

LET'S GET STARTED



Version 5

A Quick-Start Guide to Development

RFgen's powerful interactive development environment eases the burdens of almost any development effort by allowing developers to focus on the end results rather than the means to accomplish them.

LET'S GET STARTED

A QUICK-START GUIDE TO DEVELOPMENT

INTRODUCTION

The Mobile Development Studio provides an intuitive, yet powerful, interactive development environment (IDE). An application can literally be developed and become fully functional within minutes. The Mobile Development Studio has simplified the development phase and promotes a rapid deployment cycle for solutions.

This document provides simple exercises which can be performed by almost anyone. Developers should especially find this technical memo useful, because the purpose of this document is to immerse the reader into the IDE of the Mobile Development Studio. The objects introduced and addressed in this publication are Applications, Menus, and Users.

OVERVIEW

The Mobile Development Studio, by default, installs with a sample data set. For demonstration purposes, additional objects will be created in the sample data set through practical exercises throughout this document. The exercises in this document will create each object “from scratch” to demonstrate the rapid deployment capabilities of the Mobile Development Studio.

APPLICATION DATABASE

The Mobile Development Studio must (at the least) have an application database configured. The application database can be either a Microsoft Access file, SQLite file, SQL Server, or Oracle database and it will contain all of the configuration settings, data connections, and objects (Applications, Users, Menus, Transactions, Scripting Modules, Host Screens, etc.) used by the Mobile Development Studio. The application database is a crucial source and should be backed up regularly using sound data retention policies and procedures. The default installation of the Mobile Development Studio has been pre-configured with an application database that was installed when Mobile Development Studio was installed. The path for this database is "C:\ProgramData\RFgen5\RFgen.mdb".

RFgen - Select / Create Application Database

Cancel Save Changes

Configuration File RFgen

System / Company Id RFgen

Encryption Key

System Database Access

Data Access Mode Static

Session Timeout (hr) 0

Query Timeout (sec) 60

Reset on Error (hex)

Provider Name Microsoft.Jet.OLEDB.4.0

Microsoft Access Database...

%APPDATA%\RFgen5\rfgen.mdb

Database Login

Password

Extended Properties...

Property	Value

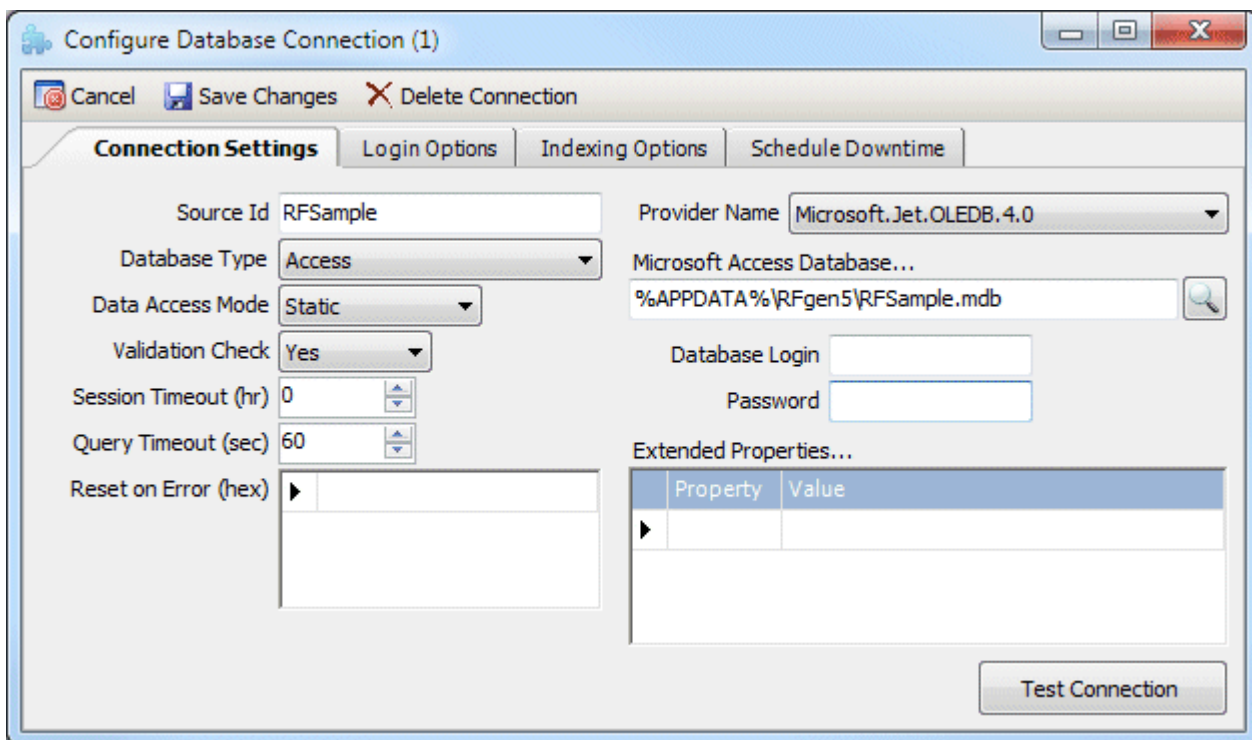
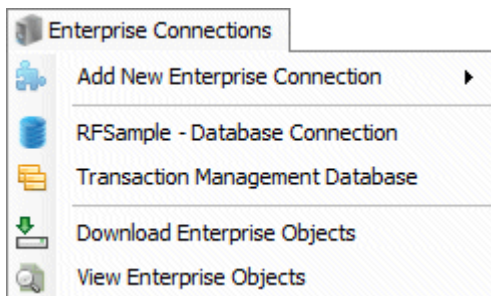
Test Connection

DATABASE CONNECTION

A database connection is a data source that allows you to connect to databases that can be of multiple database types. This connection facilitates data transfer to and from an underlying database. Creating a new ODBC system data source within your operating system is one option or you may explicitly specify the connection properties directly in the Mobile Development Studio. The data source is the actual mechanism that acts as a vehicle to facilitate communication between RFgen Software components and the target database. The default installation of the Mobile Development Studio includes a Microsoft Access database that contains sample data. The path for this sample database is generally "C:\ProgramData\RFgen5 \ RFSample.mdb".

Connection Settings

The “Connection Settings” tab is used to identify necessary parameters that will help establish a connection to the database. Note that the database connection is configured to the Microsoft Access database named “RFSample”. The “Database Type” field includes all of the other database management systems that the Mobile Development Studio can also connect to.



Login Options

Pooling enhances your investment in RFgen Software by allowing you to pool multiple RFgen devices to just one (or a few) Enterprise Resource Planning (ERP) system or Database Management System (DBMS) connection. With connection pooling enabled, a minimal number of ERP and DBMS licenses are required. On average, 10–15 remote devices can share one connection simultaneously.

Configure Database Connection (1) - [Changed]

Cancel Save Changes Delete Connection

Connection Settings **Login Options** Indexing Options Schedule Downtime

Logon Mode Automatic

Pooling Status Enabled

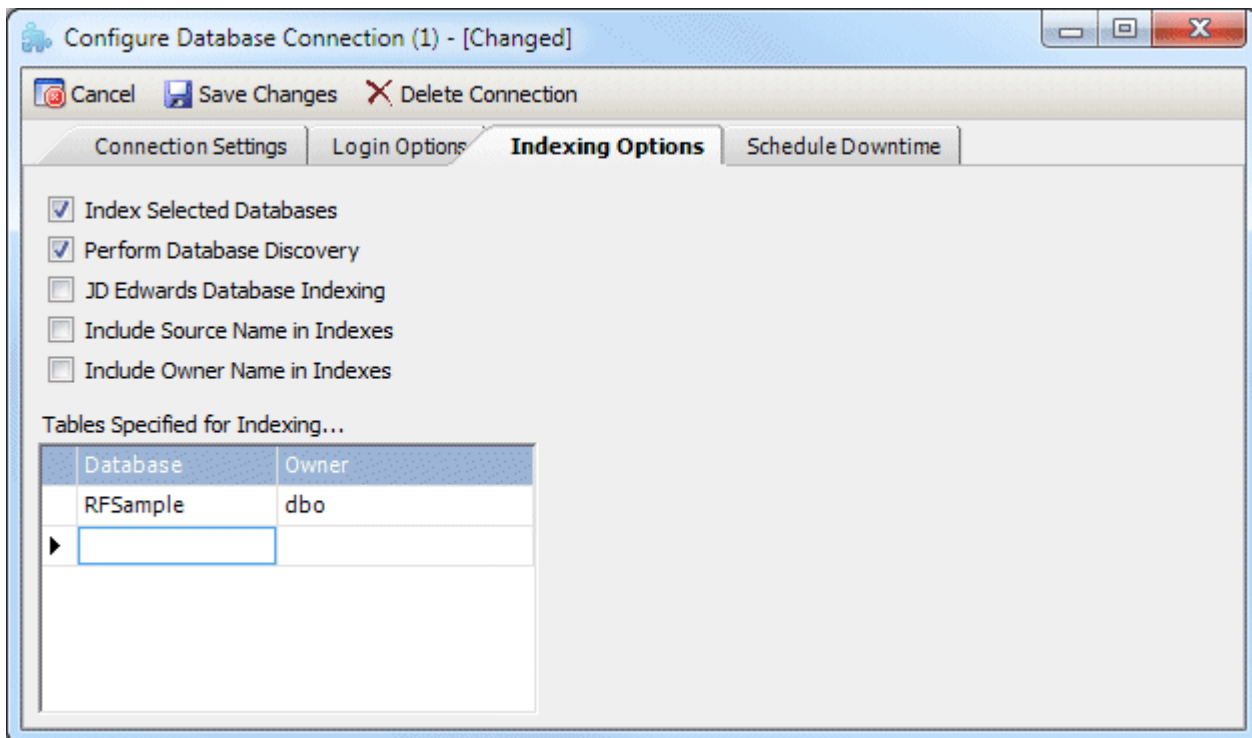
Allocated Licenses 3

Connection Pooling Named Users...

