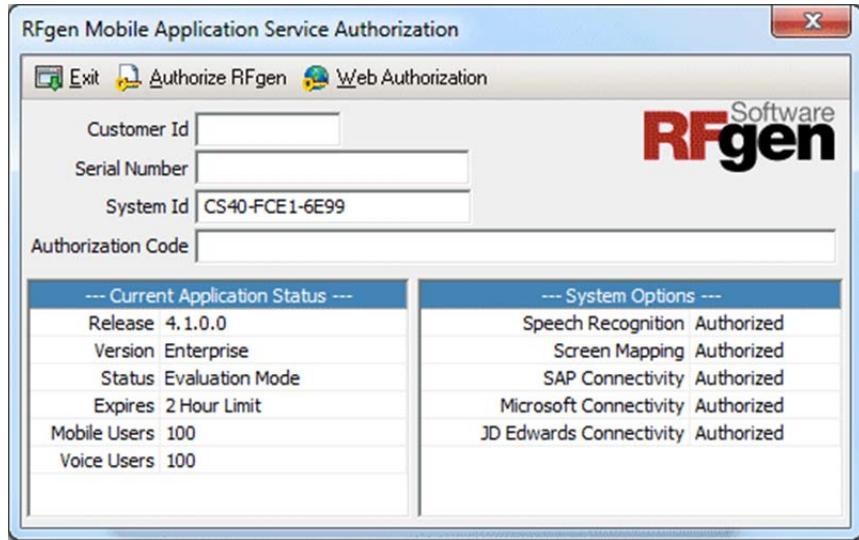
```
rfscm41w -startup  
rfscm41w -shutdown
```

When you click 'Open' the RFgen Service Manager (or double left-click the SCM icon), the Configure window shown below will appear.



To Permanently Activate the Server

The Server will not run longer than 2 hours without a permanent authorization code. To test longer than 2 hours, the RFgen service will need to be stopped and restarted. To authorize the Server for permanent operation, click on the Authorizations menu item. A window similar to the following will appear.



Enter the Customer ID, Serial Number and click the Web Authorization. Depending on System ID being generated by the software and previous System IDs having already been authorized, there is a chance a call must be made to DataMAX to get an authorization code. This is intentional. The Authorization Code is unique to this RFgen installation. Licensed users may obtain an authorization code by calling or emailing the DataMAX Software Group, Inc., RFgen support department (telephone: 916-939-4065 or email: 'support@rfgen.com'). **The 'System ID:' displayed in the window must be transmitted to DataMAX.** Enter the authorization code and click the Authorize RFgen menu option.

To Start or Stop the Server, click the appropriate button. Note that when remote data collection devices are active, clicking 'Stop Service' will terminate all device processing. **In order for remote devices to log into your RFgen-based application, the Server program (running as a service, or task) needs to be active.**

Service Configuration

The Server may be configured as either a **Windows Service**, or as an **Application/Task** (automatically or manually executed) by means of the 'Configure Service' selection; note that the default configuration setting for the Server is as a '**manual application/task**'. This means that the program needs to be started manually by clicking on the 'Start Service' box shown above. The following configuration window will appear.



The recommended configuration for running RFgen as a service is to use a local administrator to the PC where RFgen is installed. The **Domain** field contains the name of the local PC and the **Account** and **Password** are the credentials for a local administrator account. This typically avoids a problem where a domain admin account is believed to have enough rights but really does not.

The checkbox **Run as Windows Service** means RFgen will be configured in the Window's services list. Checking **Auto-Start When OS Starts** will set that service to start automatically. Otherwise it will be set to manual start.

By only checking the Auto-Start When OS Starts, RFgen will not be configured as a service but will start RFgen as an application when a user logs in to the desktop.

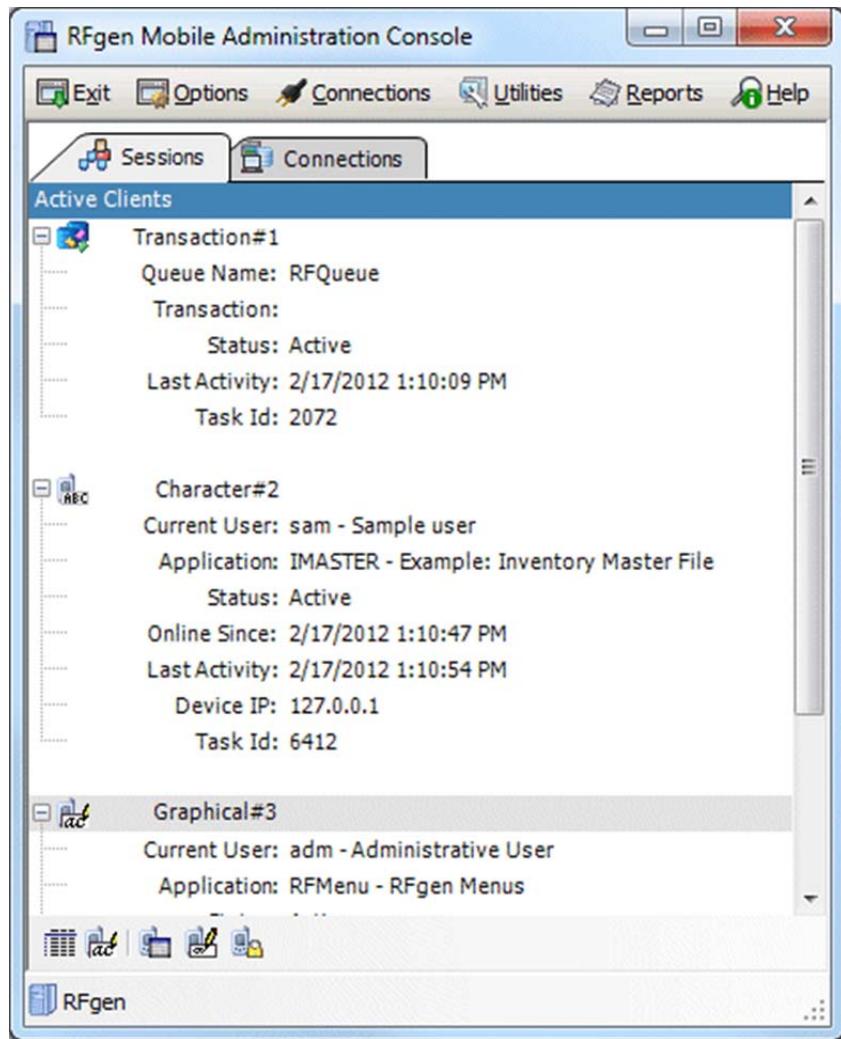
When the Server is active, the system tray icon will display a green check mark. When stopped, the icon will display a red X. When the system is shutting down, the icon may display a yellow lock representing a busy status before the service is stopped completely.

Client Sessions

Each server ‘client’ spawned by a remote device login operates as a separate multi-threaded task under the Server, with all sessions being controlled, coordinated, and serviced by the Server. Each client session may be viewed in its own window. Client sessions utilize approximately 8 megabytes each of main (server) memory. Session windows may be opened (shown/monitored), minimized (hidden), and terminated (logged off) by means of the Console.

The Mobile Administration Console

The Console allows the administrator to view and manage the remote sessions running under the Server. A window similar to the one shown below will appear when the administrator is started.

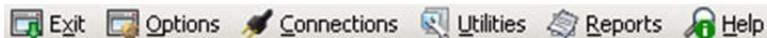


As devices log in, they are displayed in the window. If there are other computers on the network, they may also telnet into the RFgen system as data entry devices. This capability may be used as a system resource. For example, a hardwired (networked) user who connects to the RFgen system will receive the same screens that appear on the screens of remote devices.

Transaction sessions are shown for each queue that is setup. Graphical and Character sessions are displayed for each connected user and represent the type of device they are using.

The Mobile Administration Console Menu Bar

The Console menu bar selections are as follows.



These menu bar selections are functionally identical and nearly the same as those that appear for the Mobile Development Studio.

Exit – closes the Console. This does not stop the service from running, just the user interface for viewing and interacting with the service.

Options – offers all the same options as the Mobile Development Studio with one exception. In the Console the option for authorizing the product applies to the runtime service, not the development environment.

Connections – offers the same ability to view and create connections as the Mobile Development Studio, but not the options to download and view tables or business functions.

Utilities – offers only two identical options to the Mobile Development Studio: Exporting and Importing RFgen development items to and from RFgen databases or Windows files.

Reports – offers the same options as the Mobile Development Studio: Application Logs, Statistics and a View of the Transaction Manager's queues.

Help – offers the same options as well but without the VBA specific help files.

The Sessions Tab

This tab appears when the Console is first started. As data entry devices log in, they are displayed in this main window. The window displays an icon for each device logged in and can be expanded to show details of that client's session. The details are:

Current User – is blank until a user logs in and then shows the user ID and name of the operator.

Application – shows the menu or application screen name currently being viewed by the user.

Status – shows either Disabled or Active depending on the suspend status of the client connection.

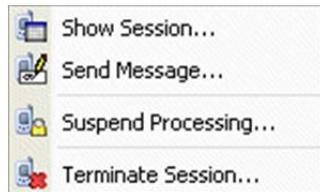
Online Since – shows when the connection was established.

Last Activity – shows when the very last keystroke was made by the user.

Device IP – is the assigned IP address of the device.

Task ID – is the process identifier of the client session executable that can be located in the processes list of the Task Manager.

To monitor or interact with an active client session, simply right-click on the client session in the Sessions tab. A selection menu will appear.

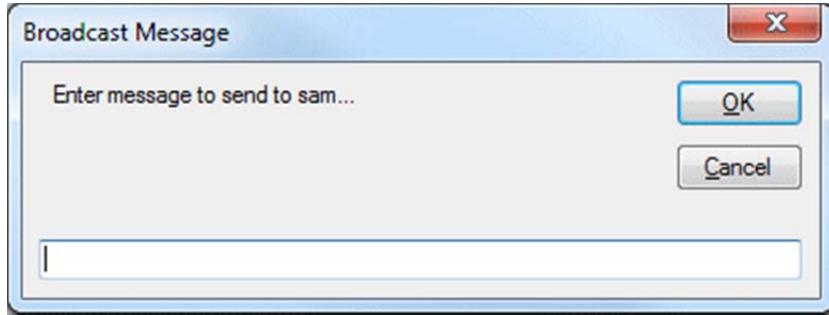


If you click on the **Show Session** selection, the selected client device screen window will appear in the right pane and can be undocked if desired, by double-clicking the Client tab. A Session Options menu item will also offer the above list of choices.



Here, the remote device is being displayed. While the session window is open, all activity for the device will appear in the window. To 'take control' of the session, click inside the screen display area and interact with the prompts.

Send Message allows a user at the system console to send a message to the device user. The following message window appears.



To temporarily stop this session from collecting data, choose **Suspend Processing** from the menu.

To terminate this session, click **Terminate Session**. This action will terminate the communication session for the remote device.

Across the bottom the screen is a toolbar that changes as different items are selected in the tree.



These icons are available from the Sessions tab. The grid table icon offers an alternate view of the tree data in a record style rather than a tree control. The following columns are available for selection.

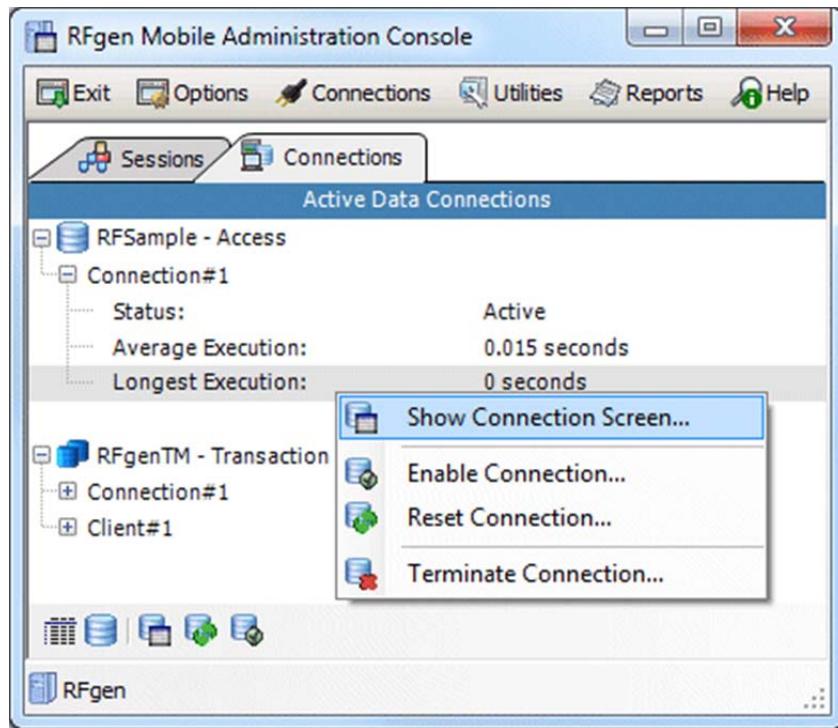
- Display Status...
- Display Average Execution...
- Display Longest Execution...
- Display Task Id...
- Display Last Activity...

The second icon is the other default icon for this tab. It has options for a number of features. It can broadcast a message to all connected devices. This creates a message box that appears on the telnet device that the user accepts before going back to their transaction. Also, there is an option to globally suspend or resume all data collection devices. The administrator also has the ability to terminate all connected sessions forcing all users to log back in again.

The last 3 icons are specific to the selected client in the tree. The options are viewing a copy of the client session on the server, sending the specific client an individual message and suspending / resuming the client.

Connections Tab

Clicking on the Connections tab displays a window that contains each data connection used by RFgen. Entire connections or individual clients attached to a connection can be managed by right-clicking on the item.



For each client session there is also the communication statistics for each data connection. Some applications may only take advantage of a few or just 1 connection depending on the work being performed. The data connection statistics are:

Status – shows either Disabled or Active depending on RFgen's ability to make the connection to the backend database or system.

Average Execution – is how long, on average, each request to the backend database or system takes.

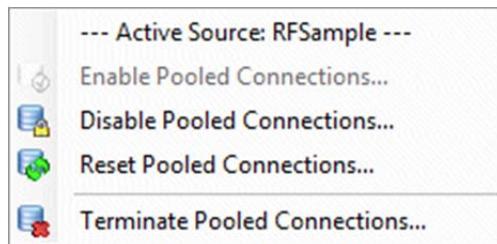
Longest Execution – is the time taken by the longest running single request to the backend database or system.

If Transaction Management is in use, the ability to suspend and resume processing of a queue is done here by right-clicking on the proper queue and choosing the proper option.

The toolbar for the Connections tab has these icons:



The first icon has the same function but the second applies to all pooled connections across all data connections. The options are:



For the selected data connection, the administrator may disable or enable the pooled connections being shared among the thin client mobile users. Resetting the pooled connections is a quick way to destroy and recreate handles to the backend connections with a minimal interruption to the users. These options should be used carefully and only in emergency situations. The Terminate Pooled Connections option tells the server that all pooled handles should be permanently destroyed. Again, this allows the administrator to have control over the environment, but should be used only when necessary.

The last 2 icons are for showing the data connection session so that commands going through that connection can be viewed and recycling the status of the connection, if the backend system seems to be unresponsive.

In the status bar, the icon indicates that the Console is connected to the RFgen server. A red circle over this icon means that the service is no longer running and may have timed out.

Mobile Devices

Windows CE or Mobile devices can be used in a few different ways, as a thin client or mobile client device. Thin client refers to having a wired or wireless connection from the mobile device to the RFgen server while using an RFgen transaction. The mobile device is most commonly using a Microsoft Windows CE or Mobile operating system but any Microsoft operating system will work. For example, tablet PCs typically have a full XP installation.

Thin Client

While in thin client mode, the user interacts with a session running on the server. Since all the processing takes place on the server, the mobile device cannot be a point of failure or lose data. If the wireless device goes out of range of the network, the mobile device screen will appear to stop since the server cannot order the screen to refresh. The telnet client on the mobile device will continue attempts at reconnecting and will then resend the last piece of data entered. Unlike standard telnet, RFgen has added a “Guaranteed Packet Delivery” system to the telnet protocol to ensure no loss of data and an always-synced application. The advantage to the thin client is real time updates to backend systems as well as complete validation data available to ensure the collected data is as accurate as possible. The disadvantage is the need for a wired, wireless, or cellular connection available while collecting data. This client does not require any authorization process.

Mobile Client

Mobile mode is where the mobile device contains a copy of the transactions, users, menus, and database values for validation as well as storage. Keep in mind there will be space limitations on the device. The mobile client can be implemented in 3 ways based on the Startup Mode configured:

1. Connected to the server as in a thin client
2. Disconnected from the server where everything takes place on the device
3. Roaming in and out of range of the server where database access is first attempted against the server, but if the client does not have a connection the local database is used.

When the mobile client is configured to start in a connected state, it is in essence a thin client. The Batch Failover property will allow this mode to either switch over to a disconnected state or remain in a connected state and wait for connectivity to return when wireless coverage is lost.

When a mobile client is configured to start in a disconnected state, all aspects of the data collection operate on the device. Applications, menus, users and others are loaded from the device, validation data on the local database is used for validation and completed transactions are queued on the device pending an upload to the server. Only an RFgen server can extract what was stored on the mobile device. The one distinction between connected (or Thin Client) and disconnected is that an extra option must be provided to tell the mobile client to upload the data to the server when it is in range. See the Device object methods to accomplish this.

When a mobile client is configured to start in a roaming state, then the Batch Failover option has no effect. If the client detects connectivity to the RFgen server, database related commands, embedded procedures and macro calls / queuing are automatically redirected to the server. If there is no connectivity to the RFgen server, all activity is directed locally just like a disconnected state. To keep data integrity, it is recommended that the Server.SendQueue command be placed in the RFgen_OnConnect event. That way all queued work will go to the server before more transactions can be performed that will automatically go to the server. It is also recommended that the Server.SyncApps command also be placed here to keep the applications, menus, users and others always up to date.

The mobile client does require an authorization code to function longer than the demo period. This is described in the Mobile Client section.

Client Network Control

On the mobile device there is an icon  that represents the RFgen CnC service. Clicking on this icon may give various options depending on the implementation. The purpose of this service is to allow requests from the server to be performed on the device when it is not in the cradle.

Some core capabilities include the ability to detect that a server upgrade has occurred and to auto-update the RFgen client environment. Further, if you've deployed RFgen in a mobile (off-line capable) environment, RFgen CnC provides support for "Application Synchronization" requests.

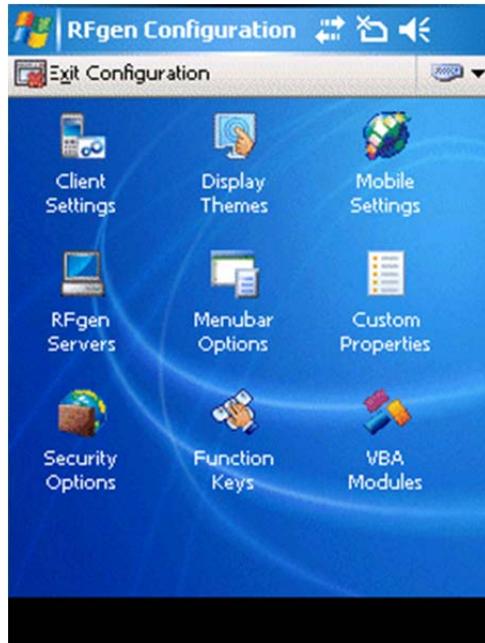
In this scenario, if you've changed any RFgen applications that are in the mobile device's profile, it can automatically detect the application / profile changes, build a custom deployment package and remotely update the device – all without the intervention of IT personnel.

RFgenCFG

This executable is installed in the RFgenMobile folder to allow the user to customize the telnet emulator differently than the options chosen in the Mobile Development Studio at the time the CAB file was created. Start by clicking the Windows Start button on the mobile device and choosing Programs.



Select the RFgenCFG program from the list.



There are multiple configuration pages available. The first one is the Client Settings page.

A screenshot of the "Client Settings" configuration page. The top navigation bar includes "Cancel", "Save Changes", and a dropdown menu. The main area is divided into two sections: "Environment Settings" and "Device Authorization".

Environment Settings	
Installation Type	Mobile Client
Startup Mode	Disconnected
Batch Failover	Disabled
Language Set	English - United States
Full Screen	Disabled
Barcode Driver	Native Wedge
Desktop Password	

Device Authorization	
Device Id	MC40-7628-E0E1
Auth Code	5N77T-5F58N-1UCRJ-X6KOK-E

Installation Type

This displays the type of installation chosen when the profile was created. If the installation type is Thin Client, then it cannot be changed. Otherwise, the user may change the mode as needed between the non-Thin Client modes available.

Startup Mode

For Mobile Installation Types, this option determines if the Mobile client will start in a connected state or a disconnected state.

Batch Failover

For Mobile Installation Types, this determines if the mobile client will ask the user if they wish to move from a connected state to a mobile mode when the thin client detects that the device is no longer in communication with the RFgen server.

Language Set

The default is English. Any language may be chosen from this menu.

Full Screen

This option determines if the display on the mobile device is in a window (smaller) or if the RFgen application is maximized for the screen (larger.)

Barcode Driver

When a particular device uses special barcode drivers and RFgen has implemented them and the profile for the installation uses it, then it will be an option here.

Desktop Password

This option is designed to prevent unauthorized users of the mobile device from leaving the RFgen client and returning to the Device desktop. Passwords are displayed with 6 asterisks (*****) regardless of actual password length.

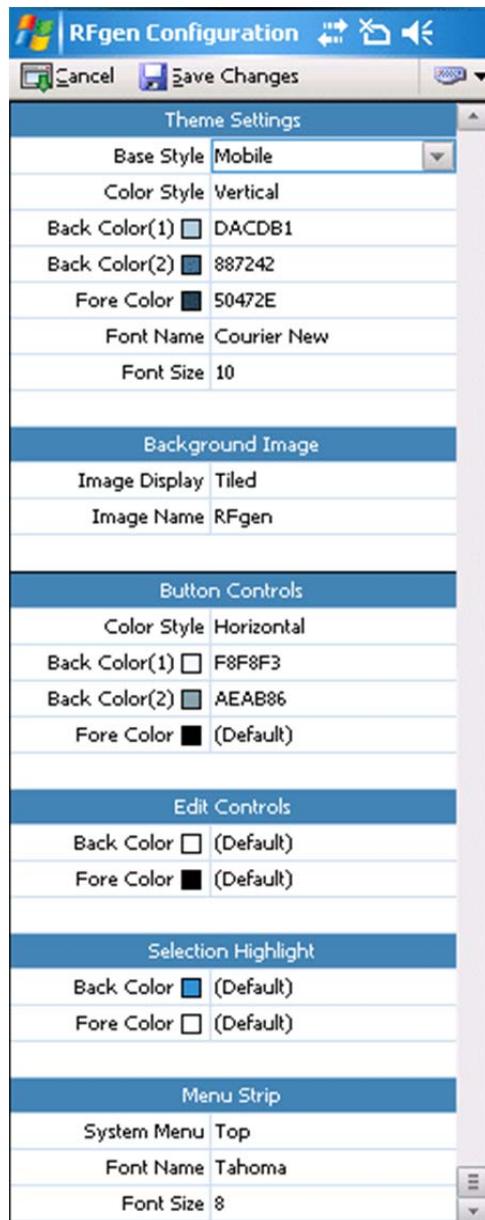
Device Id

The Device Id is the parameter generated by the device that will be used to permanently authorize it for use in the mobile mode.

Auth Code

This field will automatically fill in when the authorization process is started and completes successfully. Manually entering a code here and saving the configuration will also work if access to the server or the server's Internet access is not available.

The Display Themes are as follows:



Base Style

This toggles between the more new Mobile look and feel where the objects on the screen are rounded and the Classic option where the objects are square.

Color Style

The **Color Style** uses the Back Color (1) and Back Color (2) to create an effect of blended colors. This only applies if there is no background image. The options are Solid, Horizontal, Vertical, Diagonal Right and Diagonal Left.

The **Back Color (1 and 2)** are used with the Color Style option to create shading effects.

The **Fore Color** option changes the object's text on the screen.

The **Font Name** and **Font Size** change the font and size for each prompt on the screen.

Background Image:

Image Display will stretch, tile or move an image around on the screen for the background of each form.

The **Image Name** specified retrieves the image from the Image Resources which must contain the image first.

Button Controls:

The **Color Style** is the same as the Color Style listed in the General Theme section but applies to command button prompts only. **Back Color (1 and 2)** performs the same shading effect and the **Fore Color** is the color of the text on the button.

Edit Controls:

The **Back Color** of the data fields can be controlled here and the **Fore Color** is the color of the text inside the data field portion of the prompt.

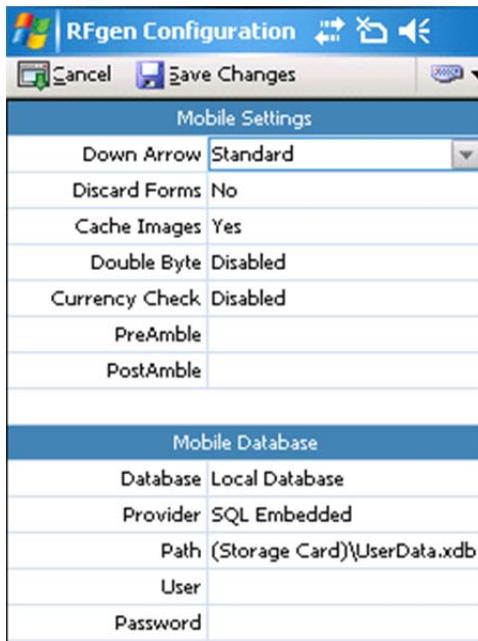
Selection Highlight:

The **Back Color** of the highlighted row in a combo box or list box is set here. The **Fore Color** is the color of the text inside the highlighted row. These should be contrasting colors for readability.

Menu Strip:

The **System Menu** bar can be displayed either at the top or bottom of the screen. It may also be disabled where it will hide the buttons on the menu bar. The **Font Name** property sets the font used on the menu bar and the **Font Size** adjusts the size of the text on the buttons.

The Mobile Settings are as follows.



The **Down Arrow** option is not checked indicating that it will not be used as an alternative for the <Enter> key.

Discard Forms sets the current application data to null when another application is called, rather than keeping the original data in memory (the default condition) should the calling application be returned to.

Cache Image will default the option for the device profiles to retrieve additional images that were not part of the initial installation when the mobile device updates itself.

The **Double Byte** option will allow Chinese, Japanese, and other double byte character sets to display and wrap properly.

Currency Check enables support for international currency formats, by using the current system regional and language option settings (e.g. \$1,234.56 becomes \$1.234,56).

Pre-Amble and **Post-Amble** filter entries are character strings that are automatically sent from a scanner. They ‘surround’ the scanned data. They are optional and neither is required.

Mobile Database:

Database

This option is to specify if a local database is to be used or not. If a database table was included In the CAB file based on the profile then this option should read as Local Database.

Provider

This field specifies which provider is used on the mobile device.

Path

The database path refers to where on the mobile device the file should be created. The file name by itself will be placed in the RFgen directory. Specifying a path such as APPS\RFSample.xdb will place the file in that location.

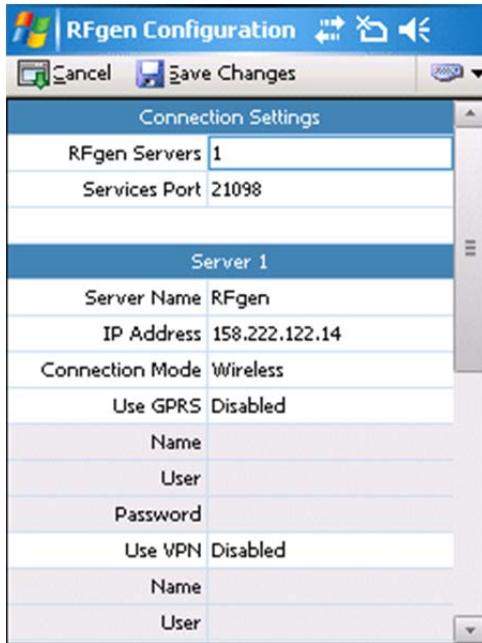
User

If a user is required to access this mobile database, enter it here.

Password

If a password is required to access this mobile database, enter it here. Passwords are displayed with 6 asterisks (*****) regardless of actual password length.

The RFgen Server configuration is as follows.



The **RFgen Servers** number references each possible server that can be configured and the details are shown on the bottom half of the screen.

Services Port is the port number the RFgen server is listening on for graphical telnet traffic.

Server Name

A telnet client can connect to any RFgen server. Adding to this list prompts the user to choose which RFgen server they wish to connect to before starting the login process. The host name is just an identifier but the IP Address is critical.

IP Address

This is the IP address of the RFgen server and can also be changed to a DNS name if the RFgen server is using DHCP to obtain various addresses.

Connection Mode

This property is either Wireless (which assumes the client is already on the private LAN or WAN) or Cellular. If cellular is chosen then there are additional options for making a cellular connection.

Depending on the platform of the mobile device, making a cellular connection can vary greatly. For example, if a VPN is configured on the device and it automatically starts the GPRS connection, then RFgen only needs to start the VPN, not both.

Note: The most effective way to determine what RFgen needs is to perform the connection manually, first to see what the working device configuration is and then tell RFgen to start 1 or both connections by name.

GPRS Setting

This is either enabled or disabled. If one is configured on the device that RFgen must start itself, then fill in the name of that connection.

Name

The name of the connection is the same name used to create the connection in the mobile device's operating system.

User and Password

A User and Password may not be necessary. If RFgen must start the cellular session then reference it by Name. Passwords are displayed with 6 asterisks (*****) regardless of actual password length.

VPN Settings

This is either enabled or disabled. If one is configured on the device that RFgen must start itself, then fill in the name of that connection.

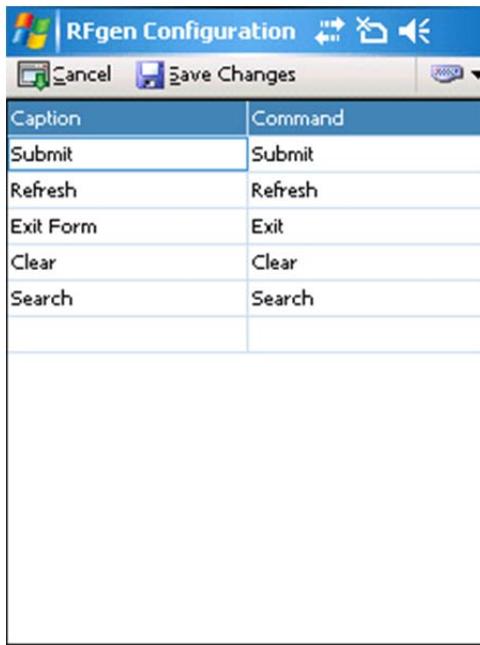
Name

The name of the VPN connection is the same name used to create the VPN in the mobile device's operating system.

User and Password

A User and Password may not be necessary. If RFgen must start the VPN session, then reference it by Name. Passwords are displayed with 6 asterisks (*****) regardless of actual password length.

The Menu Bar configuration is as follows:

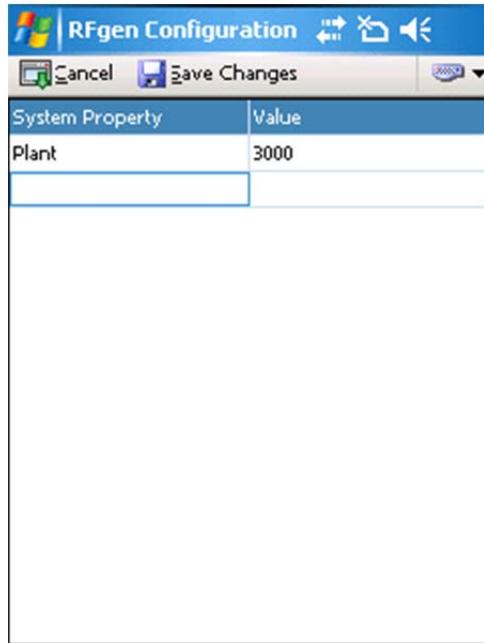


These settings control the functioning of the menu bar display that normally appears at the top or bottom of screen. Activation/Deactivation of the graphical menu bar is controlled by the 'Client Settings' page. The Caption on the left is simple text and the Command on the right is the name of the command or procedure to be executed.

When using this optional on-screen graphical menu bar:

- [Submit] means to enter the data appearing for the current prompt
- [Refresh] refreshes the display screen
- [Clear] clears the data entered for the current prompt
- [Exit] exits the current process (same as F4 normally does).

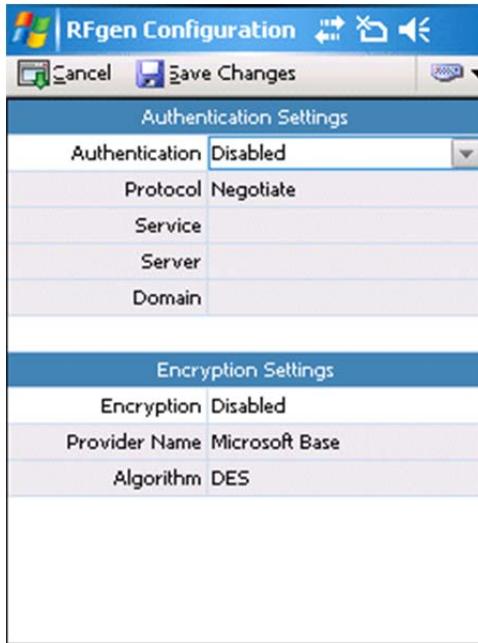
The Custom Properties screen is as follows.



System Properties

Entering a name and value in this box is like creating a global constant with a read-only value. For example, if this installation was designed for multiple warehouses but this particular installation was for a warehouse called 'Main Street' then entering the property name of 'Warehouse' and a value of 'Main' would allow the programmer to identify which warehouse was being used. The command used is `System.EnvironmentProperty`.

The Security Options are as follows.

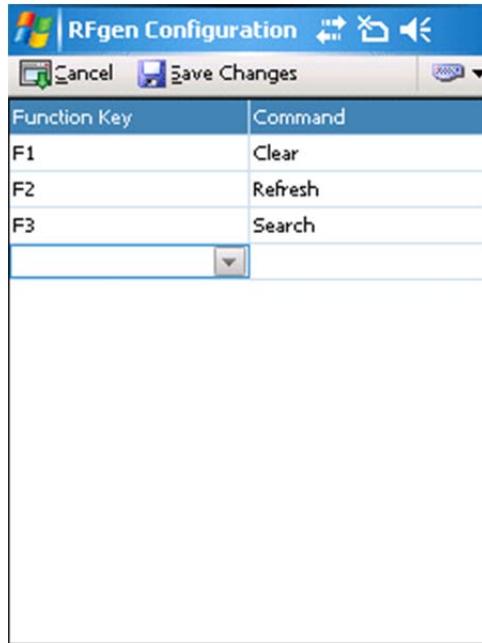


Security is broken into two categories, device authentication and data encryption settings.

Authentication is used to verify the user credentials beyond the RFgen login process. Authentication protocols currently supported are "Negotiate, NTLM, and Kerberos". If authentication is enabled then when an RFgen client first tries to connect, it will pop up a dialog box to capture user information (user id, password, and domain.) An encrypted package of this information will be sent to the configured protocol. A core Windows service on the RFgen server will attempt to authenticate the login request and accept or reject the connection. If Kerberos is used, then you must also specify the Kerberos security service name, the PC name where it is running, and the domain where the PC exists.

Encryption is used to secure the data being transmitted in the wireless environment. Microsoft provides several cryptographic choices and algorithms that are taken from what the operating system is capable of doing. The client must be configured exactly the same way as the server or it will not connect.

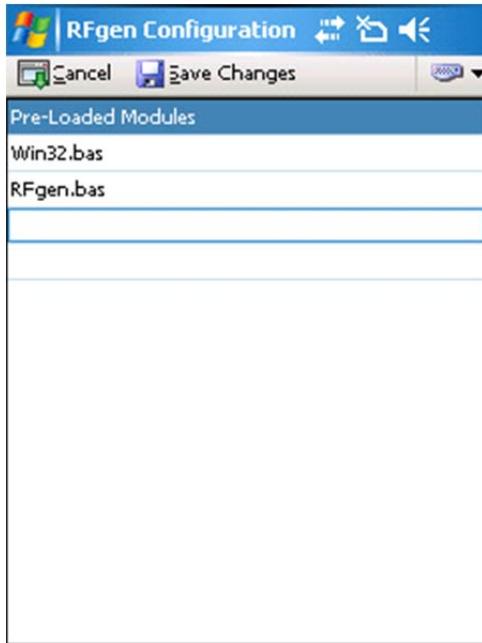
The Function Key setup is as follows.



These keys may be reset if desired. For example, changing F4 to F10 would make F10 the exit key for RFgen Applications and menus. When first using RFgen with remote devices, try using the established default settings before making function-key changes.

Note that an F4, for example, configured for Exit entered anywhere in the application will return you to your user menu. Selecting Logout or F4 on your menu will return you to your previous menu or the Login screen.

The VBA Modules configuration is as follows.



When a profile is created the BAS modules are usually part of the deployment. This screen lets the user change which modules are pre-loaded for a client session.

RFgenCE

From the Windows Start button on the mobile device, click Programs and then RFgenCE.



A splash screen displays the version of the client installed on the mobile device. This version should match the software running on the server.



After the splash screen, the RFLogin screen is the typical first form. In this case, the RFLogin screen shown is from the sample applications shipped with the software.



In the case of the mobile client, the menu may have the Authorization option. If the client has wireless access to the RFgen server and the server has Internet access (and the client is allowed to be authorized) then this option will authorize the client device and the menu option will disappear. The other options are to reset the connection between the client and server which makes sense if the client is wirelessly connected and exit the client.

This form can be bypassed and the user could be presented with a menu or another form if desired. An optional menu bar may be displayed at the top or bottom of the screen. At a minimum, there is a keyboard icon for displaying the soft keyboard for data entry and a menu icon allowing the user to reset their connection or exit the application. If a desktop password has been configured, the user must enter that password before being exited back to the desktop. If the menu bar options are not disabled, then the captions of those buttons will also appear.

Visual Basic Scripts

Visual Basic for Applications (VBA) scripts allow technical personnel to provide additional functionality to RFgen data collection applications by responding to events using Visual Basic programming statements (scripts). VBA Scripts allow the developer to enhance the capabilities offered by standard RFgen applications and other objects. In fact, developers may take total control over the client device by responding to field/system events, handling all data display functions, and even sending SQL statements through the RFgen database connection.

Programmable events include 'field' events (scripting is related to the current prompt) and application events (scripting will be executed whenever this event occurs; e.g. when an application is loaded). In addition, built-in VBA 'extensions' give the developer easy access to manipulate the client device, and quick access to the connected database.

Global User Defined Subroutines and Functions

To use global subroutines and functions: create a module file using the VBA Modules and put your 'global' subroutines and functions in that file. On the Properties tab in the RFgen Application window, you will see a list of available VBA modules. Set each module to TRUE that you want that application to be able to access.

You will then be able to call any subroutines or functions from these modules in your RFgen Applications.

Note: There are 2 built in modules that are automatically referenced by RFgen Applications. These are:

RFgen.bas – The RFgen.bas is typically used to contain functions and procedures that need to be accessible from any transaction

Win32.bas – The Win32.bas is typically used to store global variables.

Using External ActiveX files in the VBA Environment

RFgen allows the incorporation of external ActiveX .EXE or .DLL modules by declaring an Object and using the CreateObject method.

Example:

```
Dim MyObject As Object  
Set MyObject = CreateObject("name.cls")  
' name is the name of the ActiveX module ' subsequent  
usage  
MyObject.property  
MyObject.method
```

The syntax is slightly different when using a DLL compiled using Visual Basic.

```
Dim MyDLL As Object  
Set MyDLL = CreateObject("ourDDC.DDCcom")
```

Where: ourDDC is the executable name DDCcom is the public class name containing the functions you wish to call.

