Step 12. Handle Additional Problems/Issues

1. **Handle references to variables in our old 'Main'.**
   B4A allowed Public variables in 'Main' to be referenced using 'Main.variable-name'. These now give errors. The new reference is B4XPages.MainPage.variable-name.

2. **Handle Intercepts of "Close Request"**

You may have logic to intercept the return button, for instance, to warn the user that they have not saved their updates. Note that you now have not only the Android 'Return' Key to handle but also the left arrow in the Title.

This intercept usually took the form of:

```
Sub Activity_KeyPress (KeyCode As Int) As Boolean  ' Return true to consume the event
        If KeyCode = KeyCodes.keycode_back Then  ' Back button
                ' -- check for update
                If bUpdate=False Then
                        Return False  ' allows activity to be closed
                Else
                        WarnofUpdates
                        Return True   ‘ holds the activity open
                End If
        Else
                Return True        ‘ other keys pressed
        End If
End Sub
```

In B4XPages, Erel suggests using the following. Note the return codes need to be reversed (True to False, False to True). It takes care of both the Android Return key and the left arrow in the Title Bar.

```
‘ return true to close, false to cancel
Private Sub B4XPage_CloseRequest As ResumableSub
        ' check for update
        If bUpdate=False Then
                Return True
        Else
                WarnofUpdates
                Return False
        End If
End Sub
```

3. **Similar use of intercept for handling panels**

If you use panels to display fields for update, the action of the return key as well as the new left arrow cause the underlying module to be unloaded, not just the panel. If you want to remain on the underlying module (perhaps a CLV is displaying a list of departments or categories), you can use this same intercept to test if the panel was displayed:

```
‘ return true to close, false to cancel
Private Sub B4XPage_CloseRequest As ResumableSub
        ' check for panel visible
        If pnlFS91.visible Then
                pnlFS91.visible=false
                Return False
        Else
                Return True
        End If
End Sub
```

4. Check for improper inclusion of screen dynamic modifications in Activity_Create.

In B4A if you used Activity_Finish to unload an activity, you could be sloppy and include screen builds in the Activity_Create sub. I found several instances of this in my apps. The symptom is that the first time you load the class, the screen looks correct. Subsequent loads, however, always show the same values because in B4XPages the class is not unloaded and therefore the Sub **B4XPage_Created** is never re-executed.

The fix is to move any logic that builds the screen dynamically up into the Sub **B4XPage_Appear.**

5. Eliminate a 'figment' title displaying before the new title is displayed

In B4A we generally built the activity.title in the activity itself, so the title of Activity FS20 was displayed dynamically as 'All Events', 'Upcoming Events', or 'Birthdays' depending on the menu selection.

The first time display of the B4XPages class produced a 'figment' title of 'FS20' for a fraction of a second before the new title was displayed. Furthermore, if we first looked up 'Upcoming Events' and then subsequently looked up 'Birthdays' the 'Upcoming Events' title would become the 'figment' followed by the true title.

The solution was to move the title builds into the B4XMainPage from which the call to the class is started from our Main Menu CLV. In this case the build of the title takes on the form: B4XPages.SetTitle (FS20, "All Events'), etc. for the other titles. Note: you must have used B4XPages.AddPageandCreate to load the page. Otherwise, the App will crash when you attempt to set the title from B4XMainPage.

6. Voice Recognition (VR) fires the B4XPage_Appear in a module that builds a clv

This creates the situation in FS20 where a search is done twice. The solution is:

```
Sub imvVoice_Click                ' handle VR search from the module
    ' search
    ' turn on breadevents so we don't do two readevents
    bReadEvents=True
    VR.Listen 'calls the voice recognition external activity

Private Sub imvVoice_LongClick  . handle a manual search from the module
    ' fire manual search
    bReadEvents=False             ' set off so we fire readevents on return from FS20s
    B4XPages.ShowPage ("FS20s")

Private Sub B4XPage_CloseRequest As ResumableSub
    bReadEvents=False             ' set the bReadEvents off when leaving
    Return True

Private Sub B4XPage_Appear
    If bReadEvents=False Then             ' check that not yet fired
        If Starter.bManualSearch Then
            Starter.bManualSearch=False   . handle manual search from B4XMainPage
            B4XPages.ShowPage ("FS20s")
        Else
            ' read events specified
            bReadEvents=True              ' suppress a 2nd ReadEvents
            ReadEvents
        End If
    End If

Sub clvEvents_ItemClick (Index As Int, Value As Object)
    ' sub to fire FS21 as an update
    Starter.bAdd=False
    ' set index into item
```

```
        Starter.bBookx=Index
        ' save the event id
        Starter.bId=lstBId.Get (Index)
        bReadEvents=False                ' so the return will update the clv
        B4XPages.ShowPage ("FS21")
End Sub
```