###	User Id	Password
1		
2		
3		

Indexing Options

Certain DBMS such as Oracle and SQL Server often contain data structures that benefit from being indexed. Indexing fully qualifies the paths of each of the tables in the database and allows for shorter SQL queries, since the entire path of the table is no longer needed. Instead, simply the table name will suffice when it is referenced in a query. RFgen can also re-establish a connection with the DBMS should the existing connection be interrupted due to an error propagated from the DBMS, itself.



Schedule Downtime

RFgen allows you to schedule when your ERP system or DBMS will be unavailable due to a number of reasons such as maintenance, backups, patching, hot fixes, etc.

Configure Database Connection (1)

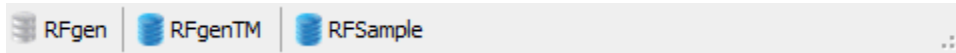
CancelSave ChangesDelete Connection

Connection SettingsLogin OptionsIndexing OptionsSchedule Downtime

From	To	Sun	Mon	Tues	Wed	Thurs	Fri	Sat
22:00	05:00	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
12:00	13:00	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

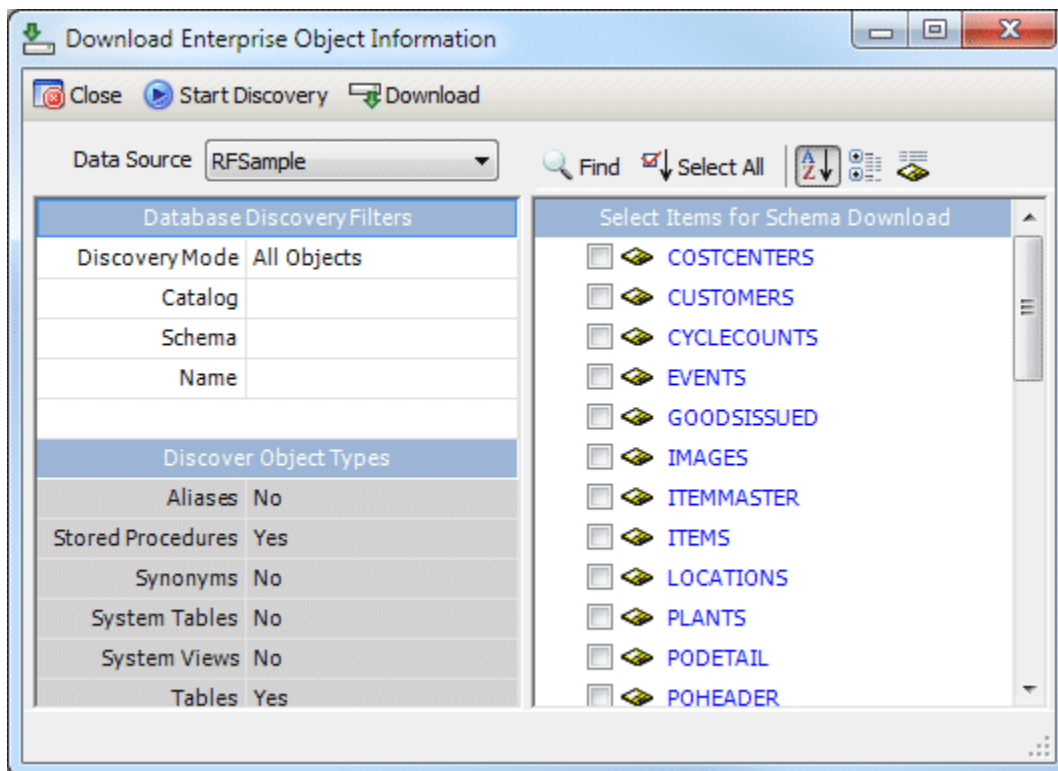
STATUS BAR AND CONNECTION “STATUS SYMBOLS”

The connectivity indicators of each data source can be easily and quickly viewed in the bottom left corner of the status bar. Incidentally, the connectivity of the transaction management connector can also be viewed as an icon in the status bar.



DOWNLOAD ENTERPRISE OBJECTS

After connecting to an ERP system and/or DBMS, it is often useful to download certain tables and/or business functions (for ERP systems). Downloading a table from an external system does not download the table data into the RFgen application database. Instead, the table structure (column definitions, data types, etc.) is downloaded into the RFgen application database. This download can be performed for each data source connection that is created in RFgen. Downloading from data source connections allows RFgen to maintain an internal mapping of which table comes from which data source connection. Since RFgen is now aware of all potential tables that will be used during development (and runtime), development efforts in the Mobile Development Studio are significantly improved. Click on Enterprise Connections and then Download Enterprise Objects.

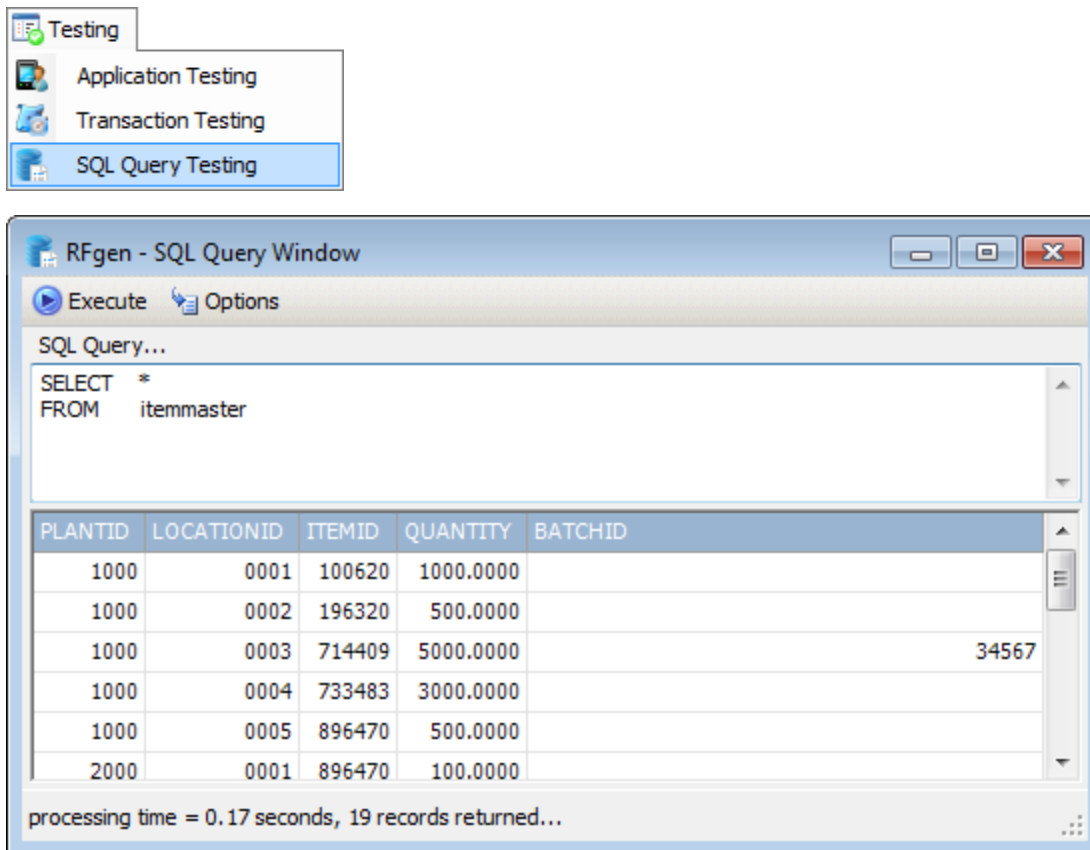


Note: Although all tables can be downloaded from a data source connection, best practices dictate that a developer should only download those tables that will be used/updated by an application during development. Tables that are not within the

scope of the development effort generally do not need to be downloaded, since their structural design is of little concern to the effort at hand.

SQL QUERY WINDOW

RFSample is a Microsoft Access database which, like most other Access databases, is ANSI SQL compliant. Therefore, SQL statements can be issued to this database to execute queries. To perform a query, simply type in your query and click the “Execute” button.