VBA Global Variables/Objects

Global variables and objects are available for use with RFgen, for storing and retrieving items related to a specific session.

The preferred method for using global variables is to declare the variables in the 'Win32.bas module, via the VBA Modules option. Any public variables declared in the Win32.bas module will be available to all RFgen applications and will maintain their values throughout the life of the RFgen session.

Examples:

```
Public sMyString As String  
Public iMyInt As Integer  
Public oMyObject As Object
```

Alternatively, you can still use the older format that is maintained for backward compatibility with prior releases.

Variables have the format: gVariant(n), where n=0 to 50,
Objects have the format: gObject(n), where n=0 to 10.

Variable and object references are shared between all defined VBA events (see below), starting at the time a user logs in, until the user logs out.

Examples:

```
gVariant(0) = "Hello"  
gVariant(1) = 567  
gVariant(2) = True  
gVariant(3) = False
```

VBA Declarations

The VBA ‘General’ object is where the developer may declare variables that are available to all events in the current application. User defined functions or subroutines may also be entered here (see above).

VBA Events

Field events are linked to and executed by individual data entry prompts in an RFgen application. Application events are global in nature (e.g., the application event occurs immediately before the corresponding field event). Edit properties are executed after the VBA event has finished. Using these events, the developer may respond to the users' actions as they choose. The events supported by RFgen are as follows:

Click

The Click event occurs when the user clicks on a prompt or button with a mouse or pen. It is typically used to determine if a user has pressed a button on the screen.

Applies to: Prompts (graphical mode only)

Syntax: Click()

GotFocus

The GotFocus event occurs as soon as the user (either moving forward or backing up) reaches the data entry field. This event is typically used to generate a default value that the user can either accept or reject.

Applies to: Applications, Prompts

Syntax: GotFocus(Rsp, AllowChange)

Rsp (String) is the default value to display on the Client device.

`AllowChange` (Boolean) is set to True to allow the user to override the default, and is set to False to generate an OnEnter event bypassing user interaction at this field.

Keypress

The Keypress event occurs when the user presses and releases a key on the keyboard. This event is typically used to capture incoming keystrokes and perform some action based upon user input. In addition, the device's telnet emulator must be capable of supporting 'Character' mode. If the emulator is set to transmit data in Line mode, RFgen will not see the data until the whole string is complete.

Applies to: Applications, Prompts

Syntax: `KeyPress(KeyAscii)`

`KeyAscii` (Integer) is the ASCII value of the character pressed on the client device.

Initialize

The Initialize event occurs when an RFgen client is initially loaded. It will occur only once, and is typically used to initialize variables, open additional database connections or create links to ActiveX objects.

Applies to: Applications (RFgen.bas)

Syntax: `Initialize()`

Load

The Load event occurs when the application is initially loaded. It will occur only once per form and is typically used to initialize variables.

Applies to: Applications (RFgen.bas)

Syntax: `Load()`

Lost Focus

The LostFocus event occurs when the prompt that had the focus is giving it up to another prompt. This would occur after the OnEnter event is finished and before the next prompt's GotFocus event is executed.

Applies to: Application, Prompts

Syntax: `LostFocus()`

OnBackup

The OnBackup event occurs when the prompt with the focus receives a command to go back to a previous prompt, as if the up arrow key was pressed.

Applies to: Application, Prompts

Syntax: OnBackup(Cancel)

Cancel (Boolean) is set to True if the backup movement should be stopped. Set it to False or do not change the default to allow the backup to continue.

OnConnect

The OnConnect event occurs when the Mobile Client in a roaming state automatically discovers it has connectivity to a network and makes a connection to the RFgen server. The event only executes after 10 seconds of continuous connectivity to the RFgen server. To maintain data integrity it may be a good idea to include the Server.SyncApps and Server.SendQueue in this event.

Applies to: RFgen (RFgen.bas)

Syntax: OnConnect()

OnCursor

The OnCursor event occurs whenever a cursor (arrow) key is pressed. This event is typically used to allow users to page through a list of data. The Up arrow is typically used to backup to the previous field (system default); however, to bypass system processing of this event, set Cursor = "" (null).

Applies to: Application, Prompts

Syntax: OnCursor(Cursor)

Cursor (String) is the character value of the key pressed. Possible values are ('U'p, 'D'own, 'R'ight, and 'L'eft).

Example:

```
Public Sub PartNo_OnCursor(Cursor As String)
    On Error Resume Next
    If Cursor = "U" Then
        'your logic
    End If
End Sub
```

OnDisconnect

The OnDisconnect event occurs when the Mobile Client in a Roaming state is disconnected from the RFgen server because the client moved out range. This event does not execute when the Server.Disconnect command is issued.

Applies to: RFgen (RFgen.bas)

Syntax: OnDisconnect()

OnEnter

The OnEnter event occurs when the user presses the Enter key on the Client device, or a default value specifies that no user input is allowed. This event is typically used to validate data entered, and/or adjust the display on the device. To reject the entry made by the user, set Cancel = True and RFgen will force the user to re-enter the field.

Applies to: Application, Prompts

Syntax: OnEnter(Rsp, Cancel, Message)

Rsp (String) is the value entered/scanned by the user on the Client device.

Cancel (Boolean) is set to False to accept the data entered and move to the next prompt, or is set to True to fail the edit check and force the user to re-enter data at the current field.

Message (String) is a message to display on the Client device.

OnEscape

The OnEscape event occurs when the Escape key is pressed. This event is typically used to capture an incoming Escape keystroke and perform some action based upon user input.

Applies to: Application, Prompts

Syntax: OnEscape()

OnFkey

The OnFkey event occurs whenever a function key (Fn) key is pressed. Function keys F1-F4 trigger pre-defined system events. However, to bypass system processing of these events set Fkey = '0' (zero) and RFgen will ignore the event.

Applies to: Application, Prompts

Syntax: OnFkey(Fkey)

Fkey (Integer) is the integer value of the key pressed. Possible values range from (1-10).

Example:

```
Public Sub Form_OnFkey(Fkey As Integer)
    On Error Resume Next
    If Fkey = 5 Then
        ' In this example we are using the F5
        ' key to execute logic.
        ' your logic
    End If
End Sub
```

OnInRange

The OnInRange event occurs when a Mobile Client notices that there is an IP address available on the network adapter. This event does not execute when using the Roaming Client because the OnConnect will execute anyway.

Applies to: RFgen (RFgen.bas)

Syntax: OnInRange()

OnLocale

The OnLocale event occurs after OnConnect (only when a client application makes a connection) and passes in the locale number based on the client device's location. In the case of the United States, number 1033 is returned. Based on this value you may set global values or default login forms (See App.ChangeLoginForm)

Applies to: Application, Prompts

Syntax: OnLocale(nDeviceLocale)

nDeviceLocale (Long) is the value of the client device's locale.

Example:

```
Public Sub RFgen_OnLocale(ByVal nDeviceLocale As Long)
    On Error Resume Next
    Select Case nDeviceLocale
        Case 1033
            App.ChangeLoginForm("RFLoginEnglish")
        Case Else
            App.ChangeLoginForm("RFLogin")
    End Select
```

End Sub

OnOutOfRange

The OnOutOfRange event occurs when a Mobile notices that the IP address is no longer available on the network adapter. This event does not execute when using the Roaming Client because the OnDisconnect will execute anyway.

Applies to: RFgen (RFgen.bas)

Syntax: OnOutOfRange ()

OnReadData

The OnReadData event occurs whenever data is successfully retrieved from a database by an RFgen application. This event is typically used to force an immediate repaint of the screen to display all linked prompts values to the user.

Applies to: Application

Syntax: OnReadData(TableName)

TableName (String) is the name of the table that was successfully queried.

Example:

```
Public Sub Form_OnReadData(TableName As String)
    On Error Resume Next
    RFRefresh ' Repaint the current screen
End Sub
```

OnRefresh

The OnRefresh event occurs when the screen is completely repainted (as when the user presses the F2 key). It is typically used to redisplay information to the user that was displayed via Screen.Print statements.

Applies to: Application

Syntax: OnRefresh()

OnReturn

The OnReturn event occurs when the user returns from a called application that had its return flag set to True. It is typically used to process data from the called application.

Applies to: Application

Syntax: OnReturn(Name)

Name (String) is the name of the called application that is returning.

OnScan

The OnScan event occurs when RFgen identifies a preamble string in the data stream coming from a client device. This event is typically used to validate that data was scanned, or to parse the data (PDF, RFID, etc.) into a more usable format. To reject the data scanned, set the ScanData = "" and RFgen will remove it from the data stream and prevent it from being entered into the current prompt.

Applies to: Application, Prompts

Syntax: OnScan(ScanData)

ScanData (String) is the value scanned by the user on the client device.

OnSearch

The OnSearch event occurs only if a linked or unlinked text box has code in the OnSearch event and the user clicks on the generated search command button. This is only possible in graphical mode. Any VBA code can be placed in this event, even if it is not specifically related to searching a data source.

Applies to: Text prompts

Syntax: OnSearch(Rsp, Cancel)

Rsp (String) The value in the textbox if any

Cancel (Boolean) Set to True to NOT run the OnEnter event and put the focus back on the prompt

OnSpeech

Using the language extension TTS.Listen, RFgen will detect that speech has occurred and execute this event.

Applies to: input prompts

Syntax: OnSpeech(Rsp, Cancel)

Rsp (String) The text representation of what was spoken. If there was a grammar substitution defined for the spoken word then the substitution is returned.

Cancel (Boolean) Setting this to True will prevent downstream OnSpeech events from executing and set the focus back to the current prompt for another attempt. The order in which they execute is RFgen_OnSpeech (located in the RFgen.Bas module), Form_OnSpeech and finally the PromptName_OnSpeech event. Setting this to True also places the contents of the Rsp variable in the text box displayed on the screen (as if it was typed or scanned input) and then the OnEnter event will execute.

OnTimer

The OnTimer event occurs at a user-defined interval when it is enabled. This event is used to perform some action based upon defined time interval.

Applies to: Application

Syntax: OnTimer()

Example:

First the timer interval is set and the timer is enabled.

```
Public Sub Form_Load()
    On Error Resume Next
    TimerInterval = 1000
    TimerEnabled = True
End Sub

'OnTimer event is given some logic.
Public Sub Form_OnTimer()
    On Error Resume Next
    'your logic
End Sub
```

OnUpdate

The OnUpdate event occurs after the last prompt has been entered and just prior to RFgen updating the database. This event is typically used to validate that data was entered and process additional updates.

Applies to: Application

Syntax: OnUpdate(Cancel)

Cancel (Boolean) is used to cancel the update.

OnVocollect

This event is only used with Vocollect data collection hardware. When the Vocollect device activates a Vocollect task on an application, a recordset is sent to RFgen for processing. If RFgen needs to communicate back to the user, RFgen passes to the Vocollect device another recordset.

Applies to: Application

Syntax: OnVocollect(rsIn, rsOut)

rsIn a DataSet object that captures the collected data coming in from the Vocollect device

rsOut a DataSet object built in the RFgen VBA code that is sent to the Vocollect device

Example:

```
Public Sub prTaskODRUpdateStatus_OnVocollect (ByVal  
rsIn As DataSet, ByVal rsOut As DataSet)  
    On Error Resume Next  
    '  
    rsIn.Param("DateTime")  
    rsIn.Param("SerialNumber")  
    rsIn.Param("Operator")  
    rsIn.Param("AssignmentId")  
    rsIn.Param("LocationId")  
    rsIn.Param("UpdateMode")  
    rsIn.Param("UpdateStatus")  
    '  
    rsOut.Param("ErrCode") = 0  
    rsOut.Param("Message") = ""  
End Sub
```

Terminate

The Terminate event occurs when an RFgen client is terminating. It will occur only once, and is typically used to close manually opened database connections or release links to ActiveX objects.

Applies to: RFgen (RFgen.bas)

Syntax: Terminate()

Unload

The Unload event occurs when the application is terminating. It will occur only once, and is typically used to release any system resources manually opened that are specific to this application.

Applies to: Application

Syntax: Unload()

VBA Language Extensions

VBA Extensions are additions to the standard Visual Basic for Applications scripting language. Using these extensions, the developer may easily control the client device data display, manipulate database records, communicate with other databases, control stored procedures, and communicate with attached or Windows-based printers. The extensions provided by RFgen have been grouped into several categories and are as follows:

Prompt Property Extensions

These properties are available at run time by using the RFPrompt methods. At design time they can be found on the Fields Properties tab.

RFPrompt().AddItem

This method is used to manually add items to a list box. You must specify the value to be returned should the user choose the selection. The display is a user-friendly description of the value behind the scenes and is optional. The location in the list where the item is to be added can also optionally be specified.

Syntax: RFPrompt(Nbr).AddItem (Value, [Display], [Location])

Nbr (Variant) refers to the prompt number or name of the list box

Value (Variant) the item to add to the list

Display (Variant) the displayed value (optional)

Location (Variant) the location to add the new item (optional)

Example:

The following uses the AddItem method in the OnEnter event in one prompt to populate the list in another prompt.

```
Public Sub TextBox1_OnEnter(RSP As String, Cancel As Boolean, ErrMsg As String)
    On Error Resume Next
    RFPrompt(4).AddItem ("2000", "2000 ABC Motor Co.")
    RFPrompt(4).AddItem ("2001", "2001 Widgets R Us")
    RFPrompt(4).AddItem ("2002", "2002 Goodfellow Inc.")
End Sub
```

RFPrompt().Autosize

For an Image control, this property set to True will resize the control to fit the image when the image is loaded.

For a Menu control, this property set to True will expand the menu object to the lowest and right-most portion of the screen. The upper left corner is static.

Syntax: RFPrompt(Nbr).Autosize = Value

Nbr (Variant) is numeric, or fieldname, and refers to a specific prompt.

Value (Boolean) turns on or off this property.

Example:

```
RFPrompt("Image1").Autosize = True
```

RFPrompt().Bitmap

This property accesses the image in the signature capture object and allows the user to write it to a file. This property can also read from a file and place the BMP formatted image in an Image control. Be sure to have pressed Enter on the signature box before trying to read the picture from it. After the OnEnter event finishes the prompt will have a value.

Syntax: RFPrompt(Nbr).Bitmap = bImage

Alternate: bImage = RFPrompt(Nbr). Bitmap

Nbr (Variant) is numeric, or fieldname, and refers to a specific prompt.

bImage (Variant) is the byte array containing the image.

Example *reading from a file*:

```
Dim bImage() As Byte
Open "C:\sig.bmp" For Binary Access Read As #1
  ReDim bImage(LOF(1))
  Get #1,,bImage
Close #1
RFPrompt(1).Bitmap = bImage  ' prompt one is an Image control
```

Example *writing to a file*:

```
Dim bImage() As Byte
bImage = RFPrompt(1).Bitmap  ' prompt one is a
signature box
Open "C:\sig.bmp" For Binary Access Write As #1
  Put #1,,bImage
Close #1
```

RFPrompt().Caption

This property accesses the caption associated with a specific prompt.

Syntax: RFPrompt(Nbr).Caption = vValue

Alternate: sValue = RFPrompt(Nbr).Caption

Nbr (Variant) is numeric, or fieldname, and refers to a specific prompt.

sValue (String) is the caption of the prompt.

vValue (Variant) is the caption to display for the prompt.

Examples:

```
RFPrompt(1).Caption = "Part Number"
```

```
RFPrompt("Part").Caption = "Part Number"
```

RFPrompt().Checked

This property accesses the status of a checkbox object associated with a specific prompt.

Syntax: RFPrompt(Nbr).Checked = Value

Alternate: Value = RFPrompt(Nbr).Checked

Nbr (Variant) is numeric, or fieldname, and refers to a specific prompt.

Value (Boolean) is the checked state of the prompt.

Examples:

```
RFPrompt(1).Checked = True
```

```
RFPrompt("Part").Checked = False
```

```
bValue = RFPrompt("Part").Checked
```

RFPrompt().Clear

This method is used to remove all items from a list box, combo box or menu.

Syntax: RFPrompt(Nbr).Clear

Nbr (Variant) is numeric, or fieldname, and refers to a specific prompt.

Example:

```
RFPrompt("List").Clear
```

RFPrompt().Defaults

This property accesses the default property associated with a specific prompt.

Syntax: RFPrompt(Nbr).Defaults = vValue

Alternate: sValue = RFPrompt(Nbr).Defaults

Nbr (Variant) is numeric, or fieldname, and refers to a specific prompt.

sValue (String) is the default expression for the prompt.

vValue (Variant) sets the default expression for the prompt.

Examples:

`RFPrompt(2).Defaults = "N"`

`RFPrompt("YesNo").Defaults = "N;O" ' with override`

The letter 'N' will be the default for prompt #2.

RFPrompt().Display3D

This property accesses the Border property associated with a specific prompt. Setting it to False makes a prompt's data field transparent to the background color.

Syntax: RFPrompt(Nbr).Display3D = Value

Alternate: Value = RFPrompt(Nbr).Display3D

Nbr (Variant) is numeric, or fieldname, and refers to a specific prompt.

Value (Boolean) is the display only state for the prompt.

Examples:

`RFPrompt(3).Display3D = True`

`RFPrompt("FieldID").Display3D = False`

RFPrompt().DisplayOnly

This property accesses the display only property associated with a specific prompt. Setting it to True makes a prompt into a display only field, while setting it to False allows users to access and change data at the prompt.

Syntax: RFPrompt(Nbr).DisplayOnly = Value

Alternate: Value = RFPrompt(Nbr).DisplayOnly

Nbr (Variant) is numeric, or fieldname, and refers to a specific prompt.

Value (Boolean) is the display only state for the prompt.

Examples:

`RFPrompt(3).DisplayOnly = True`

`RFPrompt("FieldID").DisplayOnly = True`

RFPrompt().Edits

This property accesses the Edits property associated with a specific prompt.

Syntax: RFPrompt(Nbr).Edits = vValue
Alternate: sValue = RFPrompt(Nbr).Edits
Nbr (Variant) is numeric, or fieldname, and refers to a specific prompt.
sValue (String) is the edit conditions for the prompt.
vValue (Variant) sets the edit conditions for the prompt.

Examples:

```
RFPrompt(2).Edits = "2N" ' only accept 2 numbers  
RFPrompt("FieldID").Edits = "2N"
```

RFPrompt().ErrMsg

This property accesses the error message property associated with a specific prompt.

Syntax: RFPrompt(Nbr).ErrMsg = vValue
Alternate: sValue = RFPrompt(Nbr).ErrMsg
Nbr (Variant) is numeric, or fieldname, and refers to a specific prompt.
sValue (String) is the error message to display for the prompt.
vValue (Variant) sets the error message to display for the prompt.

Examples:

```
RFPrompt(2).ErrMsg = "Must be 2 numbers."  
RFPrompt("FieldID").ErrMsg = "Must be 2 numbers."
```

RFPrompt().FieldBackColor

This property accesses the prompt's data field background color.

Syntax: RFPrompt(Nbr).FieldBackColor = vValue
Alternate: IValue = RFPrompt(Nbr).FieldBackColor
Nbr (Variant) is numeric, or fieldname, and refers to a specific prompt. Note: Using the number 0 will refer to the application, not the prompt. The application's background and all the prompt caption's backgrounds will be affected.
IValue (Long) is the color.
vValue (Variant) sets the color.

Examples:

```
RFPrompt(2).FieldBackColor = vbWhite  
RFPrompt(0).FieldBackColor = vbGreen  
RFPrompt("Part").FieldBackColor = RGB(255,255,0)  
'yellow  
RFPrompt("Part").FieldBackColor = "&HFF0000"  
'blue
```

```
RFPrompt("Part").FieldBackColor = QBColor(5)  
'magenta
```

RFPrompt().FieldFontBold

This property accesses the prompt's data field bold option.

Syntax: RFPrompt(Nbr).FieldFontBold = Value

Alternate: Value = RFPrompt(Nbr).FieldFontBold

Nbr (Variant) is numeric, or fieldname, and refers to a specific prompt. Note: Using the number 0 will refer to the application, not the prompt. All of the prompt captions will be affected.

Value (Boolean) is True or False for bold or not bold.

Examples:

```
RFPrompt(0).FieldFontBold = True  
RFPrompt("Part").FieldFontBold = True
```

RFPrompt().FieldFontItalic

This property accesses the prompt's data field italic option.

Syntax: RFPrompt(Nbr).FieldFontItalic = Value

Alternate: Value = RFPrompt(Nbr).FieldFontItalic

Nbr (Variant) is numeric, or fieldname, and refers to a specific prompt. Note: Using the number 0 will refer to the application, not the prompt. All of the prompt captions will be affected.

Value (Boolean) is True or False for italics or no italics.

Examples:

```
RFPrompt(0).FieldFontItalic = True  
RFPrompt("Part").FieldFontItalic = True
```

RFPrompt().FieldFontSize

This property accesses the prompt's data field text size.

Syntax: RFPrompt(Nbr).FieldFontSize = IValue

Alternate: vValue = RFPrompt(Nbr).FieldFontSize

Nbr (Variant) is numeric, or fieldname, and refers to a specific prompt. Note: Using the number 0 will refer to the application, not the prompt. All of the prompt captions will be affected.

vValue (Variant) is the font size.

IValue (Long) sets the font size. Default is 10.

Example:

```
RFPrompt("Part").FieldFontSize = 16
```

RFPrompt().FieldFontUnderline

This property accesses the prompt's data field underline option.

Syntax: RFPrompt(Nbr).FieldFontUnderline = Value

Alternate: Value = RFPrompt(Nbr).FieldFontUnderline

Nbr (Variant) is numeric, or fieldname, and refers to a specific prompt. Note: Using the number 0 will refer to the application, not the prompt. All of the prompt captions will be affected.

Value (Boolean) True or False for underline or no underline.

Example:

```
RFPrompt ("Part") .FieldFontItalic = True
```

RFPrompt().FieldForeColor

This property accesses the prompt's data field fore color.

Syntax: RFPrompt(Nbr).FieldForeColor = IValue

Alternate: vValue = RFPrompt(Nbr).FieldForeColor

Nbr (Variant) is numeric, or fieldname, and refers to a specific prompt. Note: Using the number 0 will refer to the application, not the prompt. All of the prompt captions will be affected.

vValue (Variant) is the color.

IValue (Long) sets the color.

Examples:

```
RFPrompt (2) .FieldForeColor = vbWhite  
RFPrompt (0) .FieldForeColor = vbGreen  
RFPrompt ("Part") .FieldForeColor = RGB(255,255,0)  
'yellow  
RFPrompt ("Part") .FieldForeColor = "&HFF0000"  
'blue  
RFPrompt ("Part") .FieldForeColor = QBColor(5)  
'magenta
```

RFPrompt().FieldId

This property returns the field ID of a specific prompt. If the prompt is linked to a database field, that field name is returned.

Syntax: Value = RFPrompt(Nbr).FieldId

Nbr (Variant) is numeric, or fieldname, and refers to a specific prompt.

Value (String) is the field id/name for the prompt.

Example:

```
Dim sValue As String  
sValue = RFPrompt(2).FieldId
```

RFPrompt().Format

This property accesses the format of a specific prompt. It is only an extension of the Format VBA command.

Syntax: RFPrompt(Nbr).Format = vValue

Alternate: sValue = RFPrompt(Nbr).Format

Nbr (Variant) is numeric, or fieldname, and refers to a specific prompt.

sValue (String) is the format mask to use when displaying data for the prompt.

vValue (Variant) sets the format mask to use when displaying data for the prompt.

Examples:

```
RFPrompt("txtTime").Format = "hh:mm"
```

c - General Date

dddddd - Long Date

ddddd - Short Date

tttt - Long Time

hh:mm AMPM - Medium Time

hh:mm - Short Time

\$#,##0.00 or (\$#,##0.00) - Currency 0.00 - Fixed

#,##0.00 - Standard 0.00% - Percent 0.00E+00 -

Scientific

Yes/No - Return "No" if zero, else return "Yes"

True/False - Return "True" if zero, else return "False"

On/Off - Return "On" if zero, else return "Off"

For further examples get help on the FORMAT command.

RFPrompt().Highlight

This property places or removes the highlight bar in a combo box or list box control when in TE (character) mode.

Syntax: RFPrompt(Nbr).Highlight = Value

Value (Boolean) is either True or False.

Example:

```
RFPrompt("PartList").Highlight = True
```

RFPrompt().ImageGetBitmap

This function retrieves a BMP object stored within the database and displays it for an image prompt.

Syntax: [bValue =] RFPrompt(Nbr).ImageGetBitmap(SQL)

Nbr (Variant) is numeric, or fieldname, and refers to a specific prompt.

SQL (String) is the ODBC call to retrieve the OLE object stored in the database.

bValue (Boolean) Optional – returns True or False for the success of the command

Example:

```
Dim SQL As String  
SQL = "Select Image from Inv where PartNo = '100620'"  
RFPrompt("PartImage").ImageGetBitmap(SQL)
```

RFPrompt().ImagePath

This property retrieves a graphic file and displays it in an image prompt. When using thin client mode the supported image types are BMP, DIB, GIF and JPG. When running in mobile mode, only BMP is supported.

Syntax: RFPrompt(Nbr).ImagePath = vValue

Alternate: sValue = RFPrompt(Nbr).ImagePath

Nbr (Variant) is numeric, or fieldname, and refers to a specific prompt.

sValue (String) is the path to the image.

vValue (Variant) sets the path to the image.

Example:

```
RFPrompt("PartImage").ImagePath = "C:\House.GIF"
```

RFPrompt().Index

This property returns the prompt number and is read only. The language extension App.PromptNo returns the same value.

Syntax: Value = RFPrompt(Nbr).Index

Nbr (Variant) is numeric, or fieldname, and refers to a specific prompt.

Value (Variant) is the variable containing the prompt's number.

Example:

```
Dim value As Variant  
value = RFPrompt("Part").Index
```

RFPrompt().LabelBackColor

This property accesses the prompt's label background color.

Syntax: RFPrompt(Nbr).LabelBackColor = IValue

Alternate: vValue = RFPrompt(Nbr).LabelBackColor

Nbr (Variant) is numeric, or fieldname, and refers to a specific prompt. Note: Using the number 0 will refer to the application, not the prompt. The background of the heading on the application will be affected.

vValue (Variant) is the color.

IValue (Long) sets the color.

Examples:

```
RFPrompt(2).LabelBackColor = vbWhite  
RFPrompt(0).LabelBackColor = vbGreen  
RFPrompt("Part").LabelBackColor = RGB(255,255,0)  
'yellow  
RFPrompt("Part").LabelBackColor = "&HFF0000"  
'blue  
RFPrompt("Part").LabelBackColor = QBColor(5)  
'magenta
```

RFPrompt().LabelFontBold

This property accesses the prompt's label bold option.

Syntax: RFPrompt(Nbr).LabelFontBold = Value

Alternate: Value = RFPrompt(Nbr).LabelFontBold

Nbr (Variant) is numeric, or fieldname, and refers to a specific prompt. Note: Using the number 0 will refer to the application, not the prompt. The application's heading will be affected.

Value (Boolean) is True or False for bold or not bold.

Examples:

```
RFPrompt(0).LabelFontBold = True  
RFPrompt("Part").LabelFontBold = True
```

RFPrompt().LabelFontItalic

This property accesses the prompt's label italic option.

Syntax: RFPrompt(Nbr).LabelFontItalic = Value

Alternate: Value = RFPrompt(Nbr).LabelFontItalic

Nbr (Variant) is numeric, or fieldname, and refers to a specific prompt. Note: Using the number 0 will refer to the application, not the prompt. The application's heading will be affected.

Value (Boolean) is True or False for italics or no italics.

Examples:

```
RFPrompt(0).LabelFontItalic = True  
RFPrompt("Part").LabelFontItalic = True
```

RFPrompt().LabelFontSize

This property accesses the prompt's label text size.

Syntax: RFPrompt(Nbr).LabelFontSize = IValue

Alternate: vValue = RFPrompt(Nbr).LabelFontSize

Nbr (Variant) is numeric, or fieldname, and refers to a specific prompt. Note: Using the number 0 will refer to the application, not the prompt. The application's heading will be affected.

vValue (Variant) is the font size.

IValue (Long) sets the font size. Default is 10.

Examples:

```
RFPrompt(0).LabelFontSize = 16  
RFPrompt("Part").LabelFontSize = 16
```

RFPrompt().LabelFontUnderline

This property accesses the prompt's label underline option.

Syntax: RFPrompt(Nbr).LabelFontUnderline = Value

Alternate: Value = RFPrompt(Nbr).LabelFontUnderline

Nbr (Variant) is numeric, or fieldname, and refers to a specific prompt. Note: Using the number 0 will refer to the application, not the prompt. The application's heading will be affected.

Value (Boolean) True or False for underline or no underline.

Examples:

```
RFPrompt(0).LabelFontItalic = True  
RFPrompt("Part").LabelFontItalic = True
```

RFPrompt().LabelForeColor

This property accesses the prompt's label fore (font) color.

Syntax: RFPrompt(Nbr).LabelForeColor = IValue

Alternate: vValue = RFPrompt(Nbr).LabelForeColor

Nbr (Variant) is numeric, or fieldname, and refers to a specific prompt. Note: Using the number 0 will refer to the application, not the prompt. The font of the heading on the application will be affected.

vValue (Variant) is the color.

IValue (Long) sets the color.

Examples:

```
RFPrompt(2).LabelForeColor = vbWhite  
RFPrompt(0).LabelForeColor = vbGreen  
RFPrompt("Part").LabelForeColor = RGB(255,255,0)  
'yellow  
RFPrompt("Part").LabelForeColor = "&HFF0000"  
'blue  
RFPrompt("Part").LabelForeColor = QBColor(5)  
'magenta
```

RFPrompt().LabelLeft

This property accesses the column position for a caption associated with a specific prompt. This value can be changed at run time if there is a need to move prompts around on the screen.

Syntax: RFPrompt(Nbr).LabelLeft = IValue

Alternate: vValue = RFPrompt(Nbr).LabelLeft

Nbr (Variant) is numeric, or fieldname, and refers to a specific prompt.

vValue (Variant) is the starting column for a prompt's label object.

IValue (Long) sets the starting column for a prompt's label object.

Examples:

```
Dim Value As Variant  
Value = RFPrompt(2).LabelLeft  
RFPrompt("Part").LabelLeft = 1
```

RFPrompt().LabelTop

This property accesses the row position for a label object associated with a specific prompt. This value can be changed at run time if there is a need to move prompts around on the screen.

Syntax: RFPrompt(Nbr).LabelTop = IValue

Alternate: vValue = RFPrompt(Nbr).LabelTop

Nbr (Variant) is numeric, or fieldname, and refers to a specific prompt.

vValue (Variant) is the row to display a prompt's label object.

IValue (Long) sets the row to display a prompt's label object.

Examples:

```
Dim Value As Variant  
Value = RFPrompt(2).LabelTop  
RFPrompt("Part").LabelTop = 3
```

RFPrompt().Length

This property accesses the Width property for a prompt object. This value can be changed at run time if there is a need to change prompt sizes on the screen.

Syntax: RFPrompt(Nbr).Length = IValue

Alternate: vValue = RFPrompt(Nbr).Length

Nbr (Variant) is numeric, or fieldname, and refers to a specific prompt.

vValue (Variant) is the display length for a prompt object.

IValue (Long) sets the display length for a prompt object.

Examples:

```
Dim value As Variant  
value = RFPrompt(2).Length  
RFPrompt("Part").Length = 10
```

RFPrompt().List

This property contains the RFgen list collection. The list collection can be generated via the AddItem method or with the DB.MakeList or the App.MakeList functions and then assigned to the list box, combo box or menu via this property.

Syntax: RFPrompt(Nbr).List = vMyList

Alternate: sMyList = RFPrompt(Nbr).List

Nbr (Variant) refers to the prompt number or name of the list box

sMyList (String) the list contained in the prompt

vMyList (Variant) the list generated with the DB.MakeList or App.MakeList functions.

Example:

The following uses the DB.MakeList function in the ListBox_GotFocus event to populate the List property of the list box.

```
Dim sMyList As String  
Dim sSQL As String  
sSQL = "select PartNo from Inventory"  
sMyList = DB.MakeList(sSQL)  
RFPrompt(4).List = sMyList
```

RFPrompt().ListCount

This function returns the number of items in the list collection.

Syntax: Value = RFPrompt(Nbr).ListCount

Nbr (Variant) refers to the prompt number or name of the list box
Value (Variant) the number of items in the list

Example:

```
Dim iValue as Integer  
iValue = RFPrompt("Listbox1").ListCount
```

RFPrompt().ListIndex

This property returns or sets the current list index property.

Syntax: RFPrompt(Nbr).ListIndex = IValue

Alternate: vValue = RFPrompt(Nbr).ListIndex

Nbr (Variant) refers to the prompt number or name of the list box

vValue (Variant) gets the item index in the list

IValue (Long) sets the item index in the list

Examples:

```
Dim iValue as Integer  
iValue = RFPrompt("Listbox1").ListIndex  
RFPrompt("Listbox1").ListIndex = 5
```

RFPrompt().PageDown

This method scrolls the contents of a list object prompt down 1 page.

Syntax: RFPrompt(Nbr).PageDown

Nbr (Variant) refers to the prompt number or name of the list box

Example: See *RFPrompt().PageUp*

RFPrompt().PageNo

This property accesses the page number property associated with a specific prompt. This property is read-only at run time.

Syntax: Page = RFPrompt(Nbr).PageNo

Nbr (Variant) is numeric, or fieldname, and refers to a specific prompt.

Page (Variant) is numeric and represents the page number on which the prompt will be displayed.

Example:

```
Dim Value as Integer  
Value = RFPrompt(6).PageNo
```

RFPrompt().PageUp

This method scrolls the contents of a list object prompt up 1 page.

Syntax: RFPrompt(Nbr).PageUp
Nbr (Variant) refers to the prompt number or name of the list box

Example:

This uses a Form_OnFkey event to activate the Listbox Page methods.

```
If Fkey = 5 Then
    RFPrompt(4).PageUp
ElseIf Fkey = 6 Then
    RFPrompt(4).PageDown
End If
```

RFPrompt().Password

This property, when set to True or False, will turn on or off asterisks (*) in the data field of a textbox that mask the data.

Syntax: RFPrompt(Nbr).Password = Value

Nbr (Variant) refers to the prompt number or name of the textbox
Value (Boolean) set to True or False

Example:

```
RFPrompt("Password").Password = True
```

RFPrompt().RemoveItem

This method is used to manually remove items from a list object prompt. If the third entry is removed all entries further down are shifted up 1 place.

Syntax: RFPrompt(Nbr).RemoveItem (Location)

Nbr (Variant) refers to the prompt number or name of the list box
Location (Variant) the location to remove the item from the list

Example:

```
RFPrompt(4).RemoveItem (3) ' Removes the third item
in the list.
```

RFPrompt().Required

This property, when set to 'True', forces users to enter data into the prompt, while setting it to False allows users to skip the field. If the prompt never gets the focus RFgen will not have a chance to use this property.

Syntax: RFPrompt(Nbr).Required = Value

Alternate: Value = RFPrompt(Nbr).Required

- Nbr (Variant) is numeric, or fieldname, and refers to a specific prompt.
Value (Boolean) is the required state for the prompt.

Examples:

```
Dim Value As Boolean  
Value = RFPrompt(1).Required  
RFPrompt("Part").Required = True
```

RFPrompt().ScrollDown

This method scrolls the contents of a list object prompt down 1 item.

Syntax: RFPrompt(Nbr).ScrollDown

Nbr (Variant) refers to the prompt number or name of the list box

Example: See RFPrompt().ScrollUp

RFPrompt().ScrollUp

This method scrolls the contents of a list object prompt up 1 item.

Syntax: RFPrompt(Nbr).ScrollUp

Nbr (Variant) refers to the prompt number or name of the list box

Example:

This uses a Form_OnFkey event to activate the List box Scroll methods.

```
Public Sub Form_OnFkey(Fkey As Long)  
On Error Resume Next  
If Fkey = 5 Then  
    RFPrompt(4).ScrollUp  
ElseIf Fkey = 6 Then  
    RFPrompt(4).ScrollDown  
End If  
End Sub
```

RFPrompt().SelLength

In graphical mode, this property sets or returns the number of characters highlighted by the mouse for a specified text prompt. For example, if the text box contained '123456' and using the mouse you drag and select only '456', the property will return 3, the number of characters selected.

Syntax: vValue = RFPrompt(Nbr).SelLength

Alternate: RFPrompt(Nbr).SelLength = iValue

- Nbr (Variant) is numeric, or fieldname, and refers to a specific prompt.
vValue (Variant) returns the numeric value
iValue (Integer) sets the numeric value

Example:

```
Dim Value as Variant  
Value = RFPrompt("Part").SelLength
```

RFPrompt().SelStart

In graphical mode, this property sets or returns the position where the highlighting starts for a specified text prompt. For example, if the text box contained '123456' and using the mouse you drag and select only '56', the property will return 4, the position where the selection started.

Syntax: vValue = RFPrompt(Nbr).SelStart

Alternate: RFPrompt(Nbr).SelStart = iValue

- Nbr (Variant) is numeric, or fieldname, and refers to a specific prompt.
vValue (Variant) returns the numeric value
iValue (Integer) sets the numeric value

Example:

```
Dim Value as Variant  
Value = RFPrompt("Part").SelStart
```

RFPrompt().Sorted

This property indicates whether the items of the list box are to be sorted alphabetically. Numbers are sorted alphabetically, not numerically. Therefore 1, 10, 100, 2, 3, 4 are numbers sorted alphabetically.

Syntax: RFPrompt(Nbr). Sorted = Value

Alternate: Value = RFPrompt(Nbr). Sorted

- Nbr (Variant) refers to the prompt number or name of the list box
Value (Boolean) set to True to sort the items in ascending order

Example:

```
RFPrompt("PartList").Sorted = True
```

RFPrompt().Stretch

For the Image control, this property set to True will stretch the image to fit the existing size of the Image control. If the control is too small and this property is set to False, only the upper left portion of the image will be visible.

Syntax: RFPrompt(Nbr).Stretch = Value
Alternate: Value = RFPrompt(Nbr).Stretch
Nbr (Variant) is numeric, or fieldname, and refers to a specific prompt.
Value (Boolean) True or False is on or off.

Example:

```
RFPrompt("Image1").Stretch = True
```

RFPrompt().Text

This property accesses the data contained by a text box object associated with a specific prompt. Other prompts also use this property such as the combo box and list box objects. The property will return the highlighted entry in the list.

Syntax: RFPrompt(Nbr).Text = Value
Alternate: Value = RFPrompt(Nbr).Text
Nbr (Variant) is numeric, or fieldname, and refers to a specific prompt.
Value (Variant) is the data collected in a prompt object.

Examples:

```
Dim Value As Variant  
Value = RFPrompt(2).Text  
RFPrompt("Message").Text = "Hello"
```

RFPrompt().TextLeft

This property accesses the column position for the data field portion of a prompt object. This value can be changed at run time if there is a need to move prompts around on the screen.

Syntax: RFPrompt(Nbr).TextLeft = lValue
Alternate: vValue = RFPrompt(Nbr).TextLeft
Nbr (Variant) is numeric, or fieldname, and refers to a specific prompt.
vValue (Variant) is the starting column for a prompt's data field.
lValue (Long) sets the starting column for a prompt's data field.

Examples:

```
Dim Value As Integer  
Value = RFPrompt(3).TextLeft  
RFPrompt("Part").TextLeft = 12
```

RFPrompt().TextTop

This property accesses the row position for a text box object associated with a specific prompt. This value can be changed at run time if there is a need to move prompts around on the screen.

Syntax: RFPrompt(Nbr).TextTop = IValue

Alternate: vValue = RFPrompt(Nbr).TextTop

Nbr (Variant) is numeric, or fieldname, and refers to a specific prompt.

vValue (Variant) is the row for a prompt's data field.

IValue (Long) sets the row for a prompt's data field.