7. VR fires the B4XPage_Appear in a module that builds a form for update

This creates the situation in FS21 where a VR data entry fires the 'Appear' sub which in turn rebuilds the form to its original values. The solution is:

```
Private Sub B4XPage_CloseRequest As ResumableSub
        bBuildForm=False                ' set the bBuildForm off when leaving
        Return True

Private Sub B4XPage_Appear
        If bBuildForm=False Then        ' check that not yet fired
                bBuildForm=True         ' suppress a 2nd BuildForm
                BuildForm
        End If
```

All instances where the module is closed

```
' return to FS20
bBuildForm=False
B4XPages.ClosePage (Me)
```

8. Startup of GPS from B4XMainPage to GPS Routines in Starter did not work.

The original GPS logic from the Forum included a call from 'Main' to startup the GPS in 'Starter' as shown:

```
        GPS1.Initialize("GPS")
        If Starter.GPS1.GPSEnabled = False Then
'               ToastMessageShow("Please enable the GPS device.", True)
                StartActivity(Starter.GPS1.LocationSettingsIntent) 'Will open the relevant settings
                        screen.
        Else
                Starter.rp.CheckAndRequest(Starter.rp.PERMISSION_ACCESS_FINE_LOCATION)
                Wait For Activity_PermissionResult (Permission As String, Result As Boolean)
                If Result Then CallSubDelayed(Starter, "StartGPS")
        End If
```

The diagnosis of this was that 'Result' was never returned. This was probably due to the fact that 'B4xMainPage is unloaded when waiting for something to happen in 'Starter'.

The solution was to move the setup of GPS to B4XMainPage so that there is no wait for as so:

```
        GPS1.Initialize("GPS")
        If GPS1.GPSEnabled = False Then
                ToastMessageShow("Please enable the GPS device.", True)
                StartActivity(GPS1.LocationSettingsIntent) 'Will open the relevant settings screen.
        Else
                rp.CheckAndRequest(rp.PERMISSION_ACCESS_FINE_LOCATION)
                Wait For B4XPage_PermissionResult (Permission As String, Result As Boolean)
                If Result Then GPS1.Start(0 , 0)
        End If
```

9. Google Billing Routines in Starter called from B4XMainPage may not work.

The standard setup for Google Billing in the forum is to place a call in 'Main' to a routine starter.RestorePurchases which in-turn issues a series of 'wait-fors' to assess the status of your purchase with Google.

When testing our App the subscription was not returned. As the routines in 'Starter' were called from "Main', we first suspected the same problem as in 8 above. We moved the logic into 'B4XMainPage' as above but it still did not work. At that point we began to suspect that the routines sent the apk name which now has an "X" appended. When we transitioned back to the old name, the routines worked. See Step 1, Item 9.

However, we are not certain the 'wait for' in the calls into 'Starter' would have worked as those for GPS did not. Your choice might be to wait until you transition back and see if the routine work. If not, you may have to move them into 'B4XMainPage' as we did with the GPS routines.

10. Handle References to other modules from some modules.

As an example, the forum approach to blue tooth is to use a class module 'BlueToothManager'. It, in turn, references modules that set up the basic blue tooth and respond to state changes. The B4A approach used the module names without quotes. The B4XPages requires quotes around the module names as shown in some of the references to our FS98 and FS98B.

```
CallSub2("FS98b", "NewMessage", BytesToString(Buffer, 0, Buffer.Length, "UTF8"))
CallSub("FS98", "DiscoverFinished")
For Each Target In Array("FS98","FS98b")
        CallSub(Target, "UpdateState")
Next
```

We also note that B4XPages does not require quotes 'Starter' when routines are called from other B4XPages, so are not sure of why they are required here.