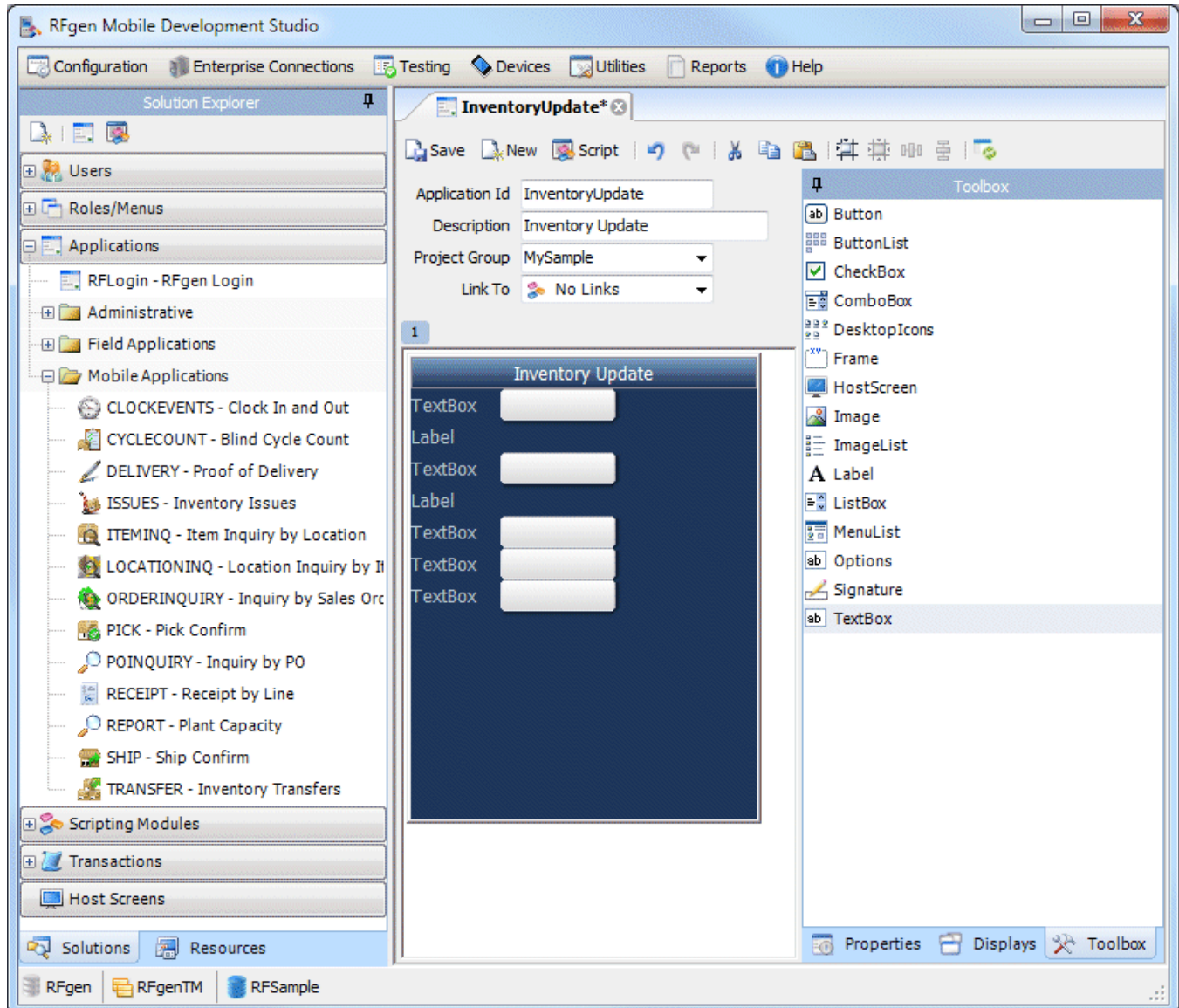


EXERCISE 1: CREATING AN APPLICATION

An application is the primary object used in RFgen to collect user input. Applications often include user input validation which serves to minimize erroneous values in the various prompts of the application. The goal of this exercise is to create a basic application that updates the inventory at a location. The “ITEMMASTER” table on the RFSample database stores the Onhand quantity for an item.

Procedure

1. In the navigation pane, right-click the “Applications” group and select “Add New Application...” from the context menu.
2. In the object design pane, provide an “Application Id”, “Description”, and “Project Group” for the new application.
3. Click on the “Caption” property under the “Heading Properties” section. Enter a title for the form.
4. Click on the “Toolbox” sub-tab. Double-click or click and drag the fields to the application to create prompts. You will need 5 textboxes and two label fields for the update inventory application.





Procedure *(continued)*

5. Arrange them to create an organized and user-friendly layout. Give each field a meaningful name. On the “Properties” sub-tab, update the following properties.
 - a. Enter the Name to be the name you want to give to each field.
 - b. The (Prompt) property represents the sequence of the fields. Provide correct numbers to the objects:
 - c. In the “Label Properties” section, update the Caption for each field.

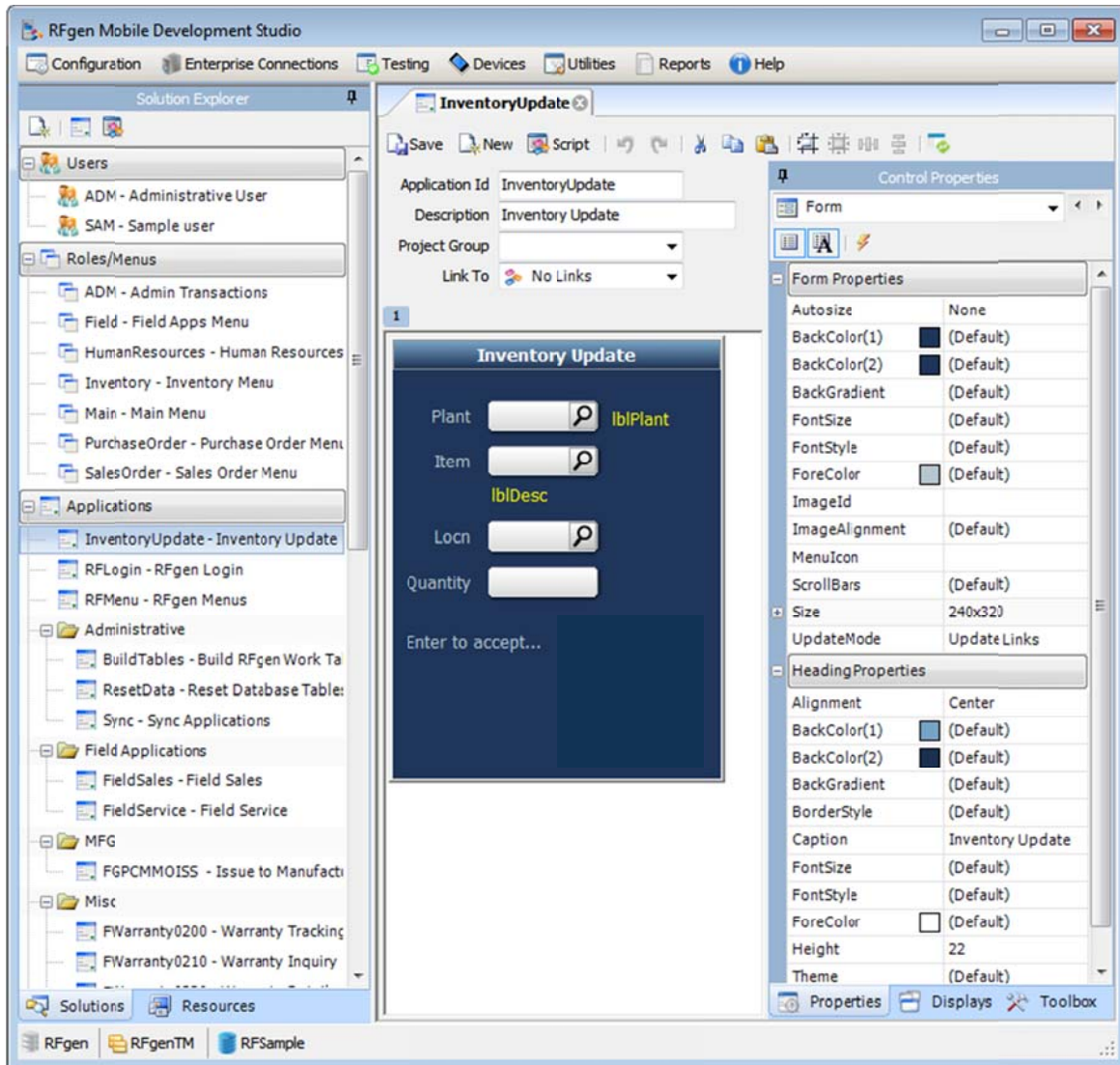
Provide these names to the objects on the screen:

Prompt	Type	Name	Caption
1	Textbox	txtPlant	Plant
2	Label	lblPlant	lblPlant
3	Textbox	txtItem	Item
4	Label	lblDesc	lblDesc
5	Textbox	txtLocn	Locn
6	Textbox	txtQty	Quantity
7	Textbox	txtAccept	Enter to accept...