Examples:

```
Dim Value As Integer  
Value = RFPrompt(3).TextTop  
RFPrompt("Part").TextTop = 5
```

RFPrompt().Type

This property will return a read-only enumeration describing what type of prompt is defined based on the prompt index name or number.

Syntax: Value = RFPrompt(Nbr).Type

Value (enFieldType) the enumeration for the prompt type. Values are:

- rfButton
- rfCheckBox
- rfComboBox
- rfForm
- rfFrame
- rfHostScreen
- rfImage
- rfLabel
- rfListBox
- rfMenu
- rfOptions
- rfScrapeField
- rfSignature
- rfTextBox
- rfVocollectTask

Nbr (Variant) is numeric, or fieldname, and refers to a specific prompt.

Example:

```
Dim oValue As enFieldType  
oValue = RFPrompt(3).Type  
If oValue = rfLabel Then Exit Sub
```

RFPrompt().ValField

This property accesses the validation field property associated with a specific prompt. When using ValTable and ValField properties, you must code them specifically with ValTable first and then ValField. When changing the validation field, you must make sure the validation table contains the new validation field. See RFPrompt().ValTable.

Syntax: RFPrompt(Nbr).ValField = vValue

Alternate: sValue = RFPrompt(Nbr).ValField

Nbr (Variant) is numeric, or fieldname, and refers to a specific prompt.

sValue (String) is the validation field in the Table specified by RFPrompt().ValTable.

vValue (Variant) sets the validation field to access whenever data is entered at the prompt.

Examples:

```
Dim Value As Variant  
Value = RFPrompt(1).ValField  
'  
RFPrompt("Part").ValTable = "Inventory"  
RFPrompt("Part").ValField = "PartNo"
```

RFPrompt().ValTable

This property accesses the validation table property associated with a specific prompt. When using ValTable and ValField properties, you must code them specifically with ValTable first and then ValField. When changing the validation table, you must make sure the validation field is contained in the new validation table. See RFPrompt().ValField.

Syntax: RFPrompt(Nbr).ValTable = vValue

Alternate: sValue = RFPrompt(Nbr).ValTable

Nbr (Variant) is numeric, or fieldname, and refers to a specific prompt.

sValue (String) is the validation table to access whenever data is entered at the prompt.

vValue (Variant) sets the validation table to access whenever data is entered at the prompt.

Examples:

```
Dim value As Variant  
value = RFPrompt(1).ValTable  
'  
RFPrompt("Part").ValTable = "Inventory"  
RFPrompt("Part").ValField = "PartNo"
```

RFPrompt().Visible

This property accesses the visible property associated with a specific prompt. Setting it to True makes a prompt visible, while setting it to False makes it invisible. Even though the prompt may be invisible, the GotFocus and OnEnter events will still be executed for this prompt. The screen must either be manually refreshed immediately or the event allowed to finish before the screen will be updated.

Syntax: RFPrompt(Nbr).Visible = Value

Alternate: Value = RFPrompt(Nbr).Visible

Nbr (Variant) is numeric, or fieldname, and refers to a specific prompt.

Value (Boolean) is the visible state for the prompt.

Examples:

```
Dim Value As Boolean  
Value = RFPrompt(1).Visible  
RFPrompt("Part").Visible = False
```

Application-Based Extensions

Application-based language extensions are a group of commands that form the backbone of an application or transaction macro. DataMAX recommends looking in this group first for general commands that are not specific to mobile devices, screen mapping or other specific functions.

CallForm

This command will transfer the user to another application.

Syntax: App.CallForm(Name, [Return], [SaveRSP], [InitScreen])

Name (String) is the name of the application to call.

Return (Boolean) Optional – set to True to return to the current application after collecting data in the called application. The default is False.

SaveRSP (Boolean) Optional – set to True to retain the value in the current prompt's text field. The default is False.

InitScreen (Boolean) Optional – set to True to delete the screens memory object prior to going to the application. This is done in case the commands App.GetValue, App.SetValue, or App.ClearValues affected the data on this application prior to calling it. The default is False.

Example:

```
App.CallForm("Cycle", False) ' This will call  
application "Cycle" without returning.
```

CallMacro

This function is used to call any transaction macro and pass in the required passing parameters. It returns True if the macro was successfully completed. These macros can be created from the Transactions Tree.

Syntax: Value = App.CallMacro(MacroName, QueueOffline, [Params])

Value (enMacroResult) – Values are:

- MacroFailed
- MacroNotProcessed
- MacroQueued
- MacroSucceeded

MacroName (String) – This is the name of the macro to be called.

QueueOffline (Boolean) – This determines whether the macro can be queued for later processing if the host is not currently available.

Params (Variant) – Optional: A series of passing parameters as required by the selected macro. Note: these parameters are the fields you defined when you wrote the macro.

Example:

```
Dim Value As Variant  
Value = App.CallMacro("SaveData",True, "Sam", "1234")
```

CallMenu

This command will transfer the user to another menu. It is typically used to log a user into RFgen when bypassing the RFgen Logon process. If the application making the menu call was deep within the menu structure, RFgen will assume that the called menu is the root menu. Going back up 1 level from a called menu will take you to the Login screen if one exists. (Note: the new menu will be called only after the Visual Basic Event has been exited.)

Syntax: App.CallMenu(Name)

Name (String) is the name of the menu to call as defined by the Menus tree.

Example:

```
App.CallMenu("Sample") ' Calls the "Sample" menu.
```

ChangeLoginForm

This function will set the default login application for the duration of the client's session. The purpose is to allow several different login applications (possibly different languages) to exist at the same time and have the client's device specify which login application should be used. In the RFgen.bas module use the OnLocale event to get the client's location and then use App.ChangeLoginForm in that event to display the proper application for the user.

Syntax: App.ChangeLoginForm (Name)

Name (String) is the default login application name for the connected Client device.

Examples:

`App.ChangeLoginForm ("RFLoginSpanish")`

`App.ChangeLoginForm ("RFLoginForkLift")`

ChangeUserPassword

This command will change the user's password to a new value if the old password is correctly provided. This is a permanent change stored in the RFgen system.

Syntax: [bValue =] App.ChangeUserPassword(OldPwd, NewPwd)

bValue (Boolean) Optional – is True or False based on the success of the command.

OldPwd (String) is the user's current password

NewPwd (String) is the new password

Example:

`App.ChangeUserPassword("Mike123", "Mike456")`

ClearValues

This command will erase the values of all fields contained in the specified application. It is used to reset the values for applications that must be called more than once.

Syntax: App.ClearValues([Name])

Name (Variant) Optional – is the name of the application's prompt to erase.

Example:

`App.ClearValues("Cycle") ' Clears the values on the application titled "Cycle".`

ClientType

This function will return the type of client connecting in to the RFgen server.

Syntax: Type = App.ClientType

Type (String) "TE", "GUI", "MOBILE", "XML", "TMQUEUE", or
"TMEVENT"

ConnAvailable

This function will return a Boolean (True / False) response that indicates whether or not the specified data connection is currently working.

Syntax: Value = App.ConnAvailable(Source)

Value (Boolean) equals True if RFgen is currently connected to the specified source; equals False if it is not available

Source (Variant) is the string representation or the numeric representation of the data connection

Examples:

```
Dim bCheck As Boolean  
bCheck = App.ConnAvailable(1)  
bCheck = App.ConnAvailable("RFSample")
```

ErrClear

Clears all property settings of the App.Errxxx objects. Use this command after trapping and dealing with an error so the next error can be trapped.

Syntax: App.ErrClear

ErrCount

This property returns the number of errors. Occasionally an ODBC driver may return multiple errors and the first one may not be the most descriptive or accurate for solving the problem. Concatenating all the errors for the message box will give the most complete description of the problem.

Syntax: Value = App.ErrCount

Value (Variant) is the number of errors in the collection.

ErrDesc

This property returns a string description of the ODBC error. Using the App.ErrCount command you may specify the number of the error in a loop and extract each error for display to the user.

Syntax: Value = App.ErrDesc([Nbr])
Value (String) is the error description.
Nbr (Long) Optional – is the error number to view. Note: this value is zero-based and defaults to 1.

ErrNative

This property returns a numeric value specifying the native database error number.

Syntax: Value = App.ErrNative([Nbr])
Value (Variant) is the database error number.
Nbr (Long) Optional – is the error number to view. Note: this value is zero-based and defaults to 1.

ErrNo

This property returns a numeric value specifying the VBA error number.

Syntax: Value = App.ErrNo([Nbr])
Value (Variant) is the VBA error number.
Nbr (Long) Optional – is the error number to view. Note: this value is zero-based and defaults to 1.

ExecuteMenuItem

This command is typically used in conjunction with voice applications to allow the user to say either the index number or name of the item on the menu to be chosen. First the OnSpeech must capture what is spoken while the focus is on the menu prompt. Then this command can be used in the OnSpeech event to execute the menu item spoken. Either the index of the entry or the display text of the menu prompt's item to be selected and passed in to this command. If speech is not used this command will still work if you were trying to automate menu selections.

Syntax: App.ExecuteMenuItem(Selection)
Selection (Variant) either the menu index or display text of the menu prompt's item to be selected and executed.

Examples:

```
App.ExecuteMenuItem("Inventory Master")
App.ExecuteMenuItem(1)
```

ExitForm

This command exits the current application and returns to the calling menu or application. The VBA event where this is called must exit before this statement is carried out. Therefore it is always best to use and Exit Sub statement immediately after this command.

Syntax: App.ExitForm

ExitSession

This command exits the current device session completely, the same as entering 'Q' at the original login screen. The VBA event where this is called must exit before this statement is carried out. Therefore it is always best to use an Exit Sub statement immediately after this command.

Syntax: App.ExitSession

GetInput

This function will retrieve data from the input device that is not associated with any RFgen prompt. By specifying a set of X/Y coordinates, the cursor will blink at that position and accept data entry until the Length property is reached. This is similar in concept to a standard InputBox control.

Syntax: Value = App.GetInput(Column, Row, [Length])

Value (Variant) is the value that is entered/scanned by the user.

Column (Integer) is the column at which to place the cursor.

Row (Integer) is the row at which to place the cursor.

Length (Integer) Optional – is a maximum data entry length.

Example:

```
Dim Value As Variant  
Value = App.GetInput(0,12,5)
```

GetString

This function will return text from the Text Resources that have been saved as part of the configuration.

Syntax: Value = App.GetString(Id, [DefaultText])

Value (String) is the text string extracted from the resources.

Id (Variant) is the key to the text string based on a specific or current locale. Use Ctrl + Spacebar to see a list of possible ID to use.

DefaultText (Variant) Optional – is the text to be used if the named ID happens to be blank. User Ctrl + Spacebar to automatically insert the text for the specified ID. If the text should be cleared from the resource at a later time, the code will still have a default value.

Examples:

```
App.GetString("Err1") ' Gets the associated text  
App.GetString("Err1", "Invalid Part")
```

GetValue

This function will get the current value of a field in the current recordset or the value of a specific RFgen prompt.

Syntax: Value = App.GetValue(Field, [Application])

Value (Variant) is the value extracted. (Note: this will always be a string value regardless of the actual field's data type.)

Field (Variant) to extract the value from an RFgen prompt, or is the name of a field in the current RFgen recordset.

Application (Variant) Optional – is the name of the application's recordset to extract data from. This is typically used to extract data from a calling application's recordset.

Examples:

```
App.GetValue(2) ' Gets value from prompt #2  
App.GetValue("PartNo") ' Gets the value from "PartNo"  
App.GetValue(2,"Cycle") ' Gets the value from the  
second prompt on the "Cycle" application
```

IpAddress

This function will return the IP Address of the current Client device.

Syntax: Value = App.IpAddress()

Value (String) is the IP Address of the connected Client device.

Example:

```
Dim sAddress As String  
sAddress = App.IpAddress
```

Locale

This function can set or return the number associated with the current locale of the connecting device. This command accepts a number (LCID) and sets the locale within RFgen. Examples would be 1033 for United States and 1041 for Japan. A web search for 'locale codes' should provide a list of LCIDs.

Syntax: Value = App.Locale

Alternate: App.Locale = Value

Value (Long) is the number associated with the current locale

Example:

```
Dim nLocale As Long
```

```
nLocale = App.Locale  
App.Locale = 1041
```

.LogError

This extension writes an entry into the RFgen General Error Log file. The purpose for the percent signs is if you use this command in a global capacity and want to pass in parameters. Each percent sign is simply replaced with the next item in the Params list.

Syntax: App.LogError(Process, ErrDesc, [Params])

Process (String) is typically the application name

ErrDesc (String) is the ErrDesc entry

Params (string array) Optional – these are parameters that will be substituted into the ErrDesc where a percent sign is used

Examples:

```
App.LogError("Application1", "Invalid Item")
```

Time..: 3/27/2006 1:37:55 PM

Process: Application1

ErrDesc: Invalid Item

```
App.LogError("Application1", "Invalid Item by user  
%", App.User)
```

Time..: 3/27/2006 1:37:55 PM

Process: Application1

ErrDesc: Invalid Item by user Sam

```
App.LogError("Application1", "Invalid Item % % % ",  
"1", "2", "3")
```

Time..: 3/27/2006 1:37:55 PM

Process: Application1

ErrDesc: Invalid Item 1 2 3

MakeList

This function builds a scrolling list of items that may then be presented to the user for selection using App.ShowList.

Syntax: App.MakeList(MyList, Value, Display, [Heading])

MyList (String) is the variable containing the list being created.

Value (Variant) is the value that is returned when the user selects a specific list item.

Display (Variant) is what the user will see for a specific list item. (Non-numeric values must be in quotes.)

Heading (Variant) Optional – is the heading for the list. (Note: it only needs to be assigned once).

Example:

```
Dim sRsp As String
Dim MYLIST As String
App.MakeList MYLIST, 2000, "2000 ABC Company",
"Select order"
App.MakeList MYLIST, 2001, "2001 Widgets R Us"
App.MakeList MYLIST, 2002, "2002 GoodFellows Inc."
sRsp = App.ShowList(MYLIST)
```

MsgBox

This function clears the screen in character mode or displays a message box when in graphical mode to the user. The 'value', 'type', 'default', and 'buttons' variables match those of the Visual Basic MsgBox function; e.g.:

Syntax: Value = App.MsgBox(Message, [Type], [Default], [Buttons])

Value (Variant) will contain an integer or string indicating which selection was made. A string is the result only when custom buttons are defined.

vbOK =OK button
vbCancel =Cancel button
vbAbort=Abort button
vbRetry =Retry button
vbIgnore = Ignore button
vbYes =Yes button
vbNo =No button

Message (String) the message text to be displayed on the device

Type (enMsgBoxTypes) Optional – the expected responses:

vbOkOnly = OK
vbOkCancel = OK, Cancel
vbAbortRetryIgnore = Abort, Retry, Ignore
vbYesNoCancel = Yes, No, Cancel
vbYesNo = Yes, No
vbRetryCancel = Retry, Cancel
vbCritical – no impact on the displayed message box
vbCustom – no impact on the displayed message box
vbExclamation – no impact on the displayed message box
vbInformation – no impact on the displayed message box
vbQuestion – no impact on the displayed message box

Default (Integer) Optional – an optional message box default:

1 = First Button
2 = Second Button
3 = Third Button

Buttons (Variant) Optional – captions for the buttons;
“[A] [B]” will set “A” as the caption for the
first button, “B” for the second.

Examples:

```
Dim Value As Integer
Value = App.MsgBox("Continue with transaction?", vbYesNo, 1)

Dim Value as Variant
Value = App.MsgBox("Continue with transaction?", vbYesNo, 1, "[Good] [Bad]")

If App.MsgBox("OK?", vbYesNo) = vbNo Then Exit Sub
```

The result in Value will be a string containing the label of the button pressed only if custom buttons are used. Otherwise the button number is returned. Defining unique buttons returns the name of the selected button.

PromptCount

This function will return the number of prompts that exist in an application. It is typically used when looping through the prompts to set properties like Visible.

Syntax: Value = App.PromptCount

Value (Variant) the number of prompts in the current application

Example:

```
Dim iVal As Integer
iVal = App.PromptCount
```

PromptNo

This property indicates the prompt number of the prompt that has the focus (read only).

Syntax: Nbr = App.PromptNo

Nbr (Integer) the number of the current prompt

SendChar

This function sends an ASCII character directly to the control on the screen as if it came from the keyboard. RFgen does not intercept the character and process it. It is similar to the VB ‘SendKeys’ command.

Syntax: App.SendKey(KeyASCII)

KeyASCII (Long) the ASCII value of the character

Example:

```
App.SendKey(43) ' this would send the * character
```

SendKey

This function sends an ASCII character to the server as if it came from the keyboard. RFgen will process it if it has special meaning. For example, sending an F key will get processed in the Form_OnFkey event if it is programmed to do so. It is similar to the VB 'SendKeys' command.

Syntax: App.SendKey(KeyCode)

KeyCode (enKeyCodeConstants)

vbKey0 – vbKey9	vbKeyA - vbKeyZ
vbKeyF1 - vbKeyF16	vbKeyNumPad0 – vbKeyNumPad9
vbKeyAdd	vbKeyBack
vbKeyCancel	vbKeyCapital
vbKeyClear	vbKeyControl
vbKeyDecimal	vbKeyDelete
vbKeyDivide	vbKeyDown
vbKeyEnd	vbKeyEscape
vbKeyExecute	vbKeyHelp
vbKeyHome	vbKeyInsert
vbKeyLButton	vbKeyLeft
vbKeyMButton	vbKeyMenu
vbKeyNumlock	vbKeyPageDown
vbKeyPageUp	vbKeyPause
vbKeyPrint	vbKeyRButton
vbKeyReturn	vbKeyRight
vbKeyScrollLock	vbKeySelect
vbKeySeparator	vbKeyShift
vbKeySnapshot	vbKeySpace
vbKeySubtract	vbKeyTab
vbKeyUp	

Example:

```
App.SendKey(vbKeyReturn) ' this sends the enter key
```

SetDisplay

This function sets an alternative application display, if one has been established. Each display may have a variation of the application such as an alternate language used to make prompt captions. The prompts may have their font sizes changed to be seen more easily on large displays from a distance. The underlying code references the prompts

by their Field ID or number so the many displays of the application only need one code base. Based on the user profile or the IP address of the connecting device, the proper display can be automatically shown to the user.

Syntax: App.SetDisplay (Display, [Col], [Row])

Display (String) the name established for the alternative application display.

Col, Row (Integer) Optional – allows the user to get a display of an application based on the screen size of the application created if the Display name was not found.

For example, if you had these displays created:

- Default 20x16
- Spanish 20x16
- Forklift 40x8
- Forklift 20x4

Then the command App.SetDisplay("Forklift",20,4) would bring up the last display.

SetFocus

This command is used to reposition the focus to a specific prompt. App.SetFocus should be the last statement for an event since subsequent statements will be ignored.

If the optional SaveRSP parameter is used the data entered for the current prompt will be saved in the current record, prior to the App.SetFocus. The SaveRSP flag does not apply to the Rsp variable in the OnEnter event but will only save the text in the textbox itself. To alter the textbox value in code use the RFPrompt(1).Text property and give it a value.

The ValidateRSP flag will check the text value against the Edits property and also execute the OnEnter event. This makes the most sense when the SetFocus method is used in an event other than the OnEnter event. If the SetFocus method is used in the OnEnter event the ValidateRSP flag will not process the edits again or run through the OnEnter event a second time. Be sure not to cause one SetFocus command to execute another SetFocus command as this may lead to unintended results.

Syntax: [Value =]App.SetFocus (FieldId, [SaveRSP], [ValidateRSP])

Value (Boolean) Optional – Returns True or False depending on the success of the command

FieldId	(Variant) is the prompt's Field Id or number receiving the focus.
SaveRSP	(Boolean) Optional – set to True to save any changes to the current prompts value before switching focus to the new prompt.
ValidateRSP	(Boolean) Optional – set to True to call the edit checks and to re-run the OnEnter event.

Examples:

`App.SetFocus(5) ' Cursor will go to prompt #5`

`App.SetFocus("PartNo") ' Focus goes to PartNo prompt`

`App.SetFocus("PartNo", True, True) ' Cursor will go to "PartNo", save current Text property and execute edits and OnEnter event`

SetMenu

This function will set the default menu for the current Client device. This is typically used to set an initial menu for a non-RFgen defined user (i.e., you are bypassing RFgen's normal login process.)

Syntax: `App.SetMenu(Name)`

Name (String) is the default menu name for the connected Client device.

Example:

`App.SetMenu("Sample")`

SetOption

This function allows multiple environmental functions to be implemented.

Syntax: `SetOption(Option,Value)`

Option (enAppOptions) is a predefined option presented as a list of options by RFgen.

Value (Variant) is specific to the type of system option chosen

SetOption(SetBarcodeScanner,Value)

This command turns on or off the barcode scanner of the mobile device. Not all manufacturers are supported. Using the Device Profile options in the Mobile Development Studio, choose the manufacturer first and then the Barcode Driver option if a non-native driver exists.

Syntax: `SetOption(SetBarcodeScanner,Value)`

Value (Boolean) is True or False for enabling or disabling the scanner.

SetOption(SetDefaultBackColor,Value)

For the CE device, this command will set the Back Color of all applications until it is changed. Examples of valid values are:

vbRed	red
RGB(255,255,0)	yellow
"&HFF0000"	blue
QBColor(5)	magenta

SetOption(SetDefaultFontSize,Value)

For the CE device, this command will set the Font Size of all the text on the applications until it is changed. An example of a valid value would be the number 12.

SetOption(SetDefaultForeColor,Value)

For the CE device, this command will set the Fore Color of all the text on the applications until it is changed. Examples of valid values are the same as the Back Color option.

SetOption(SetListBackColor, Value)

This command sets the back color of the App.ShowList displayed on a CE device. Examples of valid values are:

vbRed	red
RGB(255,255,0)	yellow
"&HFF0000"	blue
QBColor(5)	magenta

SetOption(SetListFontSize,Value)

This command sets the font size of the text when the App.ShowList is displayed on a CE device.

SetOption(SetListForeColor, Value)

This command sets the fore color of the App.ShowList displayed on a CE device. Examples of valid values are:

vbRed	red
RGB(255,255,0)	yellow
"&HFF0000"	blue
QBColor(5)	magenta

SetOption(SetMsgBoxBackColor, Value)

This command sets the back color of the App.MsgBox displayed on a CE device. Examples of valid values are:

vbRed	red
RGB(255,255,0)	yellow

“&HFF0000”	blue
QBColor(5)	magenta

SetOption(SetMsgBoxFontSize,Value)

This command sets the font size of the text in an App.MsgBox displayed on a CE device.

SetOption(SetMsgBoxForeColor, Value)

This command sets the fore color of the App.MsgBox displayed on a CE device. Examples of valid values are:

vbRed	red
RGB(255,255,0)	yellow
“&HFF0000”	blue
QBColor(5)	magenta

SetOption(ShowHorizontalScrollBar, True)

In the graphical environment, this command places a scroll bar across the bottom of the screen allowing the user to scroll the screen left and right. This scroll bar will not disappear until the same command with a “False” option is given and will stay on the screen even after leaving the application that activated it.

SetOption(ShowVerticalScrollBar, True)

In the graphical environment, this command places a scroll bar along the right side of the screen allowing the user to scroll the screen up and down. This scroll bar will not disappear until the same command with a “False” option is given and will stay on the screen even after leaving the application that activated it.

SetOption(MenuMsgBackupText, "Previous Menu")

When the option to go back to the previous menu is displayed on the submenu, entering any string in the last parameter can customize the menu item’s text. The default is “<Back>”.

SetOption(MenuMsgBackupVisible, True)

This command either displays or hides the menu option that takes the user back from a submenu to the previous menu or main menu. To turn off or hide the menu option, use the “False” option.

SetOption(MenuMsgLogoutText, "Login Screen")

When the option to go back to the RFgen Login screen is displayed on the main menu, entering any string in the last parameter can customize the menu item’s text. The default is “<Logout>”.

SetOption(MenuMsgLogoutVisible,True)

This command either displays or hides the menu option that takes the user back to the RFgen Login screen. To turn off or hide the menu option, use the “False” option.

SetValue

This command will set the value of a field in the current recordset or of a specific RFgen prompt.

Syntax: App.SetValue(Field, Value, [Name])

Field (Variant) to set the value or an RFgen prompt, or is the name of a field in the current RFgen recordset.

Value (Variant) is the value to insert into the current recordset. (Note: all values are treated as strings until passed to the database.)

Name (Variant) Optional – is the name of the application's recordset to update. This is typically used to insert data into a calling application's recordset.

Examples:

```
App.SetValue(1, "FX1") ' Puts "FX1" in prompt #1.  
App.SetValue("PartNo", "FX1", "Cycle")
```

Puts "FX1" in "PartNo" prompt on "Cycle" application.

ShowList

This function causes a scrolling list to be displayed on the Client device and allows the user to make a selection. (Maximum entries: 32,767.)

Syntax: Value = App.ShowList(MyList, Force Display)

Value (Variant) is the value that is returned when the user selects a specific list item.

MyList (String) is the list to be returned for display (i.e., set RSP = Value to display the list on the Client device.)

Force Display (Boolean) Use True or False to specify sending this command immediately rather than at the end of the current event. The default is False.

Example: See *App.MakeList* and *DB.MakeList*

Sleep

This command will cause the Client device to sleep for the specified number of seconds. (Note: any activity by the user will terminate the sleep mode.) The Visual Basic Wait command may be a viable alternative.

Syntax: App.Sleep(Seconds)

Seconds (Long) is the number of seconds to suspend program operation. This is usually used to flash a message to the user, and then to erase it. (Note: The new timer feature is recommended for time related programming.)

Example:

```
App.Sleep(1)
```

TimerEnabled

This command will enable / disable the ‘Form_OnTimer’ event within RFgen. It is a good idea to always disable the application level timer event when exiting the application. Use the Form_Unload event or just before any App.CallForm command.

Note: see the Mobile Development Studio sample ‘FieldService’ VBA script for an example.

Syntax: App.TimerEnabled = Value

Alternate: Value = App.TimerEnabled

Value (Boolean) is the state of the RFgen timer event. Set to ‘True’ to enable timer functions.

Example: See *TimerInterval*

TimerInterval

This command sets the interval for the Form_OnTimer event.

Syntax: App.TimerInterval = Value

Alternate: Value = App.TimerInterval

Value (Long) is the number of milliseconds between timer events; i.e., for 10 seconds, set TimerInterval = 10000.

Example:

```
App.TimerEnabled = True
```

```
App.TimerInterval = 1000
```

User

This function can set or return the user of the current Client device. The default login screen sets this user. This id will then be used by RFgen SQL pre-processor and by the Console panel.

Syntax: sUser = App.User()

Alternate: App.User = sUser

sUser (String) is the current user of the connected Client device.

Examples:

```
Dim sUser As String  
sUser = App.User  
App.User = "SAM"
```

UserProperty

This function can set or return the value of the specified property on the currently logged in user's profile. This command references the App.User command to get the correct user and then looks up the parameter specified.

Syntax: Value = App.UserProperty(Property)

Alternate: App.UserProperty(Property) = Value

Value (Variant) is the value stored under the property's value on the Users profile.

Property (Variant) is the property added to the current user's profile

Examples:

```
Dim UserAge As Variant  
UserAge = UserProperty("Age")  
UserProperty("Language") = "English"
```

SQLNum

This command is not part of the App object but falls into the same category. This command will take a specified number, remove any decimals and replace commas with decimals. If you know a certain number format will not work in an SQL statement because the database expects English formatted numbers, use this command.

Syntax: Result = SQLNum(Value)

Result (String) the converted string value suitable for use in SQL statements.

Value (Variant) a string or numeric value containing a non-English number representation that needs to be converted.

Example:

```
Dim sValue As String  
sValue = SQLNum("123.456,78") ' returns "123,456.78"
```

Screen Display Extensions

Screen display commands typically interact with the user at the GUI level. These commands are used to print or remove text from the screen or gather information about the screen size.

Bell

This command will cause the Client device terminal to beep. For telnet devices, the parameter is the number of times to sound the beep. For WinCE devices, this is the waveform to play. Windows CE specifies 5 default waveforms whose values are 0, 16, 32, 48 and 64. Each of these sounds must be setup on the CE device.

Syntax: Screen.Bell([Nbr])

Nbr (Integer) Optional – is the bell sound.

Examples:

```
Screen.Bell(3)  ' Telnet mode - bell rings 3 times.  
Screen.Bell(0)  ' Graphical mode - OK waveform will  
play.  
Screen.Bell(16) ' Graphical mode - Hand waveform will  
play.  
Screen.Bell(32) ' Graphical mode - Question waveform  
will play.  
Screen.Bell(48) ' Graphical mode - Exclamation  
waveform will play.  
Screen.Bell(64) ' Graphical mode - Asterisk waveform  
will play.
```

Clear

This command will clear the Client screen until RFgen needs to refresh the screen or unless a Screen.Refresh command is used. That will bring back any prompts that were in the area. Any text placed there with a Screen.Print command will be removed permanently.

Syntax: Screen.Clear([SendNow])

SendNow (Boolean) Optional – clears the screen immediately if this option is set to True.

ClearEOL

The Screen.ClearEOL command will clear the Client screen from the current position to the end-of-line. Using a Screen.Refresh will bring back any prompts that were in the area. Any text placed there with a Screen.Print command will be removed permanently.

Syntax: Screen.ClearEOL

ClearEOP

This command will clear the text boxes of all user-entered fields on the devices terminal screen from the current position to the bottom of the

screen. Using a Screen.Refresh will bring back any prompts that were in the area. Any text placed there with a Screen.Print command will be removed permanently.

Syntax: Screen.ClearEOP

DrawLine

This command will draw a vertical line, horizontal line, or create a box on device screen. If you want to display boxes within boxes, you must draw them from the inside to the outside. This command works in both character and graphical mode.

Syntax: Screen.DrawLine(StartCol, StartRow, EndCol, EndRow)

StartCol (Long) is the column (x coordinate) in which to start drawing the line/box.

StartRow (Long) is the row (y coordinate) in which to start drawing the line/box

EndCol (Long) is the column (x coordinate) in which to finish drawing the line/box.

EndRow (Long) is the row (y coordinate) in which to finish drawing the line/box.

Examples:

```
Screen.DrawLine(0,10,20,10) ' Draws a horizontal line  
Screen.DrawLine(0,0,0,16)   ' Draws a vertical line  
Screen.DrawLine(0,0,20,16) ' Draws a box around the  
screen
```

Height

Returns the height in pixels or rows for the current application. (See Screen.Width).

Syntax: Value = Screen.Height

Value (Variant) equals the number of available rows in the current application or the height in pixels.

Print

This command will move the cursor to a specified row and column, print text in normal or reverse video, and optionally clear to the end-of-line. In graphical mode, the permanent objects, prompts on the screen, will take precedence and the text will display behind the prompt. Making a prompt invisible and then using Screen.Print may be an alternate solution. As with all 'screen-printing' commands, you cannot use this command in the Form_Load event because the application is not created until the Form_Load event is finished.

Syntax: Screen.Print(Column, Row, Text, [Reverse], [ClearEOL], [SendNow])
Column (Long) is the column (x coordinate) in which to place the cursor.
Row (Long) is the row (y coordinate) in which to place the cursor.
Text (Variant) is the text to print at the current row and column.
Reverse (Boolean) Optional – set to True to print text in reverse video.
This only applies to a character based device, not graphical.
ClearEOL(Boolean) Optional – set to True to clear to the end-of-line after printing text.
SendNow(Boolean) Optional – set to True to send the print packet to the Client device immediately

Example:

```
Screen.Print(0, 10, "F4 to Exit", , True, True)
```

Refresh

This command will clear the Client screen and then repaint all standard RFgen prompts and data. (Note: any text displayed using Screen.Print statements will not be redisplayed.)

Syntax: Screen.Refresh

ResetCursor

This command is for internal use only and has no use for the user.

ReverseOff

This command is for internal use only and has no use for the user.

ReverseOn

This command is for internal use only and has no use for the user.

Width

Returns the width in pixels or columns for the current application. (See Screen.Height).

Syntax: Value = Screen.Width

Value (Variant) equals the number of available columns in the current application or the width in pixels.

Example:

```
If App.ClientType = "TE" Then  
RFPrompt("Description").Length = Screen.Width
```

In this case, the text box containing the description will be the same width as the current application. Since the graphical mode would return pixels, setting the length of a textbox to a very large number would not be appropriate.

Soft Input Panel Extensions

The soft input panel (SIP) is the small keyboard window that pops up when a field on the mobile (graphical capable) device allows for text input. The display and characteristics of this panel can be controlled from the RFgen script.

GetCurrentType

This method returns a textual representation of the current input panel such as “Keyboard” or “Transcriber”.

Syntax: [Value =] SIP.GetCurrentType

Value (String) – is the variable containing the result.

Example:

```
Dim sVal as String  
sVal = SIP.GetCurrentType
```

In this case sVal will have a value like “Keyboard”.

GetTypes

This method returns a pipe (|) delimited list of available input types that are supported on the device. Not all devices will have the same list of available types. Here are a few examples:

Letter Recognizer	Block Recognizer	Phone Keypad
Compact QWERTY	Full QWERTY	WinCE Handwriting
WinCE Keyboard	Keyboard	Kana
Kensaku	Romaji	Tegaki

Syntax: List = SIP.GetTypes

List (String) – Contains a list of all available types of input that are supported on the device.

Example:

```
Dim sList as String
```

```
sList = SIP.GetTypes
```

Mode

This property is used to set one or modes together to create the proper input. For standard English use the Roman mode. For other variations a combination may be required.

Syntax: [OK =] SIP.Mode(Mode)

OK (Boolean) – Optional – Returns True if the command was successful.

Mode (enSIPMode) – are the individual properties that can combine to make up the proper input panel. They are:

- enSIP_CHARCODE
- enSIP_CHINESE
- enSIP_FULLSHAPE
- enSIP_HANGUL
- enSIP_JAPANESE
- enSIP_KATAKANA
- enSIP_LANGUAGE
- enSIP_NATIVE
- enSIP_ROMAN

Examples:

```
Dim OK as Boolean  
OK = SIP.Mode(enSIP_ROMAN)  
OK = SIP.Mode(enSIP_ROMAN or enSIP_FULLSHAPE or  
enSIP_KATAKANA or enSIP_NATIVE)
```

In the second example, several modes are ORed together to create the Katakana input panel.

SetType

This method sets the current input panel. The Input Method parameter can either be text as it is returned in the GetTypes method or it can be a zero-based number referring to the same list. This list is the same as the dropdown list on the mobile device where you choose between the styles of recognition.

Syntax: [OK =] SIP.SetType(IM)

OK (Boolean) – Optional – Returns True if the command was successful.

IM (Variant) – set to a number or a string representing one of the available recognition styles.

Examples:

```
Dim OK as Boolean  
OK = SIP.SetType(0)' 0 may not represent the keyboard  
OK = SIP.SetType("Keyboard")
```

Show

This method is used to display or hide the SIP.

Syntax: [OK =] SIP.Show(Show)

OK (Boolean) – Optional – Returns True if the command was successful.

Show (Boolean) – set to True for the SIP to display, set to False for the SIP to be hidden.

Example:

```
Dim OK as Boolean  
OK = SIP.Show(True)
```

Server-Based Extensions

Server-based commands are used by the mobile device when not in a Thin-client state to send and receive data to and from the RFgen server.

CallMacro

This function is used to call a screen mapping or transaction macro and pass the macros required parameters. Using this function gives you the ability to “store-and-forward” transactions while the host is off-line for backup or other reasons. It returns ‘True’ if the macro was successfully completed.

Syntax: [OK =] Server.CallMacro(MacroName, QueueOffline, [Params])

OK (enMacroResults) Optional – Returns 1 of the following 4 values:

- MacroFailed
- MacroNotProcessed
- MacroQueued
- MacroSucceeded

MacroName (String) – This is the name of the macro to be called.

QueueOffline (Boolean) – This determines whether the macro can be queued for later processing if the host is not currently available.

Params (Variant) – Optional: A series of passing parameters as required by the selected macro. Note: these parameters are the fields you defined when you record the macro.

Example:

```
Dim OK As enMacroResults  
OK = Server.CallMacro("IMASTER", True, "Sam", "1234")
```

CommandTimeout

This command limits the number of seconds any Server command waits for a response from the server. All Server commands will return with a response or failure message no later than the specified number of seconds. Set the parameter to zero (0) to disable this feature and go back to the RFgen default for each command.

Syntax: Server.CommandTimeout(Timeout)

Timeout (Long) parameter to specify a number of seconds

Example:

```
Server.CommandTimeout(10)
```

Connect

This command connects to the remote RFgen server via TCP/IP. Default values are stored in the registry when the files are installed.

Syntax: [OK =] Server.Connect([Connection], [PortNum])

OK (Boolean) Optional – return a True if the CE device is able to make a connection to the host within 30 seconds or less.

Connection (Variant) Optional – to specify either an IP address or the connection number (1-4) as listed in RFgenCFG. If this parameter is not specified, the default from a previous use of Server.Connect or the registry will be used.

PortNum (Long) Optional – to specify which server port number should be used. If the server's port number is not specified, the registry setting will be used.

Examples:

```
Dim bOK as Boolean  
bOK = Server.Connect  
bOK = Server.Connect("192.168.0.1", 21097)
```

Disconnect

This command disconnects from the remote RFgen server. You should always use this command when you know the connection is no longer needed. Using the Form_Unload event may be a good place.

Syntax: Server.Disconnect

ExecuteSQL

This function will execute a pass-through SQL statement on the host through the connection maintained by RFgen. Any results from the statement will be returned in a text-delimited object. (Note: this means that the item is a static view of the database and cannot be updated.)

Syntax: [OK =] Server.ExecuteSQL(SQL, [Columns], [Rows])

OK (Boolean) Optional – is a return value; a value of True means the SQL statement processed normally.

SQL (String) is the SQL statement to be sent to the database. As this is a pass through statement, its syntax must be understood by the database, as no pre-processing will occur.

Columns (Variant) – Optional – is a string representation of the columns returned by the SQL statement.

Rows (Variant) – Optional – is a string representation of the static result of an SQL statement.

Example:

```
Dim sSQL As String  
Dim sCols As String  
Dim sRows As String  
Dim bOK As Boolean  
sSQL = "select * from Inventory"  
bOK = Server.ExecuteSQL(sSQL, sCols, sRows)
```

In the case of an insert or an update, the sCols and sRows variables would not be necessary.

GetTable

This command executes the SQL statement on the remote server and copies the results to an existing local table.

Syntax: [OK =] Server.GetTable(SQL, LocalTable, [KeyFields])

OK (Boolean) Optional – is a return value; a value of True means the SQL statement processed normally.

SQL (String) is the SQL statement to be sent to the database. As this is a pass through statement, its syntax must be understood by the database, as no pre-processing will occur.

LocalTable (String) is the name of the table already created on the CE device.

KeyFields (Variant) Optional – is 1 or more key fields to represent a unique record. One key would be represented by “PartNo” and multiple keys look like “PartNo,Onhand”.

Example:

```
Dim bOK as Boolean  
bOK = Server.GetTable("Select * from Inventory",  
"Inv2", "PartNo")
```

Note: In the event that you need to update data using the Key field you should pay close attention to how the table has been populated. To optimize update performance you should try to match the order in which the data was populated in the table initially. For instance assume table INVENTORY has a unique ID column. Also assume that you will need to update a small subset of rows on the device database using the Server.GetTable() language extension. You would want to initially populate the table using the GetTable function with a SQL statement similar to:

```
Server.GetTable("SELECT * FROM INVENTORY ORDER BY Id", "INVENTORY") .
```

To optimize performance while updating records you would want to match the data that way it was downloaded by using the ORDER BY clause in your update like:

```
Server.GetTable("SELECT Id, Column1, Column2 FROM INVENTORY WHERE LastWrite > '07/22/2010' ORDER BY Id", "INVENTORY", "Id")
```

The ORDER BY clause will ensure that the records in the initial download and the updates are in the same order and will optimize the databases ability to seek to the next record to update.

