MICRO FOCUS

Microfocus VBA Add-On for Rumba+ Desktop

Quick Start Guide

Micro Focus
The Lawn
22-30 Old Bath Road
Newbury, Berkshire RG14 1QN
UK
http://www.microfocus.com

2018-08-14

Contents

About VBA Macros

Visual Basic for Applications (VBA) macros use Visual Basic to create macros. The macros have a file extension of `.vb`.

This guide introduces you to Rumba+ Desktop VBA macros.

For more information, see the Help available in the VBA Editor.

Prerequisites:

- Rumba+ Desktop 9.5 SP1 or later
- Micro Focus Rumba+ Desktop VBA Add-On

Macro Operations

In Rumba+ Desktop

The three basic macro operations are described in the main Rumba+ Help. These are:

- Recording
- Editing
- Running

In the VBA Editor

There are two main Rumba interaction object types in the VBA editor:

RumbaApplicationObject

Used to access and handle multiple emulation sessions (Rumba+ Desktop tabs).

RumbaEmulationSessionObject

Used to perform emulation session related tasks such as connecting, typing, screen text retrieving, and navigation sequence execution.

Working in Rumba+

This section describes how to create VBA macros in Rumba+ Desktop.

Basic operations using RumbaEmulationSessionObject

RumbaEmulationSessionObject is used to access and handle multiple host session tasks such as:

- Connecting
- Typing
- Retrieving screen text
- Executing navigation sequences.

For a full list of **RumbaEmulationSessionObject** methods, properties and events, see the VBA Editor Help.

Creating a RumbaEmulationSessionObject instance

1. Open a mainframe session in Rumba+.

   A **Session1 (Mainframe Display)** object is created in the project under **Rumba Objects**. This object is a **RumbaEmulationSessionObject**, which means you can access all of its methods and properties.
2. Double-click the object.
3. Type **Me.**.

   When you press the period (**.**) key, a list appears showing the available methods and properties for **RumbaEmulationSessionObject**.

Connection

This example shows how to establish a connection, setting the host name, port, then connecting:

```
Sub SetConnection()
    HostName = "csimvs"
    Port = 2023
    Connect
End Sub
```

Simple navigation

After the connection is made, you can perform some navigation steps: such as setting the cursor position, typing text, sending emulation keys, and waiting for a screen to arrive:

```
     Sub NavigateToTele()
    WaitScreen Me, "Application:", SearchOnlyAt, False,
DefaultConnectionTimeout, 3, 2, 3, 15
    SetCursorPosition 25, 7
    TypeText "cics"
    SendKey "Enter"
    WaitScreenTimeout Me, DefaultScreenTimeout
    SendKey "Clear"
    WaitScreenTimeout Me, DefaultScreenTimeout
    SendKey "Tab"
```

```
    TypeText "tele"
    SendKey "Enter"
    WaitScreen Me, "Account Number : ", SearchOnlyAt, False,
DefaultScreenDataTimeout, 6, 26, 6, 44
End Sub
```

Basic Operations Using RumbaApplicationObject

Use **RumbaApplicationObject** to access and handle multiple emulation sessions (Rumba+ tabs).

For a full list of **RumbaApplicationObject** methods, properties and events, see the VBA Editor Help.

Opening a session

When working in the VBA Editor launched from Rumba+, you have access to **RumbaApplicationObject**.

For example, to open a new AS/400 session:

```
...
    Application.CreateSession RumbaSessionType_AS400Display
...
```

or, to open a saved session profile:

```
...
    Application.OpenSessionProfile "C:\work\MF1.rsdm"
...
```

Getting the active session

Getting the active session for further work

```
...
    Dim RumbaActiveSession As RumbaEmulationSessionObject
    Set RumbaActiveSession =
Application.GetSession(Application.ActiveSessionID)
    ' further work with the session object
    RumbaActiveSession.Connect
...
```

Getting a session by its tab name

```
...
    Dim RumbaActiveSession As RumbaEmulationSessionObject
    Set RumbaActiveSession = Application.GetSessionByName("MF1")
    ' further work with the session object
    RumbaActiveSession.Connect
...
```

Getting a session by its tab index

```
...
    Dim RumbaActiveSession As RumbaEmulationSessionObject
    Set RumbaActiveSession = Application.GetSession (2)
    ' further work with the session object
    RumbaActiveSession.Connect
```

```
...
```

Events

When working under the **Application** object, you can access **RumbaApplicationObject** events.