6. Select each prompt and use the border around the selected prompt to resize the prompt. To resize, simply click and drag a border. Note that on the “Properties” sub-tab, the “Width” property can also be adjusted to change the width of each prompt.
7. Select the control portion (not label) of the “txtPlant” prompt. Click on the “Properties” sub-tab. Change the value of the “Width” property to “125”.
8. Hold down the “Ctrl” key and click to select the control portion of the following prompts in this order: “txtPlant”, “txtItem”, “txtLocn”, and “txtQty”. In the toolbar, click on the sizing options button  and select the “Make Same Width” entry.
9. Clear the selection by clicking in the blank space on the form and the hold down the “Ctrl” key and click to select the control portion of the following prompts in this order:

“txtPlant”, “txtItem”, “txtLocn”, and “txtQty”. In the toolbar, click on the sizing options button  and select the “Align Right” entry.

10. Finally, click the “Save” button in the toolbar. Make any additional adjustments you wish to match your form to the one shown below.

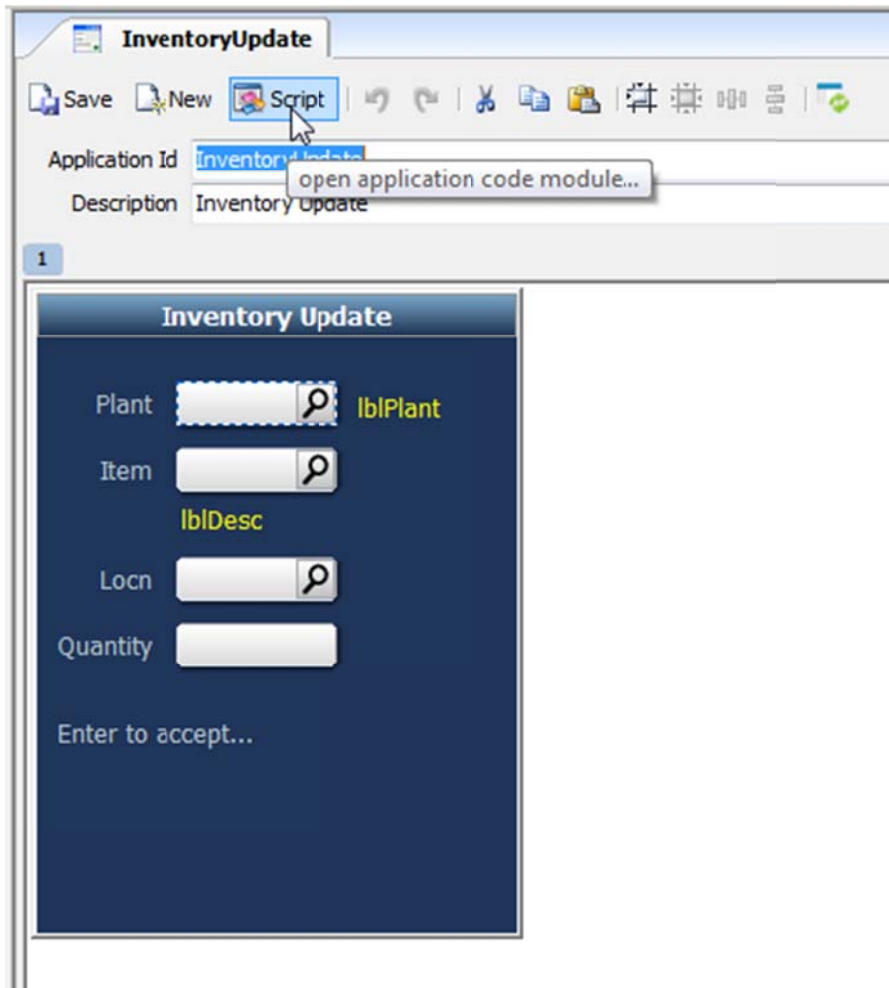


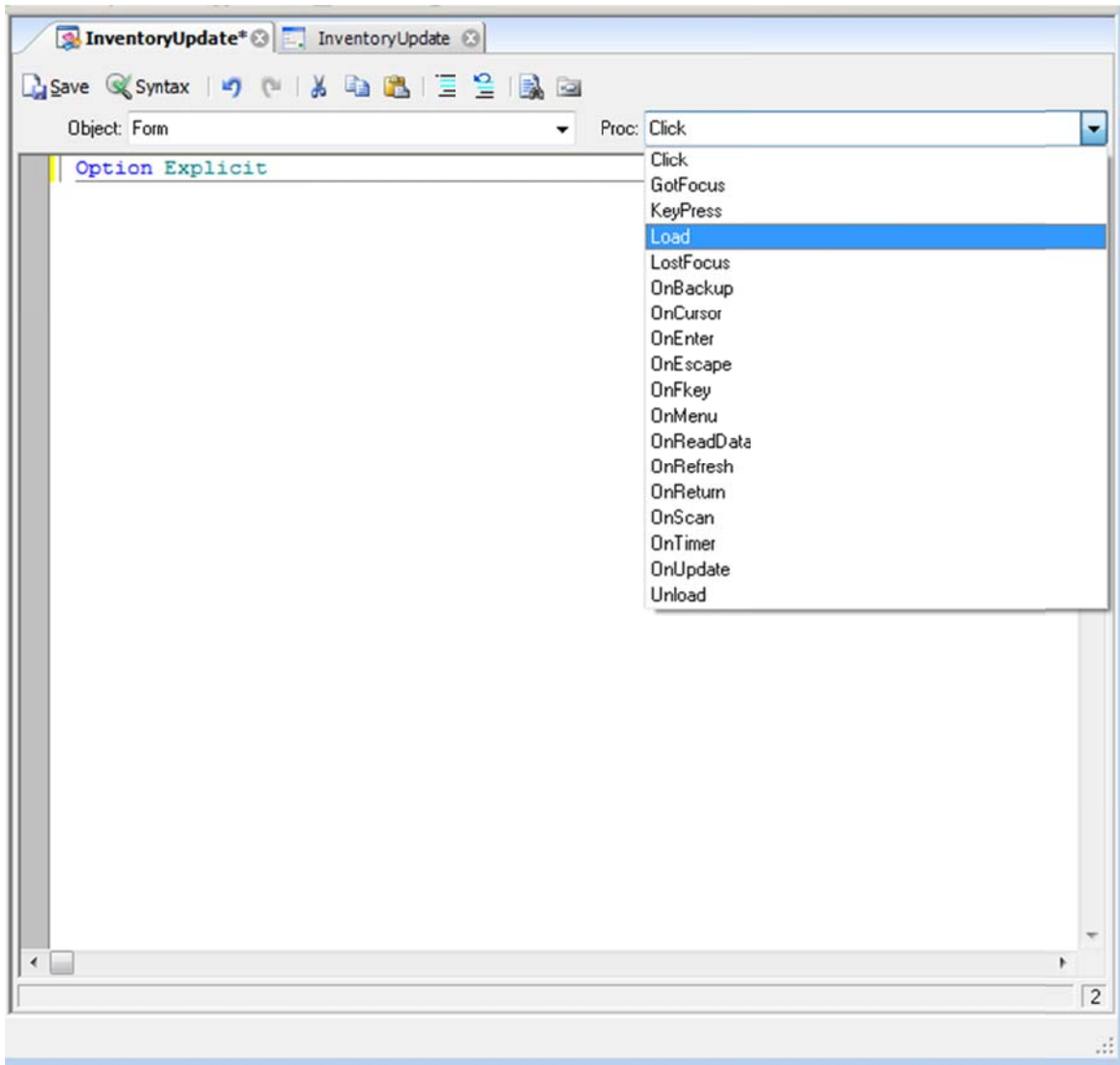
EXERCISE 2: SCRIPTING THE APPLICATION

All the business logic for the mobile app is handled by adding VBA script for displaying and updating data. Each prompt on the application, and the application itself, has associated scripting for the different events. Application events take precedence over prompt events; e.g., events linked to the application will occur prior to the event firing for a prompt.

Procedure

1. In the Form Designer pane, select the “Script” button. This will open the scripting window.
2. Select the “Object” from the drop down for which you want to add a validation. This includes each prompt on the application and the form itself.
3. Select an associated event from the “Proc” dropdown.
4. After scripting, click on the “Syntax” option which checks the script for any syntax errors.
5. Click on “Save” to save the script.





Application Code

Sample Code for Inventory Update Mobile Application

Form Load Event

```
Private Sub Form_Load()  
    On Error Resume Next  
        lblPlant.Caption = ""  
        lblDesc.Caption = ""  
        txtAccept.Visible = False  
End Sub
```

Plant Validation

```
Private Sub txtPlant_OnEnter(ByRef Rsp As String, ByRef Cancel As Boolean, ByRef ErrMsg As String)

    On Error Resume Next

    Dim sName As String

    Call Validate_Plant(Rsp, sName)

    If sName = "" Then

        ErrMsg = "Plant not found."

        Cancel = True

        Exit Sub

    End If

    lblPlant.Caption = sName

End Sub
```

You will notice that the Validate_Plant function is called. It has already been written and is stored globally in the RFgen.Bas Scripting Module section. Other functions from here will also be used throughout this example.