IsConnected

This command returns a Boolean value of True or False depending on if the Server.Connect command was previously successfully executed.

Syntax: OK = Server.IsConnected

OK (Boolean) a value of True means there is already an open connection to use.

Example:

```
Dim bOK as Boolean  
bOK = Server.IsConnected
```

Ping

This command returns a True / False regarding the ability to reach the server. Based on the result you may choose to continue with the Server.Connect or go to a disconnected state. Default values are stored in the registry when the files are installed.

Syntax: OK = Server.Ping([Server])

OK (Boolean) returns a True if the mobile device could connect to the server.

Server (Variant) Optional – to specify the name or IP address of the server. If this is not specified, the registry settings will be used.

Example:

```
Dim bOK As Boolean  
bOK = Server.Ping("192.168.123.45")
```

QueueMacro

This function is used to queue any transaction macro and pass its required passing parameters directly to the server while in mobile mode. Unlike the TM.CallMacro with the queue property set to True, this command will not check for a valid host connection, or move to a linked screen. It returns 'True' if the macro was successfully queued. These macros can be created on the Transactions tree.

Syntax: [OK =] Server.QueueMacro(MacroName, [Params])

OK (Long) Optional – Returns the sequence number of the macro when it is successfully queued.

MacroName (String) – This is the name of the macro to be called.

Params (Variant) – Optional – A series of passing parameters as required by the selected macro. Note: these parameters are the fields you defined when you wrote the macro.

Example:

```
Dim lOK As Long  
lOK = Server.QueueMacro("ChangeUser", "Sam", "1234")
```

SendQueue

This function sends any locally queued transactions to the remote RFgen server queue for processing. Upon successfully transferring the local queue to the server, the local queue is cleared.

Syntax: [OK =] Server.SendQueue([DestQueue])

OK (Boolean) Optional – Returns True if the queued macros were successfully sent to the host for processing.

DestQueue (Variant) Optional – an option to send the queue to a queue with a different name

Example:

```
Dim bOK As Boolean  
bOK = Server.SendQueue
```

SendTable

This function executes the SQL statement locally and copies the results to an existing table on the remote RFgen server.

Syntax: [OK =] Server.SendTable(SQL, RemoteTable, [KeyFields])

OK	(Boolean) Optional – is a return value; a value of True means the SQL statement processed normally.
SQL	(String) is the SQL statement to be sent to the database. As this is a pass through statement, its syntax must be understood by the database, as no pre-processing will occur.
RemoteTable	(String) is the name of the table already created on the CE device.
KeyFields	(Variant) Optional – is one or more key fields to represent a unique record. One key would be represented by “PartNo” and multiple keys look like “PartNo,Onhand”.

Example:

```
Dim bOK as Boolean  
bOK = Server.SendTable("Select * from Inv2",  
"Inventory", "PartNo")
```

SetHost

This function sets the default settings for connecting to the RFgen host server. Typically Server.Connect follows this command.

Syntax: Server.SetHost(ServerName, Port)

ServerName (Variant) Optional – is the name or IP address of the RFgen server

Port (Long) Optional – is the port that facilitates the data portion of the communication.

Example:

```
Server.SetHost("192.168.1.100", 21097)
```

ShowProgress

This command enables or disables a popup progress box for all Server commands. The elapsed number of seconds and the current activity are displayed.

Syntax: Server.ShowProgress(Show)

Show (Boolean) True or False for turning on or off the progress bar

Example:

```
Server.ShowProgress(True)
```

SyncApps

This command will update a mobile device configured for mobile operation with changes made to application related items such as menus, users, applications, macros etc. If an administrator changes a device profile to include or exclude items, this command will compare what is on the device with what is in the profile and request any changed or missing items. Depending on the size of the change the Server.ShowProgress command may be useful.

Syntax: [OK =] Server.SyncApps([Profile])

OK (Boolean) Optional – returns True or False for the success of the command

Profile (Variant) Optional – is the name of the profile used to compare what is on the device with the server list of objects.

Example:

```
Server.SyncApps
```

```
Server.SyncApps("CEProfile1")
```

WriteFile

This command will write a file to the server hard drive and can be used from both the Thin client mode and Mobile client mode.

Syntax: [OK =] Server.WriteFile(FileName, Data, Overwrite)

OK (Boolean) Optional – returns True or False for the success of the command

FileName (String) is the path and file name for the file being created or overwritten.

Data (Variant) the contents of the file which can be either a string or byte array

Overwrite (Boolean) set to True, RFgen will overwrite an existing file, False will return a failure because the file already existed

Example:

```
Dim bOK As Boolean
```

```

Dim sData As String
Dim sPath As String

sPath = "C:\MyFile.txt"
sData = "Sample Text"
bOK = Server.WriteFile(sPath, sData, True)

```

System Level Extensions

System level commands get or set environmental properties for users, data connections or other global system settings.

ConnectionProperty

This function will return a characteristic of a data source. Some properties only refer to certain types of data connections. For example, dcDatabase would refer to an ODBC connection where dcHostPort would refer to a screen mapping connection. This is read-only.

Syntax: Value = SYS.ConnectionProperty(Source, Property)

Value (Variant) set to the value of the specified property

Source (Variant) is the string representation or the numeric representation of the database, screen mapping host or ERP session

Property	(enDCProps)	the following list of properties is available
	dcAutoWrap	(SM) Wrap text at end-of-line (VT220 only)
	dcBackColor	(SM) Back color of telnet emulator
	dcClient	(SAP) Application server's Client number
	dcCodePage	(SM) Language code page for TN5250 and TN3270 only
	dcConnectMode	(ALL) Returns ODBC, SM, or ERP
	dcConnectUsing	(ALL) Returns ADO for ODBC connections, Emulation Type for SM connections and the ERP type for ERP connections
	dcCursorType	(ODBC) The cursor type or locking state used to make the connection (eg: Optimistic / Pessimistic)
	dcDatabase	(ODBC) The type of database such as Access, Oracle, DB2, etc.
	dcDeviceName	(SM) Display Device name for TN5250 and TN3270 only
	dcDSN	(ODBC) Database DSN entry in Control Panel.

dcErrorCodes	(ODBC)	Database Reset Codes added to the connection's list.
dcFontSize	(SM)	Font size used to display the host screen in the RFgen telnet emulator
dcForeColor	(SM)	Fore color used to display the host screen in the RFgen telnet emulator
dcGWHost	(SAP)	Gateway Server
dcGWService	(SAP)	Gateway Number
dcIncludeSystem	(ODBC)	Returns 1 if View System Files option is checked
dcLanguage	(ERP)	Language used to log in to the ERP system
dcLocalEcho	(SM)	Echo Characters Locally option for VT220 only
dcLoginMode	(SM & ERP)	Returns Session Default Login Mode option or Auto Logon for ERP (0 = Automatic, 1 = Manual)
dcMenuLocked	(SM)	Returns 1 if the data connection is locked to a specific menu rather than the default of any main menu
dcPadFields	(SM)	Auto Fill Input Fields option
dcPassword	(ODBC & ERP)	Default password used by RFgen to login to the database or ERP system
dcPooled	(ALL)	Maximum number of pooled connections
dcProvider	(ODBC)	Returns the database driver provider
dcRouter	(SAP)	Router Connection string
dcSendCRLF	(SM)	Returns Send Carriage Return Line Feed as Enter key option (for VT220 only)
dcServer	(ERP)	Server Name or Environment setting
dcServicePort	(SM)	Telnet port RFgen uses to connect to the host
dcSourceId	(ALL)	RFgen name of the data connection entry
dcStartMenu	(SM)	Returns Session Default Main Menu option
dcSystem	(SAP)	System Number option

dcTimeout	(ODBC)	Max Timeout value
dcTrimFields	(SM)	Auto-Trim Output Fields option
dcUnicode	(SAP)	Returns ASCII or Unicode setting
dcUser	(ODBC & ERP)	Default user ID used by RFgen to login to the database or ERP system
dcVersion	(ERP)	Returns the Version option
dcViewOwner	(ODBC)	Returns 1 if View Owner option is checked
dcViewSource	(ODBC)	Returns 1 if View Source option is checked

Examples:

```
Dim Value As Variant
Value = SYS.ConnectionProperty(1, dcUser)
Value = SYS.ConnectionProperty("RFSample", dcUser)
```

DeleteProperty

This function will delete a property (or all properties) contained within the collection specified by the key. SYS SetProperty is used to add new properties to a collection.

Syntax: SYS.DeleteProperty(Key, ID)

Key (Variant) is the main key. It is the name of the collection of properties that are to be grouped together.

ID (Variant) is the property key. If an '*' (asterisk) is used, all properties within the collection will be deleted.

Example:

```
SYS.DeleteProperty("Counts", "LastIssue")
SYS.DeleteProperty("Counts", *)
```

DisableTimeout

This function will allow the session to continue even when the server's Client Inactivity Timeout value has been reached. In essence, this session will never time out.

Syntax: SYS.DisableTimeout(Value)

Value (Boolean) set to True to enable (eliminate the Client Inactivity Timer) or False to return to normal monitoring.

Example:

```
SYS.DisableTimeout(True)
SYS.DisableTimeout(False)
```

EnvironmentProperty

This function will return the value of the assigned property set in the Options / Application Environment / User Environment Properties grid.

Syntax: Value = SYS.EnvironmentProperty(Property)

Value (Variant) is the value stored in the System Level Settings for the Property specified.

Property (Variant) is the property specified in the Environment Properties grid.

Example:

```
Dim Plant As Variant
Plant = SYS.EnvironmentProperty("PlantName")
```

GetConnection

This is a specialized method similar to the EmbeddedProc object that could be used for getting the connection object to Microsoft Dynamics. The Dynamics client must be installed on the same machine. Use the ERP.BeginTrans and ERP.CommitTrans before and after the use of this method. This exposes the business functions but the programmer must know how to use them.

Syntax: Value = SYS.GetConnection(Source)

Value (Object) An object declared as Axapta3 or Object, if you choose to use late binding. It will return the ADO, RDO, OLEDB, ERP, or Web connection object

Source (Variant) data source name (DSN) or the data source number

Example:

```
Dim oConn As Axapta3
Dim oRecord As IAxaptaRecord

Set oConn = SYS.GetConnection("Dynamics")

Set oRecord =
oConn.CreateRecord("INVENTJOURNALTABLE")
If oRecord Is Nothing Then
    App MsgBox "Error: The Axapta Record object could
not be created."
    Exit Sub
End If
App MsgBox "Record company: " & oRecord.company
```

GetProperty

This function will return a property value that has been set using SYS SetProperty. SYS.GetProperty and SYS SetProperty can be used in place of making function calls to an INI file or to the System Registry. The keys and data are stored in the RFgen.mdb file.

Syntax: Value = SYS.GetProperty(Key, ID)

Value (Variant) is the property value.

Key (Variant) is the main key. It is the name of the collection of properties that are to be grouped together.

ID (Variant) is the property key. It is the reference to the value being returned. If an '*' (asterisk) is used as the ID, a Chr(1) delimited list of properties contained within the collection will be returned.

Example:

```
Dim TransfersCt As Variant  
Dim IssuesCt As Variant  
TransfersCt = SYS.GetProperty("Counts",  
"LastTransfer")  
IssuesCt = SYS.GetProperty("Counts", "LastIssue")
```

SendMessage

This function sends a message to a specific device number currently connected to the RFgen server. Device numbers of users can be obtained by using the SYS.UserList command.

Syntax: SYS.SendMessage(DevNo, Message)

DevNo (Integer) the ID of the mobile device

Message (String) the message to send to the other device

Example:

```
SYS.SendMessage(4, "Please see Sam immediately.")
```

SetProperty

This function will save a property value for future retrieval using SYS.GetProperty. Property values will persist between sessions because they are saved in the RFgen.mdb file. SYS.GetProperty and SYS SetProperty can be used in place of making function calls to an INI file or to the System Registry. Use this command with caution. The RFgen.mdb database is not designed for heavy adding and deleting of properties. Microsoft Access tends to grow over time as records are added and deleted.

Syntax: SYS SetProperty(Key, ID, Value)

Key (Variant) is the main key. This will be the identifier for all like properties and their values.

ID (Variant) is the sub key. This key is referenced to obtain its value

Value (Variant) is the property value.

Example:

```
Dim IssuesCt as Long  
IssuesCt = 200  
SYS SetProperty("Counts", "LastTransfer", 500)  
SYS SetProperty("Counts", "LastIssue", IssuesCt)
```

UserList

This function will return the device number, user and application currently in use. The purpose of this command is to log or locate a user doing a transaction, possibly for sending just the one device a message over the network. (See SYS.SendMessage)

Syntax: UserList = SYS.UserList([OnlyLoggedIn])

UserList (String) contains the device number, logged in user and application of all devices in use at the time the command was issued.

OnlyLoggedIn (Boolean) Optional – when set to True only returns entries for devices that have a logged in user. The default is False and this returns all connected devices to the RFgen server regardless of a user logged in. This could be used to generate a list of all connected devices so that they may all receive a message on the screen. The default is False.

Examples:

```
Dim UserList As Variant  
UserList = SYS.UserList  
UserList = App.ShowList(SYS.UserList(True), True)
```

ValidateWinUser

This function will validate user credentials against the Windows domain or local system.

Syntax: OK = SYS.ValidateWinUser(Domain, User, Password)

OK (Boolean) returns True if the validation was a success

Domain	(String) Enter the domain to be searched. If the local PC is used, specify a single period for the domain.
User	(String) Enter the user name to be validated.
Password	(String) Enter the password to be validated.

Example:

```
Dim bOK As Boolean
SYS.ValidateWinUser(".", App.User, gsPass)
```

In this example the password was saved in a global string variable when it was used on the login screen.

Database Related Extensions

Database related commands send and receive data to and from ODBC data connections. When operating in a mobile environment, the SQL statements are directed to the local database on the mobile device.

BeginTrans

This function begins a new transaction. RFgen will track any changes made to the database until either a DB.CommitTrans or DB.RollbackTrans are executed. You cannot have a second DB.BeginTrans for the same data source before committing or rolling back the first one. On a CE device, there is only 1 data connection and it must be committed or rolled back before another DB.BeginTrans is used.

Syntax: Value = DB.BeginTrans (Source)

Value (Boolean) indicates success or failure

Source (Variant) data source name (DSN) or the data source number

CommitTrans

This function saves any changes and ends the current transaction.

Syntax: Value = DB.CommitTrans (Source)

Value (Boolean) indicates success or failure

Source (Variant) data source name (DSN) or the data source number

Count

This function will return the number of rows contained in a specified rows item returned from the DB.Execute function or any Dynamic Array.

Syntax: Nbr = DB.Count(Rows)

Nbr (Variant) is the number of rows contained in the Records item.

Rows (String) is the string representation of a static recordset generated by a SQL statement using the DB.Execute function.

Example:

```
Dim sSQL As String
Dim sCols As String
Dim sRows As String
Dim Nbr As Integer
sSQL = "select * from Inventory"
DB.Execute(sSQL, sCols, sRows)
Nbr = DB.Count(sRows)
```

Execute

This function will execute a pass-through SQL statement on the host through the connection maintained by RFgen. Any results from the statement will be returned in a text-delimited object. (Note: this means that the item is a static view of the database and cannot be updated.)

Syntax: ErrNo = DB.Execute(SQL, [Columns], [Rows])

ErrNo (Variant) is a return value; a value of '0' (zero numeric) means the SQL statement processed normally.

SQL (String) is the SQL statement to be sent to the database. As this is a pass through statement, its syntax must be understood by the database, as no pre-processing will occur.

Columns (Variant) Optional – is a string representation of the columns returned by the SQL statement. This is optional because Insert, Update, and other statements do not return data.

Rows (Variant) Optional – is a string representation of the static rows of an SQL statement. This is optional because Insert, Update, and other statements do not return data.

Example:

```
Dim sSQL As String
Dim sCols As String
Dim sRows As String
sSQL = "select * from Inventory"
DB.Execute(sSQL, sCols, sRows)
```

In the case of an insert or an update, the sCols and sRows variables would not be necessary because no recordset is returned.

Extract

This function will extract from a recordset the 1 value at the specified column and row. A specific column and row intersect at a single value.

Syntax: Value = DB.Extract(Columns, Rows, RecNo, FieldNo)
Value (Variant) is the value extracted. (Note: this will always be a string value regardless of the actual field's data type.)
Columns (String) is the string variable used to hold the list of columns in the DB.Execute command.
Rows (String) is the string variable used to hold the rows of data in the DB.Execute command.
RecNo (Integer) is the row number within the retrieved recordset to extract data from.
FieldNo (Variant) is the column number (numeric), or column name (String) within the retrieved recordset to extract data from.

Example:

```
Dim sSQL As String
Dim sCols As String
Dim sRows As String
Dim Part_num As String
sSQL = "select * from Inventory"
DB.Execute(sSQL, sCols, sRows)
Part_num = DB.Extract(sCols, sRows, 1, "PartNo")
```

MakeList

This function executes a pass-through SQL 'select' statement against the database and converts the results into a scrolling list of items when used with App.ShowList.

Syntax: MyList = DB.MakeList(SQL, [Columns], [Normalize])
MyList (String) is the list to be returned for display (i.e., set RSP = Value to display the list on the Client device.)
SQL (String) is the SQL 'SELECT' statement to be sent to the database.
Columns (Boolean) Optional – when set to True will return all the columns as the potential key, not just the first column. Default is False.
Normalize (Boolean) Optional – when set to True will trim the spaces from the data so that it will display consistently. Default is False.

Example:

```
Dim sSQL As String
Dim sMyList As String
sSQL = "select PartNo from Inventory"
```

```
sMyList = DB.MakeList(sSQL, True, True)
Rsp = App.ShowList(sMyList)
```

OpenResultset

This function will execute a pass-through SQL statement on the host through the connection maintained by RFgen. Any results from the statement will be returned in a RDO Resultset object. (Note: this item defaults to a static view of the database and cannot be updated. Also, under the configuration of the database, the 'Connect Using...' setting must match the declaration of the recordset; ADO versus RDO.)

Syntax: Results = DB.OpenResultset(SQL, [CursorType], [LockType])

Results (rdoResultset, adoRecordset) is a snapshot type resultset generated from an SQL statement.

SQL (String) is the SQL statement to be sent to the database. This is a pass through statement; its syntax must be understood by the database, as no pre-processing will occur.

CursorType (Variant) Optional – indicates the type of cursor used by the recordset object.

LockType (Variant) Optional – indicates the type of lock placed on records during editing.

ADO Cursor Types

- 0 Provides a static copy of the records (you can't see additions, changes or deletions by other users). You can only move forward through the recordset. Forward-only is the ADO default cursor type.
- 1 Existing records at time of creation are updateable. You can't see additions or deletions. All types of movement are enabled.
- 2 Dynamic requires more overhead, because updates are immediate and all types of movement are enabled. The dynamic cursor isn't currently supported by the Microsoft Jet OLE DB Provider, and therefore defaults to a keyset cursor when adOpenDynamic is applied to a Jet database.
- 3 Provides a static copy of the records (you can't see additions, changes or deletions by other users), but all types of movement are enabled.

ADO Lock Types

- 1 This value indicates read-only records where the data cannot be altered.
- 2 This value indicates pessimistic locking, record by record. The provider does what is necessary to ensure successful editing

of the records, usually by locking records at the data source immediately after editing.

This lock type is supported by the Microsoft® OLE DB Provider for AS/400 and VSAM and the Microsoft OLE DB Provider for DB2. However, the OLE DB Provider for AS/400 and VSAM internally maps this lock type to adLockBatchOptimistic.

- 3 This value indicates optimistic locking, record by record. The provider uses optimistic locking, locking records only when the Update method is called.
This lock type is not supported by the OLE DB Provider for DB2.
- 4 This value indicates optimistic batch updates and is required for batch update mode.
This option is not supported by the OLE DB Provider for DB2.

Example:

Using rdoResultset, the RDO language extensions must be enabled.

```
Dim rs As rdoResultset
Dim sSQL As String
Dim sDesc As Variant
sSQL = "select * from Inventory where PartNo =
'100620'"
Set rs = DB.OpenResultset(sSQL)
sDesc = rs!Description
```

RedirectDataSource

This command will disable RFgen's auto processing of SQL statements to determine their correct source and will instead route all SQL traffic that is intended for the specified ODBC data source to always go to the other specified data source. To clear a redirection simply set the source and destination to the same data connection.

Syntax: Value = DB.RedirectDataSource(FromSource, ToSource)

Value (Boolean) indicates success or failure

FromSource (Variant) is the name or number of the data source to redirect.

ToSource. (Variant) is the name or number of the data source to receive the other database's SQL statements.

Example:

```
Dim bValue as Boolean
bValue = DB.RedirectDataSource("CAPlant", "NVPlant")
```

RollbackTrans

This function cancels any changes made during the current transaction and ends the transaction.

Syntax: Value = DB.RollbackTrans (Source)

Value (Boolean) indicates success or failure

Source (Variant) data source name (DSN) or the data source number

SaveBitmap

This command will save the byte array of a bitmap picture directly to a database. The record in the database must already exist.

Syntax: Value = DB.SaveBitmap(TableName, Field, Data, [Where])

Value (Boolean) indicates success or failure

TableName (String) is the name of the table in the database

Field (String) is the field name within the table

Data (Byte) is the byte array containing the image

Where (Variant) Optional – a SQL parameter that updates or inserts the image to a specific record

Example:

```
Dim bVal as Boolean
Dim bImage() as Byte
Dim nSize As Long

Open "C:\MyPic.bmp" For Binary Access Read As #1
    nSize = LOF(1)
    ReDim bImage(nSize - 1)
    Get #1, , bImage
Close #1
'
bVal = DB.SaveBitmap("Inventory", "Image", bImage,
"PartNo='100620'")
```

UseDataSource

This command will disable RFgen's auto processing of SQL statements to determine their correct source and will instead route all SQL traffic to the specified data source.

Syntax: Value = DB.UseDataSource(Source)

Value (Boolean) indicates success or failure of the language extension

Source (Variant) is the name or number of the data source to which all future SQL traffic will be routed.

Mobile Device Extensions

RFgen supports commands that are used specifically on a mobile installation of RFgen. These commands provide a method of synchronizing the server, sending and receiving data, queuing transactions on the host, etc. For the queuing commands, the server's Transaction Manager and the client's Transaction Manager must be enabled.

RFgen also supports the ability to load and execute COM objects on the Windows CE device. COM objects are loaded and run by executing the language extensions `ceObject.Create`, `ceObject.Execute`, and `ceObject.Release`. (`ceObject` is an object you must declare as a variable first and then use that variable.) The objects must be derived from the `IDispatch` interface and must be compatible with VBA scripting environments.

ClickAndSkipPrompts

This method will turn on or off the user's ability to click in to prompts in a random sequence. The line "Device.ClickAndSkipPrompts(False)" will allow the user to click in the very next prompt or any prompt that already contains data. All other prompts will not receive focus if they are clicked. Using `Device.ClickAndSkipPrompts(True)` will set RFgen back to its default behavior.

Syntax: `Device.ClickAndSkipPrompts(Enabled)`
Enabled (Boolean) True or False turns this feature on or off.

ClickCoordinates

This command will return the last set of X, Y coordinates relative to the control that was clicked. For example, after clicking on an image control, a call to this command will return X, Y coordinates with 0, 0 being the upper left corner of the image control. If there is a label or caption for the control, its area is included as possible click space. Note: this will not work on some controls that require clicks such as a signature capture box.

Syntax: `Device.ClickCoordinates(X, Y)`
X (Variant) the horizontal axis pixel in the last click event
Y (Variant) the vertical axis pixel in the last click event

Example:

```
Dim X as Variant  
Dim Y as Variant
```

`Device.ClickCoordinates (X, Y)`

ForceLocal

This property will tell the Mobile Client in a Roaming state that it should only look to either the server or local database for all data retrieval and updates. For example, if Mobile Client in a Roaming state was connected to the RFgen server and the ForceLocal was set to False, when the user executes a TM.QueueMacro it would actually be queued on the server not the local device. DB.Execute would look to the server automatically to retrieve the latest data rather than the locally stored data on the device.

Syntax: `Device.ForceLocal = Value`

Value (Boolean) set to True or False to force data access to either the server or local database

Example:

`Device.ForceLocal = True`

GetGPSInfo

This command will retrieve GPS data from an attached GPS receiver on the mobile device. If no receiver is available or no signal is received the output will be zeros. This command would need to be called repeatedly to update the screen if the device is in motion. Use the Form_OnTimer event to continuously populate variables.

Note: The CE operating system GPS APIs must exist on the device. Mobile 2005 and above have these APIs by default. To configure the GPS settings on the mobile device choose Settings / System Tab / External GPS configuration, set the Program COM port to 'none' and set the Hardware COM port to the proper number based on the manufacturer's documentation.

Syntax: [OK =] `Device.GetGPSInfo([Longitude], [Latitude], [Heading], [Altitude], [Speed])`

OK (Boolean) Optional – the success or failure of the command.

Longitude (Variant) Optional – returns the longitude as a float value

Latitude (Variant) Optional – returns the latitude as a float value

Heading (Variant) Optional – returns the heading as a value (0-360)

Altitude (Variant) Optional – returns the altitude in meters

Speed (Variant) Optional – returns the speed in knots (nautical miles)

Examples:

`Dim bOK as Boolean`

`Dim vLong as Variant`

`Dim vLat as Variant`

```

Dim vHead as Variant
Dim vAlt as Variant
Dim vSpeed as Variant

bOK = Device.GetGPSInfo (vLong, vLat, vHead, vAlt,
vSpeed)
bOK = Device.GetGPSInfo (, , , vAlt)

```

For converting meters to US feet use the following conversion:

```
vAlt = vAlt * 3.28083989501312
```

For converting knots to US MPH use:

```
vSpeed = vSpeed * 1.15077945
```

For converting knots to KPH use:

```
vSpeed = vSpeed * 1.852
```

For converting the heading into a direction use:

```

Select Case vHead
    Case Is < 22.5: sText = "N"
    Case Is < 67.5: sText = "NE"
    Case Is < 112.5: sText = "E"
    Case Is < 157.5: sText = "SE"
    Case Is < 202.5: sText = "S"
    Case Is < 247.5: sText = "SW"
    Case Is < 292.5: sText = "W"
    Case Is < 337.5: sText = "NW"
    Case Else: sText = "N"
End Select

```

GoOffline

This command returns a True / False regarding the attempt to close the socket connecting the mobile client to the server. This will make the device go from THIN client mode to a MOBILE disconnected state.

Syntax: [OK =] Device.GoOffline([User], [Menu], [Form])

OK (Boolean) Optional – returns a True if the CE device successfully closes the socket to the server. Since the thin client is shut down with this command, evaluating the response variable will never be reached unless there is a failure.

User (Variant) Optional – If a user is specified then the device will set the current user and load the menu associated with the user bypassing the login screen.

Menu (Variant) Optional – Specifying a specific menu requires that the optional User parameter be filled in. The User

parameter will still load the default menu for that user but then the Menu parameter will load, in addition. This means that if the user backs out of the specified menu RFgen will display the user's default menu.

Form (Variant) Optional – If the form is specified then the form will be loaded after the specified menu or the default menu. The Form depends on the UserID being filled in.

Examples:

```
Dim bOK as Boolean  
bOK = Device.GoOffline  
bOK = Device.GoOffline("Sam")  
bOK = Device.GoOffline("Sam", , "IMASTER")  
bOK = Device.GoOffline("Sam", "BasicApps", "IMASTER")
```

GoOnline

This command returns a True / False regarding the status of opening a socket connection with the server. This will make a disconnected MOBILE client an online THIN client to the server. If the user is going online for the first time the server will present the login screen. If the user goes online with the configured client inactivity timeout value they would see the session as they left it when going mobile. This command will not work for Mobile clients with a startup mode of Disconnected.

Syntax: [OK =] Device.GoOnline([Server], [Port])

OK (Boolean) Optional – return a True if the CE device is successfully connected to the server. Since the mobile client is shut down with this command, evaluating the response variable will never be reached unless there is a failure.

Server (Variant) Optional – parameter to specify which server name or IP address should be used. If this is not specified, the registry settings will be used.

Port (Long) Optional – parameter to specify which server port number should be used. If this is not specified, the registry settings will be used.

Example:

```
Dim bOK as Boolean  
bOK = Device.GoOnline("192.168.123.45", 21098)
```

IsOffline

This command returns a True / False regarding the status of an open socket connection with the server. This command is typically used to evaluate which state the client is in and then set variables that would

vary between the server and the mobile device such as file path information.

Syntax: OK = Device.Offline

OK (Boolean) return a True if the CE device is currently not connected to the server.

Example:

```
Dim bOK as Boolean  
bOK = Device.Offline
```

IsOnline

This command returns a True / False regarding the status of an open socket connection with the server. This command is typically used to evaluate which state the client is in and then set variables that would vary between the server and the mobile device such as file path information.

Syntax: OK = Device.Online

OK (Boolean) return a True if the CE device is connected to the server.

Example:

```
Dim bOK as Boolean  
bOK = Device.Online
```

Platform

This command returns an enumeration indicating the platform of the connected device.

Syntax: Value = Device.Platform

Value (Enumeration)

0 = Device_None	(Application Testing)
1 = Device_WinCE	(CE / Mobile Device)
2 = Device/Desktop	(Windows Desktop Client)
3 = Device_Android	(Android Device)
4 = Device_iOS	(Apple Device)
5 = Device_Vocollect	(Vocollect Talkman)
6 = Device_TN	(Standard Telnet Client)
7 = Device_SOA	(SOA Service Connection)

Example:

```
Dim vValue as Variant  
vValue = Device.Platform
```

PlaySound

This command plays any CE supported sound file by specifying the path to the file.

Syntax: Device.PlaySound(Path)

Path (String) the full path to the sound file to be played.

Example:

```
Device.PlaySound("\Program Files\RFgen  
Mobile\ERROR.WAV")
```

PrinterOff

This command is obsolete and has been replaced with Device.SendCommPort. It will turn off transparent print mode and redirect all text received by the Client device back to the standard display. Note: Device.PrinterOff will not function in the GUI mode, Consider using Device.SendCommPort instead.

Syntax: Device.PrinterOff

Example: See *Device.Send*

PrinterOn

This command is obsolete and has been replaced with Device.SendCommPort. It will turn on transparent print mode and redirect all text received by the client device to the attached serial printer. Note: Device.PrinterOn will not function in the GUI mode. Consider using Device.SendCommPort instead.

Syntax: Device.PrinterOn

Example: See *Device.Send*

Send

This command is obsolete and has been replaced with Device.SendCommPort. It will send raw, unformatted text to the Client device; it is typically used to send text to an attached barcode printer or other auxiliary device. Used with Device.PrinterOn and Device.PrinterOff only.

Syntax: Device.Send(Packet)

Packet (Variant) is the text stream that is to be sent to the attached barcode printer.

Example:

```
Device.PrinterOn  
Device.Send("100620 20lb Super Green Lawn Food")  
Device.PrinterOff
```

Refer to www.rfgen.com, "Frequently Asked Questions" for help on printing barcodes.

SendCommPort

Sends data to the device serial port. This command replaces the commands Device.PrinterOn and Device.PrinterOff that are still supported but are now obsolete. RFgen will automatically turn transparent print mode on or off as required.

Syntax: Device.SendCommPort(Packet)

Packet (Variant) is the text stream to be sent to the serial port.

SetCameraOption

This command changes some default settings for supported cameras. Since different cameras may use different settings, RFgen does not know the values for some IDs.

Syntax: Device.SetCameraOption(Option, Value)

Option (enCameraOptions) this ID contains either
IMAGE_BRIGHTNESS, IMAGE_CONTRAST, or
IMAGE_FLASH

Value (Variant) the camera's expected value for the named ID

Example:

```
Device.SetCameraOption (IMAGE_FLASH, False)
```

SetCommMode

This command changes the phonebook option located in the RFgenCFG.exe application. The option to connect using WiFi or GPRS can be set and then upon the next connection attempt the new method will be used.

Syntax: Device.SetCommMode(Mode)

Mode (enConnectMode) can be 1 of these 3 options:
ConnIsGPRS
ConnIsUndefined
ConnIsWiFi

Example:

```
Device.SetCommMode (ConnIsGPRS)
```

SetCommPort

This command will enable or disable the serial port on a 'CE-based' data device.

Syntax:	Device.SetCommPort(Enabled, Port, BaudRate, ByteSize, StopBits, Parity, FlowControl, PollingInterval, [IsUnicode])
Enabled:	(Boolean) True / False, enable or disable the serial port
Port	(Integer) 1,2,3,etc - the port number to activate
BaudRate	(Long) 9600, 19200, etc.
ByteSize	(Integer) 7, 8, etc.
StopBits	(enStopBits) Select the appropriate stopbits from the drop-down list (1,1.5 or 2)
Parity	(enParity) Select the appropriate Parity from the drop-down list (none, even, odd, etc.)
FlowControl	(enFlowControl) Select the appropriate FlowControl from the drop-down list (CTS, DSR, XONXOFF, etc.)
PollingInterval	(Long) RFgen will poll the serial port and look for data coming in using this timing interval in milliseconds.
IsUnicode	(Boolean) Optional – alerts RFgen that the attached COM device sends and receives in UNICODE. The default is False.

Example:

```
Device.SetCommPort(True, 1, 9600, 8, 1, NONE, NONE,  
1000, False)
```

TakePicture

This command will retrieve an image from supported cameras and will fill a byte array with a BMP type image. (See also DB.SaveBitmap)

Syntax: [OK =] Device.TakePicture(Image)
OK (Boolean) Optional – the success or failure of the command.
Image (Byte) stores the byte array of the BMP image

Example:

```
Dim bOK As Boolean  
Dim szImage() As Byte  
bOK = Device.TakePicture(szImage)  
RFPrompt(1).Bitmap = szImage   'prompt 1 is an image  
control  
DB.SaveBitmap(sTableName, sField, szImage, sWhere)
```

WriteFile

This command will write data to a file on the mobile device. String data or binary data can be written.

Syntax: [OK =] Device.WriteLine(Path, Data, Overwrite)

OK (Boolean) Optional – the success or failure of the command.

Path (String) specifies where on the mobile device the file should be placed

Data (Variant) contains the data to be written to the file. If it is string data it will be saved in Unicode format otherwise it will be left as is. A byte array is also allowed.

Overwrite (Boolean) set to True, RFgen will overwrite an existing file, False will return a failure because the file already existed

Example:

```
Dim bOK As Boolean  
Dim sData As String  
Dim sPath As String  
  
sPath = "\Program Files\MyFile.txt"  
sData = "Sample Text"  
bOK = Device.WriteLine(sPath, sData, True)
```

DeviceObject

This object is declared as a variable type and used to execute COM objects from the CE device. It is not part of the Device object directly but is a subset of CE functionality specifically for calling COM objects.

The following examples would all start with:

```
Dim [WithEvents] oMyObj As DeviceObject
```

If the object supports events include the WithEvents statement in the Dim statement. If WithEvents is declared as part of the Dim statement, this event would also be available from the script window's drop-down:

```
Private Sub oMyObj_OnEvent(ByVal EventName As String,  
ByRef rsData As DataSet)  
  
End Sub
```

Create

This command loads a COM object stored on the CE device. Calling Create multiple times on the same program id will increment the reference count. Create will only register for events, if specified, on the first call to Create.

The App.ErrNo object should be inspected to determine if Create was successful.

Syntax: [OK =] oMyObj.Create

OK (Boolean) Optional – the success or failure of the command.

Example: (See *Execute*)

Execute

This command executes the COM object stored on the CE device. Execute will first determine if the COM object has been loaded. If the object has not been loaded then Execute will attempt to load the object without events. If the object could be loaded successfully, the method executes. The object will remain loaded after this method completes execution.

The App.ErrNo object should be inspected to determine if Execute was successful.

Syntax: [OK =] oMyObj.Execute(Method, [Params])

OK (Boolean) Optional – the success or failure of the command.

Method (String) Method is a text string that represents the method that you wish to execute.

Params (Variant) Optional – parameter array where you specify the parameters to the method. The parameters must be specified in the appropriate order that the method expects.

Example:

```
Dim bOK as Boolean
Dim oMyObj As DeviceObject
Dim vErr As Variant
Dim sParam1 As String
Dim sParam2 As String

Set oMyObj = New DeviceObject
oMyObj.Name = "DeviceObjectTest.ioInterface"
oMyObj.Create
sParam1 = "Value1"
sParam2 = "Value2"
bOK = oMyObj.Execute("Method1", sParam1, sParam2)
vErr = oMyObj.LastError
If vErr <> "" Then
    App MsgBox "COM failure: " & CStr(vErr)
Else
```

```
    App MsgBox "COM object returned: " &
oMyObj.ReturnValue
End If
oMyObj.Release
```

LastError

This property returns the last error generated from the Execute method.

Syntax: Value = oMyObj.LastError

Value (Variant) is the error number returned from the COM object

Example: *(See Execute)*

Name

This property sets the program ID of the object that you wish to load. The object must have been successfully registered on the device for the Create function to succeed.

Syntax: oMyObj.Name = ProgID

Alternate: ProgID = oMyObj.Name

ProgID (String) a string indicating the program ID of the object that you wish to execute.

Example: *(See Execute)*

Release

This command releases the COM object. Calling Release multiple times on the same program ID will decrement the reference count. Release will free the object when the reference count reaches 0.

The App.ErrNo object should be inspected to determine if Release was successful.

Syntax: [OK =] oMyObj.Release

OK (Boolean) Optional – the success or failure of the command.

Example: *(See Execute)*

ReturnValue

This property returns any value being passed back from the COM object

Syntax: [Value =] oMyObj.ReturnValue

Value (Variant) Optional – the value returned by the COM object.

Example: (See *Execute*)

Transaction Management Extensions

Commands relating to RFgen Transaction Management capabilities and queuing are called from the TM object.

AbortTrans

This function can be used inside a Transaction macro to halt processing of the transaction if conditions warrant it. This command will notify the RFgen Transaction Manager that the current macro processing was aborted for a reason other than failure. The Transaction Manager will then keep this transaction at the top of the queue and try again on the next cycle. For example, if an SM command comes back with a failed status or the BeginTrans command fails, TM.AbortTrans can be executed to give the transaction macro's Boolean value a False value. In the application's script, you will know if the called macro was successful.

Syntax: TM.AbortTrans

GetItems

This method loads a set of macros to be evaluated from the Transaction Management tables. Selecting the category, date and user, this command will return a DataSet containing the macros and their data. If a list of macros with a more complex selection criteria is desired use TM.GetItemsEx.

Syntax: [OK =] TM.GetItems(ItemStatus, TranDate, UserID, rsTran)

OK (Boolean) Optional – True or False for the success of the command

ItemStatus (enItemStatus) There are 4 options for the ItemStatus parameter:

tmAllItems all macros in all queues, in the queue, failed, or completed

tmCompleted all macros in all queues that were completed successfully

tmFailed all macros in all queues that were completed unsuccessfully

tmInProcess all macros in all queues that are not yet processed

TranDate (Variant) specifies a single day to be retrieved

UserID (String) specifies the user that performed the transaction

rsTran (DataSet) the variable declared as a DataSet that contains the list of macros and their data. For more information on how the DataSet object is used see the DataSet section of the manual.

Example:

```
Dim oList As DataSet  
TM.GetItems(tmFailed, Date, "SAM", oList)
```

This command will return all the failed macros on the current day that was performed by the user Sam.

GetItemsEx

This method loads a set of macros to be evaluated from the Transaction Management tables. Specify a SQL statement that will retrieve the desired recordset and take advantage of the complexity allowed by ODBC. Knowledge of the table and field names is required.

Syntax: [OK =] TM.GetItemsEx(SQL, rsTran)

OK (Boolean) Optional – True or False for the success of the command

SQL (String) specifies the SQL statement for retrieving a list of macros

rsTran (DataSet) the variable declared as a DataSet that contains the list of macros and their data. For more information on how the DataSet object is used see the DataSet section of the manual.