Add the following code to the **Application** object to show a message box each time an active session opens:

```
Private Sub RumbaApplicationObject_OnOpenSession(ByVal sessionID As Long)
    MsgBox "Opened"
End Sub
```

Add the following code to the Application object to show a message box each time an active session changes:

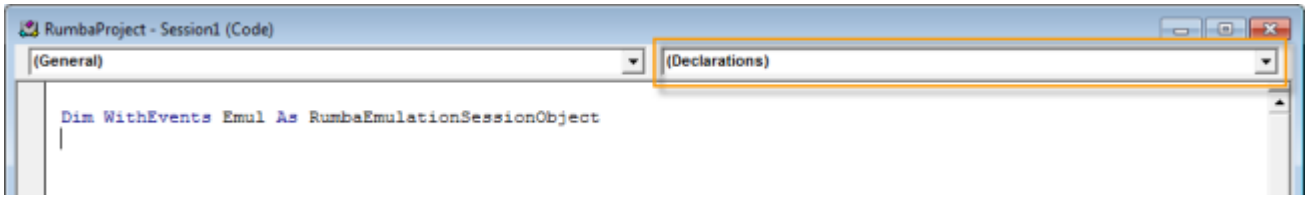
```
Private Sub RumbaApplicationObject_OnActiveSessionChange(ByVal
previousActiveSessionID As Long, ByVal currentActiveSessionID As Long)
MsgBox "Active session changed."
End Sub
```

The events defined under the **Application** object are persistent. This means that the events are triggered for any instance of Rumba (even without the VBA editor being open), and also when launched from an external application such as Microsoft Excel. See *Working with External Applications* on page 10.

More events are available using the **RumbaEmulationSessionObject** object. Defining the object using the **WithEvents** keyword provides another set of events.

```
Dim WithEvents Emul As RumbaEmulationSessionObject
```

The **Procedure** list shows the available events:



The following example shows how to use the **OnCursorPositionChange** event to display a message box each time the cursor changes position:

```
Private Sub Emul_OnCursorPositionChange(ByVal Row As Long, ByVal Column As
Long)
    MsgBox "The new cursor position is: " & Row & ", " & Column
End Sub
```

The following example shows how to use the **OnConnectionStateChange** event to display a message box each time a successful connection is established:

```
Private Sub Emul_OnConnectionStateChange(ByVal oldState As
RumbaAutomation.RumbaConnectionState, ByVal newState As
RumbaAutomation.RumbaConnectionState)
    If (newState = RumbaConnectionState_Connected) Then
        MsgBox "Connection established."
    End If
End Sub
```

Persistency

Any code written in the **Session** object is saved in the Rumba+ session profile.

Any code written in the **Application** object is saved in the `AppData\RumbaProject.apc` file.

Working with External Applications

This section describes how to create VBA macros in external applications such as Microsoft Excel.

Setting up your environment

Referencing the Rumba+ Automation Objects Library

1. From the VBA Editor menu bar, select **Tools** > **References**.

   The **References** window appears.
2. Check **Rumba Automation Objects Library** from the **Available References** list.
3. Click **OK**.

Adding RumbaGlobals and ReflectionLegacySession

To add **RumbaGlobals** and **ReflectionLegacySession**, you must first export them from the Rumba+ VBA Editor, then import them into the Microsoft VBA Editor.

1. Open the VBA Editor from Rumba+: Select **Tools** > **New Macro** > **VBA**.
2. In the **Project** explorer, expand the **Modules** list, right-click **RumbaGlobals** and select **Export File** from the pop-up menu.
3. Save the file.
4. In the **Project** explorer, expand the **Class Modules** list, right-click **ReflectionLegacySession** and select **Export File** from the pop-up menu.
5. Save the file.
6. Close this instance of the VBA editor.
7. Open the VBA Editor from Excel or another Office program: Select **Developer** tab **> Code** group **> Visual Basic**.

   Note: If the **Developer** tab is not visible, you can enable it:

   1. Select **File** > **Options** > **Customize Ribbon**
   2. Under **Customize the Ribbon and Main Tabs**, check **Developer**.
   3. Click **OK**.
8. In the VBA Editor, select **File** > **Import File**
9. Select `RumbaGlobals.bas`.
10. Click **OK**.
11. In the VBA Editor, select **File** > **Import File**
12. Select `ReflectionLegacySession.cls`.
13. Click **OK**.