Plant Search

```
Private Sub txtPlant_OnSearch(ByRef Rsp As String, ByRef Cancel As Boolean)

    On Error Resume Next

    Dim sList As String

    sList = DB.MakeList("select * from Plants")

    Rsp = App.ShowList(sList)

    Cancel = (Rsp = "")

End Sub
```

Item Validation

```
Private Sub txtItem_OnEnter(ByRef Rsp As String, ByRef Cancel As Boolean, ByRef
ErrMsg As String)

    On Error Resume Next

    Dim sDesc As String

    Call Validate_Item(Rsp, sDesc)

    If sDesc = "" Then

        ErrMsg = "Item not found."

        Cancel = True

        Exit Sub

    End If

    lblDesc.Caption = sDesc

End Sub
```

Item search

```
Private Sub txtItem_OnSearch(ByRef Rsp As String, ByRef Cancel As Boolean)

    On Error Resume Next

    Dim sList As String

    sList = DB.MakeList("select * from Items")

    Rsp = App.ShowList(sList)

    Cancel = (Rsp = "")

End Sub
```

Location Search

```
Private Sub txtLocn_OnSearch(ByRef Rsp As String, ByRef Cancel As Boolean)

    On Error Resume Next

    Dim sList As String

    sList = DB.MakeList("select * from Locations")

    Rsp = App.ShowList(sList)

    Cancel = (Rsp = "")

End Sub
```

Location Validation

```
Private Sub txtLocn_OnEnter(ByRef Rsp As String, ByRef Cancel As Boolean, ByRef
ErrMsg As String)

    On Error Resume Next

    Dim sName As String

    Call Validate_Locn(Rsp, sName)

    If sName = "" Then

        ErrMsg = "Location not found."

        Cancel = True

        Exit Sub

    End If

End Sub
```

Quantity Validation

```
Private Sub txtQty_OnEnter(ByRef Rsp As String, ByRef Cancel As Boolean, ByRef ErrMsg As String)

    On Error Resume Next

    If IsNumeric(Rsp) = False Then

        ErrMsg = "Quantity must be a number."

        Cancel = True

        Exit Sub

    End If

End Sub
```

Enter to accept...

```
Private Sub txtAccept_GotFocus(ByRef Rsp As String, ByRef AllowChange As Boolean)

    On Error Resume Next

    txtAccept.Visible = True

End Sub

Private Sub txtAccept_LostFocus()

    On Error Resume Next

    txtAccept.Visible = False

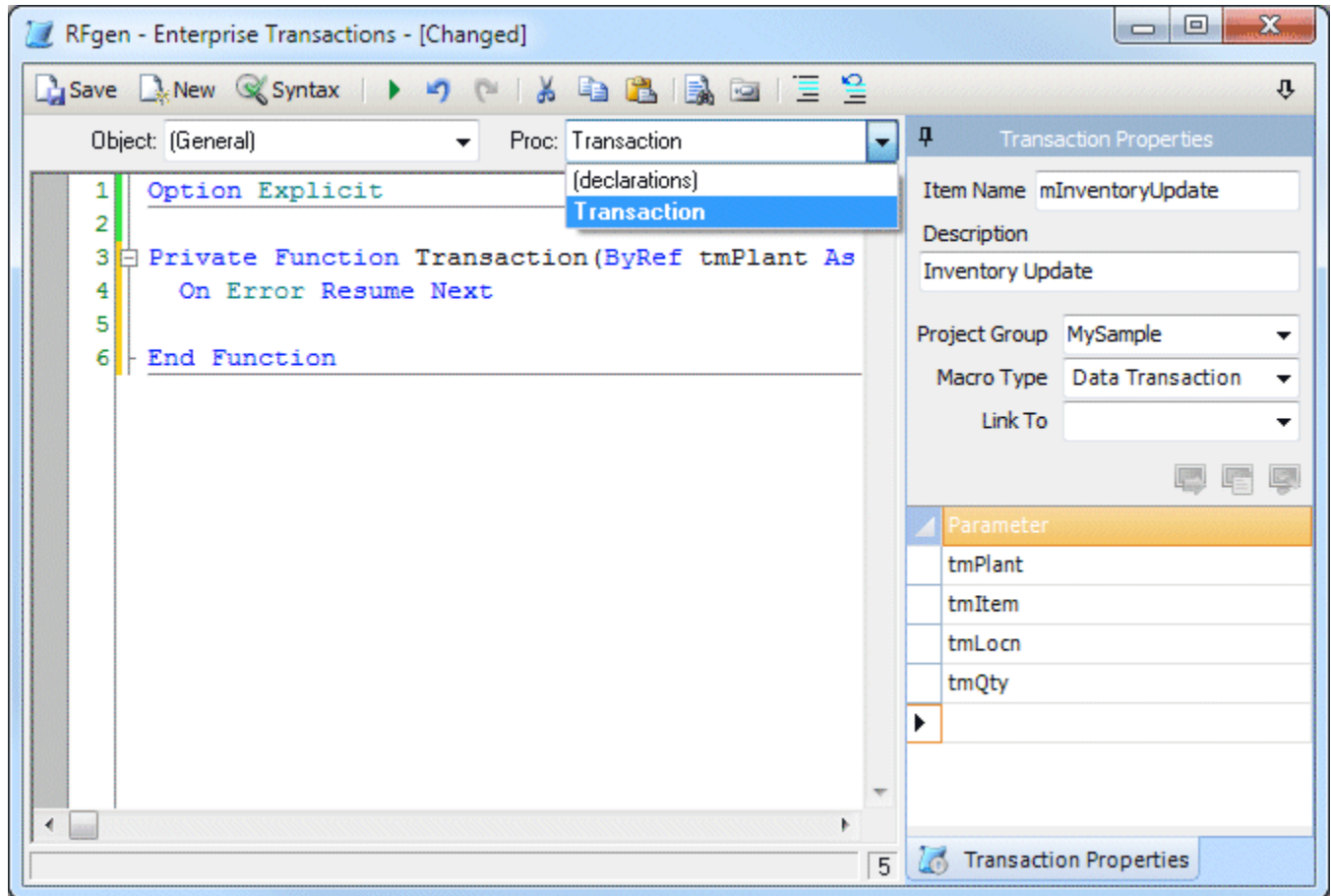
End Sub
```

EXERCISE 3: CREATING A TRANSACTION MACRO

All the business logic for updating the backend system is typically done in a Transaction Macro rather than in the form. This is for separating the user collecting the data from the logic required to update the system. This provides better organization and the ability to queue the update without making the user wait for the backend system to process the transaction.

Procedure

1. In the navigation pane, right-click the “Transactions” group and select “Add New Transaction...” from the context menu. This would open the Transaction script window.
2. In the object design pane, provide an “Item Name”, “Description”, and “Project Group” for the new application.
3. In the Parameters section, enter the parameters you need to pass for updating inventory.
4. From the “Proc” dropdown, select the “Transaction” event and write the logic to update the inventory.
5. After scripting, click on the “Syntax” option which checks the script for any syntax errors.
6. Click on “Save” to save the script.



Macro Code

Sample code for updating inventory in ITEMMASTER table:

```
Private Function Transaction(ByRef tmPlant ... ) As Boolean

    On Error Resume Next

    Dim sSQL As String

    Dim vResult As Variant

    sSQL = "Update ItemMaster set Quantity = " & tmQty & _
        " where PlantID = '" & tmPlant & "' and " & _
        "ItemID = '" & tmItem & "' and " & _
        "LocationID = '" & tmLocn & "'"

    vResult = DB.Execute(sSQL)

    If vResult = 0 Then

        Transaction = True

    Else

        Transaction = False

    End If

    ' Alternate

    Transaction = (DB.Execute(sSQL) = 0)

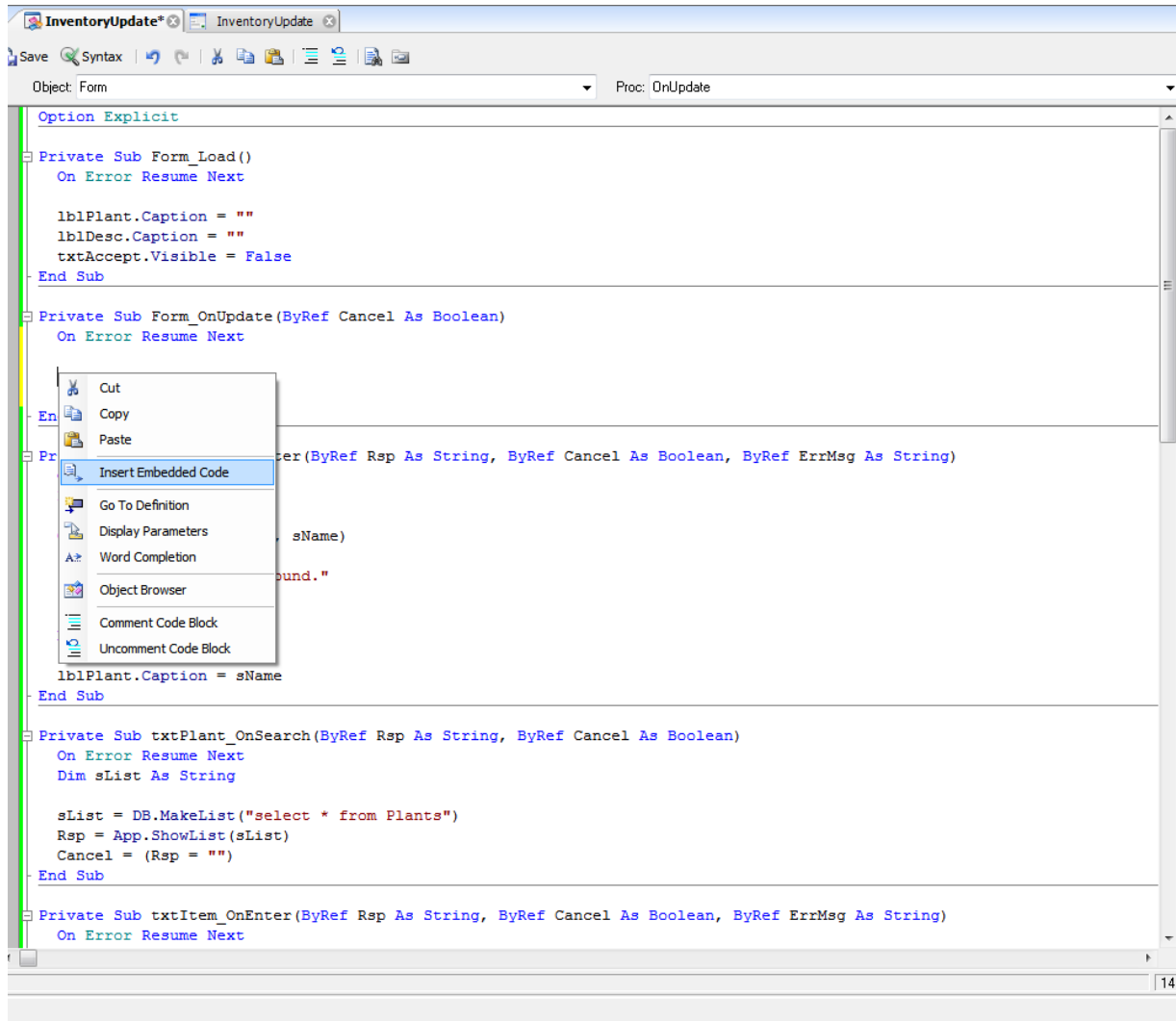
End Function
```