The table names required depend on the name of the queue configured in the Transaction Management setup. For the RFQueue the tables created are:

RFQueue – used to store the InProcess items

RFQueue_Completed – used to store successfully completed items

RFQueue_Failed – used to store failed transaction macros

Substitute the RFQueue name for alternate queues that may have been defined. The fields that can / should be referenced through SQL within the tables are shown below. The Data Type values were taken from the default RFgenTM.mdb (MS Access) database.

RFQueue table

<u>Field Name</u>	<u>Data Type</u>	<u>Description</u>
SeqNo	Number	the sequence number
TranDate	Number	when the macro was queued (20111231)
TranTime	Number	when the macro was queued (235959)

Name	Text	name of the macro
FormId	Text	form that queued the macro
UserId	Text	user that queued the macro

RFQueue_Completed table and RFQueue_Failed table

The table design is the same as RFQueue but includes additional columns:

Field Name	Data Type	Description
ExecDate	Number	when the macro was queued (20111231)
ExecTime	Number	when the macro was queued (235959)

Example:

```
Dim oList As DataSet
TM.GetItemsEx("Select * from RFQueue where UserId =
'SAM'", oList)
```

MacroName

This returns the name of the macro currently in focus in the object's list of macros.

Syntax: Value = TM.MacroName
Value (String) the name of the macro

MoveQueue

This function will take an RFgen queue from one database and guarantee its delivery to another database. If another instance of RFgen is monitoring the second database, that instance will become responsible for executing the queued transactions. The RFgen Transaction Management database connection must be capable of seeing both databases.

Syntax: Value = TM.MoveQueue(FromQueue, ToQueue)
Value (Boolean) Optional – the success or failure to move the queue from one database to another.
FromQueue (String) the source queue to be moved
ToQueue (String) the destination queue to receive the transactions

Example:

```
Dim Value As Boolean
Value = TM.MoveQueue("RFQueue", "HostQueue")
Value = TM.MoveQueue("RFQueue",
"AlternateDB.HostQueue")
```

QueueMacro

This function is used to queue Data Transaction macros and pass any required parameters. It returns the sequence number of the macro after it has been successfully queued. These macros can be created on the Transactions tree.

Syntax: [Seq =] TM.QueueMacro(MacroName, [Params])

Seq (Long) Optional – the sequence number of the macro after it has been successfully queued.

MacroName (String) is the name of the macro to be called.

Params (Variant) Optional – A series of passing parameters as required by the selected macro. Note: these parameters are the fields you defined when you wrote the macro.

Example:

```
Dim lSeq As Long  
lSeq = TM.QueueMacro("ChangeUser", "Sam", "1234")
```

QueueName

This function will return the name of the RFgen queue currently being processed if this command is executed from within the macro itself.

Syntax: Value = TM.QueueName

Value (String) is the name of the queue

Example:

```
Dim Value As String  
Value = TM.QueueName
```

SeqNo

This function will return the sequence number of the transaction currently being processed if this command is executed from within the macro itself.

Syntax: Value = TM.SeqNo

Value (Long) is the sequence number

Example:

```
Dim Value As Long  
Value = TM.SeqNo
```

Enterprise Resource Planning Extensions

Commands relating to RFgen ERP capabilities and queuing are called from the ERP object.

BeginTrans

This command is used to retrieve an ERP connection from the RFgen managed pool and keep it for an unspecified amount of time. It would typically be used to execute a sequence of ERP commands against a pooled connection. Note: If the connection is not pooled, or will call only a single business function, this command is not needed. The first ERP data connection is the default Source value.

Syntax: [Value =] ERP.BeginTrans ([Source])

Value (Boolean) Optional – A True/False notification that the command processed successfully

Source (Variant) Optional – data source name (DSN) or the data source number

Example:

```
Dim Value As Boolean  
Value = ERP.BeginTrans(1)
```

CommitTrans

This command is used to release an ERP connection back to the RFgen managed pool once the process is finished with it. Note: You must always use this function paired with ERP.BeginTrans, otherwise you will deplete the pool of connections and prevent other users from having ERP access. The first ERP data connection is the default Source value.

For an SAP system, this command will also execute BAPI_TRANSACTION_COMMIT.

Syntax: [Value =] ERP.CommitTrans ([Source])

Value (Boolean) Optional – A True/False notification that the command processed successfully

Source (Variant) Optional – data source name (DSN) or the data source number

Example:

```
Dim Value As Boolean  
Value = ERP.CommitTrans(1)
```

LogOff

This function is used to logoff a non-pooled ERP connection. Note: this function does not need to be called by the user. RFgen will call it automatically when shutting down the RFgen session. The first ERP data connection is the default Source value.

Syntax: [Value =] ERP.LogOff (Source)

Value (Boolean) Optional – A True/False notification that the command processed successfully

Source (Variant) data source name (DSN) or the data source number

Example:

```
Dim Value As Boolean  
Value = ERP.LogOff(1)
```

LogOn

This is used to logon the ERP connection and specify a user / password sequence for a non-pooled ERP connection (optional).

Note: the user does not always require this function as RFgen calls it automatically when a session starts.

Syntax: [Value =] ERP.LogOn (Source, [UserId], [UserPwd], [Options])

Value (Boolean) Optional – A True/False notification that the command processed successfully

Source (Variant) data source name (DSN) or the data source number

UserId (Variant) Optional – The login name to be used to connect to the ERP system

UserPwd (Variant) Optional – The login password to be used to connect to the ERP system

Options (Variant) Optional – SAP only additional parameters that can be added to the logon string. Separate multiple parameters with the pipe (|) symbol.

CLIENT	- SAP Client
LANG	- Logon language
SYNSR	- SAP System number
ASHOST	- SAP application server
MSHOST	- SAP message server
GWHOST	- Gateway host
GWSERV	- Gateway service
R3NAME	- R/3 name
GROUP	- Group of SAP application servers
TPHOST	- Host of the external server program
TYPE	- Type of remote host (2 = R/2, 3 = R/3, E = External)
TRACE	- Enable RFC trace (1=on, 0 = off)
CODEPAGE	- Initial code page in SAP notation
LCHECK	- Enable logon check at open time (1=on, 0 = off)
QOSDISABLE	- Disable Quality of Service check (1=disable, 0 = enable)
GRT_DATA	- Additional data for GUI
USE_GUIHOST	- Redirect remote GUI to this Host

USE_GUISERV	- Redirect remote GUI to this Service
USE_GUIPROGID	- Server program id that will start the remote GUI
SNC_MODE	- Enable Secure Network Communications (1=on, 0 = off)
SNC_PARTNERNAME	- SNC partner (p:CN=R3, O=XYZ-INC, C=EN)
SNC_QOP	- SNC Level of security (1-9)
SNC_MYNAME	- SNC Name - overrides default SNC partner
SNC_LIB	- SNC service library path
DEST	- R/2 destination
SYSTEM	- Name of the system as used in the system landscape definition
LOGONMETHOD	- Authentication method used to log on to the destination (UIDPW, SAPLOGONTICKET, X509CERT)

Examples:

```
Dim Value As Boolean
Value = ERP.LogOn("SAP", "User345", "Pass345")
Value = ERP.LogOn(1, "User345", "Pass345",
"CLIENT=800|TYPE=3")
Value = ERP.LogOn("JDE", "JDEUser", "JDEPass1")
```

MakeList

This function executes a pass-through SQL 'select' statement against the ERP system's database and converts the results into a scrolling list. You may use App.ShowList to display the list to the user. The first ERP data connection is assumed as the source.

Syntax: MyList = ERP.MakeList(Sql, [Columns], [Normalize])

MyList	(String) is the list to be returned for display (i.e., set RSP = MyList to display the list on the Client device.)
Sql	(String) is the SQL 'SELECT' statement to be sent to the database.
Columns	(Boolean) Optional – when set to True will return all the columns as the potential key, not just the first column. Default is False.
Normalize	(Boolean) Optional – when set to True will trim the spaces from the data so that it will display consistently. Default is False.

Example:

```
Dim sSQL As String
Dim sMyList As String
sSQL = "select PartNo from Inventory"
sMyList = ERP.MakeList(sSQL,True,True)
Rsp = App.ShowList(sMyList)
```

ReadData

This command executes a read-only SQL statement against the ERP system. Note: In the case of SAP, the business function RFC_READ_TABLE is used and it is not an officially released BAPI and has significant limitations. Therefore it may not work with all versions of SAP. If this is the case, please contact DataMAX for a solution.

Syntax: [Value =] ERP.ReadData(SQL, Cols, Rows)

Value (Boolean) Optional – A True/False notification that the command processed successfully

SQL (String) the SQL statement to be executed on the ERP table

Cols (String) a variable containing the columns of the resulting data

Rows (String) a variable containing the rows of resulting data

SAP Limitations – since the RFC_READ_TABLE function is used, SQL statements must specify at least 1 field. Using an asterisk (*) or a function will not be accepted. Examples are “select *”, “select count(*)”, select SUM(QTY) …”, etc. Only a comma-delimited list of field names is acceptable. Finally you may only specify 1 table in the SQL statement, no joins.

For example, if you would like to retrieve the Material numbers from a table, such as the Material Documents table within SAP, specify the following:

Example:

```
Dim Value As Variant  
Value = ERP.ReadData("select MATNR from MSEG", sCols,  
sRows)
```

RollbackTrans

This command is used to undo executed functions against an ERP connection assuming the ERP system is capable of undoing those functions. The first ERP data connection is the default Source value.

For an SAP system, this command will also execute BAPI_TRANSACTION_ROLLBACK.

Syntax: [Value =] ERP.RollbackTrans ([Source])

Value (Boolean) Optional – A True/False notification that the command processed successfully

Source (Variant) Optional – data source name (DSN) or the data source number

Example:

```
Dim Value As Boolean  
Value = ERP.RollbackTrans(1)
```

SetHardRelease

This function (exclusively for JDE connections) is to be called at the beginning of any transaction macro that will be affected by the resource leak in JD Edwards. It sets an internal flag that will release the data pointers when an ERP.CommitTrans or ERP.RollbackTrans is called. At this point the internal flag is toggled off.

Syntax: ERP.SetHardRelease

SetSession

If there is more than 1 ERP connection and a transaction may need them independently but at the same time, ERP.SetSession will direct embedded business functions to the specified session, ignoring the RFgen defaults. The 'DataSource' method will overwrite this function if it is included in the VBA code.

Syntax: [Value =] ERP.SetSession (Source)

Value (Boolean) Optional – A True/False notification that the command processed successfully

Source (Variant) data source name (DSN) or the data source number. Setting this value to 0 will re-enable RFgen's automatic determination of which data connection should be used based on which connection was used to download a particular business function being called.

Example:

```
Dim Value As Boolean  
Value = ERP.SetSession(1)
```

Printer Extensions

Printer commands specifically interact with creating and delivering data to tethered or IP addressable printers.

Activate

This command will redirect output to the specified Windows printer. Specifying the printer by name should be exactly the same as it appears in the Windows Control Panel under the Printers section.

Syntax: Printer.Activate(Name)

Name (Variant) is the printer name in the 'Printers Window'. Click Start/ Settings/Printers. Note: You may use the printer number if desired.

Example:

```
Printer.Activate("HPDeskJet 697C")
```

Copies

This property accesses the copy count parameter associated with a specific Windows printer. If it is not specified, the default is 1 copy.

Syntax: Printer.Copies = Value

Value (Long) is the number of copies to print.

Example:

```
Printer.Copies = 10
```

Note: there is a special condition with the Copies, Orientation and PrintQuality commands. Since RFgen manages the beginning and ending of a print job, the VBA scripts must not use one of these commands after the Printer.Print command. These 3 commands will start a new print job and all settings defined up to this point will be lost.

This is valid:

```
Printer.Copies = 10  
Printer.Orientation = PrintHorizontal  
Printer.Print "Sample Text"  
Printer.EndDoc
```

This is not because the Orientation line will lose the 2 lines above it.

```
Printer.Copies = 10  
Printer.Print "Sample Text"  
Printer.Orientation = PrintHorizontal  
Printer.EndDoc
```

EndDoc

This command will close the print job on the selected Windows Printer and start the printing process.

Syntax: Printer.EndDoc

Note: All printer settings such as font, copies, orientation and others are reset to defaults after this command. They must be re-issued to take effect on the next print job.

FontBold

This property accesses the font bold parameter associated with a specific Windows printer.

Syntax: Printer.FontBold = Value

Value (Boolean) determines whether text is printed using the bold font option.

Example:

```
Printer.FontBold = True
```

FontItalic

This property accesses the font italic parameter associated with a specific Windows printer.

Syntax: Printer.FontItalic = Value

Value (Boolean) determines whether text is printed using the italic font option.

Example:

```
Printer.FontItalic = True
```

FontName

This property accesses the font name parameter associated with a specific Windows printer.

Syntax: Printer.FontName = Value

Value (String) specifies the type of font to use.

Example:

```
Printer.FontName = "Arial"
```

FontSize

This property accesses the font size parameter associated with a specific Windows printer.

Syntax: Printer.FontSize = Value

Value (Long) is the font size to use.

Example:

```
Printer.FontSize = 12
```

FontStrikeThru

This property accesses the font strike thru parameter associated with a specific Windows printer.

Syntax: Printer.FontStrikethru = Value

Value (Boolean) determines whether text is printed using the strike thru font option.

Example:

```
Printer.FontStrikeThru = True
```

FontUnderline

This property accesses the font underline parameter associated with a specific Windows printer.

Syntax: Printer.FontUnderline = Value

Value (Boolean) determines whether text is printed using the underline font option.

Example:

```
Printer.FontUnderline = True
```

GetName

This function returns the name for the specified Windows printer number. You may use this command within a loop to get a list of names for the user to pick from. See Printer.Activate for setting a printer.

Syntax: Name = Printer.GetName(Index)

Name (String) is the Windows name of the requested printer.

Index (Long) is the Windows printer number.

Example:

```
Dim Name As String  
Name = Printer.GetName(1)
```

NewPage

This command will send a page eject character to the selected Windows Printer.

Syntax: Printer.NewPage

Orientation

This function accesses the orientation parameter associated with a specific Windows printer.

Syntax: Printer.Orientation = Value

Value (enPrintOrientation) is the text orientation to use.

PrintHorizontal (Portrait)

PrintVertical (Landscape)

Example:

```
Printer.Orientation = PrintHorizontal
```

Note: there is a special condition with the Copies, Orientation and PrintQuality commands. Since RFgen manages the beginning and ending of a print job, the VBA scripts must not use one of these commands after the Printer.Print command. These 3 commands will start a new print job and all settings defined up to this point will be lost.

This is valid:

```
Printer.Copies = 10
Printer.Orientation = PrintHorizontal
Printer.Print "Sample Text"
Printer.EndDoc
```

This is not because the Orientation line will lose the 2 lines above it.

```
Printer.Copies = 10
Printer.Print "Sample Text"
Printer.Orientation = PrintHorizontal
Printer.EndDoc
```

PageWidth

This function sets the printer print width to a specific value. The value is reset to 0 after an EndDoc command is issued.

Syntax: Printer.PageWidth = Value

Value (Long) is the printer width.

Examples:

```
Printer.PageWidth = 80
Printer.PageWidth = 132
```

Print

This command will print the text on the selected Windows printer. Each Print statement will be on a new line.

Syntax: Printer.Print(Text)

Text (String) is the text stream that is to be sent to the network printer.

Example:

```
Printer.Print("20lb Super Green Lawn Food")
```

PrintQuality

This property accesses the print quality parameter associated with a specific Windows printer.

Syntax: Printer.PrintQuality = Value

Alternate: Value = Printer.PrintQuality

Value (enPrintQuality) is the print quality desired.

DraftQuality

HighQuality

LowQuality

MediumQuality

Example:

```
Printer.PrintQuality = MediumQuality
```

Note: there is a special condition with the Copies, Orientation and PrintQuality commands. Since RFgen manages the beginning and ending of a print job, the VBA scripts must not use one of these commands after the Printer.Print command. These 3 commands will start a new print job and all settings defined up to this point will be lost.

This is valid:

```
Printer.Copies = 10
Printer.Orientation = PrintHorizontal
Printer.Print "Sample Text"
Printer.EndDoc
```

This is not because the Orientation line will lose the 2 lines above it.

```
Printer.Copies = 10
Printer.Print "Sample Text"
Printer.Orientation = PrintHorizontal
Printer.EndDoc
```

PrintRaw

This command will send a byte stream to the selected printer. A typical example would be sending escape sequences.

Syntax: Printer.PrintRaw(Data)

Data (Variant) is the byte stream or string that is to be sent to the printer. If the passed value is an actual byte array it will be sent to the printer unmodified. If it is not a byte array RFgen will convert it to an SBCS string and send that.

Example:

```
Dim sData As String  
sData = "Print Me"  
Printer.PrintRaw(sData)
```

Converts “Print Me” to Unicode and sends it

Screen Mapping Extensions

Screen mapping commands deal specifically with placing and scraping text to and from a green screen / legacy system. The following details all of the RFgen Screen Mapping VBA Extensions.

BeginTrans

This function is provided to allow a series of commands to be sent to a single host session when connection pooling is enabled. It basically gets a connection from the pool and holds it until SM.CommitTrans is called. If connection pooling is disabled, it has no effect. It returns True if a connection handle was available and removed from the pool.

Syntax: [OK =] SM.BeginTrans([Screen])

OK (Boolean) Optional – Returns True if a connection handle is available for use.

Screen (Variant) Optional – the name of a screen macro to be called so that upon connection the host will attempt to go to this screen. If the screen macro does not exist the pooled handle will be released and the function will return a False.

Example:

```
Dim bOK As Boolean  
bOK = SM.BeginTrans
```

CommitTrans

This subroutine basically returns the current handle to the connection pool (if it was previously retrieved by using the SM.BeginTrans function)

and makes it available to other client sessions. It returns True if the handle was successfully released back to the pool.

Syntax: [OK =] SM.CommitTrans

OK (Boolean) Optional – Returns True if the handle was released back to the connection pool.

Example:

```
Dim bOK As Boolean  
bOK = SM.CommitTrans
```

Connected

This function is used to determine if the host is available for direct input or if transactions should be queued for later input. It also evaluates both the current state of the socket connection to the host and the state of any data sent but not yet received. If there is non-received data in the send buffer, this command will mark the connection as closed. It returns True if the host is currently available.

Syntax: IsConnected = SM.Connected

IsConnected (Boolean) Returns True if the host is currently available (e.g. the telnet connection is currently valid).

Example:

```
Dim bIsConnected As Boolean  
bIsConnected = SM.Connected
```

CurScreen

This function returns the name of the current screen in the host session if it can be identified. RFgen identifies the screen by comparing the host screen to the Text Identifier grid and looks for an ID match. If a match is found, it returns the screen name otherwise this function returns an empty string.

Note: This function only works if the current screen has been defined as a macro that is linked to the main menu macro that is currently in use. For example if MainMenu1 links to TransactionScreen1 and Screen2 and MainMenu1 is either configured in the screen mapping connector as the default menu or the program performed the SM.SetBase("MainMenu1") command, then only Screen1 and Screen2 can be identified. If Screen3 is linked to MainMenu2 and you are on Screen3 but MainMenu1 is your base menu, Screen3 cannot be identified. Use the SM.GetText command to verify your location.

Syntax: sScreenName = SM.CurScreen([Page])

sScreenName (String) Returns the name of the current screen or an empty string if screen is unknown.
Page (Variant) Optional – returns the page of the current screen

Example:

```
Dim vPage As Variant  
Dim sScreenName As String  
sScreenName = SM.CurScreen(vPage)
```

FindText

This function is used to determine if a specified text string is currently displayed on the host screen. It can be used to look for the text in a specific screen location or to search the entire host screen for the text. It returns True if it is found. Optionally, it can also return the screen coordinates where the text was found.

Syntax: Found = SM.FindText(Text, StartCol, StartRow, [FoundCol], [FoundRow])

Found (Boolean) Returns True if the specified text string was found.

Text (String) Text string to be searched for.

StartCol (Integer) the screen column to search for the text. Specify a column position of '-1' (minus 1) to search for the text in any screen column position.

StartRow (Integer) the screen row to search for the text. Specify a row position of '-1' (minus 1) to search for the text in any screen row.

FoundCol (Variant) Optional – the column position where the text was found

FoundRow (Variant) Optional – the row position where the text was found.

Example:

```
Dim bFound As Boolean  
bFound = SM.FindText("UserID", 3, 5)
```

GetArea

This command gets the text off the screen in any rectangular area.

Syntax: [OK =] SM.GetArea(StartCol, StartRow, EndCol, EndRow, AreaText, [TrimData], [AppendData])

OK (Boolean) Optional – Returns True / False depending on the success of the command

StartCol	(Integer) start column position coordinate
StartRow	(Integer) start row position coordinate
EndCol	(Integer) end column position coordinate
EndRow	(Integer) end row position coordinate
AreaText	(String) variable containing the captured text
TrimData	(Boolean) Optional – trims each line (row) of data. Default is False
AppendData	(Boolean) Optional – appends the new block of data to the data variable rather than replaces it. Default is False

Example:

```
Dim sText As String
SM.GetArea(5, 5, 10, 10, sText, True)
SM.GetArea(5, 11, 10, 15, sText, True, True)
SM.GetArea(5, 16, 10, 20, sText, True, True)
```

GetAttribute

This command returns an integer value describing the characteristics of an X,Y coordinate from a host screen. This command does not work with a VT host, only 5250 and 3270.

Syntax: Value = SM.GetAttribute(Col, Row)

Value	(Integer) contains the value in the table below
-1	- error was returned
0	- Normal
1	- Bold
2	- Reverse Video
4	- Underline
8	- Half
16	- Protected
32	- Hidden
64	- Graphic

Col (Integer) is the column position in question.

Row (Integer) is the row position in question.

Example:

```
Dim Value As Integer
Value = SM.GetAttribute(5,18)
```

GetBackColor

This command returns an integer value representing the back color located at the specified coordinates. This command does not work with a VT host, only 5250 and 3270.

Syntax: Value = SM.GetBackColor(Col, Row)

Value	(Integer) contains the integer value of the back color -1 - Error was returned 0 - Black 1 - Blue 2 - Green 3 - Cyan 4 - Red 5 - Purple 6 - Yellow 7 - Gray 8 - Light Blue 9 - Light Green 10 - Light Cyan 11 - Light Red 12 - Light Purple 13 - Light Yellow 14 - Light Gray 15 - White
Col	(Integer) Is the current column position in question.
Row	(Integer) Is the current row position in question.

Example:

```
Dim Value As Integer
Value = SM.GetBackColor(3, 22)
```

GetCursor

This method is used to determine where the cursor is currently located on the host screen.

Syntax: SM.GetCursor(Col, Row)

Col	(Variant) Returns the current column position of the cursor.
Row	(Variant) Returns the current row position of the cursor.

Example:

```
Dim Col As Variant
Dim Row As Variant
SM.GetCursor(Col, Row)
```

GetForeColor

This command returns an integer value representing the fore color located at the specified coordinates. This command does not work with a VT host, only 5250 and 3270.

Syntax: Value = SM.GetForeColor(Col, Row)

Value	(Integer) contains the integer value of the fore color -1 - Error was returned 0 - Black 1 - Blue 2 - Green 3 - Cyan 4 - Red 5 - Purple 6 - Yellow 7 - Gray 8 - Light Blue 9 - Light Green 10 - Light Cyan 11 - Light Red 12 - Light Purple 13 - Light Yellow 14 - Light Gray 15 - White
Col	(Integer) Is the current column position in question.
Row	(Integer) Is the current row position in question.

Example:

```
Dim Value As Integer
Value = SM.GetForeColor(3,22)
```

GetPage

This method returns the page number of the current screen. If you designed a screen mapping transaction that uses more than 1 host screen and you identified those multiple screens within RFgen, this method returns the arbitrarily assigned page number.

Given a 3 page host transaction, RFgen will not know how to return from page 3 to page 1 automatically. This command lets you determine which page you are on and then you can include in the macro how to navigate back to page 1.

Syntax: SM.GetPage(ScreenName, Page)

ScreenName (String) The string name of the page in question.

Page (Variant) The variable that will contain the result of the inquiry.

Example:

```
Dim vPageNum As Variant
SM.GetPage ("Count_Txn", vPageNum)
```

GetText

This method is used to retrieve text from the host screen. Note: you can retrieve the entire screen buffer by specifying a starting position of 1,1 and a length of 2,000.

Syntax: SM.GetText(Col, Row, Length, ScreenText, [Trim])

Col (Integer) The starting screen column position to get text.
Row (Integer) The starting screen row position to get the text.
Length (Integer) The number of screen columns to retrieve text from. Note: on DBCS systems, this may not be the number of characters returned as some characters require 2 screen columns.
ScreenText (String) The text being returned.
Trim (Boolean) Optional – if True, the string will be returned with all padding removed. Default is False.

Example:

```
Dim sText As String  
SM.GetText(3, 6, 10, sText, True)
```

GoToScreen

This function attempts to navigate the host menu system to move to the requested application screen. It will return True if the desired screen is successfully displayed. Its method of operation (simplified) is as follows:

1. Test if the current screen is the requested screen.
2. Call the current screens “ReturnToMainMenu” method.
3. Call the requested screens “GoToScreen” method.

If the screen you are trying to get to is not part of the linked macros from the current main menu then RFgen will not navigate properly. The current main menu is defined in the screen mapping connector or by using the SM.SetBase command.

Syntax: [OK =] SM.GoToScreen(ScreenName)

OK (Boolean) Optional – Returns True if the host session was successfully moved to the requested screen.
ScreenName (String) The name of the screen, as identified in RFgen “Host Screens Tab”, to activate on the host session.

Example:

```
Dim bOK As Boolean  
bOK = SM.GoToScreen("UserLogin")
```

IsScreen

This function returns a True or False by comparing the specified screen and optionally the page number against the host screen's current page.

Syntax: [IsScreen =] SM.IsScreen(ScreenName, [Page])

IsScreen (Boolean) Optional – Returns True if the requested screen is the current active screen

ScreenName (String) The name of the screen, as identified in RFgen "Hosts tree", to be evaluated.

Page (Integer) Optional – The page number of the identified screen. This is setup when the page macro was recorded.

Example:

```
Dim bIsScreen As Boolean  
bIsScreen = SM.IsScreen("invTrans", 2)
```

LogOff

This function is used internally to logoff the session just prior to termination of the process. It works by sending the user to the base screen, as identified in RFgen "Host Screens Tab", and executes the 'LogOff' macro. It returns True if the session was logged off properly. Note: this function is not required to be called by the user as RFgen calls it automatically when the user disconnects.

Syntax: [OK =] SM.LogOff

OK (Boolean) Optional – Returns True if the session was successfully logged off.

Example:

```
Dim bSuccess As Boolean  
bSuccess = SM.LogOff
```

LogOn

This function is used to log in to the host system. It returns True if the session was logged in properly. The host system must already be logged off or you must use the SM.LogOff command. A user and password may be specified but the host does not use this information. This is for validation later in the programming. Note: this function is not always required to be called by the user as RFgen calls it automatically.

Syntax: [OK =] SM.LogOn([User], [Password], [Menu])

OK (Boolean) Optional – Returns True if the session was successfully logged on.

User (Variant) Optional – value accessed by SM.SessionUser

Password (Variant) Optional – value accessed by SM.SessionPwd

Menu (Variant) Optional – value specifying the menu to log in to

Example:

```
Dim bSuccess As Boolean  
bSuccess = SM.LogOn  
bSuccess = SM.LogOn("UserName", "Password",  
"MainMenu")
```

PadInput

This command adds spaces to the end of a string until the total length of the string is the size specified in the command. Using a number too small will truncate the string.

Syntax: SM.PadInput(Text, Length)

Text (String) the string of characters to be padded

Length (Integer) the number of spaces to add to the end of the string

Example:

```
Dim PartNo as String  
PartNo = "12345"  
SM.PadInput(PartNo, 8)
```

Result is “12345 “ with 3 spaces at the end

PingHost

This command sends a single ping packet to host and waits for response. It works against the current connection and can be used in transaction or navigation macros, used against a local connection, or you can get a pooled connection (SM.BeginTran) and then use it to test the connection.

Syntax: OK = SM.PingHost

OK (Boolean) returns True / False if the server can be reached

Example:

```
Dim bOK As Boolean  
bOK = SM.PingHost
```

ResetConnection

This command will reset the active host session in an effort to re-establish a locked-up connection.

Syntax: SM.ResetConnection

SendCTRL

This function sends the <CTRL> code with the specified character. It returns True if the key was successfully entered. Note: reasons for a negative or False value might be that keyboard entry is inhibited, or that the cursor was not in an input field.

Syntax: [OK =] SM.SendCTRL(Key)

OK (Boolean) Optional – Returns True if the key was successfully sent to the host session.

Key (String) Valid characters are: A-Z, space, [, /,], ~, ?, 0-9

Example:

```
Dim bOK As Boolean  
bOK = SM.SendCTRL("C")
```

SendCTRLAlt

This function sends the <CTRL> code with the specified character to a specific location in host session's screen. It returns True if the key was successfully entered. Reasons for a negative or False value might be that keyboard entry is inhibited, or that the coordinates were not an input field. Note: the ability to move the cursor to a specific location is only supported for tn5250 and tn3270 sessions.

Syntax: [OK =] SM.SendCTRLAlt(Col, Row, Key)

OK (Boolean) Optional – Returns True if the key was successfully sent to the host session.

Col (Integer) The screen column position to input the key.

Row (Integer) The screen row position to input the key.

Key (String) Valid characters are: A-Z, space, [, /,], ~, ?, 0-9

Example:

```
Dim bOK As Boolean  
bOK = SM.SendCTRLAlt(8, 12, "C")
```

SendKey

This function sends the selected key to the current cursor location in the host session's screen. It returns True if the key was successfully entered. Note: reasons for a negative or False value might be that keyboard entry is inhibited, or that the cursor was not in an input field.

Syntax: [OK =] SM.SendKey(Key)

OK (Boolean) Optional – Returns True if the key was successfully sent to the host session.

Key (enCommandKeys) Select from the drop-down list the desired key to send to the host session. Options are:

KeyAttention	KeyBackspace
KeyBackTab	KeyBreak
KeyClear	KeyCursorDown
KeyCursorLeft	KeyCursorRight
KeyCursorUp	KeyDelete
KeyDo	KeyDuplicate
KeyEnd	KeyEnter
KeyEscape	KeyF1 – KeyF24
KeyFieldAdvance	KeyFieldBack
KeyFieldErase	KeyFieldExit
KeyFieldMark	KeyFieldMinus
KeyFieldPlus	KeyFind
KeyHelp	KeyHome
KeyInsert	KeyPA1 – KeyPA3
KeyPageDown	KeyPageUp
KeyRemove	KeyReplace
KeyReset	KeySelect
KeySystemRequest	KeyTab

Example:

```
Dim bOK As Boolean  
bOK = SM.SendKey(KeyEnter)
```

SendKeyAlt

This function sends the selected key to a specific location in host session's screen. It returns True if the key was successfully entered. Reasons for a negative or False value might be that keyboard entry is inhibited, or that the coordinates were not an input field. Note: the ability to move the cursor to a specific location is only supported for tn5250 and tn3270 sessions.

Syntax: [OK =] SM.SendKeyAlt(Col, Row, Key)

OK (Boolean) Optional – Returns True if the key was successfully sent to the host session.

Col (Integer) The screen column position to input the key.

Row (Integer) The screen row position to input the key.

Key (enCommandKeys) Select from the drop-down list the desired key to send to the host session. For a list see SM.SendKey.

Example:

```
Dim bOK As Boolean  
bOK = SM.SendKeyAlt(8, 12, KeyEnter)
```

SendText

This function is used to send text to the current cursor location in the host session's screen. It returns True if the text was successfully entered. Note: reasons for a negative or False value might be that keyboard entry is inhibited, that the cursor was not in an input field, or that the text exceeded the input field's length.

Syntax: [OK =] SM.SendText(Text, [PadSize], [PadChar])

OK (Boolean) Optional – Returns True if the text was successfully sent to the host session.

Text (String) The desired text to send to the host session.

PadSize (Integer) Optional – total length of padded string

PadChar (Variant) Optional – the character to use for padding

Examples:

```
Dim bOK As Boolean  
bOK = SM.SendText("SAM")  
SM.SendText("SAM", 10, " ")
```

SendTextAlt

This function is used to send text to a specific cursor location in the host session's screen. It returns True if the text was successfully entered. Reasons for a negative or False value might be that keyboard entry is inhibited, that the coordinates were not an input field, or that the text exceeded the input field's length. Note: the ability to move the cursor to a specific location is only supported for tn5250 and tn3270 sessions.

Syntax: [OK =] SM.SendTextAlt(Col, Row, Text, [PadSize], [PadChar])

OK (Boolean) Optional – Returns True if the text was successfully sent to the host session.

Col (Integer) The screen column position to input the text.

Row (Integer) The screen row position to input the text.

Text (String) The desired text to send to the host session.

PadSize (Integer) Optional – total length of padded string

PadChar (Variant) Optional – the character to use for padding

Example:

```
Dim bOK As Boolean  
bOK = SM.SendTextAlt(10, 8, "SAM")
```

SessionID

This function is used return the current session or pool number. If Connection Pooling is disabled, the Transaction Manager may have to establish its own connection if macros are queued. If this is the case the

returned integer will be 0. If Connection Pooling is enabled, the pool number (1, 2, 3 etc.) will be returned.

Syntax: Value = SM.SessionID

Value (Integer) stores the pool number used by the client

Example:

```
Dim Value as Integer  
Value = SM.SessionID
```

SessionPwd

This function is used to set or return the Password associated with a host session. For pooled connections the password can be defined under Connections / Connection X (Screen Mapping) / Pooling option. For non-pooled connections the SM.SessionPwd can assign the password.

Syntax: Value = SM.SessionPwd

Alternate: SM.SessionPwd = Value

Value (String) variable containing the password

Example:

```
Dim Value as String  
Value = SM.SessionPwd
```

SessionUser

This function is used to set or return the User ID associated with a host session. For pooled connections the user can be defined under the Connections / Connection X (Screen Mapping) / Connection Pooling option. For non-pooled connections the SM.SessionUser can assign the user ID.

Syntax: Value = SM.SessionUser

Alternate: SM.SessionUser = Value

Value (String) variable containing the user ID

Example:

```
Dim Value as String  
Value = SM.SessionUser
```

SetBase

This function is used to switch to an alternate Main Menu. This command must be run while on a menu screen and not an application screen.

Syntax: [OK =] SM.SetBase(Menu)
OK (Boolean) Optional – returns True / False depending on the
 success of the command
Menu (String) the name of the new base menu

Example:

```
SM.SetBase ("MainMenu")
```

SetCursor

This function is used to move the cursor to a specified location on the host screen. It returns True if the cursor was successfully moved to the requested location. Note: the ability to move the cursor to a specific location is only supported for tn5250 and tn3270 sessions.

Syntax: [OK =] SM.SetCursor(Col, Row)
OK (Boolean) Optional – Returns True if the cursor was
 successfully moved in the host session.
Col (Integer) The desired new column position for the cursor.
Row (Integer) The desired new row position for the cursor.

Example:

```
Dim bOK As Boolean  
bOK = SM.SetCursor(2, 9)
```

SetDelay

This function is typically used for debugging purposes against a vt220 host session. It allows users to set a delay, in milliseconds, for all screen mapping VBA extensions that change what is on the host screen. This allows the user to watch in slow motion, all screen updates as the result of a macro. Note: the user could also accomplish this by single stepping through the macro's VBA code in the test environments VBA debug window.

Syntax: SM.SetDelay(MilliSeconds)
MilliSeconds (Long) The delay to set in milliseconds.

Example:

```
SM.SetDelay(1000)
```

SetSession

This function is used when you are connecting to multiple hosts via Screen Mapping. It allows you to specify which screen mapping session

(RFgen Connection) is active. Note: the first RFgen screen mapping connection is set as the active session by default.

Syntax: [OK =] SM.Session(Source)
OK (Boolean) Optional – Returns True if the requested session is defined as a valid Screen Mapping Connection.
Source (Variant) The RFgen connection number or the name of the host session (as displayed in the RFgen status bar).

Example:

```
Dim bOK As Boolean  
bOK = SM.Session(2)
```

SetTimeout

This function is used with all other SM commands, providing a maximum length of time before the other SM commands fail after no response from the host system. The internal default is 5 seconds.

Syntax: SM.setTimeout(Seconds)
Seconds (Long) The number of seconds

Example:

```
SM.setTimeout (10)
```

WaitForCursor

This function is used to time your commands to the host session. With it you can delay sending text or keys to the host session, or retrieving data from the host session until the cursor is in a specific location. It basically allows you to wait for the cursor to arrive at a specific position in the host screen for a certain amount of time. Once the cursor reaches the desired location this function will immediately return with a value of True, otherwise it will timeout and return False.

Syntax: [OK =] SM.WaitForCursor(Col, Row, Seconds)
OK (Boolean) Optional – Immediately returns True if the cursor stopped at the desired location.
Col (Integer) The column position to wait for cursor.
Row (Integer) The row position to wait for cursor.
Seconds (Integer) The maximum number of seconds to wait for the cursor to reach the requested position.

Example:

```
Dim bOK As Boolean  
bOK = SM.WaitForCursor(6, 14, 5)
```

WaitForCursorMove

This function is also used to time your commands to the host session. With this command, you specify only an amount of time in seconds. If the cursor has changed positions within that time, a True result is returned. Otherwise it will timeout and return False.

Syntax: [OK =] SM.WaitForCursorMove(Seconds)

OK (Boolean) Optional – Immediately returns True if the cursor has changed locations.

Seconds (Integer) – The maximum number of seconds to wait for the cursor to change locations.

Example:

```
Dim bOK As Boolean  
bOK = SM.WaitForCursorMove(5)
```

WaitForHost

This function is used to time your commands to a vt220 host session. With it you can delay sending text or keys to the host session, or retrieving data from the host session until the host has responded to the last command sent. Whenever input data is sent to a host session, RFgen sets a switch that indicates whether the host has responded. This function basically allows you to wait for a certain amount of time until the host processes and responds to the last input command. Once the host response has been received and processed by RFgen this function will immediately return with a value of True, otherwise it will timeout and return False.

Note: There is a difference with the VT environment where this command will return a 'True' after the last command finishes even if the host is not ready for the keyboard input. This is because in the VT environment, the keyboard is never locked and in the 5250 and 3270 environments, this commands waits for the keyboard to be unlocked before returning any values.

Syntax: [ReplyReceived =] SM.WaitForHost(Seconds)

ReplyReceived (Boolean) Optional – Immediately returns True once the host responds to your last input command.

Seconds (Integer) The maximum number of seconds to wait for the host to respond to your last input command.

Example:

```
Dim bReplyReceived As Boolean  
bReplyReceived = SM.WaitForHost(5)
```

WaitForScreen

This function is used to confirm that we have reached the desired host application screen. With it you can positively confirm that the 'GoToScreen' or 'ReturnToMainMenu' functions have succeeded. It allows you to wait for a certain amount of time for the desired screen to be positively identified. Once the ID match has occurred, this function will immediately return with a value of True, otherwise it will timeout and return False.

Syntax: [OK =] SM.WaitForScreen(ScreenName, Seconds)

OK	(Boolean) Optional – Immediately returns True if the host session is now in the desired screen.
ScreenName	(String) The host screen that we wish to confirm is active.
Seconds	(Integer) The maximum number of seconds to wait for the screen to be identified.

Example:

```
Dim bOK As Boolean  
bOK = SM.WaitForScreen("InventoryMovements", 5)
```

WaitForText

This function is used to time your commands to the host session. With it you can delay sending text or keys to the host session, or retrieving data from the host session until a specific text string has appeared on the screen. It allows you to wait for a text string to appear anywhere on the screen, or only at a specific position. Once the text appears, this function will immediately return with a value of True, otherwise it will timeout and return False.