Examples

Launching Rumba+ and creating a session

The following example launches Rumba+, creates a new mainframe session, sets some connection settings, then connects to the specified host:

```
Sub RunRumba()
    Dim App As RumbaApplicationObject
    Dim Emul As RumbaEmulationSessionObject
    Dim SessID As Integer

    ' Create Rumba Application object:
    Set App = CreateObject("MicroFocus.Rumba")
    'Create Mainframe Display session:
    SessID = App.CreateSession(RumbaSessionType_MainFrameDisplay)
    ' Get Emulation Session object:
    Set Emul = App.GetSession(SessID)
    ' Set connection parameters and connect:
    Emul.HostName = "csimvs"
    Emul.Port = 23
    Emul.Connect
End Sub
```

Working with an existing instance of Rumba+

This example searches an existing instance of Rumba+ and, if found, obtains the emulation object:

```
Sub UseExistingRumbaSession()
    Dim App As RumbaApplicationObject
    Dim Emul As RumbaEmulationSessionObject

    rumbaDescriptors = RunningRumbaObjectDescriptors
    If Len(Join(rumbaDescriptors)) > 0 Then
        ' A Rumba instance was found
        Set App = GetObject(rumbaDescriptors(LBound(rumbaDescriptors)))
        Set Emul = App.GetSession(App.ActiveSessionID)
        ' Perform session operations using Emul
    End If
End Sub
```

Enhanced capabilities using ReflectionLegacySession

The Micro Focus Rumba+ VBA Add-On supports a subset of the Micro Focus Reflection for IBM 14.x methods and properties. These extended methods and properties are manifested in the **ReflectionLegacySession** object, and provide additional emulation functionality, such as:

- Cursor information
- Screen Selection
- Clipboard manipulaton
- Field and text searching options
- Connection information
- Multiple screen wait options

For details of these methods and properties, see the *Reflection for IBM 14.x - Programming Reference*. For more information, see the VBA Editor Help.

This example launches Rumba+, creates a new mainframe session, sets some connection settings, then connects to the specified host. Notice how an instance of the **ReflectionLegacySession** object is created and used.

```
Dim App As RumbaApplicationObject
Dim Emul As RumbaEmulationSessionObject
Dim ReflectionSession As ReflectionLegacySession
Dim SessID As Integer

Sub RunRumbaAndGetData()
    ' Create Rumba Application object:
    Set App = CreateObject("MicroFocus.Rumba")
    'Create Mainframe Display session:
    SessID = App.CreateSession(RumbaSessionType_MainFrameDisplay)
    ' Get Emulation Session object:
    Set Emul = App.GetSession(SessID)
    ' Set connection parameters
    Emul.HostName = "csimvs"
    Emul.Port = 23

    ' Create a Reflection legacy session object and connect
    Set ReflectionSession = CreateReflectionLegacySession(Emul, App)
    ReflectionSession.Connect
End Sub
```

Passing data from the screen to Excel

The following example launches Rumba+, opens a session, connects, then reads data from the screen and sets it to a range of cells in an Excel sheet:

```
Dim App As RumbaApplicationObject
Dim Emul As RumbaEmulationSessionObject
Dim ReflectionSession As ReflectionLegacySession
Dim SessID As Integer

Sub RunRumbaAndGetData()
    Set App = CreateObject("MicroFocus.Rumba")
    SessID = App.CreateSession(RumbaSessionType_MainFrameDisplay)
    Set Emul = App.GetSession(SessID)
    Emul.HostName = "csimvs"
    Emul.Port = 23
    Set ReflectionSession = CreateReflectionLegacySession(Emul, App)
    ReflectionSession.Connect

    ' Get data from the emulation screen to the Excel cells
    ReflectionSession.WaitForDisplayString "Application", "10", 3, 2
    GetData
End Sub

Sub GetData()
    Set Durations = Range("A1:A3")
    For i = 1 To 3
        DataText = ReflectionSession.GetDisplayText(i + 4, 10, 7)
        Set Duration = Durations.Cells(i, 1)
        Duration.Value = DataText
    Next
End Sub
```