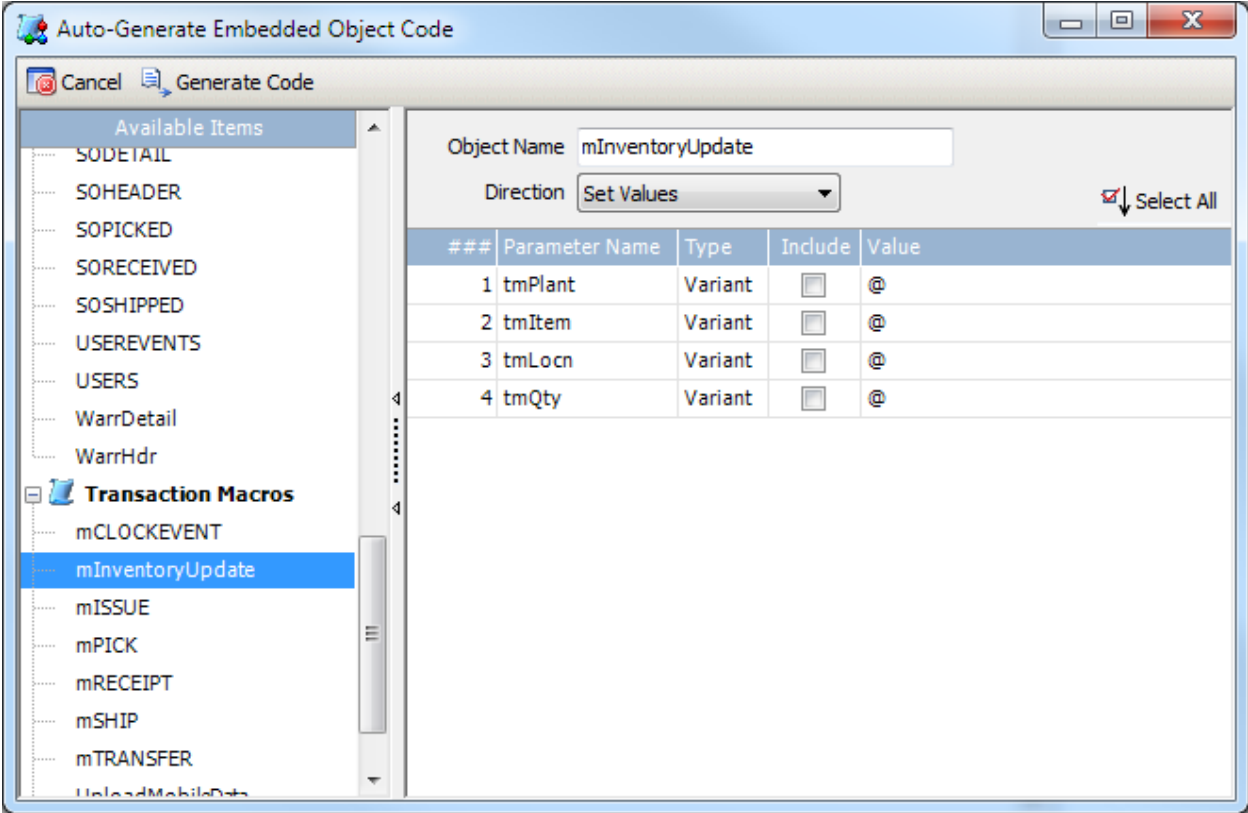
Form OnUpdate – Embed Macro

The Transaction macro needs to be called from the form when the user finishes collecting data in the form.

Procedure

1. Go back to the application script.
2. Select the “Form” object from the Object drop down. Select the “OnUpdate” event from the Proc dropdown.
3. Right click and select the option to “Insert Embedded Code”. The Auto Generate Embedded Object Code pop up window is displayed.
4. From the Available Items list, select the Transaction Macro that you just created. Select all the parameters and click on “Generate Code”.
5. Update the new code to pass the value from the application prompts.
6. After scripting, click on the “Syntax” option which checks the script for any syntax errors.
7. Click on “Save” to save the script.





The embedded code will have additional lines. Remove and replace the code to add data from the screen fields. Final code will look something like this:

```
Private Sub Form_OnUpdate(ByRef Cancel As Boolean)

    On Error Resume Next

    Dim emmInventoryUpdate As New EmbeddedProc
    '
    emmInventoryUpdate.Name = "mInventoryUpdate"
    emmInventoryUpdate.DataSource = "RFSample"
    '
    emmInventoryUpdate.Param("tmPlant") = txtPlant.Text
    emmInventoryUpdate.Param("tmItem") = txtItem.Text
    emmInventoryUpdate.Param("tmLocn") = txtLocn.Text
    emmInventoryUpdate.Param("tmQty") = txtQty.Text
    '
    If Not emmInventoryUpdate.Execute Then
        App.MsgBox("Failed to update database.")
    End If

    lblPlant.Caption = ""
    lblDesc.Caption = ""

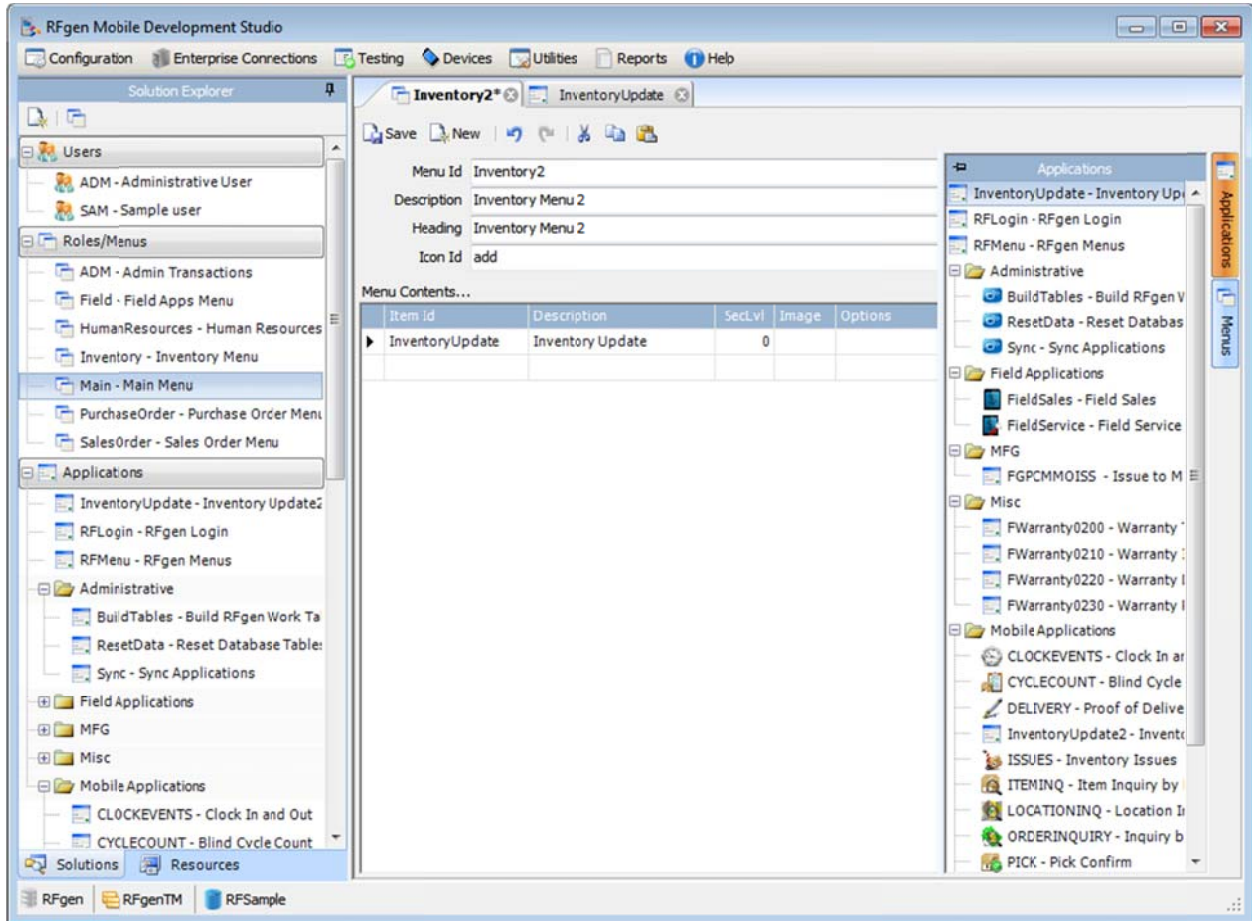
End Sub
```