Syntax: [Found =] SM.WaitForText(Text, Seconds, [Col], [Row])

Found	(Boolean) Optional – Immediately returns True once the specified text appears on the screen.
Text	(String) The text to find.
Seconds	(Integer) The maximum number of seconds to wait for the screen to be identified.
Col	(Integer) Optional – the column position to look for the text. Use '-1' for any Column
Row	(Integer) Optional – the row position to look for the text. Use '-1' for any row.

Example:

```
Dim bFound As Boolean
```

```
bFound = SM.WaitForText("Enter Next Task Code:", 5, -1, -1)
```

WaitForWrite

This function waits for a specified number of seconds for data to be entered at a specific location and returns a True or False. If data was written within the wait time, True is returned. If the number of seconds expires first, False is returned.

Syntax: [OK =] SM.WaitForWrite(Col,Row,Seconds)

OK (Boolean) Optional – Returns True if data was written at the specified coordinates within the specified time frame
Col (Integer) The column position to watch.
Row (Integer) The row position to watch.
Seconds (Integer) The maximum number of seconds to wait for data to be written.

Example:

```
Dim bOK As Boolean  
bOK = SM.WaitForWrite(24,13,5)
```

Text to Speech Extensions

The TTS commands are for listening and interpreting what the user is speaking and for specifying how the system will talk to the user. There are a number of commands that can also be used by the user to improve the speech playback experience on the fly. Using a global word list and some special or reserved words like "Next", "Back", "Faster", "Slower", "Confidence Up", "Confidence Down", "Volume Up", "Volume Down", "Silence Up", "Silence Down", the user can use voice commands to change RFgen's behavior. Moving the prompt focus up or down, changing volume, changing the read back rate, loosening or restricting the speech clarity required for input, etc. are all possibilities.

ActiveGrammar

This property will allow you to enable prompt grammar only, global grammar only or both. An example may be to turn off the global grammar at any time to improve the accuracy of the recognition of the speech at the prompt level. If the global grammar has many words but the prompt only is expecting a Yes or No then you can turn off the global grammar and use the TTS.AddGrammar(sePrompt, "Yes", "No") command to add just the expected words. If the system has only these 2 words to use for comparison rather than a long list of allowed words the accuracy will be increased.

Syntax: TTS.ActiveGrammar = Value

Value (enGrammarMode) is either seEnableAll, seGlobalOnly, or
sePromptOnly

Example:

```
TTS.ActiveGrammar = seEnableAll
```

AddGrammar

Grammar is a list of allowable commands added to either the global list of commands or the prompt level list that are not listed in the global grammar. For example, if the prompt is expecting only a Yes or No and those words do not exist in the global grammar then they can be added to the global list or just at the prompt level. They will remain allowable commands until removed using the TTS.RemoveGrammar command. This command is also used in conjunction with TTS.ActiveGrammar to improve accuracy.

Syntax: TTS.AddGrammar(Type, Words)

Type (enGrammarType) specifies either an add to the Global list or the Prompt list. (Options are seGlobal and sePrompt)

Words (ParamArray) is the list of words to be added

Example:

```
TTS.AddGrammar(sePrompt, "Yes", "No", "Skip")
```

AddGrammarFile

This command allows the user to load a pre-defined grammar list and add it either to the prompt level list or the global list. First you must create a Speech Resource under the Resources tab and give it an ID.

Syntax: TTS.AddGrammarFile(Type, ID)

Type (enGrammarType) specifies an add to the Global list or the Prompt list. (Options are seGlobal and sePrompt)

ID (String) is the Resource ID of the pre-defined grammar list

Example:

```
TTS.AddGrammarFile(seGlobal, "English_All_Letters")
```

ClearGrammar

This command clears all words stored in the prompt list or the global list. To clear both lists just use this command 2 times. They can be added back again later with the TTS.AddGrammar command if desired.

Syntax: TTS.ClearGrammar(Type)

Type (enGrammarType) specifies either an add to the Global list or the Prompt list. (Options are seGlobal and sePrompt)

Example:

`TTS.ClearGrammar(seGlobal)`

ClearProfile

This command clears the active user's speech profile in memory and optionally deletes the profile from the user configuration.

Syntax: TTS.ClearProfile>Delete)

Delete (Boolean) False means that only the memory is cleared of user specific speech training, where True will clear memory and delete the user specific speech training files from the master database.

Example:

`TTS.ClearProfile(True)`

ConfidenceLevel

This property requires that the speech input from a user meet a minimum confidence level threshold before it is accepted and returned as valid input. The default is 5,000 within a range of 0 – 10,000. If the speech engine determines that the user input evaluates to less than 5,000 it will be ignored. Using this property to lower the threshold means the speech engine should not be as critical when comparing speech input to the defined grammar list of acceptable words. Increasing this value would make the speech engine more critical and require the user to be clearer when speaking.

Syntax: TTS.ConfidenceLevel = Value

Alternate: Value = TTS.ConfidenceLevel

Value (Integer) specifies a number between 0 - 10,000

Example:

`TTS.ConfidenceLevel = 4000`

DemoMode

This property allows the user to either talk at the same time the system is speaking or force the listening of the system to wait until all speaking is complete. If there is the possibility that the speakers are loud enough for the microphone to hear and therefore interfere with data collection, this command can be set to a synchronous mode that prevents listening and speaking from happening at the same time. If the users are using headsets and can anticipate the transaction's request for input, while the

instructions are being given the user could speak the answer and possibly gain efficiency. In the Asynchronous mode there is the possibility the user could get out of sync or too far ahead of the system. This can be remedied by using a combination of both modes forcing the user to wait at some point.

Syntax: TTS.DemoMode = Value

Alternate: Value = TTS.DemoMode

Value (Boolean) is either True or False. True is synchronous

Example:

```
TTS.DemoMode = True
```

DisplayGrammar

This property will return the currently active grammar in memory. This is most useful when performing training (TTS.Train) on the words and then using TTS.SaveProfile to commit the active user's profile to the server.

Syntax: List = TTS.DisplayGrammar

List (String) contains a list of understood grammar from both the prompt level and global level

Example:

```
Dim sList As String  
sList = TTS.DisplayGrammar
```

Enabled

This is a read-only command to dynamically determine if speech is enabled and available for the client user. It is equivalent to looking at the Enable RFgen Speech Support check box in the configuration of the Speech Recognition.

Syntax: Value = TTS.Enabled

Value (Boolean) returns True if speech support is available

Example:

```
Dim bValue As Boolean  
bValue = TTS.Enabled
```

GetInput

This command is intended to function like an input box would, asking and waiting for an answer from the user. It has an optional timeout value that defaults to 10 seconds. If the user does not respond within that time the command will return an error and the code will continue to execute. This command returns 1 of 4 enumerated values as output.

Syntax: [Value =] TTS.GetInput(Rsp, [Seconds])
Value (Boolean) Optional – is True for False for the success of the command
Rsp (String) is the response from the user
Seconds (Integer) Optional - is the number of seconds to wait for user input. The default is 10 seconds unless the TTS.WaitTime command was already used to change the default value.

Example:

```
Dim bOK As Boolean  
Dim Rsp As String  
bOK = TTS.GetInput(Rsp, 30)
```

LoadProfile

This command will clear and reload the active user's speech profile from the user configuration into memory.

Syntax: [Value =] TTS.LoadProfile

Value (Boolean) Optional – either True or False based on the success of the command

Example:

```
Dim bValue As Boolean  
bValue = TTS.LoadProfile
```

PauseTime

This property will change the default wait time between all succeeding message units (time between each spoken word in a sentence). The default value is 400 milliseconds.

Syntax: TTS.PauseTime = Value

Alternate: Value = TTS.PauseTime

Value (Integer) is a number between 0 and 1800

Note: the value must be in 200 millisecond increments like 200, 400, 600...

Example:

```
TTS.PauseTime = 1000
```

ReadMode

The read mode determines the way in which the system will split the input text into message units. Each message unit will then be separately

processed and pronounced by the system. The ‘Sentence’ mode is the default.

Syntax: TTS.ReadMode = Mode

Alternate: Mode = TTS.ReadMode

Mode (enReadMode) is an enumeration containing options for ‘seCharacter’, ‘seList’, ‘seSentence’ and ‘seWord’.

Example:

```
TTS.ReadMode = seSentence
```

ReadRate

This property will determine how fast the speech is spoken.

Syntax: TTS.ReadRate = Value

Alternate: Value = TTS.ReadRate

Value (Integer) is a number between 1 and 100. The default is 50.

Example:

```
TTS.ReadRate = 75
```

RemoveGrammar

At any time after the use of TTS.AddGrammar, this command can remove the extra allowable grammar words from the system’s vocabulary.

Syntax: TTS.RemoveGrammar(Type, Words)

Type (enGrammarType) selects either Prompt or Global level grammar. (Options are seGlobal and sePrompt)

Words (ParamArray) a list of words to be removed. Specifying nothing for this parameter will have no effect.

Example:

```
TTS.RemoveGrammar(sePrompt, "Yes", "No")
```

If the TTS.AddGrammar command was used to add many different words including Yes and No, this command would remove just the Yes and No and leave the rest available.

RemoveGrammarFile

At any time after the use of TTS.AddGrammarFile, this command can remove the Speech Resource from the system’s vocabulary.

Syntax: TTS.RemoveGrammarFile(Type, ID)

Type (enGrammarType) selects either Prompt or Global level grammar. (Options are seGlobal and sePrompt)
ID (String) is the Resource ID of the pre-defined grammar list

Example:

```
TTS.RemoveGrammarFile(seGlobal,  
"English_All_Letters")
```

Repeat

This command will repeat whatever was the last text spoken.

Syntax: TTS.Repeat

Example:

```
TTS.Repeat
```

SaveProfile

This command will take the current speech profile stored in memory and replace the profile stored on the server for the active user. The active user is determined by the login process or the App.User command.

Syntax: [Value =] TTS.SaveProfile

Value (Boolean) Optional – is True or False based on the success of the command

Example:

```
Dim bValue As Boolean  
bValue = TTS.SaveProfile
```

SetLanguage

Depending on the purchased language packs and the pre-configured voices available for that language, the user may set the language spoken and the voice (tone and pitch) for how the system will speak.

Syntax: TTS.SetLanguage(Language, Voice)

Language (String) is a string that names the desired language

Voice (String) is a string that names the voice pattern to be used

Example:

```
TTS.SetLanguage("American English", "Jill")
```

Speak

This command is used to speak text to the user. When having acronyms spoken like “TTS” you would want to place a space between the letters

like T T S so that it is pronounced correctly. The use of commas and periods can also influence the pausing while saying different words.

Syntax: TTS.Speak(Sentence)
Sentence (String) is a word or phrase to be spoken

Examples:

```
TTS.Speak("I am the quick brown fox.")  
TTS.Speak("I am saying T T S.")  
TTS.Speak("I am saying, T, T, S, slowly.")
```

Spell

This command is used to speak each letter on the provided text. In theory setting the TTS.ReadMode to Character would have the same effect.

Syntax: TTS.Spell(Word)
Word (String) is a string that the user wants to hear spoken character by character

Example:

```
TTS.Spell("Hello")
```

This should spell out each letter as if it were being read H E L L O.

StopSpeak

This command will stop the system from speaking the remainder of the text for the current and all other TTS.Speak commands that came before the TTS.StopSpeak command.

This only applies if the TTS.DemoMode is not changed to True. In DemoMode only 1 line of code runs at a time and there is no way to reach the TTS.StopSpeak command until all other lines have been completed. In the case of a TTS.Speak, all the text will be spoken.

Syntax: TTS.StopSpeak

Example:

```
TTS.StopSpeak
```

TestMic

This command is used to begin recording through the device's microphone for a specific duration and then send the captured stream of sound to the Mobile Development Studio to be played back through the speakers. Since this command is designed for testing and requires real

time communication with the server, it has no use in mobile mode. It can optionally write a log file that would help diagnose poor recognition.

Syntax: TTS.TestMic([Duration], [SavePath])

Duration (Variant) Optional – a number of seconds that should be recorded and played back. If this value is omitted then the default timeout value configured in the speech options will be used.

SavePath (String) Optional – the path where the system can write a log file containing debug information that DataMAX can use to diagnose poor speech recognition. The file will be generated by the system and should be sent to DataMAX for interpretation.

Examples:

`TTS.TestMic(5)`

`TTS.TestMic(5, "C:\")`

TrailingSilence

This property is used to tell the speech engine how long it should wait before determining when the user has finished speaking. The range is 0 – 1800 milliseconds and the default is 400. For speech that is slow this number can be increased to prevent the speech engine from returning the input too early.

Syntax: TTS.TrailingSilence = Value

Alternate: Value = TTS.TrailingSilence

Value (Integer) a value between 0 – 1800 milliseconds of wait time

Example:

`TTS.TrailingSilence = 800`

Train

This command is used to train the speech engine to better recognize a user's speech pattern. This is not word training where a word or a list of words becomes available but only giving the speech engine a better understanding on how the user says various sounds in general. Performing training in different environments may influence the stored speech recognition.

This new understanding is saved with the user's profile and downloaded to any thin client device that user uses after they log in. This command will modify the current profile in memory only so you must use the TTS.SaveProfile to commit the changes after the training.

Syntax: [Value =] TTS.Train(Word, [Timeout])

Value (Boolean) Optional – True is the success of the training of the specified word

Word (String) is the word that you would like to use for improving the speech engine adaptations to a user's speech pattern.

Timeout (Variant) Optional – Timeout is specified in milliseconds. The default is 10 seconds. The timeout is how long the system will wait for an utterance from the user.

Example:

For the best understanding on how the user says the numbers 1, 2, 3 and 4 the training should look like:

```
TTS.Train("1234")
TTS.Train("4321")
TTS.Train("1423")
TTS.SaveProfile
```

TrimInput

This property is used to tell the speech engine to remove all spaces from the captured speech result. When the OnSpeech event fires or the TTS.GetInput command is used, the resulting data may contain spaces. For example, entering a part number of 1234 must be spoken as 1 2 3 4. In this case, the spaces need to be removed for proper validation. If there is a case where the user speaks something where spaces are desired, set this command to False. Note: there is a configuration option to set this value globally.

Syntax: TTS.TrimInput = Value

Alternate: Value = TTS.TrimInput

Value (Boolean) True will make the speech engine deliver data with no spaces; False does not alter the data

Example:

```
TTS.TrimInput = True
```

Volume

The volume is how loud RFgen will speak the text. The range 0 – 100 and is recommended that it be set at the maximum or loudest level. This volume is still subject to the volume setting of the operating system and / or hardware device. Since it is usually much easier to access the volume controls of the device (buttons on the side of the device or a speaker icon on the desktop) volume can be more easily adjusted for comfort there.

Syntax: TTS.Volume = Value

Alternate: Value = TTS.Volume
Value (Integer) a number between 0 and 100

Example:
`TTS.Volume = 100`

WaitTime

This property sets the timeout value used by RFgen to wait for input either from the TTS.GetInput command or the TTS.Listen command. Since it is optional for TTS.GetInput you can use TTS.WaitTime once and it will be in effect globally. The default value is 10 seconds.

Syntax: TTS.WaitTime = Value
Alternate: Value = TTS.WaitTime
Value (Integer) is a number of seconds

Example:
`TTS.WaitTime = 45`

Database List Object

A list is a full screen display of selected items; i.e., the data entry device screen is cleared and the contents of the list are displayed. A list box is different in that it displays selected items in the original prompt space allocated for the data in the application.

List

This object is an advanced method of creating a scrolling list of items that may then be presented to the user for selection.

`Dim oMyList as New List`

This object has the following methods:

AddListEntry

This method adds an item to the list

Syntax: oMyList.AddListEntry(SelValue, ColValues)
SelValue (Variant) The value to be returned when this item is chosen
ColValues (param array of Variants) a comma separated list of values to be displayed in each column

Example:
`Dim oMyList as New List`

```
oMyList.AddListEntry(123, "1st Col Description", "2nd  
Col Description")
```

Clear

This method clears the list's contents.

Syntax: oMyList.Clear

MaxRows

This property limits how many rows will be allowed in the list. If the database will return several thousand records, you may want to limit this list to 500. An alternative is to further restrict the search criteria if using SQL.

Syntax: oMyList.MaxRows = nNum

Alternate: nNum = oMyList.MaxRows

nNum (Long) is a numeric value to limit the rows in the list

ReturnAllRows

This property when set to True instructs the list to return all the values for all the columns instead of the first value of the first column when a row is selected.

Syntax: oMyList.ReturnAllRows = Value

Alternate: Value = oMyList.ReturnAllRows

Value (Boolean) a True or False value (*default = False*)

SetReportColumn

This method formats the returned data to fit the device's screen and desired layout. This statement should be used for each column to be displayed.

Syntax: oMyList.SetReportColumn(Index, Heading, Length, Align, TrimSpaces, [Decimals])

Index (Long) the column number to be affected by the formatting

Heading (String) the title that will appear across the top of the column

Length (Long) the width of the column

Align (String) how to align the column; either Left or Right

TrimSpaces (Boolean) True means that leading and trailing spaces will be removed from the display of the data in this column

Decimals (Long) Optional – if the value is numeric and the database does not store decimals, use this to position a decimal at a position from the right side. A comma will be used for large numbers. This field is locale specific and will use the appropriate characters for each region

Example:

```
Dim oMyList as New List  
oMyList.SetReportColumn(1, "My Heading", 10, "L", True, 2)  
oMyList.SetReportColumn(2, "Num", 3, "R", False)
```

ShowEmptyList

This property displays the blank list to the user even when there are no entries. The whole screen will be temporarily cleared and the user must press enter to acknowledge there were no entries.

Syntax: oMyList.ShowEmptyList = Value

Alternate: Value = oMyList.ShowEmptyList

Value (Boolean) is either True or False.

Example:

```
Dim oMyList as New List  
oMyList.ShowEmptyList = True
```

ShowList

This function displays the list to the user. The whole screen will be temporarily cleared and the list will be displayed until a choice is made.

Syntax: Value = oMyList.ShowList

Value (String) is a Chr(1) delimited list of the values in the columns based on the row selected.

Example:

```
Dim oMyList as New List  
sValue = oMyList.ShowList
```

SQL

This property will contain the SQL statement to be used in creating the list.

Syntax: oMyList.SQL = SQL

Alternate: SQL = oMyList.SQL

SQL (String) the SQL statement to be executed

Example:

```
Dim sSQL As String  
Dim oMyList as New List  
sSQL = "Select * from Inventory"  
oMyList.SQL = sSQL
```

Embedded Procedure Object

The following section describes:

- Embedded Procedures – ERP and other stored procedures, macros, etc. that allow functions to be executed without using a front-end interface
- Properties and Methods – The properties of the function are the values sent and received from the backend system and the methods are commands used to manipulate the function such as ‘execute’ or ‘clear’ that operate on the object itself.

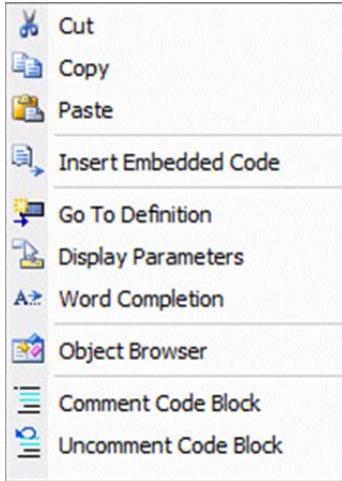
Embedded Procedures

An Embedded Procedure refers to embedding VBA code that performs a previously created task. The user may generate code for:

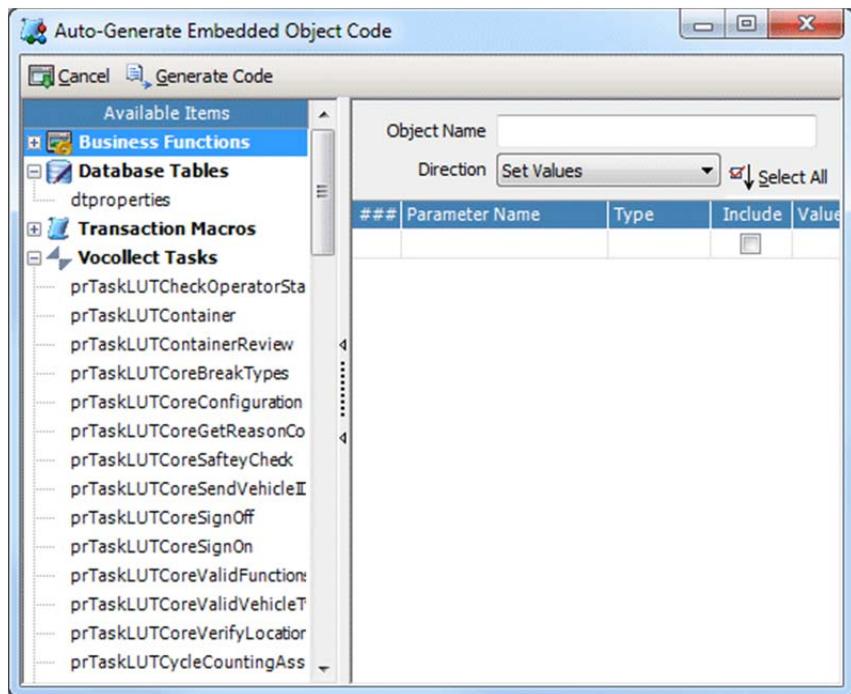
- Business Functions – typically used with ERP systems
- Business Objects – an object variable used by Microsoft Axapta
- Database Table – used to generate Insert SQL statements
- Stored Procedures – found in ODBC compliant databases
- Transaction Macros – to be queued or executed
- Vocollect Recordsets – defined Vocollect resources
- Web Service Objects – based on the Web Service Connector

Embedded Procedures are intended to be an automated approach to writing VBA code that specifies parameters and then executes the procedure. Although you can write the code yourself, RFgen has a code generator that makes this process much easier.

In the VBA environment right-click and select “Insert Embedded Code”.

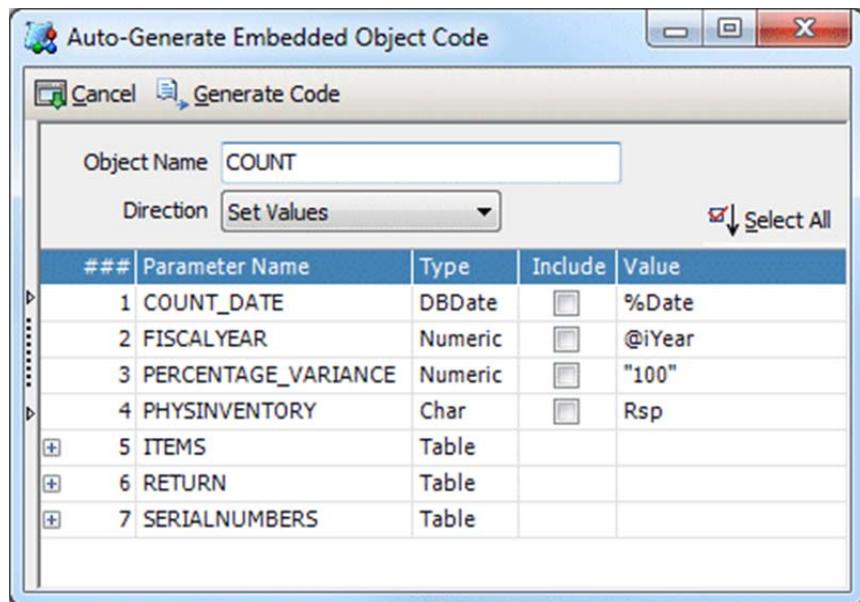


Next, select the type of object to embed from the list on the left.



A list is shown of all previously downloaded objects of that type. If the item does not appear in the list, it must first be downloaded from the backend data source.

To select the desired business function double-click on the listed function. Alter any default values that may be required. Any default values that were assigned after downloading the function (under the menu item Data Link – Edit Tables / Business Functions) will appear blue. These default values will be assumed when the function executes. You can at this time override the default value, which will then turn red if it indicates that a default was changed. Any entries that do not have a value will remain black and are only used for the immediate pasting of the Embedded Procedure code. The value column is the only editable field.



When the code is generated, only the lines with non-default values are displayed. RFgen knows to send all parameters to the function, but unless there is a non-default value, the code window is not needlessly filled with every available parameter. A basic embedded procedure will look similar to the following:

```
Dim iYear As Integer
Dim Rsp As Variant
Dim emCOUNT As New EmbeddedProc
'
emCOUNT.Name = "BAPI_MATPHYSINV_COUNT"
emCOUNT.DataSource = "SAP"
```

```

        ,
emCOUNT.Param("COUNT_DATE") = App.GetValue("Date")
emCOUNT.Param("FISCALYEAR") = iYear
emCOUNT.Param("PERCENTAGE_VARIANCE") = "100"
emCOUNT.Param("PHYSINVENTORY") = Rsp
,
If Not emCOUNT.Execute Then
End If

```

As you can see, the parameters are filled in with the default values. If you choose, you could replace “2011” with RFPrompt(“Year”).Text.

There are several options when filling in the Value column. When only an '@' symbol is used, the name of the parameter itself will be declared as a variable to be used or replaced later. If a value is wrapped with double quotes RFgen will place it in the code as a literal like the “2011” value in the above example. If any string begins with a percent sign the value will be used inside an App.GetValue or App.SetValue command. The Hungarian notation for Integer (i), Long (n), Boolean (b), Variant (v) and String (s) placed in front of the value will ensure the variable declaration as that type. Variant is the default type if the system cannot determine the type or a standard variable name is used. Any standard string by itself will be made into a variable.

Embedded Procedure Properties and Methods

The following list of properties and methods are used to setup a procedure and / or be evaluated after the procedure executes.

ClassObject

When using Microsoft ERP systems, there are functions that can be called that return class objects. This property can then be set to that returned object and methods on that object can then be called.

Syntax: emProc.ClassObject = oClass

oClass (object) the name of the class object required by the method.

Example:

```
emProc.ClassObject = oMyClassObject
```

Clear

After the programmer or the Execute method has altered parameters, this command clears all parameters so it may be used again. This only effects the emProc.Param() values.

Syntax: emProc.Clear

ColumnCount

For a given table, this function will contain the number of parameters or columns as they appear when downloaded.

Syntax: Value = emProc.ColumnCount(ParamId)

Value (Variant) contains the number of columns or parameters in the specified table.

ParamId (Variant) the name of the table usually referred to by name and in double quotes.

Example:

```
Dim Value As Long  
Value = emProc.ColumnCount ("RETURN")
```

ColumnName

For a given table or function, this function will display the name of a parameter when referenced by its index value. If there are no tables use the property ParamName.

Syntax: Value = emProc.ColumnName(Table, [Index])

Value (String) contains the name of the parameter

Table (Variant) the table or function name that contains parameters

Index (Long) Optional – the number of the parameter for which the name is retrieved. Left off and the complete list will be returned.

Example:

If the embedded procedure had these parameters declared:

```
emProc.Param ("PLANT", "SIGN")      = "I"  
emProc.Param ("PLANT", "OPTION")     = "EQ"  
emProc.Param ("PLANT", "PLANT_LOW")  = "3000"
```

then emProc.ColumnName("PLANT", 3) would return "PLANT_LOW"

DataSource

This property indicates the name of the data source to connect to. This is generally setup in the HOSTS file assigning an IP address to a name.

Syntax: emProc.DataSource = Value

Value (String) name of the data source

Example:

```
emProc.DataSource = "SAP1"
```

DebugLog

This property contains the path to the log file. The log file will contain all function calls by appending to this file.

Syntax: emProc.DebugLog = Value

Alternate: Value = emProc.DebugLog

Value (String) the path to a text file

Example:

```
emProc.DebugLog = "\Program Files\RFgen\Status.txt"
```

DisableParam

Only used for SAP connectivity, this method allows the user to request that SAP not send data back in tables that will not be used. If a BAPI returns 5 tables of data but only 1 will be used, the other 4 can be turned off to increase the speed SAP returns with results.

Syntax: emProc.DisableParam(ParamID, Disabled)

ParamID (Variant) the name of the table that is not necessary

Disabled (Boolean) set to True if the table should be ignored.

Execute

After all values are correctly set, this command sends the values to the ERP system for evaluation. The function being executed is contained in the emProc.Name property. The properties of the procedure are then filled in with the results.

Syntax: [Value =] emProc.Execute

Value (Boolean) Optional – contains a True / False based upon its success.

ExecuteMethod

For Microsoft ERP systems, the user may specify a specific method to be executed. After all values are correctly set, this command sends the values to the ERP system for evaluation. The function being executed is contained in the emProc.Name property. The properties of the procedure are then filled in with the results.

Syntax: [Value =] emProc.ExecuteMethod(Name)

Value (Boolean) Optional – contains a True / False based upon its success.

Name (String) the name of the method to execute

LogMode

This property can be set to log nothing, everything or just errors as it relates to executing business functions, stored procedures or macros. All messages are written to the RFgen error log.

Syntax: emProc.LogMode = Value

Alternate: Value = emProc.LogMode

Value (enLogMode) an enumeration containing 3 possible values:

LogNever – nothing is written to the log

LogAlways – all data, successes and failures are logged

LogFailure – only embedded procedure failures are logged

Example:

```
emProc.LogMode = LogAlways
```

Name

This property contains the function name to be executed.

Syntax: emProc.Name = Value

Alternate: Value = emProc.Name

Value (String) the name of the business function

Example:

```
emProc.Name = "BAPI_GOODSMVT_CREATE"
```

Param

This property controls all the business function's parameter values.

Syntax: emProc.Param(ParamId, [ColName], [RowNo]) = Value

Alternate: Value = emProc.Param(ParamId, [ColName], [RowNo])

Value (Variant) the result of the parameter's value or the value being placed in the parameter

ParamId (Variant) the name of the table usually referred to by name and in double quotes.

ColName (Variant) Optional – the name of the column within a table parameter

RowNo (Long) Optional – the row number within a table parameter

Examples:

```
emProc.Param("YEAR") = "1994"
```

- or -

```
Dim Value As Variant
```

```
Value = emProc.Param("RESULTS", "ID", 1)
```

ParamCount

For a given business function, this function will contain the number of parameters as they appear when downloaded. Each table will count as 1 parameter.

Syntax: Value = emProc.ParamCount

Value (Variant) contains the number of parameters for the specified business function.

Example:

```
Dim Value as Long  
Value = emProc.ParamCount
```

ParamEx

For SAP systems only, this property controls all the business function's parameter values and allows for nested parameters, like a table parameter that contains another table.

Syntax: emProc.ParamEx(ParamId, ColName, [RowNo]) = Value

Alternate: Value = emProc.ParamEx(ParamId, ColName, [RowNo])

Value (EmbeddedParam) the result of the parameter's value or the value being placed in the parameter

ParamId (Variant) the name of the table usually referred to by name and in double quotes.

ColName (Variant) the name of the column within a table parameter

RowNo (Long) Optional – the row number within a table parameter

Examples:

```
Dim sError As Variant  
sError = emProc.ParamEx("RETURN", "MESSAGE", 1)
```

In the case of nested parameters, specify the parameter ID and the column that contains the nested table and then use “dot” notation to extend the statement.

```
Dim Value As Variant  
Value = emProc.ParamEx("ParamId",  
"Col1").Param("SubParam", "SubCol")
```

This notation can be used indefinitely to set or obtain data from structures within other structures. The properties available after the ParamEx property are:

```
emParam.ParamEx("ParamID", "Col").ColumnCount  
emParam.ParamEx("ParamID", "Col").ColumnName
```

```
emParam.ParamEx("ParamID", "Col").Param  
emParam.ParamEx("ParamID", "Col").ParamEx  
emParam.ParamEx("ParamID", "Col").RowCount
```

ParamName

For a given function, this function will display the name of a parameter when referenced by its index value. If there are tables that contain parameters, use the property ColumnName.

Syntax: Value = emProc.ParamName([Index])

Value (String) contains the name of the parameter

Index (Long) Optional – the number of the parameter for which the name is retrieved. Left off a full list is returned.

Queue

This function returns a Boolean value depending on if the transaction could be queued. This is in place of executing the business function. The Transaction Manager will add this business function to the queue and execute it when the host is available and when it gets to the top of the queue. (Also see QueueSeqNo)

Syntax: Value = emProc.Queue

Value (Boolean) contains a True if the transaction was successfully queued.

Example:

```
Dim Value as Boolean  
Value = emProc.Queue
```

QueueName

Multiple queues are allowed in the transaction manager process (as configured in Configure \ System Options \ Service Options.) This property will return the name of the queue currently in use. It can also be used to set which queue processes the transaction.

Syntax: Value = emProc.QueueName

Alternate: emProc.QueueName = Value

Value (String) contains the name of the queue

Examples:

```
Dim sValue as String  
sValue = emProc.QueueName  
- Or -  
emProc.QueueName = "AltQueue"
```

QueueOffline

This property sets or returns whether the embedded procedure is set to queue if the host is offline.

Syntax: Value = emProc.QueueOffline

Alternate: emProc.QueueOffline = Value

Value (Boolean) contains a True or False depending on if the transaction can be or should be queued.

Examples:

```
Dim bValue as Boolean  
bValue = emProc.QueueOffline  
- or -  
emProc.QueueOffline = True
```

QueueSeqNo

This property returns the sequence number given to the queued function using the emProc.Queue method.

Syntax: Value = emProc.QueueSeqNo

Value (Long) contains the sequence number generated by the Transaction Manager when the function was queued.

Example:

```
Dim nValue As Long  
emProc.Queue  
nValue = emProc.QueueSeqNo
```

RowCount

For a given table, this function will contain the number of rows returned from the function given the passed parameters.

Syntax: Value = emProc.RowCount(ParamId)

Value (Variant) contains the number of rows in the specified table.

ParamId (Variant) the name of the table, usually referred to by name and in double quotes.

Example:

```
Dim nValue as Long  
nValue = emProc.RowCount ("RETURN")
```

DataSet Object

This object gives the user information about a stored recordset populated by other RFgen language extensions.

To use this object, start with a declaration similar to:

```
Dim oData as New DataSet
```

AddNew

This method creates an additional entry in the DataSet. As an example, when the TM.GetItems method populates a DataSet from a list of items in a queue, this method is used internally to grow the DataSet until the list is complete. If this object is being used for other purposes the AddNew method may be used to grow the DataSet as needed.

Syntax: oData.AddNew

Clear

This method clears the object of any previously loaded information

Syntax: oData.Clear

IsEOF

This method (End-Of-File) returns a True if the current pointer in the recordset is beyond the last entry or if there are no entries contained in the object.

Syntax: [OK =] oData.IsEOF

OK (Boolean) Optional – returns the True or False

MoveFirst

This selects (moves the pointer to) the first entry in the object's list of values.

Syntax: [OK =] oData.MoveFirst

OK (Boolean) Optional – returns True or False for the success of the command

MoveLast

This selects (moves the pointer to) the last entry in the object's list of values.

Syntax: [OK =] oData.MoveLast

OK (Boolean) Optional – returns True or False for the success of the command

MoveNext

This selects (moves the pointer to) the next entry in the object's list of values.

Syntax: [OK =] oData.MoveNext

OK (Boolean) Optional – returns True or False for the success of the command

MovePrevious

This selects (moves the pointer to) the previous entry in the object's list of values.

Syntax: [OK =] oData.MovePrevious

OK (Boolean) Optional – returns True or False for the success of the command

MoveTo

This selects (moves the pointer to) a specific entry in the recordset.

Syntax: [OK =] oData.MoveTo(Row)

OK (Boolean) Optional – returns True or False for the commands success

Row (Long) the row number to move the object's pointer

Param

This property returns the values stored in the named columns of the DataSet.

Syntax: Param(ParamID, [RowNo]) = Value

Alternate: Value = Param(ParamID, [RowNo])

Value (Variant) the stored value given a parameter ID

ParamID (Variant) then name of a column in the DataSet

RowNo (Variant) Optional – if the DataSet is a table containing multiple rows of data, this parameter will move the data pointer to the specified row before retrieving the data.

Examples:

```
Dim vData As Variant  
oData.Param("ErrMsg") = "Wrong value."  
vData = oData.Param("DeviceNo")  
vData = oData.Param("FormId")  
vData = oData.Param("IPAddress")  
vData = oData.Param("ExecDate")
```

ParamCount

For a given row that is a table, this function will contain the number of columns returned.

Syntax: Value = oData.ParamCount

Value (Variant) contains the count of the columns in the DataSet

Example:

For example, a table named Inventory has 2 fields, Part and Quantity, and has 200 records.

```
Dim iValue As Integer  
iValue = oData.ParamCount      ' will return 2.
```

ParamName

This property is a collection of parameter IDs. In the case of the TM.GetItems use of the DataSet object the parameter IDs are:

- | | | | |
|---------------|------------|----------------|---------------|
| (1) QueueName | (5) Source | (9) UserId | (13) ExecTime |
| (2) SeqNo | (6) Name | (10) DeviceNo | (14) Status |
| (3) TranDate | (7) Record | (11) IPAddress | (15) ErrMsg |
| (4) TranTime | (8) FormId | (12) ExecDate | |

If the DataSet object is used for another purpose the parameter names would be different. See the ParamCount property to get a count of the parameter IDs.

Syntax: Value = oData.ParamName([Index])

Value (String) the name of the parameter at the specified index

Index (Long) Optional – the index of the parameter to be evaluated

Example:

When using the TM.GetItems method, to retrieve the name of the macro for the first row in the DataSet:

```
Dim oData As New DataSet  
Dim sMacroName As String  
TM.GetItems(tmCompleted, Date, App.User, oData)  
oData.MoveFirst  
sMacroName = oData.Param(6)  
sMacroName = oData.Param("Name")
```

RowCount

For a given record set, this function will get the number of rows returned in the object.

Syntax: Value = oData.RowCount
Value (Variant) the number of rows in the DataSet

Example:

When using the TM.GetItems method, the number of queue entries returned would be returned.

```
Dim oData As New DataSet
Dim iValue As Integer
TM.GetItems(tmCompleted, Date, App.User, oData)
iValue = oData.RowCount
```

SchemalId

This property returns the Task ID of a Vocollect task that is currently loaded in the DataSet.

Syntax: Value = oData.SchemalId
Value (String) the task Id of a Vocollect task

```
Dim sValue As String
sValue = oData.SchemaId
```

Dynamic Array Extensions

RFgen supports a special kind of string variable called a 'dynamic array'. The word 'dynamic' means that this type of variable is designed to allow stored data to grow or shrink in size. In addition, the stored data is easily accessible and changeable. Dynamic arrays allow volumes of data to be organized, stored, referenced, counted, and manipulated by use of just 1 single string variable. Access to the data is instantaneous via the RFgen Dynamic Array VBA extensions that follow.

Dynamic Array Structure

Dynamic Arrays are special string variables that use low end ANSI characters (1, 2, and 3) as 'delimiters' to separate data into Fields using Chr(1), Subfields using Chr(2), and Sub-subfields using Chr(3). These delimiters were chosen because data being entered or scanned are never expected to contain them. To use Dynamic Arrays in your RFgen programming, you do not need to reference these characters, RFgen simply uses them internally to manage stored data.

Why Use RFgen Dynamic Arrays?

1. There are no limits to the amount of data that may be stored

2. The alternative is to declare and use subscripted variables, the number of which may be insufficient
3. RFgen has been optimized to access and manipulate this type of data instantaneously.

For illustrative purposes, the ‘Field’ and ‘Subfield’ delimiters are represented in the examples below as follows:

Dynamic Array Field delimiter (Chr(1)) as a ‘&’
Dynamic Array Subfield delimiter (Chr(2)) as a ‘#’
Dynamic Array Sub-Subfield delimiter (Chr(3)) as a ‘@’

DCount

The DCount function (Delimiter Count) may be used to determine the number of occurrences of a specified delimiter within a string, or a string variable. The command will also add 1 to the count to actually represent the number of values or fields stored within the string variable. This makes the DCount command really return the number of available fields in the record so that extra logic is not necessary when determining the length of a loop structure used to manipulate the contents of the string.