EXERCISE 4: CREATING A MENU

A menu is a logical grouping or collection of applications and menus (submenus). Applications and menus can be launched from a menu. Menus usually are grouped based on functionality and are composed of similar applications and submenus. For example, there may be separate menus within an organization for Accounts Payable, Inventory, Manufacturing, and Human Resources.

Procedure

1. In the navigation pane, right-click the "Menus" group and select "Add New Menu..." from the context menu.
2. In the object design pane, provide a "Menu ID", "Description", "Heading", and "Icon Id" for the new menu.
3. To add entries to this menu, click on the "Applications" side-tab and double-click the application which you created in the previous exercise.
4. Finally, click the "Save" button in the toolbar.

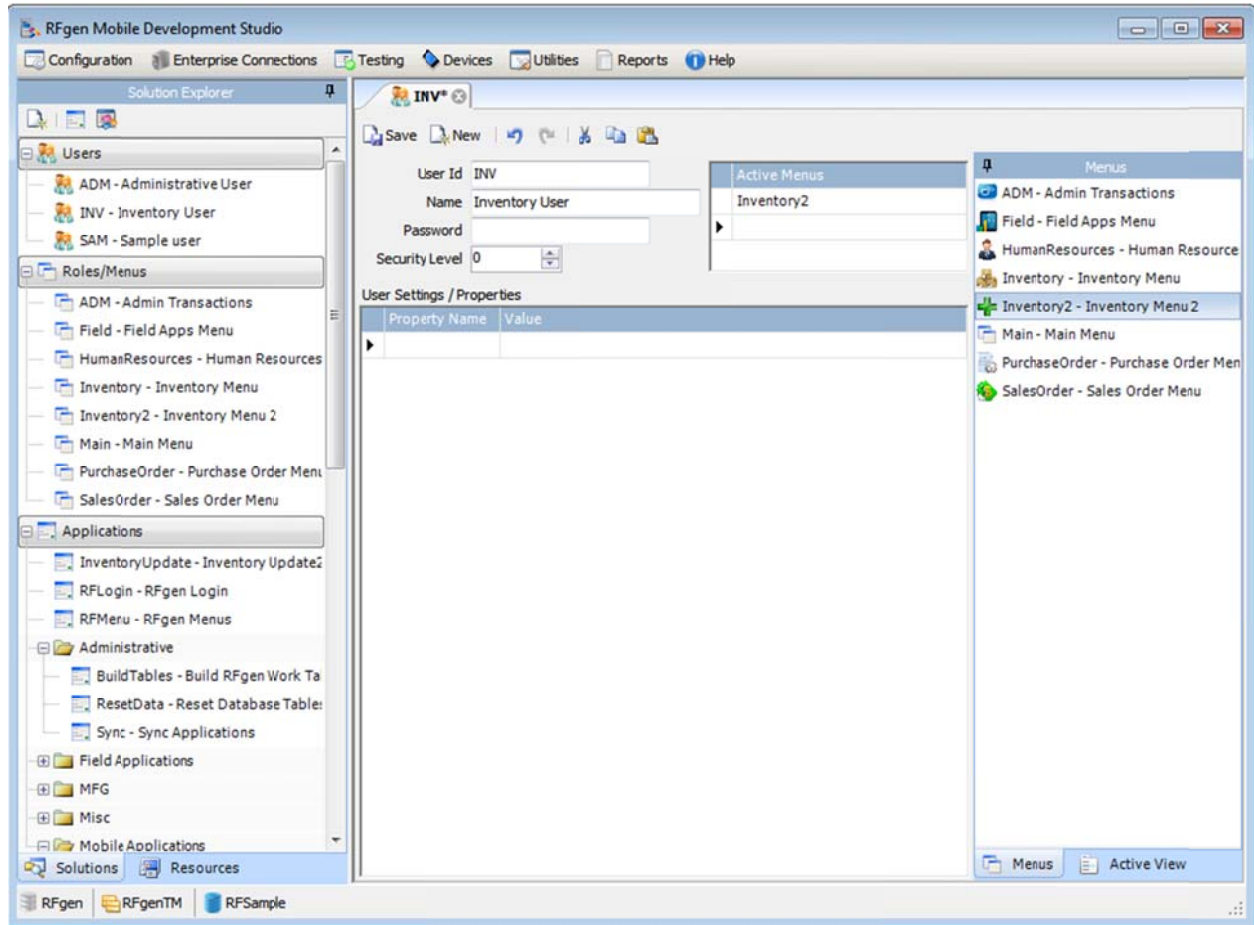


EXERCISE 5: CREATING A USER

A user is a profile that is generally created for each user that must login to use RFgen. Users can be identified by their real name or an ID number. Users should also have different startup menus based on their roles within the organization. To ensure accountability and promote security, passwords should be set for each user.

Procedure

1. In the navigation pane, right-click the “Users” group and select “Add New User...” from the context menu.
2. In the object design pane, provide a “User Id”, “Name”, and “Password” (optional) for the new user.
3. Click on the “Menus” sub-tab. Select the menu which you created in the previous exercise by double-clicking it.
4. Finally, click the “Save” button in the toolbar.

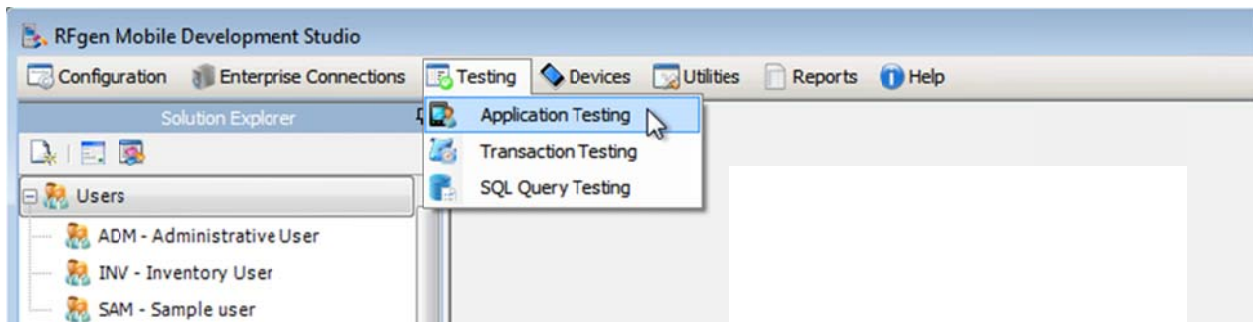


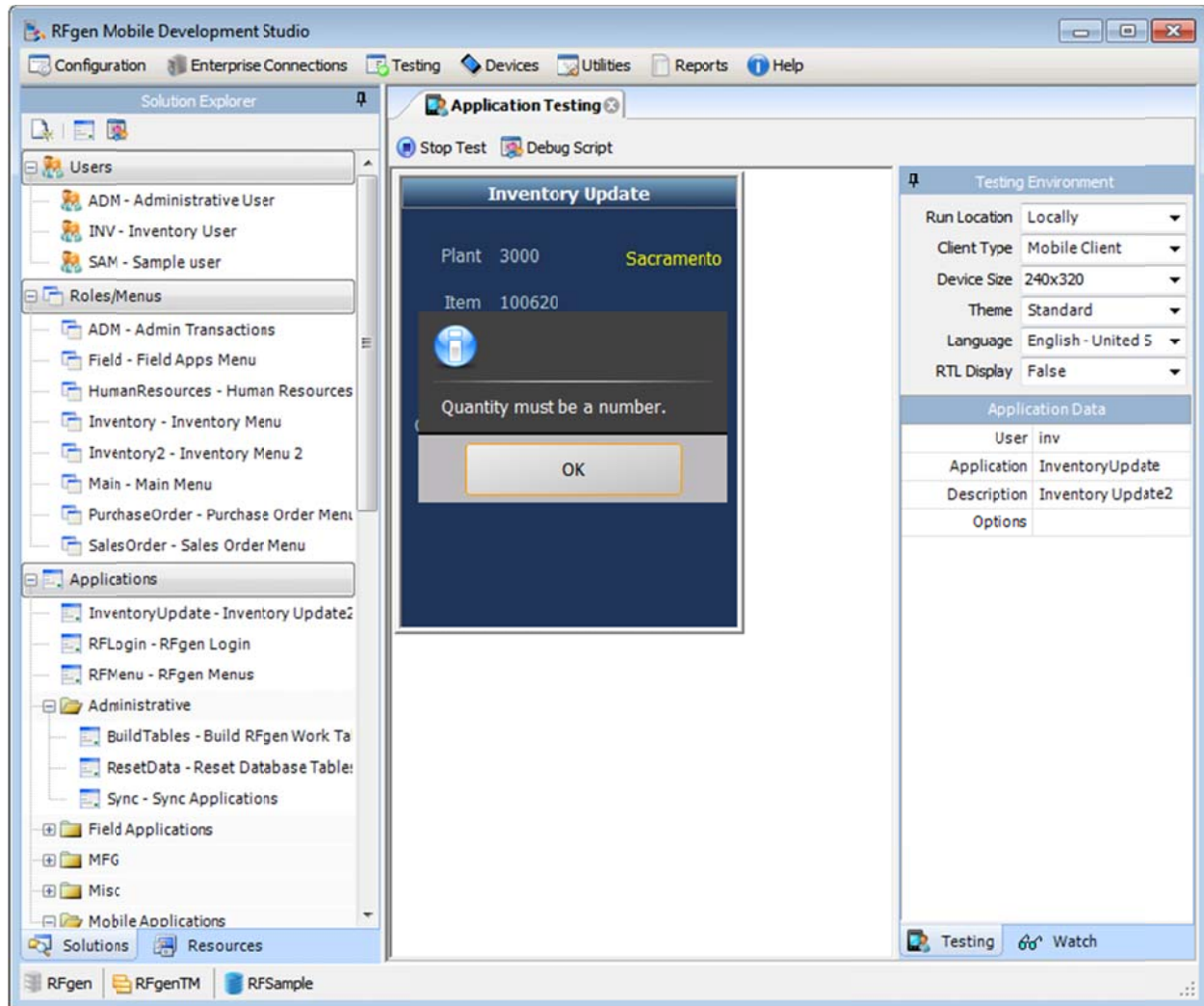
EXERCISE 6: TESTING

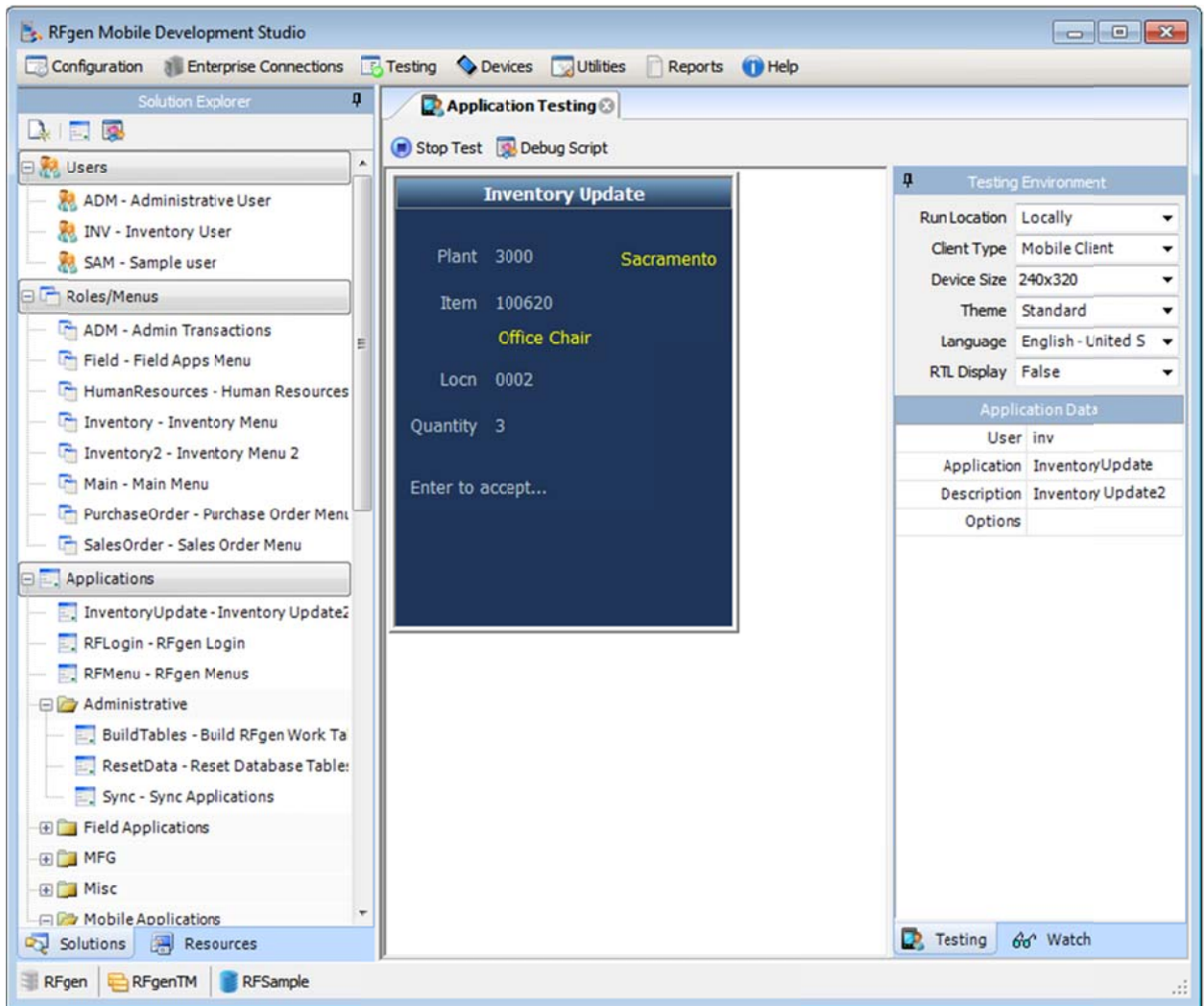
Testing should always be performed before releasing a solution to a production environment. In this exercise you will test all that has been created and configured to ensure intended functionality.

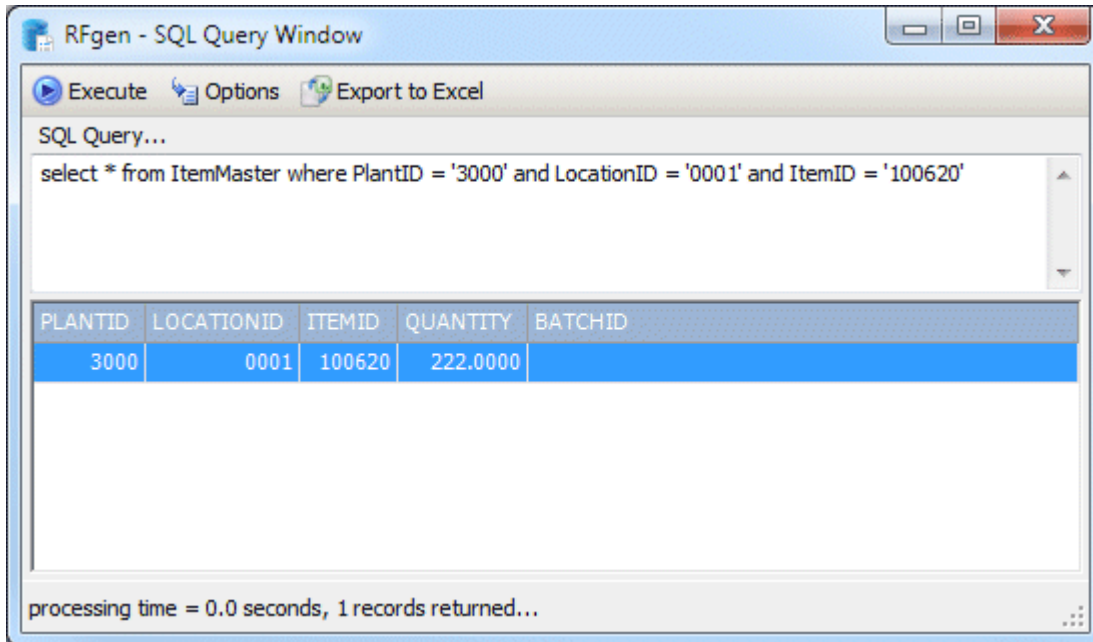
Procedure

1. In the menu bar, select “Testing” > “Application Testing”.
2. In the object design pane, click the “Start Test” button in the toolbar.
3. Login with the user you created and navigate to the application assigned to the user’s startup menu.
4. Ensure that all positions and widths for each prompt render correctly.
5. Try typing a non-numeric value in the Quantity prompt to test the validation.
6. Enter the correct values and submit the transaction and check to see if the quantity was updated for the item.









NOTE TO THE DEVELOPER

The exercises and subject matter covered in this document take a simplistic and ideal approach to development. The Mobile Development Studio provides enhanced flexibility and refined control by implementing Visual Basic for Application (VBA) script. VBA scripting can be implemented to add intricate logic for solutions that are more elaborate and demanding.