Syntax: Nbr = DCount(VAR, DELIM)

Nbr (Variant) is a number of delimiters found in the variable plus 1.
VAR (Variant) is the string value to be evaluated
DELIM (Variant) is a specified delimiter character, or multi-character string to count; e.g., to count dynamic array delimiters use Chr(1), Chr(2), Chr(3).

Examples:

```
Dim Nbr As Long
Dim sNames As String
sNames = "John&Mike&Albert"
Nbr = DCount(sNames,Chr(1)) ' Nbr = 3

sNames = "&Mike&Albert"
Nbr = DCount(sNames,Chr(1)) ' Nbr = 3

sNames = "&Mike&"
Nbr = DCount(sNames,Chr(1)) ' Nbr = 3
```

Note that +1 has been added to the actual count of Chr(1) (represented here with a character '&') count of 2. Also note that just because the fields are empty, they are valid slots where data may be inserted.

Del

The Delete function deletes a field, subfield, or sub-subfield from a dynamic array.

Syntax: REC = Del(REC, FLD, [SUB], [SSUB])

REC (Variant) is the dynamic array variable name

FLD (Variant) is the array field number to search

SUB (Variant) Optional – is the array subfield number

SSUB (Variant) Optional – is the array sub-subfield number

Examples:

```
Dim sNames As String
sNames = "John&Mike&Albert"      ' Three records of names
sNames = Del(sNames, 3)            'sNames becomes
"John&Mike"

sNames = "John&Mike&Albert"      ' Three records of names
sNames = Del(sNames, 2)            'sNames becomes
"John&Albert"
```

This next example shows each record with 3 values; Name, Age and Language.

```
sNames =
"John#31#English&Mike#34#Spanish&Albert#40#English"
sNames = Del(sNames, 1)
'sNames becomes
"Mike#34#Spanish&Albert#40#English"

sNames =
"John#31#English&Mike#34#Spanish&Albert#40#English"
sNames = Del(sNames, 2)
'sNames becomes
"John#31#English&Albert#40#English"
```

You can also delete portions of a record.

```
sNames =
"John#31#English&Mike#34#Spanish&Albert#40#English"
sNames = Del(sNames, 2, 1)
'sNames becomes
"John#31#English&34#Spanish&Albert#40#English"

sNames =
"John#31#English&Mike#34#Spanish&Albert#40#English"
sNames = Del(sNames, 2, 2)
```

```

'sNames is now
"John#31#English&Mike#Spanish&Albert#40#English"

sNames =
"John#31#English&Mike#34#Spanish&Albert#40#English"
sNames = Del(sNames, 2, 3)
'sNames becomes
"John#31#English&Mike#34&Albert#40#English"

```

Using the Sub-subfield concept, maybe the language has sub categories like Canadian English and United Kingdom English.

```

sNames = "John#31#Canadian English@UnitedKingdom
English&Mike...
sNames = Del(sNames, 1, 3, 1)
'sNames becomes "John#31# UnitedKingdom
English&Mike...

```

Here everything between the Chr(1) delimiters is considered the first field. The 2 versions of English are a part of the third sub-field. So reading the DEL statement: “Using the first field, look at the third sub-field (all the languages) and delete the first sub-subfield.

A Dynamic Array is like a database stored as a single string. Everything between the Chr(1) delimiters are fields referenced by the first number. If there are any Chr(2) delimiters, each piece makes up the complete record. If there are any Chr(3) delimiters, they make up the complete subfield.

Ext

The Extract function returns a field, subfield, or sub-subfield value from a dynamic array.

Syntax: VAL = Ext(REC, FLD, [SUB], [SSUB])

VAL (Variant) is the string value extracted from the array.

REC (Variant) is the dynamic array variable name

FLD (Variant) is the array field number to search

SUB (Variant) Optional – is the array subfield number

SSUB (Variant) Optional – is the array sub-subfield number

Examples:

```

Dim sNames As String
Dim sValue As String
sNames = "John&Mike&Albert" ' Three records of names
sValue = Ext(sNames, 2)      'sValue becomes "Mike"

```

```
sNames = "John&Mike&Albert" ' Three records of names
sValue = Ext(sNames, 3)      'sValue becomes "Albert"
```

This next example shows each record with 3 values; Name, Age and Language.

```
sNames =
"John#31#English&Mike#34#Spanish&Albert#40#English"
sValue = Ext(sNames, 2, 3) 'sValue becomes "Spanish"
```

Here Mike#34#Spanish is the second record and the third subfield value was retrieved.

Using the Sub-subfield concept, maybe the language has sub categories like Canadian English and United Kingdom English.

```
sNames = "John#31#Canadian English@UnitedKingdom
English&Mike...
sValue = Ext(sNames, 1, 3, 1) 'sValue becomes
"Canadian English"
```

There are times when the Extract function can be very helpful. For example, if you create a comma-delimited list of items and must parse through it, writing code to manipulate the string and keep track of the pointer can be extensive. Instead use a statement like:
sList = Replace(sList, ", , Chr(1))

This replaces the comma with the Chr(1) character. Then in the loop you only need to EXT(sList, i) to get each entry out. Use the DCount command to get the length of the list.

FixLeft

This function pads a raw string to the left with a specified character until the specified length is reached. If the optional 3rd parameter is not specified, spaces are used. (This is not specifically a Dynamic Array command but is commonly used to manipulate them.)

Syntax: Value = FixLeft(StringData, Chars, [PadChar])
Value (Variant) is the padded string value
StringData (Variant) is the string containing the raw data
Chars (Variant) is the total size of the resulting string
PadChar (Variant) Optional – is the pad character to use. If none is specified, spaces are used

Example:

```
Rsp=FixLeft(Rsp, 8, "A")
```

If Rsp was “123” then Rsp becomes “123AAAAA” because it puts the character ‘A’ as many times as necessary after the Rsp value until the length of Rsp is 8, left justifying the StringData value.

FixRight

This function pads a raw string to the right with a specified character until the specified length is reached. If the optional 3rd parameter is not specified, spaces are used. (This is not specifically a Dynamic Array command but is commonly used to manipulate them.)

Syntax: Value = FixRight(StringData, Chars, [PadChar])

Value (Variant) is the padded string value

StringData (Variant) is the string containing the raw data

Chars (Variant) is the total size of the resulting string

PadChar (Variant) Optional – is the pad character to use. If none is specified, spaces are used

Example:

```
Rsp=FixRight(Rsp, 8, "A")
```

If Rsp was “123” then Rsp becomes “AAAAA123” because it puts the character ‘A’ as many times as necessary before the Rsp value until the length of Rsp is 8, left justifying the StringData value.

Ins

The Insert function inserts a value into a field, subfield, or sub-subfield of a dynamic array.

Syntax: REC = Ins(REC, FLD, [SUB], [SSUB], STR)

REC (Variant) is the dynamic array string variable name

FLD (Variant) is the array field number to search

SUB (Variant) Optional – is the array subfield number

SSUB (Variant) Optional – is the array sub-subfield number

STR (Variant) is data or variable name, which will be inserted at the dynamic array location (FLD, SUB, SSUB).

Examples:

```
Dim sNames As String  
sNames = Ins(sNames, 2, 0, 0, "John")
```

Inserts string 'John' into dynamic array sNames. If sNames was originally null, then after the insert, field2 contains 'John'; i.e., sNames = '&John'. Here, field1 is null.

Note: Suppressing the zeros [i.e., sNames = INS(sNames, 2, "John")] gives the same result.

```
sNames = Ins(sNames, 2, "Mike")
```

If data already exists for field2 (John), then an additional insert would move field2 data to field3; i.e., sNames = '&Mike&John'. Here, field1 is null, field2 = 'Mike', and field3 = 'John'.

Using the subfields the following string can be created one element at a time using 9 Insert statements:

```
"John#31#English&Mike#34#Spanish&Albert#40#English"
```

```
sNames = Ins(sNames, 1, "John")
sNames = Ins(sNames, 1, 2, "31") ' First record,
second field
sNames = Ins(sNames, 1, 3, "English") ' First record,
third field

sNames = Ins(sNames, 2, "Mike")
sNames = Ins(sNames, 2, 2, "34") ' Second record,
second field
sNames = Ins(sNames, 2, 3, "Spanish") 'Second record,
third field

sNames = Ins(sNames, 3, "Albert")
sNames = Ins(sNames, 3, 2, "40") ' Third record,
second field
sNames = Ins(sNames, 3, 3, "English") 'Third record,
third field
```

If you used insert statements and the sub-subfield values you can get this:

```
"John#31#Canadian English@UnitedKingdom English&Mike"
```

by

```
sNames = Ins(sNames, 1, "John")
sNames = Ins(sNames, 1, 2, "31") ' First record,
second field
```

```
sNames = Ins(sNames, 1, 3, 1, "Canadian English") ' First record, third  
field, first sub-field
```

```
sNames = Ins(sNames, 1, 3, 2, "UnitedKingdom  
English") ' First record, third field, second sub-  
field  
sNames = Ins(sNames, 2, "Mike")
```

LField

The LField function searches a string from the left to extract a sub-string by using a specified delimiter character.

Syntax: VAL= LField(StringData, Delimiter, StartField, [NumberFields])
VAL (Variant) is the resulting sub-string value located in StringData
StringData (Variant) is a string or string variable to be searched
Delimiter (Variant) is a specified delimiter character
StartField (Variant) Optional – is the Delimiter to start from when searching the StringData
NumberFields (Variant) specifies the number of sub-fields to retrieve.

Examples:

```
If VAR="111|222|333", then:  
VAL= LField(VAR, "|", 1) returns VAL='111'  
VAL= LField(VAR, "|", 3) returns VAL='333'
```

Note: If StartField = 1, then the LField function will return a sub-string which starts at the beginning of VAR, up to the first occurrence of Delimiter.

Locate

The Locate function may be used to find the index of a field, subfield, or sub-subfield within a dynamic array.

Syntax: Nbr = Locate(VAL, StringData, [FLD], [SUB], [Option])
Nbr (Variant) is an integer that indicates the location of VAL, or 0 (zero) if VAL is not located.
VAL (Variant) is the string value to be located
StringData (Variant) is the dynamic array variable that will be searched
FLD (Variant) Optional – is the array field number to search
SUB (Variant) Optional – is the array subfield number to search
Option (Variant) Optional – the column within the row to search

Examples:

The ‘&’ character is used like the Chr(1) delimiter to separate the different rows. The ‘#’ character is used like the Chr(2) delimiter to separate different values within a record, In this case, the name and the age and the language. The ‘@’ symbol is used like Chr(3) to delimit between a sub-field. In this case, the difference between 2 languages.

```
Dim Nbr As Integer
Dim sNames As String
sNames = "John#31#Canadian English@UnitedKingdom
English&Mike"

Nbr = Locate("Mike", sNames)  ' Nbr = 2
Nbr = Locate("John", sNames)  ' Nbr = 0 since there
is depth to record 1
Nbr = Locate("John", sNames, 1) ' Nbr = 1
Nbr = Locate("31", sNames, 1)   ' Nbr = 2 since 1st
record was specified
Nbr = Locate("UnitedKingdom English", sNames, 1, 3)

Nbr = 2 since 1st record and 3d subfield was specified
```

If you do not know the row number where your data is then you can specify 0 in place of the Field and Sub values to make RFgen search the entire array.

```
Nbr = Locate("31", sNames, 0, 0, "V2")  ' Nbr = 1
```

Assuming you know the layout of the array (Name, Age, Language) then you can tell RFgen to locate the “31” anywhere in the array and specifically look at column 2 for the match. This will return which row is the first with the data.

If your columns are Chr(3)-delimited and the rows are still Chr(1)-delimited then use an “S” to find the row:

```
sNames = "Names" & "John@Mike@Steve@Rob" &
"Addresses"
```

```
Nbr = Locate("Steve", sNames, 0, 0, "S3")  ' Nbr = 2
```

You still need to know the format of the array so, in this case, Steve was found in the known third position of the unknown second row.

LocateAdd

The LocateAdd function will add a value to the dynamic array only if the value being added does not already exist. If the value does exist but has

associated subfields the new value will still be added to the end of the list. This function does not interact with SUB and SSUB fields and therefore treats a whole record with subfields as 1 string during the compare.

Syntax: LocateAdd(VAL, Info, [DLM])

VAL (Variant) is the value to be located or added to the Dynamic Array

Info (Variant) is the dynamic array variable that will be searched

DLM (Variant) Optional – is the delimiter to use when searching

Examples:

```
Dim sNames As String
```

```
sNames = "John#31#Canadian English@UnitedKingdom
```

```
English&Mike"
```

```
LocateAdd("Mike", sNames) ' Since Mike already exists  
nothing is added
```

```
LocateAdd("John", sNames) ' Since John is associated  
with subfields in the first record, it will again be  
added to the end of the string:
```

```
"John#31#Canadian English@UnitedKingdom  
English&Mike&John"
```

```
LocateAdd("Albert", sNames, "X")
```

This adds “Albert” to the end of the list and uses the letter ‘X’ to separate the last entry and Albert.

LocateDel

The LocateDel function will remove a value from the dynamic array and report the location in the list where it was. If the value does not exist, 0 is returned. This function does not interact with SUB and SSUB fields and therefore treats a whole record with subfields as 1 string during the compare.

Syntax: Nbr = LocateDel(VAL, Info, [DLM])

Nbr (Variant) the position in the dynamic array where the VAL was found and deleted

VAL (Variant) the value to be located and deleted from the Dynamic Array

Info (Variant) is the dynamic array variable that will be searched

DLM (Variant) Optional – is the delimiter to use when searching

Examples:

```
Dim Nbr As Integer
```

```

Dim sNames As String
sNames = "John#31#Canadian English@UnitedKingdom
English&Mike"

Nbr = LocateDel("Mike", sNames) ' Nbr = 2, the second
record
Nbr = LocateDel("31", sNames, Chr(2))

```

Nbr = 2, since the age is a subfield specifying the Chr(2) delimiter finds "31" in the second position of the first record.

Rep

The Replace function replaces a field, subfield, or sub-subfield in a dynamic array.

Syntax: StringData = Rep(StringData, FLD, [SUB], [SSUB], STR)
StringData (Variant) is the dynamic array string variable name
FLD (Variant) is the array field number to search
SUB (Variant) Optional – is the array subfield number
SSUB (Variant) Optional – is the array sub-subfield number
STR (Variant) is data or variable name, which will be inserted at the dynamic array location (FLD, SUB, SSUB).

Examples:

```

Given : "John#31#Canadian English@UnitedKingdom
English&Mike"
StringData = Rep(StringData, 2, "Albert") ' removes
Mike and returns:
"John#31#Canadian English@UnitedKingdom
English&Albert"
StringData = Rep(StringData, 1, "Fred") ' removes
whole first record and returns: "Fred&Albert"

```

Using subfields, given:

```

"John#31#Canadian English@UnitedKingdom
English&Mike"
StringData = Rep(StringData, 1, 2, "80") ' using
the first record and the second subfield, 31 is
replaced with 80 giving:
"John#80#Canadian English@UnitedKingdom English&Mike"

StringData = Rep(StringData, 1, 3, 2, "US English")
' using the first record, the third subfield and the
second sub subfield, United Kingdom English is
replaces with US English giving:
"John#80#Canadian English@US English&Mike"

```

RField

The RField function searches a string from the right to extract a sub-string by using a specified delimiter character.

Syntax: VAL= RField(StringData, Delimiter, StartField, [NumberFields])
VAL (Variant) is the resulting sub-string value located in StringData
StringData (Variant) is a string or string variable to be searched
Delimiter (Variant) is a specified delimiter character
StartField (Variant) is the Delimiter to start from when searching the StringData
NumberFields (Variant) Optional – specifies the number of sub-fields to retrieve.

Examples:

```
If VAR="111|222|333", then:  
VAL= RField(VAR, "|", 1) returns VAL='333'  
VAL= RField(VAR, "|", 3) returns VAL='111'
```

Note: If StartField = 1, then the RField function will return a sub-string which starts at the end of VAR, up to the first occurrence of Delimiter.

Socket Object

The Socket object is provided to the programmer for creating and managing their own WinSock connection from within RFgen. This object is only a pass-through to the standard Windows WinSock control so additional documentation about how this works can be easily found on the Internet. There are 3 sections for this object, properties, methods and events.

This object is declared in a similar manner to:

```
Dim WithEvents oSocket As Socket
```

The properties for the Socket object are as follows:

BytesReceived

This function returns the number of bytes currently in the receive buffer. This is a read-only property and is unavailable at design time. The value returned is a long integer.

Syntax: Value = oSocket.BytesReceived
Value (Long) the number of bytes

Example:

```
Dim WithEvents oSocket As Socket  
Dim nBytes As Long  
nBytes = oSocket.BytesReceived
```

LocalHostName

The LocalHostName function returns the name of the local host system. This is read-only property and is unavailable at the design time. The value returned is a string.

Syntax: Value = oSocket.LocalHostName
Value (String) the name of the local host

Example:

```
Dim WithEvents oSocket As Socket  
Dim sName As String  
sName = oSocket.LocalHostName
```

LocalIP

The LocalIP function returns the local host system IP address in the form of a string, such as 11.0.0.127. This property is read-only and is unavailable at design time.

Syntax: Value = oSocket.LocalIP
Value (String) the IP address of the local host

Example:

```
Dim WithEvents oSocket As Socket  
Dim sIP As String  
sIP = oSocket.LocalIP
```

Protocol

This property can get or set the protocol used to either TCP or UDP.

Syntax: lValue = oSocket.Protocol

Alternate: oSocket.Protocol = oValue

oValue (enWinsockProtocols) the numeric value representing TCP or UDP. TCP = 0, UDP = 1. There are built in constants to represent these value as well. They are sckTCPPProtocol and sckUDPPProtocol.

lValue (Long) the numeric value representing TCP or UDP. TCP = 0, UDP = 1.

Examples:

```
Dim WithEvents oSocket As Socket
```

```
Dim wProtocols As enWinsockProtocols  
oSocket.Protocol = sckTCPProtocol  
wProtocols = oSocket.Protocol
```

RemoteHost

The RemoteHost property returns or sets the remote host. This can be both read from and written to and is available both in design time and runtime. The value returned is a string and can be specified either as an IP address or as a DNS name.

Syntax: Value = oSocket.RemoteHost

Alternate: oSocket.RemoteHost = Value

Value (String) the name of the host system the client will be connecting to.

Example:

```
Dim WithEvents oSocket As Socket  
Dim sName As String  
sName = oSocket.RemoteHost
```

RemoteHostIP

This property returns or sets the remote host IP address.

Syntax: Value = oSocket.RemoteHostIP

Alternate: oSocket.RemoteHostIP = Value

Value (String) the IP address of the host system the client will be connecting to.

Example:

```
Dim WithEvents oSocket As Socket  
Dim sIP As String  
sIP = oSocket.RemoteHostIP
```

RemotePort

This property returns or sets the remote port number.

Syntax: Value = oSocket.RemotePort

Alternate: oSocket.RemotePort = Value

Value (Long) the port number used to access the host system the client will be connecting to.

Examples:

```
Dim WithEvents oSocket As Socket  
Dim nPort As Long  
oSocket.RemotePort = 11908
```

```
nPort = oSocket.RemotePort
```

State

This property returns the state of the control as expressed by an enumerated list. This is read-only property and is unavailable at design time.

Syntax: Value = oSocket.State

Value (Long) the number assigned to the different socket states

- 0 = sckClosed
- 1 = sckOpen
- 2 = sckListening
- 3 = sckConnectionPending
- 4 = sckResolvingHost
- 5 = sckHostResolved
- 6 = sckConnecting
- 7 = sckConnected
- 8 = sckClosing
- 9 = sckError

Example:

```
Dim WithEvents oSocket As Socket
Private Sub oSocket_OnConnect()
    On Error Resume Next
    '
    If oSocket.State <> sckConnected Then
        App MsgBox ("Connect failed.")
End Sub
```

The methods for the Socket object are as follows:

sktClose

This method terminates a TCP connection from either the client or server applications. If the object is declared using the WithEvents option, then an OnClose event is available in the script environment but will not fire with the use of this method but only if the connection is closed by the server.

Syntax: oSocket.sktClose

Example:

```
Dim WithEvents oSocket As Socket
If oSocket.State = sckConnected Then oSocket.sktClose
```

sktConnect

This method requests a connection to a remote computer. If the object is declared using the WithEvents option, then an OnConnect event is available in the script environment.

Syntax: oSocket.sktConnect([RemoteHost], [RemotePort])

RemoteHost (Variant) Optional – allows a connection to a host that is not specified in the RemoteHost property

RemotePort (Variant) Optional – allows a connection to a host that is not specified in the RemotePort property

Example:

```
Dim WithEvents oSocket As Socket
oSocket.Protocol = sckTCPProtocol
oSocket.RemoteHost = "127.0.0.1"
oSocket.RemotePort = 21097
oSocket.sktConnect

Private Sub oSocket_OnConnect()
    On Error Resume Next
    '
    Do
        Loop Until oSocket.State <> sckConnecting
        If oSocket.State <> sckConnected Then App MsgBox
        "Connect failed."
    End Sub
```

sktGetData

This method retrieves the current block of data from the buffer and then stores it in a variable of the variant type. If the object is declared using the WithEvents option, then an OnDataArrival event is available in the script environment.

Syntax: oSocket.sktGetData vData, [Type], [MaxLen]

vData (Variant) Where retrieved data will be stored after the method returns successfully. If there is not enough data available for requested type, vData will be set to Empty.

Type (Variant) Optional – specifies the type of data being retrieved

Byte = vbByte

Integer = vbInteger

Long = vbLong

Single = vbSingle

Double = vbDouble

Currency = vbCurrency

Date = vbDate

	Boolean	= vbBoolean
	SCODE	= vbError
	String	= vbString
	Byte Array	= vbArray + vbByte
MaxLen	(Variant)	Optional – Specifies the desired size when receiving a byte array or a string. If this parameter is missing for byte array or string, all available data will be retrieved. If provided for data types other than byte array and string, this parameter is ignored.

Example:

```
Dim WithEvents oSocket As Socket

Private Sub oSocket_OnDataArrival(ByVal bytesTotal As Long)
    On Error Resume Next
    '
    Dim sData As String
    oSocket.sktGetData(sData, vbString, bytesTotal)
End Sub
```

sktPeekData

This method operates in a fashion similar to the sktGetData method. However, it does not remove data from the input queue. It is used to see what data is coming before using the sktGetData method to retrieve and delete the data from the receive buffer. This method works only for TCP connections.

Syntax:	oSocket.sktPeekData vData, [Type], [MaxLen]	
vData	(Variant)	Where retrieved data will be stored after the method returns successfully. If there is not enough data available for requested type, vData will be set to Empty.
Type	(Variant)	Optional – specifies the type of data being retrieved
	Byte	= vbByte
	Integer	= vbInteger
	Long	= vbLong
	Single	= vbSingle
	Double	= vbDouble
	Currency	= vbCurrency
	Date	= vbDate
	Boolean	= vbBoolean
	SCODE	= vbError
	String	= vbString
	Byte Array	= vbArray + vbByte

MaxLen (Variant) Optional – Specifies the desired size when receiving a byte array or a string. If this parameter is missing for byte array or string, all available data will be retrieved. If provided for data types other than byte array and string, this parameter is ignored.

Example:

```
Dim WithEvents oSocket As Socket

Private Sub oSocket_OnDataArrival(ByVal bytesTotal As Long)
    On Error Resume Next
    '
    Dim sData As String
    oSocket.sktPeekData(sData, vbString, bytesTotal)
End Sub
```

sktSendData

This method dispatches data to the remote computer. When a UNICODE string is passed in, it is converted to an ANSI string before being sent out on the network.

Syntax: oSocket.sktSendData(Data)
Data (Variant) Data to be sent. For binary data, byte array should be used.

Example:

```
Dim WithEvents oSocket As Socket
oSocket.sktSendData("Hello world.")
```

The events for the Socket object are as follows:

OnClose

Occurs when the connection has been closed by the remote host. This does not occur when the sktClose method has been called.

Example:

```
Dim WithEvents oSocket As Socket
Private Sub oSocket_OnClose()
    On Error Resume Next
End Sub
```

OnConnect

Occurs when a connection is successfully established

Example:

```
Dim WithEvents oSocket As Socket
Private Sub oSocket_OnConnect()
    On Error Resume Next
End Sub
```

OnDataArrival

Occurs when data arrives and is used to process incoming data.

Example:

```
Dim WithEvents oSocket As Socket
Private Sub oSocket_OnDataArrival(ByVal bytesTotal As Long)
    On Error Resume Next
End Sub
```

OnError

Occurs when there is an error, such as a PC not existing.

Example:

```
Dim WithEvents oSocket As Socket
Private Sub oSocket_OnError(ByVal Number As Long,
                           ByVal Description As String)
    On Error Resume Next
End Sub
```

OnSendComplete

Occurs when the data has been sent

Example:

```
Dim WithEvents oSocket As Socket
Private Sub oSocket_OnSendComplete()
    On Error Resume Next
End Sub
```

OnSendProgress

Occurs when the data is being sent

Example:

```
Dim WithEvents oSocket As Socket
Private Sub oSocket_OnSendProgress(ByVal bytesSent As Long,
                                   ByVal bytesRemaining As Long)
    On Error Resume Next
End Sub
```

Web Object

The Web object is provided to the programmer for managing the Web data connector configured as one of the data connections.

This object is declared in a similar manner to:

```
Dim oWeb As Web
```

The properties for the Web object are as follows:

ConnectTimeout

The ConnectTimeout property returns or sets the timeout to connect for the HTTP request. When the language extension is created it will initialize this value to what is configured in the data connection configuration.

Syntax: Value = oWeb.ConnectTimeout

Alternate: oWeb.ConnectTimeout = Value

Value (Long) the number of milliseconds the system will wait for a connection

Examples:

```
Dim oWeb As Web
Dim nTimeout As Long
nTimeout = oWeb.ConnectTimeout
oWeb.ConnectTimeout = 10000
```

Data

The Data property returns or sets the data portion of the HTTP request. This is likely to be the collected data requiring validation or the return values that come from the web server.

Syntax: oWeb.Data = Value

Alternate: Value = oWeb.Data

Value (String) the data to be sent as part of the request or returned from the server

Example:

```
Dim oWeb As Web
Dim sPart As String
Dim sQty As String
sPart = "100620"
sQty = "100"
oWeb.Data = sPart & " | " & sQty
```

DataSource

The DataSource property returns or sets an index of the data connection that should be used. If a DataSource is not specified the language extension will use the first data connection that is configured as a WEB connection.

Syntax: Value = oWeb.DataSource

Alternate: oWeb.DataSource = Value

Value (Variant) the number or name of the RFgen data connection object.

Examples:

```
Dim oWeb As Web
Dim vConnID As Variant
oWeb.DataSource = 2
oWeb.DataSource = "MyWebServer"
vConnID = oWeb.DataSource
```

HeaderValue

The HeaderValue property returns or sets what the HTTP or HTTPS header values are.

Syntax: oWeb.DataSource(Index) = Value

Alternate: Value = oWeb.DataSource(Index)

Value (Boolean) contains a True for False for the success of the HeaderValue assignment.

Index (String) the property name for the HTTP header

Example:

```
Dim oWeb As Web
oWEB.HeaderValue("Content-Type") = "text/xml;
charset=utf-8"
oWEB.HeaderValue("SOAPAction") =
"""/xml.namespaces.xerox. com/im
/xip/services/xcllmswebservice/wsdl/getDistributionRe
furbInfo"""
```

ReceiveTimeout

The ReceiveTimeout property returns or sets the timeout to receive a reply to the HTTP request. If a reply has not been completely received before the timeout then the request will be terminated. When this language extension is created it will initialize this value to what is configured in the data connection configuration.

Syntax: Value = oWeb.ReceiveTimeout

Alternate: oWeb.ReceiveTimeout = Value

Value (Long) the number of milliseconds the system will wait for a connection

Examples:

```
Dim oWeb As Web
Dim nTimeout As Long
nTimeout = oWeb.ReceiveTimeout
oWeb.ReceiveTimeout = 20000
```

Reply

The Reply property returns the data portion of the HTTP response. This is where the data will be held when the request returns. Keeping the Data and Reply properties separate allows Execute to be called multiple times without requiring the Data property to be reset.

Syntax: Value = oWeb.Reply

Value (String) the data being returned to the client from the server

Example:

```
Dim oWeb As Web
Dim sValue As String
sValue = oWeb.Reply
```

Request

The Request property returns or sets the path to the ASP page (if executed against a Windows server) that is then appended to the URL that is used for connecting.

Syntax: oWeb.Request = Value

Alternate: Value = oWeb.Request

Value (String) part of the URL that is the path to the ASP page

Example:

```
Dim oWeb As Web
Dim sRequest As String
oWeb.Request =
"/data/DoWork.asp?num=100620?MySQLTable"
sRequest = oWeb.Request
```

SendTimeout

The SendTimeout property returns or sets the timeout value to send the HTTP request. If the send is not complete before the timeout then the request will be terminated. When the language extension is created it

will initialize this value to what is configured in the data connection configuration.

Syntax: Value = oWeb.SendTimeout

Alternate: oWeb.SendTimeout = Value

Value (Long) the number of milliseconds the system will wait for submitting a HTTP request

Example:

```
Dim oWeb As Web
Dim nTimeout As Long
nTimeout = oWeb.SendTimeout
oWeb.SendTimeout = 30000
```

The methods for the Web object are as follows:

Execute

The Execute method will execute the configured oWeb object. vData is optional and is equivalent to the XML data in a SOAP request.

Syntax: Value = oWeb.Execute([Data])

Value (Boolean) is True or False depending on the success of the submission to the web server.

Data (Variant) Optional – an object that could contain the individual properties.

Example:

```
Dim oWeb As Web
If oWeb.Execute Then
    RFPrompt("txtReply").Text = oWeb.Reply
End If
```

Login

The Login method is for future use.

Logout

The Logout method is for future use.

Stored Procedure Extensions

These *obsolete* commands relate specifically to stored procedures and have been replaced with the Embedded Procedure structure.

CallAction (not used with Sybase)

This function will call the stored procedure as specified, and will return the output from the procedure.

Syntax: `RetVal = SP.CallAction(Name, Params)`

`RetVal` (Variant) is a return value number from the stored procedure. Typically '0' (zero) specifies that no errors occurred during processing.

`Name` (String) is the name of the stored procedure to call.

`Params` (Variant) passing parameters as required by the stored procedure. For stored procedures that return values as passing parameters, any output parameters must be declared as Variant and be set to empty.

Example:

```
Dim vRetVal As Variant  
vRetVal = SP.CallAction ("SPname", Rsp)
```

CallProc (must be used with Sybase)

This function will call the stored procedure as specified, and will return the output from the procedure. An unlimited number of passing parameters Pn (e.g., P1, P2, etc.) may be specified.

Syntax: `ErrNo = SP.CallProc(Name, Options, Columns, Rows, Params)`

`ErrNo` (Variant) is a return value from the stored procedure. Typically '0' (zero) specifies that no errors occurred during processing.

`Name` (String) is the name of the stored procedure to call.

`Options` (String) are the options for the stored procedure to use.

Allowable options are:

A = Action - no rows returned

S = Select - rows are returned

O = Open - an alternative required for some databases (i.e., for Sybase: 'A' and 'S' are not useful, use only 'O').

Note: each type may optionally be suffixed by the letters 'Y' or 'N' (for Yes or No) to specify additional possible database requirements, as follows:

1st letter (Y or N): a 'Return' value is specified in the stored procedure.
1st letter defaults (if not specified) are 'Y' for type 'A', 'N' for type 'S', and 'N' for type 'O',

2nd letter (Y or N): RFgen is to declare the passing parameters as 'input' or 'output'. The 2nd letter default (if not specified) is 'Y' for all 3 type ('A', 'S', and 'O'). A tip: consider 'N' for Oracle databases).

Examples:

```
Type='A'  
Type='S'  
Type='ONN'
```

Columns (String) is a string representation of the columns contained in the associated Records item (below).

Rows (String) is a string representation of any results returned by the stored procedure.

Params (Variant) passing parameters as required by the stored procedure. For stored procedures that return values as passing parameters, any output parameters must be declared as Variant and be set to empty.

Example:

```
Dim sColumns As String  
Dim sRows As String  
Dim vPn As Variant  
Dim vErrNo As Variant  
vPn = Rsp ' User must provide value  
vErrNo= SP.CallProc("byroyalty", "SNY", sColumns,  
sRows, vPn)
```

The Records variable now contains a resultset from which values can be extracted, using DB.Extract.

CallSelect (not used with Sybase)

This function will call the stored procedure as specified, and will return the output from the procedure.

Syntax: ErrNo = SP.CallSelect(Sql, Columns, Rows, [Params])

ErrNo (Variant) is a return value from the stored procedure. Typically '0' (zero) specifies that no errors occurred during processing.

Sql (String) is the name of the stored procedure to call as a string or string variable, along with any passing parameters for the procedure (i.e., "GetCust '1001'").

Columns (String) is a string representation of the columns contained in the associated Records item (below).

Rows (String) is a string representation of any results returned by the stored procedure.

Params (Variant) Optional – additional parameters for the CallSelect

Example:

```
Dim vErrNo As Variant  
Dim sSQL As String  
Dim sColumns As String  
Dim sRows As String  
sSQL = "SPQuery" ' Name of stored procedure  
vErrNo = SP.CallSelect (sSQL, sColumns, sRows)
```

Database Stored Procedure Object

This object is *obsolete* and has been replaced with the Embedded Procedure structure.

The StoredProcedure Object is used to execute a stored procedure. It features the following set of properties and methods with which you can manipulate the object and its content.

StoredProc Properties and Methods:

CommandText

Sets or returns a value containing a provider command such as a stored procedure call.

Syntax: StoredProcedure.CommandText = Value
Alternate: Value = StoredProcedure.CommandText
Value (String) stored procedure name

Example:

See Execute

CommandTimeout

Indicates how long to wait while executing a command before terminating the attempt and generating an error.

Syntax: StoredProcedure.CommandTimeout = Value
Alternate: Value = StoredProcedure.CommandTimeout
Value (Long) time in seconds, default is 30

Example:

See Execute

CommandType

Specifies the type of command prior to execution to optimize performance.

Syntax: `StoredProc.CommandType = oValue`
Alternate: `IValue = StoredProc.CommandType`
`IValue` (Long) numeric value for the command type
`oValue` (enCommandTypes) Sets 1 of the following values:

<u>Constant</u>	<u>Description</u>
<code>dbCmdText</code>	Evaluates CommandText as a textural definition of a command.
<code>dbCmdStoredProc</code>	Evaluates CommandText as a stored procedure that will return records.
<code>dbCmdExecuteNoRecords</code>	Evaluates CommandText as a stored procedure that will return no records.
<code>dbCmdUnknown</code>	The type of command in the CommandText property is not known.
<code>dbExecuteNoRecords</code>	The stored procedure does not return records.
<code>dbManualDeclare</code>	This will execute the stored procedure directly without checking the syntax.
<code>dbOpenNoRecords</code>	For connection to Sybase.

Example:
See Execute

CreateParameter

Creates a new Param Object with the specified properties.

Syntax: `CreateParameter (Index, [Datatype], [Direction], [Size], [Value])`
`Index` (Long) Represents the parameter's index
`Datatype` (enDataTypes) Optional – the parameter's data type
`Direction` (enDirections) Optional – the parameter's direction
`Size` (Long) Optional – the parameter's length
`Value` (Variant) Optional – the parameter's value

Example:
See Execute

Data

A string representation of any results returned by the stored procedure.

Syntax: `Value = StoredProc.Data`
`Value` (String) results of the stored procedure

Example:
See Execute

DataSource

This property indicates which data source to connect.

Syntax: `StoredProc.DataSource = Value`

Alternate: `Value = StoredProc.DataSource`

Value (String) name of the data source

Example:

See Execute

Dict

A string representation of the columns returned by the SQL statement.

Syntax: `Value = StoredProc.Dict`

Value (String) representation of the columns returned

Example:

See Execute

Execute

Executes the stored procedure in the CommandText Property.

Syntax: `[OK =]StoredProc.Execute`

OK (Boolean) True or False based on the success of calling the stored procedure regardless of the outcome

Example:

```
Dim ddict As Variant
Dim ddata As Variant
Dim spobj As StoredProc
Set spobj = New StoredProc
spobj.DataSource = "Oracle"
spobj.CommandText = "byroyalty"
spobj.CommandType = dbCmdStoredProc
spobj.Param(1).DataType = dbInteger
spobj.Param(1).Direction = dbParamInput
spobj.Param(1).Value = 100
spobj.Execute
ddict = spobj.Dict
ddata = spobj.Data
```

Param

The Param object represents a parameter or argument associated with a StoredProc object based on a parameterized stored Procedure. The

Param object represents in/out arguments and the return values of stored procedures.

Syntax: Value = StoredProc.Param(Index).Property
Value (StoredParam) data value of the parameter
Index (Long) index of the parameter
Property (object) the available objects are listed below

Example:

See Execute

Param().Datatype

This property indicates the data type of the Param Object.

Syntax: StoredProc.Param(Index).Datatype = oValue
Alternate: IValue = StoredProc.Param(Index).Datatype
Index (Long) index of the parameter
IValue (Long) the numeric value of an enDataTypes value
oValue (enDataTypes) sets 1 of the following values:

<u>Constant</u>	<u>Description</u>
dbBigInt	An 8-byte signed integer.
dbBinary	A binary value.
dbBoolean	A Boolean value.
dbBSTR	A null-terminated char str (Unicode).
dbChar	A String value.
dbCurrency	A currency value. Currency is a fixed-point number with 4 digits to the right of the decimal point. It is stored in an 8-byte signed integer scaled by 10,000.
dbDate	A Date value. A date is stored as a Double, the whole part of which is the number of days since December 30, 1899, and the fractional part of which is the fraction of a day.
dbDBDate	A date value (yyyymmdd).
dbDBTime	A time value (hhmmss).
dbDBTimeStamp	A date-time stamp (yyyymmddhhmmss plus a fraction in billionths).
dbDecimal	An exact numeric value with a fixed precision and scale.
dbDouble	A double-precision floating point value.
dbEmptyNo	value was specified.
dbError	A 32-bit error code.
dbGUID	A globally unique identifier (GUID).

dbIDispatch	A pointer to an IDispatch interface on an OLE object.
dbInteger	A 4-byte signed integer.
dbIUnknown	A pointer to an unknown interface on an OLE object.
dbLongVarBinary	A long binary value.
dbLongVarChar	A long String value.
dbLongVarWChar	A long null-terminated string value.
dbNumeric	An exact numeric value with a fixed precision and scale.
dbSingle	A single-precision floating point value.
dbSmallInt	A 2-byte signed integer.
dbTinyInt	A 1-byte signed integer.
dbUnsignedBigInt	An 8-byte unsigned integer (DBType_UI8).
dbUnsignedInt	A 4-byte unsigned integer.
dbUnsignedSmallInt	A 2-byte unsigned integer.
dbUnsignedTinyInt	A 1-byte unsigned integer.
dbUserDefine	A user-defined variable.
dbVarBinary	A binary value.
dbVarChar	A string value.
dbVariant	An Automation Variant.
dbVarWChar	A null-terminated Unicode chr string.
dbWChar	A null-terminated Unicode chr string.

Param().Direction

This property indicates whether the Parameter represents an input parameter, an output parameter, or both, or if the parameter is the return value from a stored procedure.

Syntax: `StoredProc.Param(Index).Direction = oValue`

Alternate: `IValue = StoredProc.Param(Index).Direction`

Index (Long) index of the parameter

IValue (Long) the numeric value of an enDirections value

oValue (enDirections) sets or returns 1 of the following values:

<u>Constant</u>	<u>Description</u>
dbParamInput	Default, indicates an input parameter
dbParamOutput	Indicates an output parameter
dbParamInputOutput	Indicates both an input and output parameter
dbParamReturnValue	Indicates a return value.

Param().NumericScale

This property indicates the scale of the numeric value by specifying the number of decimal places to which the number will be resolved. Not all

database types support this parameter. If it is not supported, the ODBC driver will ignore this setting. The default is 0.

Syntax: `StoredProc.Param(Index).NumericScale = Value`

Alternate: `Value = StoredProc.Param(Index).NumericScale`

Index (Long) index of the parameter

Value (Long) the number of places to resolve the parameter's value

Param().Precision

This property indicates the degree of precision for numeric values by specifying the maximum number of digits used for a parameter. Not all database types support this parameter. If it is not supported, the ODBC driver will ignore this setting. The default is 0.

Syntax: `StoredProc.Param(Index).Precision = Value`

Alternate: `Value = StoredProc.Param(Index).Precision`

Index (Long) index of the parameter

Value (Long) the maximum number of integers that will make up the size of the value

Param().Size

This property indicates the maximum size, in bytes or characters of the Param Object. Use the size property to determine the maximum size for values written to or read from the value property of the Param Object.

Syntax: `StoredProc.Param(Index).Size = Value`

Alternate: `Value = StoredProc.Param(Index).Size`

Index (Long) index of the parameter

Value (Long) size in bytes or characters

Param().Value

This property indicates the value assigned to the Param object. Use the Value property to set or return parameter values with Param Objects.

Syntax: `StoredProc.Param(Index).Value = Value`

Alternate: `Value = StoredProc.Param(Index)`

Value (Variant) value of the dbParam

Index (Long) index of the parameter

ParamCount

This function indicates the number of parameters associated with the current stored procedure object. A parameter's index may be zero-based so the count may appear 1 less than expected.

Syntax: Value = StoredProc.ParamCount

Value (Variant) the number of parameters associated with this stored procedure.

Prepared

This property indicates whether or not to save a compiled version of a command before execution. The prepared property is used to have the provider save a prepared (or compiled) version of the query specified in the CommandText property before the object's first execution. If the property is False, the provider will execute the object directly without creating a compiled version.

Syntax: StoredProc.Prepared = Value

Alternate: Value = StoredProc.Prepared

Value (Boolean) set to True to save a compiled version before execution

Results

This property represents the entire set of records from a base table or the results of an executed command.

Syntax: Value = StoredProc.Results

Value (rdoResultset, adoRecordset) recordset object containing any rows returned by the stored procedure.

Initialization Files

Initialization files are used to make changes to the default behavior of RFgen such as resetting data connections based on usage, indexing databases, preventing server-side dialog boxes from stopping the RFgen service and altering connectivity parameters for mobile devices.

RFgen.ini

The RFgen.ini file provides configuration parameters for the following features. In all cases, the headings and settings are case sensitive.

Note: this file must be created using Notepad and stored in the RFgen directory for it to work.

[Options]	
ForceReset=0	set to a positive number to force RFgen to reset all data connections every “nn” minutes. Setting this value to 0 disables the reset function. It only impacts pooled data connections.

The TranLimit command will let the Transaction Manager process a specified number of transactions in the queue and then destroy and restart the client process in charge of processing the queue. This allows the system to release the memory for the process and debug memory leaks in the operating system.

TranLimit=x	set x equal to the number of transactions that will be processed before the Transaction Management process as well as its data connections are shut down and recreated. The data connections that are pooled will not be reset.
-------------	---

This command will have RFgen remove a connection from the pool, destroy it, and re-create it upon next use. This allows the system to release the memory for the process and debug memory leaks in the operating system.

ResetUDCx=y	set x equal to the data connector. Set the y value to the number of transactions that
-------------	---

will be processed before the reset takes place. The reset will only impact pooled connections.

In the case of some DB2 ODBC connections, some tables may not be visible to RFgen because of settings for the DB2 database. To make these tables visible, add an entry with the following format. You may have as many schemas as needed. These entries are only examples.

[AS400]	DB2 Data source name in RFgen
Schema1=S104WL5M.RB1	specific database / library index
Schema2=S104WL5M.QSYS	specific database / library index

To check JDE connectivity before a business function is executed, use the CheckX0010 option with the value of 1 and its related parameters.

[JDE]	
CheckX0010=0	perform a GetNextNumber test before BSFN execution to assure JDE connection state, 0 = off, 1 = on
SystemCode=41	parameter for X0010 call
NextNumberingIndexNo=2	parameter for X0010 call
CompanyKey=00001	parameter for X0010 call
DocumentType=IA	parameter for X0010 call

RFgen will monitor the JDE.log file for designated strings and then perform a series of reset based on finding one of the search strings.

CheckLog=1	perform a lookup within the JDE.log file (last 500 characters) for various test strings. If found then it will stop and restart all RFgen child processes, 0 = off, 1 = on
LogText1=RESETNOW	string 1 to look for in JDE.log
LogText2=JDB9900400	string 2 to look for in JDE.log

If one of the LogText parameters is found the server will:

1. Delete the log file. JDE creates a new file every time it starts.
2. Suspend all active client connections temporarily
3. Close all transaction manager client(s).
4. Close all data connections.
5. Start all data connections.
6. Start the transaction manager client(s).
7. Resume the active client connections.

RFgen can be made to reset every process after a certain number of transactions go through the queue. This includes thin client device connections, ODBC, SM, and ERP connections.

TranLimit=1000	maximum number of transactions to process before resetting all RFgen child processes (data connections and all connected users).
----------------	--

ERPDIALOGS.ini

The ERPDIALOGS.ini file provides the ability to automatically respond to dialog boxes created by software (such as JD Edwards) running on the RFgen (Communications) server.

Syntax:

Search, Title, Message, Reset, Close, Text
Search 1 = Search on Title, 2 = Search on Message
Title Window Title
Message Message Text (only if 'Search' = 2)
Reset 0 = Don't reset connection, 1-5 = Connection number to reset
Close 1 = Close Window, 2 = Click button
Text Text on button to click (only if 'Close' = 2) (include "&" to represent underscore, e.g., "E&xit" for "Exit").

Examples:

```
1, Database Password Entry, ,1,1,  
1, Database Error,,1,1,  
1, OneWorld Error,,1,1,  
1, OneWorld,,1,1,  
1, PeopleSoft Error,,1,1,  
1, PeopleSoft,,1,1,  
1, Remote Job,,1,1,  
1, Database Password Entry, ,1,1,  
1, Database Error,,1,1,  
1, OneWorld Error,,1,1,  
1, OneWorld,,1,1,  
1, PeopleSoft Error,,1,1,  
1, PeopleSoft,,1,1,  
1, Remote Job,,1,1,  
2, ,JDB9103 - Fetch failed, 1, 2, OK  
2, ,to Activate the component and correct the  
problem, 1, 2, Switch To...  
1, RFDB,, 1, 2, Retry
```

Even if 'Search' = 2, a window title must be included if one exists.

GPRS.ini

This ini file is designed to make the RFgen mobile client dial a connection so that data can be sent or retrieved to the server while in a disconnected state. This file is not intended to establish a connection for a thin client connection. If a GPRS connection is desired for a thin client connection simply establish it first, then any VPN connections if required and then launch the RFgen thin client.

In a typical implementation Server.Connect and Server.Disconnect are used both at the beginning of the process, to retrieve validation data, and at the end to submit collected data. When the connect command is executed the mobile client checks the phonebook setting in the RFgenCFG.exe program and uses either WiFi or GPRS to make the connection. If GPRS is selected the ini file tells RFgen what settings to use. If RFgen needs to establish the VPN connection as well then those settings can be documented as well.

Note that the GPRS and VPN connections must first be setup in the CE environment and RFgen simply references and starts those connections.

Using any desired method, create and deploy a file called GPRS.ini to the install directory (by default: \Program Files\RFgenCE). It contains the following settings:

```
[GPRS]
Enabled=True
Name=GPRS
User=
Pwd=
```

```
[VPN]
Enabled=True
Name=My Work Network
User=
Pwd=
```

The **Enabled** parameter just tells RFgen if it should make that connection.

The **Name** parameter is the name used in the setup of the connection in the operating system's configuration.

The **User** and **Pwd** fields, if needed, are stored in the ini file as well.

SQL Reserved Words

The following list includes words reserved by RFgen for use in internally generated applications SQL statements. Accordingly, these words should not be used as table names or as field/column names, not even when enclosed by square brackets, eg “[date]”. Words followed by an asterisk (*) are reserved for future use (for example, LEVEL, OUTER, TABLEID).

A

ADD	ANY
ALL	AS
ALPHANUMERIC	ASCII
ALTER	AUTOINCREMENT
AND	AVG

B

BETWEEN	BOOLEAN
BINARY	BY
BIT	BYTE

C

CHAR, CHARACTER	COUNTER
COLUMN	CREATE
CONSTRAINT	CURRENCY
COUNT	

D

DATABASE	DISALLOW
DATE	DISTINCT
DATETIME	DISTINCTROW
DELETE	DOUBLE
DESC	DROP

E-F

EQV	FLOAT4
EXISTS	FOREIGN
FLOAT, FLOAT8	FROM

G-H

GENERAL	GUIDE
GROUP	HAVING

I

IEEEDOUBLE	INSERT
IEEE SINGLE	INT, INTEGER, INTEGER1
IGNORE	INTEGER2
IMP	INTEGER4
IN	INTO
INDEX	IS
INNER	

J-L

JOIN	LOGICAL, LOGICAL1
KEY	LONG
LEFT	LONGTEXT
LEVEL*	LONGBINARY
LIKE	

M-N

MAX	NOT
MEMO	NULL
MIN	NUMBER
MOD	NUMERIC
MONEY	

O-P

OLEOBJECT	OWNERACCESS
ON	PARAMETERS
OPTION	PERCENT
OR	PIVOT
ORDER	PRIMARY
OUTER*	PROCEDURE

R-S

REAL	SMALLINT
REFERENCES	SOME
RIGHT	STDEV

SELECT	STDEVP
SET	STRING
SHORT	SUM
SINGLE	

T

TABLE	TIMESTAMP
TABLEID*	TOP
TEXT	TRANSFORM
TIME	

U-Y

UNION	VARGBINARY
UNIQUE	VARP
UPDATE	WHERE
VALUE	WITH
VALUES	XOR
VAR	YESNO
VARCHAR	

Index

A

- AbortTrans, 288
- Activate, 297
- ActiveGrammer, 320
- ActiveX files in VBA, 204
- AddGrammar, 320
- AddGrammarFile, 321
- AddListEntry, 330
- AddNew, 342
- Administration
 - Services, 23
- ADO and RDO
 - Language
 - Extensions, 28
- Alpha Only Validation, 105
- App Object
 - Extensions, 235
 - App..LogError, 242
 - App.CallForm, 235
 - App.CallMacro, 236
 - App.CallMenu, 236
 - App.ChangeLoginFor m, 237
 - App.ChangeUserPass word, 237
 - App.ClearValues, 237
 - App.ClientType, 238
 - App.ConnAvailable, 238
 - App.ErrClear, 238
 - App.ErrCount, 238
 - App.ErrDesc, 239
 - App.ErrNative, 239
 - App.ErrNo, 239
 - App.ExecuteMenuSel ection, 239
 - App.ExitForm, 240
- App.ExitSession, 240
- App.GetInput, 240
- App.GetString, 241
- App.GetValue, 241
- App.IpAddress, 241
- App.Locale, 242
- App.MakeList, 243
- App.MsgBox, 243
- App.PromptCount, 244
- App.PromptNo, 245
- App.SendChar, 245
- App.SendKey, 245
- App.SetDisplay, 246
- App.SetFocus, 246
- App.SetMenu, 247
- App.SetOption, 247
- App.SetValue, 250
- App.ShowList, 250
- App.Sleep, 251
- App.TimerEnabled, 251
- App.TimerInterval, 251
- App.User, 252
- App.UserProperty, 252
- Application Based Extensions, 235
- Application
 - Construction Area, 110
- Application Logs, 73
- Application Screens, 157
- Application Statistics, 74
- Application Testing, 52
- Applications, 79
- Applications List, 82
- Authorize the Server, 143, 144

Authorizing
Development, 8
Auto UnDock VBA, 16

B

Back Color, 125
Basic Implementation
 Steps, 6
Batch Failover, 124
Before You Begin, 3
BeginTrans, 271, 292
Bell, 253
Branch/Goto
 Validation, 108
Button Object, 86
BytesReceived, 357

C

Calculation Defaults,
 99
CallAction, 368
CallForm, 235
CallMacro, 236, 258
CallMenu, 236
CallProc, 369
CallSelect, 370
ceObject Object
 Extensions, 277
ChangeLoginForm,
 237
ChangeUserPassword
 , 237
Character String
 Extraction Default,
 99
Character String
 Validation, 104
CheckBox Object, 86
ClassObject, 336
Clear, 253, 330, 336,
 343
ClearEOL, 254
ClearEOP, 254

ClearGrammar, 321
ClearProfile, 321
ClearValues, 237
Click, 205
ClickAndSkipPrompts,
 277
ClickCoordinates, 278
Client Inactivity
 Timeout, 13
Client Network
 Control, 185
Client Sessions, 147
ClientType, 238
ColumnCount, 336
ColumnName, 337
ComboBox Object, 86
CommandText, 371
CommandTimeout,
 259, 371
.CommandType, 371
CommitTrans, 272,
 292, 303
Concatenation
 Default, 99
Confidence Level, 21
ConfidenceLevel, 321
Configure the Server,
 143
Configuring
 Application
 Environment
 Options, 11
Configuring
 Connection Pooling,
 33
Configuring Desktop
 Options, 15
Configuring ERP
 Connection, 38
Configuring Master
 Database, 9
Configuring Mobile
 Device Theme, 17
Configuring ODBC
 Data Sources, 35

Configuring ODBC
Database
Connections, 30
Configuring RFgen
Software, 8
Configuring Scheduled
Down Time, 33
Configuring Screen
Mapping
Connection, 37
Configuring Security
Options, 19
Configuring Speech
Recognition
Options, 20
Configuring TCP/IP
Services Options, 22
Configuring
Transaction
Management, 24
Configuring Visual
Basic Options, 28
Configuring Visual
SourceSafe
Integration, 29
Configuring Web
Connection, 42
ConnAvailable, 238
Connect, 259
Connected, 303
Connection Pooling,
33, 39, 43
Connection Statistics,
74
ConnectionProperty,
265
Connections Tab, 153
ConnectTimeout, 365
Contains String
Validation, 107
Controller Failover
Monitoring, 23
Controls, 112
Copies, 297
Count, 272
Create, 286
CreateParameter, 372
CurScreen, 304

D

Data, 365, 372
Data Range
Validation, 106
Database Path, 128
Database Related
Extensions, 271
DataSet Object, 342
DataSet.AddNew, 342
DataSet.Clear, 343
DataSet.IsEOF, 343
DataSet.MoveFirst,
343
DataSet.MoveLast,
343
DataSet.MoveNext,
343
DataSet.MovePrevious
s, 343
DataSet.MoveTo, 344
DataSet.Param, 344
DataSet.RowCount,
344
DataSet.ParamName,
345
DataSet.RowCount,
345
DataSet.Schemadl,
346
DataSource, 337, 366,
373
Date Validation, 107
DB Object Extensions,
271
DB.BeginTrans, 271
DB.CommitTrans, 272
DB.Count, 272
DB.Execute, 272
DB.Extract, 273
DB.MakeList, 273

DB.OpenResultset,
 274
DB.RedirectDataSource
 e, 276
DB.RollbackTrans,
 276
DB.SaveBitmap, 276
DB.UseDataSource,
 277
dbBigInt, 374
dbBinary, 374
dbBoolean, 374
dbBSTR, 374
dbChar, 374
dbCmdExecuteNoRec
 ords, 372
dbCmdStoredProc,
 372
dbCmdText, 372
dbCmdUnknown, 372
dbCurrency, 374
dbDate, 374
dbDBDate, 374
dbDBTime, 374
dbDBTimeStamp, 374
dbDecimal, 374
dbDouble, 374
dbEmptyNo, 374
dbError, 374
dbExecuteNoRecords,
 372
dbGUID, 374
dbIDispatch, 375
dbInteger, 375
dbIUnknown, 375
dbLongVarBinary, 375
dbLongVarChar, 375
dbLongVarWChar,
 375
dbManualDeclare, 372
dbNumeric, 375
dbOpenNoRecords,
 372
dbParamInput, 375
dbParamInputOutput,
 375
dbParamOutput, 375
dbParamReturnValue,
 375
dbSingle, 375
dbSmallInt, 375
dbTinyInt, 375
dbUnsignedBigInt, 375
dbUnsignedInt, 375
dbUnsignedSmallInt,
 375
dbUnsignedTinyInt,
 375
dbUserDefine, 375
dbVarBinary, 375
dbVarChar, 375
dbVariant, 375
dbVarWChar, 375
dbWChar, 375
DCount, 347
DebugLog, 337
Declarations, 205
Del, 347
DeleteProperty, 267
DemoMode, 322
Design Mode, 16
Design Tab Overview,
 78
Device Object
 Extensions, 277
Device Profiles, 121
Device.ClickAndSkipP
 rompts, 277
Device.ClickCoordinat
 es, 278
Device.ForceLocal,
 278
Device.GetGPSInfo,
 278
Device.GoOffline, 280
Device.GoOnline, 280
Device.Offline, 281
Device.Online, 281
Device.Platform, 282

Device.PlaySound,
282
Device.PrinterOff, 282
Device.PrinterOn, 282
Device.Send, 283
Device.SendCommPort,
283
Device.SetCameraOption, 283
Device.SetCommMode, 284
Device.SetCommPort,
284
Device.TakePicture,
285
Device.WriteFile, 285
DeviceObject, 285
DeviceObject.Create,
286
DeviceObject.Execute,
286
DeviceObject.LastError,
287
DeviceObject.Name,
287
DeviceObject.Release
, 288
DeviceObject.ReturnValue, 288
Devices Menu, 58
Dict, 373
DisableParam, 338
DisableTimeout, 268
Discard Form Data
When Chaining, 12
Disconnect, 260
DisplayGrammar, 322
Displaying Data From
Other Tables, 98
Double Byte Support,
12
Down Arrow as Enter
Key, 12

Download
Tables/Business
Functions, 43
DrawLine, 254
Dynamic Array
Structure, 346

E

Echo Mode, 20
embedded methods,
158
Embedded Procedure,
336
Embedded
Procedures, 332,
333
emProc.ClassObject,
336
emProc.Clear, 336
emProc.ColumnCount,
336
emProc.ColumnName,
337
emProc.DataSource,
337
emProc.DebugLog,
337
emProc.DisableParam
, 338
emProc.Execute, 338
emProc.ExecuteMethod,
338
emProc.LogMode, 338
emProc.Name, 339
emProc.Param, 339
emProc.ParamCount,
339
emProc.ParamEx, 340
emProc.ParamName,
340
emProc.Queue, 341
emProc.QueueName,
341

emProc.QueueOffline,
 341
emProc.QueueSeqNo,
 342
emProc.RowCount,
 342
Enabled, 323
Encrypting Master
 Database, 10
Encryption Key, 10
EndDoc, 298
Enterprise Resource
 Planning
 Extensions, 292
Environment
 Properties, 15, 197
EnvironmentProperty,
 268
ERP Business
 Functions
 Download, 46
ERP Object
 Extensions, 292
ERP.
 SetHardRelease,
 296
ERP.BeginTrans, 292
ERP.CommitTrans,
 292
ERP.LogOff, 293
ERP.LogOn, 293
ERP.MakeList, 294
ERP.ReadData, 295
ERP.RollbackTrans,
 296
ERP.Session, 296
ErrClear, 238
ErrCount, 238
ErrDesc, 239
ErrNative, 239
ErrNo, 239
Escape Processing
 Delay, 13
Events, 113, 205

Execute, 272, 286,
 338, 368, 373
ExecuteMenuItem
 239
ExecuteMethod, 338
ExecuteSQL, 260
ExitForm, 240
ExitSession, 240
Export Items from
 RFgen Database, 66
Ext, 349
Extract, 273
Extraction Default, 99

F

Find in Application
 Scripts, 70
FindText, 304
FixLeft, 350
FixRight, 350
Font Name, 125
Font Size, 125
FontBold, 298
FontItalic, 298
FontName, 299
FontSize, 299
FontStrikeThru, 299
FontUnderline, 299
ForceLocal, 278
Fore Color, 125
Forms, 82
Forms Tab, 84
Frame Object, 87
Full Screen, 124, 188

G

GetArea, 305
GetAttribute, 305
GetBackColor, 306
GetConnection, 268
GetCurrentType, 256
GetCursor, 306
GetForeColor, 307

GetGPSInfo, 278
GetInput, 240, 323
GetItems, 288
GetItemsEx, 289
GetName, 299
GetPage, 307
GetProperty, 269
GetString, 241
GetTable, 261
GetText, 308
GetTypes, 257
GetValue, 241
Global User Defined
 Subroutines and
 Functions, 203
Global
 Variables/Objects,
 204
GoOffline, 280
GoOnline, 280
GotFocus, 205
Goto Validation, 108
GoToScreen, 308
Graphical Services, 22
Greater Than
 Validation, 106
Import Items to RFgen
 Database, 66
Index Validation, 107
Indexing, 34
Initialization Files -
 ERPDialogs.ini, 380
Initialization Files -
 GPRS.ini, 381
Initialization Files -
 RFgen.ini, 378
Initialize, 206
Ins, 351
Install Applications on
 Device, 58
Installing additional
 files, 136
Integer Only
 Validation, 105
International Currency
 Support, 12, 192
Introduction, 0
IpAddress, 241
IsConnected, 262
IsEOF, 343
IsOffline, 281
IsOnline, 281
IsScreen, 309

H

HeaderValue, 366
Height, 255
Help Menu, 77
High level, 158
Host Screen Macro,
 176
Hosts, 79
Hosts List, 121
Hosts Tree, 167
HostScreen Object, 87
Hot-Key, 160

I

Image Object, 87
Image Resources, 136

K

Keypress, 206

L

Label Object, 88
Language Packs, 6
Language Set, 16,
 124, 188
Last/Prior Entry
 Default, 100
LastError, 287
Less Than Validation,
 106
LField, 353
List Object, 330

List.AddListEntry, 330
List.Clear, 330
List.MaxRows, 330
List.ReturnAllRows,
 330
List.SetReportColumn,
 331
List.ShowEmptyList,
 331
List.ShowList, 332
List.SQL, 332
ListBox Object, 88
Lists (Listing Choices)
 Default, 100
Load, 206
Loading the Software,
 4
LoadProfile, 323
Locale, 242
LocalHostName, 357
LocalIP, 358
Locate, 353
LocateAdd, 354
LocateDel, 355
LogError, 242
Login, 368
LogMode, 338
LogOff, 293, 309
Logoff Session, 152
LogOn, 293, 310
Logout, 368
LostFocus, 206
Low level, 158

M

MacroName, 290
Main Menu, 157
MakeList, 243, 273,
 294
Making Screen
 Mapping Work, 156
MaxRows, 330
MDAC, 7
Medium level, 158

Menu List, 80
Menu Object, 88
Menus, 79
Microsoft Data Access
 Components, 36
Mobile Administration
 Console, 148
Mobile Administration
 Console Menu Bar,
 150
Mobile Administration
 Console Overview, 2
Mobile and Wireless
 Client, 6
Mobile Application
 Server, 143
Mobile Application
 Server Overview, 2
Mobile Client, 123,
 184
Mobile CnC Services,
 23
Mobile Development
 Studio Development
 Tools, 78
Mobile Development
 Studio Menu Bar, 50
Mobile Development
 Studio Overview, 0
Mobile Device
 Extensions, 277
Mobile Devices, 184
Mode, 257
MoveFirst, 343
MoveLast, 343
MoveNext, 343
MovePrevious, 343
MoveQueue, 290
MoveTo, 344
MsgBox, 243

N

Name, 287, 339

Network Application
Testing, 54
NewPage, 300
Not Condition
Validation, 107
Not Equal To
Validation, 106
Numeric Only
Validation, 105

Param, 339, 344, 373
Param().Datatype,
374
Param().Direction,
375
Param(
).NumericScale, 375
Param().Precision,
376
Param().Size, 376
Param().Value, 376
ParamCount, 339,
344, 376
ParamEx, 340
ParamName, 340, 345
Password, 128
Pattern Match
Validation, 105
Pattern Not Allowed
Validation, 105
PauseTime, 324
Ping, 262
PingHost, 310
Platform, 282
PlaySound, 282
Post-Amble, 14, 193
Pre-Amble, 14, 193
Prepared, 377
Primary Hardware, 16
Print, 255, 301
Printer Extensions,
297
Printer Object
Extensions, 297
Printer.Activate, 297
Printer.Copies, 297
Printer.EndDoc, 298
Printer.FontBold, 298
Printer.FontItalic, 298
Printer.FontName, 299
Printer.FontSize, 299
Printer.FontStrikeThru,
299
Printer.FontUnderline,
299

O

OnBackup, 206
OnClose, 363
OnConnect, 207, 363
OnCursor, 207
OnDataArrival, 364
OnDisconnect, 207
OnEnter, 208
OnError, 364
OnEscape, 208
OnFkey, 208
OnInRange, 209
OnLocale, 209
OnOutOfRange, 210
OnReadData, 210
OnRefresh, 210
OnReturn, 210
OnScan, 211
OnSearch, 211
OnSendComplete,
364
OnSendProgress, 364
OnSpeech, 211
OnTimer, 212
OnUpdate, 212
OnVoCollect, 212
OpenResultset, 274
Options Object, 88
Orientation, 300

P

PadInput, 310
PageWidth, 301

Printer.GetName, 299
Printer.NewPage, 300
Printer.Orientation,
 300
Printer.PageWidth,
 301
Printer.Print, 301
Printer.PrintQuality,
 301
Printer.PrintRaw, 302
PrinterOff, 282
PrinterOn, 282
PrintQuality, 301
PrintRaw, 302
Prior Entry Default,
 100
Prompt Property
 Extensions, 215
PromptCount, 244
PromptNo, 245
Protocol, 358
Provider, 127

Q

Queue, 341
QueueMacro, 262,
 291
QueueName, 291, 341
QueueOffline, 341
QueueSeqNo, 342

R

Read Mode, 21
ReadData, 295
ReadMode, 324
ReadRate, 324
ReceiveTimeout, 366
RedirectDataSource,
 276
Refresh, 255
Release, 288
Re-Map, 160

Remote Application
 Explorer, 60
Remote Database
 Explorer, 62
Remote SQL Explorer,
 63
RemoteHost, 358
RemoteHostIP, 359
RemotePort, 359
RemoveGrammar, 325
RemoveGrammarFile,
 325
Rep, 356
Repeat, 325
Replace in Application
 Scripts, 71
Reply, 367
Report All Errors, 13
Reports Menu, 73
Request, 367
ResetConnection, 311
ResetCursor, 255
Resources Tab
 Overview, 121
Response Timeout, 22
Results, 377
ReturnAllRows, 330
ReturnValue, 288
ReverseOff, 256
ReverseOn, 256
RFgen Sessions
 window, 151
RFgen SourceSafe
 Browser, 64
RFgen.bas, 79
RFgen.mdb, 4, 10, 67
RFgenCE, 200
RFgenCFG, 186
RField, 356
RFPrompt().AddItem,
 215
RFPrompt().Autosize,
 215
RFPrompt().Bitmap,
 216

RFPrompt().Caption,	216	RFPrompt().LabelFont	
216		Size, 225	
RFPrompt().Checked,	217	RFPrompt().LabelFont	
217		Underline, 225	
RFPrompt().Clear, 217		RFPrompt().LabelFore	
RFPrompt().Defaults,	217	Color, 226	
217		RFPrompt().LabelLeft,	
RFPrompt().Display3D	, 218	226	
, 218		RFPrompt().LabelTop,	
RFPrompt().DisplayOn	ly, 218	226	
ly, 218		RFPrompt().Length,	
RFPrompt().Edits, 218		227	
RFPrompt().ErrMsg,	219	RFPrompt().List, 227	
219		RFPrompt().ListCount,	
RFPrompt().FieldBack		228	
Color, 219		RFPrompt().ListIndex,	
RFPrompt().FieldFont		228	
Bold, 220		RFPrompt().PageDow	
RFPrompt().FieldFontI		n, 228	
talic, 220		RFPrompt().PageNo,	
RFPrompt().FieldFont		228	
Size, 220		RFPrompt().PageUp,	
RFPrompt().FieldFont		229	
Underline, 221		RFPrompt().Password,	
RFPrompt().FieldFore		229	
Color, 221		RFPrompt().RemoveIt	
RFPrompt().FieldId,	221	em, 229	
221		RFPrompt().Required,	
RFPrompt().Format,	222	230	
222		RFPrompt().ScrollDow	
RFPrompt().Highlight,	222	n, 230	
222		RFPrompt().ScrollUp,	
RFPrompt().ImageGet		230	
Bitmap, 223		RFPrompt().SelLength	
RFPrompt().ImagePat		, 231	
h, 223		RFPrompt().SelStart,	
RFPrompt().Index,	223	231	
223		RFPrompt().Sorted,	
RFPrompt().LabelBack		231	
Color, 224		RFPrompt().Stretch,	
RFPrompt().LabelFont		232	
Bold, 224		RFPrompt().Text, 232	
RFPrompt().LabelFont		RFPrompt().TextLeft,	
Italic, 225		232	

RFPrompt().TextTop,
233
RFPrompt().Type, 233
RFPrompt().ValField,
234
RFPrompt().ValTable,
234
RFPrompt().Visible,
235
RollbackTrans, 276,
296
RowCount, 342, 345

S

SaveBitmap, 276
SaveProfile, 325
Schemald, 346
SCM icon, 143
Screen Display
Extensions, 253
Screen Mapping, 156
Screen Mapping
Configuration, 161
Screen Mapping
Design
Considerations, 158
Screen Mapping
Extensions, 302
Screen Mapping
Keyboard/Special
Key Configuration,
160
Screen Mapping
Levels, 158
Screen Mapping
Logon Security
Considerations, 159
Screen Mapping
Programming
Philosophy, 157
Screen Mapping
Recording Options,
181

Screen Mapping
Runtime
Environment
Variables, 161
Screen Mapping
Scheduled
Downtime, 164
Screen Mapping
System Integrity
Considerations, 159
Screen Mapping
Theory Of
Operation, 157
Screen Object
Extensions, 253
Screen.ReverseOff,
256
Screen.ReverseOn,
256
Screen.Bell, 253
Screen.Clear, 253
Screen.ClearEOL, 254
Screen.ClearEOP,
254
Screen.DrawLine, 254
Screen.Height, 255
Screen.Print, 255
Screen.Refresh, 255
Screen.ResetCursor,
255
Screen.Width, 256
Scripts Menu Bar, 112
Send, 283
Send Keys Selection,
169
Send Message, 152
SendChar, 245
SendCommPort, 283
SendCTRL, 311
SendCTRLAlt, 311
SendKey, 245, 312
SendKeyAlt, 312
SendMessage, 269
SendQueue, 263
SendTable, 263

SendText, 313
SendTextAlt, 313
SendTimeout, 367
Sentence Pause, 21
SeqNo, 292
Server Based
 Extensions, 258
Server Object
 Extensions, 258
Server.CallMacro, 258
Server.CommandTime
 out, 259
Server.Connect, 259
Server.Disconnect,
 260
Server.ExecuteSQL,
 260
Server.GetTable, 261
Server.IsConnected,
 262
Server.Ping, 262
Server.QueueMacro,
 262
Server.SendQueue,
 263
Server.SendTable,
 263
Server.SetHost, 264
Server.ShowProgress,
 264
Server.SyncApps, 264
Server.WriteFile, 265
Service Configuration,
 145
Session Shutdown
 Delay, 13
SessionID, 314
SessionPwd, 314
Sessions Tab, 150
SessionUser, 315
Set Default, 101
SetBase, 315
SetCameraOption,
 283
SetCommMode, 284
SetCommPort, 284
SetCursor, 315
SetDelay, 316
SetDisplay, 246
SetFocus, 246
SetHardRelease, 296
SetHost, 264
SetLanguage, 326
SetMenu, 247
SetOption, 247
SetProperty, 270
SetReportColumn,
 331
SetSession, 296, 316
SetTimeout, 316
SetType, 258
SetValue, 250
Show, 258
Show Session, 151
ShowEmptyList, 331
ShowList, 250, 332
ShowProgress, 264
Signature Object, 89
SIP.GetCurrentType,
 256
SIP.GetTypes, 257
SIP.Mode, 257
SIP.SetType, 258
SIP.Show, 258
Skip the Current Entry
 Default, 101
sktClose, 360
sktConnect, 360
sktGetData, 361
sktPeekData, 362
sktSendData, 363
Sleep, 251
SM Host Macros, 79
SM Object Extensions,
 302
SM Transactions, 79
SM.BeginTrans, 302
SM.CommitTrans, 303
SM.Connected, 303
SM.CurScreen, 304

SM.FindText, 304
SM.GetArea, 305
SM.GetAttribute, 305
SM.GetBackColor,
 306
SM.GetCursor, 306
SM.GetForeColor, 307
SM.GetPage, 307
SM.GetText, 308
SM.GoToScreen, 308
SM.IsScreen, 309
SM.LogOff, 309
SM.LogOn, 310
SM.PadInput, 310
SM.PingHost, 310
SM.ResetConnection,
 311
SM.SendCTRL, 311
SM.SendCTRLAlt, 311
SM.SendKey, 312
SM.SendKeyAlt, 312
SM.SendText, 313
SM.SendTextAlt, 313
SM.SessionID, 314
SM.SessionPwd, 314
SM.SessionUser, 315
SM.SetBase, 315
SM.SetCursor, 315
SM.SetDelay, 316
SM.SetSession, 316
SM setTimeout, 316
SM.WaitForCursor,
 316
SM.WaitForCursorMov
 e, 317
SM.WaitForHost, 317
SM.WaitForScreen,
 318
SM.WaitForText, 318
SM.WaitForWrite, 319
SMBeginTrans, 302
SMCallMacro, 160
SMWaitForCursor,
 160
SMWaitForHost, 160

SMWaitForScreen,
 160
SMWaitForText, 160
Socket Object, 357
Socket Services, 22
Soft Input Panel
 Extensions, 256
SP Object Extensions,
 368
SP.CallAction, 368
SP.CallProc, 369
SP.CallSelect, 370
Speak, 326
Speech Rate, 21
Speech Resources,
 137
Spell, 326
SQL, 332
SQL Query Testing,
 57
SQL Reserved Words,
 383
SQL Statement (List)
 Default, 102
SQLNum, 252
Start and Stop the
 Server, 143
Start Menu Macro,
 168
Start Test, 53
Startup Mode, 124
State, 359
Stop Test, 54
StopSpeak, 327
Stored Procedure
 Download, 45
Stored Procedure
 Extensions, 368
StoredProcedure Properties
and Methods, 371
StoredProcedure.Command
 Text, 371
StoredProcedure.Command
 Timeout, 371

StoredProc.Command
 Type, 371
StoredProc.CreatePar
 ameter, 372
StoredProc.Data, 372
StoredProc.DataSource
 e, 373
StoredProc.Dict, 373
StoredProc.Execute,
 373
StoredProc.Param,
 373
StoredProc.Param(
).Datatype, 374
StoredProc.Param(
).Direction, 375
StoredProc.Param(
).NumericScale, 375
StoredProc.Param(
).Precision, 376
StoredProc.Param(
).Size, 376
StoredProc.Param(
).Value, 376
StoredProc.ParamCou
 nt, 376
StoredProc.Prepared,
 377
StoredProc.Results,
 377
String Validation, 107
Strip Validations, 108
Support Files
 Required, 7
SyncApps, 264
SYS.ConnectionPrope
 rty, 265
SYS.DeleteProperty,
 267
SYS.DisableTimeout,
 268
SYS.EnvironmentProp
 erty, 268
SYS.GetConnection,
 268
SYS.GetProperty, 269
SYS.SendMessage,
 269
SYS SetProperty, 270
SYS.UserList, 270
SYS.ValidateWinUser,
 271
System Date Default,
 97
System Level
 Extensions, 265
System Object
 Extensions, 265
System Time Default,
 98

T

Tab Width, 28
TakePicture, 285
TCP/IP Direct, 12
Telnet, 143, 149
Telnet Services, 22
Terminate, 213
Terminate a session,
 152
Test the Main Menu
 Macros, 175
Testing Menu, 52
TestMic, 327
Text Default, 97
Text Resources, 140
Text to Speech
 Extensions, 320
Text Validation, 104
TextBox Object, 89
Thin Client, 123, 184
TimerEnabled, 251
TimerInterval, 251
TM Object Extensions,
 288
TM.AbortTrans, 288
TM.GetItems, 288
TM.GetItemsEx, 289
TM.MacroName, 290

TM.MoveQueue, 290
TM.QueueMacro, 291
TM.QueueName, 291
TM.SeqNo, 292
Trailing Silence, 21
TrailingSilence, 328
Train, 328
Transaction Macros,
 79
Transaction
 Management
 Extensions, 288
Transaction Testing,
 55
Transactions, 79
Transactions List, 119
Translate Default, 98
Trasnaction Module
 Editing Menu Bar,
 120
TrimInput, 328
TTS Object
 Extensions, 320
TTS.ActiveGrammar,
 320
TTS.AddGrammar,
 320
TTS.AddGrammarFile,
 321
TTS.ClearGrammar,
 321
TTS.ClearProfile, 321
TTS.ConfidenceLevel,
 321
TTS.DemoMode, 322
TTS.DisplayGrammar,
 322
TTS.Enabled, 323
TTS.GetInput, 323
TTS.LoadProfile, 323
TTS.PauseTime, 324
TTS.ReadMode, 324
TTS.ReadRate, 324
TTS.RemoveGrammar
 , 325

TTS.RemoveGrammar
 File, 325
TTS.Repeat, 325
TTS.SaveProfile, 325
TTS.SetLanguage,
 326
TTS.Speak, 326
TTS.Spell, 326
TTS.StopSpeak, 327
TTS.TestMic, 327
TTS.TrailingSilence,
 328
TTS.Train, 328
TTS.TrimInput, 328
TTS.Volume, 329
TTS.WaitTime, 329

U

Undo Save/Delete
 Action, 72
Unload, 213
UseDataSource, 277
User, 128, 252
User Default, 104
UserList, 270
UserProperty, 252
Users, 79
Users List, 79
Utilities Menu, 64

V

Validate Application
 Scripts, 70
ValidateWinUser, 271
VB Event Click, 113
VB Event GotFocus,
 114
VB Event Keypress,
 114
VB Event Load, 114
VB Event LostFocus,
 114

VB Event OnBackup,
 114
VB Event OnConnect,
 114
VB Event OnCursor,
 114
VB Event
 OnDisconnect, 114,
 116
VB Event OnEnter,
 115
VB Event OnEscape,
 115
VB Event OnFkey,
 115
VB Event OnLocale,
 114, 115, 116
VB Event
 OnReadData, 115
VB Event OnRefresh,
 115
VB Event OnReturn,
 115
VB Event OnScan,
 115
VB Event OnSearch,
 115
VB Event OnSpeech,
 115
VB Event OnTimer,
 116
VB Event OnUpdate,
 116
VB Event OnVocollect,
 116
VB Event UnLoad,
 116
VBA Declarations, 205
VBA Events, 205
VBA Global
 Variables/Objects,
 204
VBA Language
 Extensions, 215
VBA Module Editing
 Menu Bar, 117
VBA Module Editing
 Window Fields, 118
VBA Modules, 79
VBA Modules List, 116
View Downloaded
 Tables/Business
 Functions, 47
View Transaction
 Queues, 75
Viewing ERP
 Business Functions,
 49
Visual Basic Scripts,
 203
Vocollect Services, 23
Vocollect Tasks, 141
Voice, 20
Volume, 21, 329
VT220 Key Mapping,
 166

W

WaitForCursor, 316
WaitForCursorMove,
 317
WaitForHost, 317
WaitForScreen, 318
WaitForText, 318
WaitForWrite, 319
WaitTime, 329
Web Object, 365
Width, 256
Win32.bas, 79
Windows (Graphical)
 Clients, 3
Windows Desktop
 Client, 6
Windows Service, 145
WriteFile, 265, 285

