Brekeke PBX Active Library

Version 1.0.0.2

Developer’s Guide

Brekeke Software, Inc.

Version

Brekeke PBX Active Library v1.x Developer's Guide

Revised July 30, 2007

Disclaimer

Brekeke Software, Inc. reserves the right to change any information found in this document without any written notice to the user.

Trademark Acknowledgement

- *LINUX is a registered trademark of Linus Torvalds in the United States and other countries.*
- *Red Hat is a registered trademark of Red Hat Software, Inc.*
- *Windows is a trademark or registered trademark of Microsoft Corporation in the United States and other countries.*
- *Mac is a trademark of Apple Computer, Inc., registered in the U.S. and other countries.*
- *Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries.*
- *Other logos and product and service names contained in this document are the property of their respective owners.*

Table of Contents

1. Brekeke PBX Active Library

Brekeke PBX Active Library (PAL) is a Windows Control Library that allows the creation of companion applications for the Brekeke PBX. Some possible applications that may be developed with the Brekeke PAL include Operator Consoles, Speed Dial Panels, Phone Call Recorders, and Line Status Panels.

1.1 What's New in BETA Version 1.0.0.2

The following changes were introduced since the last version release (BETA 1.0.0.0):

- This version REQUIRES Brekeke PBX Version 2.1.1.3 or later.
- The pal.properties file is no longer used.
- To configure Brekeke PAL, set the values for new control properties: localPort, pbxAddress, pbxPort, and webService. Read all about Brekeke PAL properties in Section 2.5.
- The new initialize() method MUST be called.

1.2 System Requirements

Development with Brekeke PAL requires the following:

- Windows Platform
- Brekeke PBX Version 2.1.1.3 or later, Evaluation License or Standard License with PAL Option enabled
- Java Runtime Environment (JRE) Version 1.5
- Visual Studio 2005 or Visual Express for Development (.NET development)

✓ *Note: The JRE is still needed even though we are developing a Windows Application. Brekeke Software is built using Java. The Brekeke PAL component uses a Java ActiveX Control Bridge mechanism to communicate with the PBX software.*

1.3 Installation

Prepare the development environment as follows:

1. Download and install the Java Runtime Environment Version 1.5 if the development system does not already have it.
2. Uninstall any previous version of Brekeke PAL (see Section 1.4 below).
3. Install Brekeke PBX (see Section 1.2 for Version requirement). The Brekeke PBX software does not have to be installed on the same machine as the development environment, but the Brekeke PAL Component must be able to interact with the PBX software via network connections. Note down the IP Address of the machine where the Brekeke PBX Software was installed.
4. Run the PAL Installer, setup.exe. The installer will copy all the required files to c:\program files\brekeke\pal and <jre_home>axbridge\bin subdirectories. The dlls will also be automatically registered.

1.4 UnInstall

If, for some reason, the Brekeke PAL needs to be uninstalled:

1. Go to the Windows Control Panel.
2. Select Add or Remove Programs.
3. Remove the Brekeke PAL.

2. Developing with Brekeke PAL

Creating a companion application for the Brekeke PBX is now very simple. Read completely through Sections 2.1 to 2.8 for the basic development steps. Read Section 3 to learn about all the PAL Control's capabilities.

2.1 Verify your Brekeke PBX and IP Phone Setup

Applications developed using Brekeke PAL are usually software companions to desktop phones or softphones. Make sure your desktop IP phones or softphones are registered with the Brekeke PBX. Please consult the proper Brekeke Manuals if you need help with this. Place some test calls to make sure the phone system and the Brekeke PBX are running properly.

2.2 Configure Web Service Security at the Brekeke PBX

The PAL Control makes use of the built-in Brekeke Web Service which is controlled using the PBX Administration Tool. Only Clients with IP Addresses that match the regular expression pattern defined in the PBX Administration Tool will be allowed to consume from the web service.

To define the valid client ip addresses:

1. Browse to the PBX Administration tool.
2. Login with Admin privileges.
3. Select Options from the menu.
4. Enter a regular expression for the text box labeled "Valid Client IP Pattern". For example, "192.168.*.*"

2.3 Add PAL to your Visual Studio Component ToolBox

To add PAL to your project toolbox:

1. Select Choose Toolbox Items from the Tools menu or right-click on the Toolbox and select Choose Items from its shortcut menu.
2. Select and add the Brekeke PAL Control by browsing for the PAL.dll file in the c:\program files\brekeke\PAL subdirectory.

2.4 Add the PAL Control to the Project Form

To create an instance of the Brekeke PAL Control:

1. Open the project's form in the Forms Designer. In the Toolbox, you will see the new PAL control.
2. Drag the PAL control onto your form. An instance of the control is created and added to the Component Tray. The Brekeke Logo will appear on your form. This logo can be hidden by changing the hide property of the control to TRUE. The default control name will be Pal1.

2.5 Set PAL Control Properties

Customize the control by setting some properties. Right click on the Brekeke PAL control and select “Properties” to see the properties sheet for the control.

2.5. 1 TimerInterval Property

Set the timerInterval under the Misc tab of the properties for PAL1. This step is optional. This property controls the polling rate to detect notification events. The default setting is 250 milliseconds.

2.5.2 LocalPort

The port on the local machine through which Brekeke PAL will communicate with the Brekeke PBX Server.

2.5.3 PbxAddress

The IP Address of the Brekeke PBX Server.

2.5.4 PbxPort

The port number used by the Brekeke PBX Server.

2.5.5 WebService

The URL of the Brekeke PBX Web Service. The Brekeke PBX Web Service is usually in http://<host>/pbx/services/pal where <host> is the host server name or host ip address of your Brekeke PBX Server.

2.6 Initialize the Control

After configuring the Control’s properties, the control must be initialized prior to calling any other methods or functions. The Brekeke PAL Control is initialized by calling the “initialize” function.

2.7 Subscribe for Notifications

Brekeke PAL uses a subscribe-notify mechanism to receive information from the PBX. In your form's Load event, add call the subscribe function (see Section 3.X).

There are two types of notifications that you can subscribe for:

- Line status notifications will allow you to see the state of a phone
- Voicemail notifications will allow you to see if voicemail is available

For example,

```
Dim UserArray() As String = {"4001","4007","4008"}

Private Sub Form1_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.Load
        PAL1.subscribeStatus("4008", UserArray, 60 )
        PAL1.subscribeVmail("4008", UserArray, 60)
End Sub
```

The first parameter is your desktop extension. The second parameter is a string array of extension numbers for which you want to receive notifications about line status and voice mail status. The last parameter indicates how many seconds before the subscription expires (in seconds).

2.8 Monitor events

If your subsciption was successful, Brekeke PBX will start sending notification events to the PAL control. PAL will generate notify_status events for line status notification events. For voicemail notification events, PAL will generate notify_mail events. Monitor for these events to display line status and voicemail changes.

For example,

```
'we received a voicemail notification
Private Sub Pal1_notify_vmail(ByVal user As String) Handles Pal1.notify_vmail
    If (user = AppUser) Then
        vmailCountLabel.Text = Pal1.vmailCount(user)
    End If
End Sub
```

2.9 Complete Your Application

Use the functions of the Brekeke PAL control to create your application. Please see the next section for a full description of all available functions.

2.10 Terminology

This section explains some terms used in PAL function calls and development.

2.10.1 Talker ID

When one agent calls another agent, each agent is considered a talker and is assigned a Talker ID (TID). Each agent can be involved in multiple conversations, hence each can have multiple Talker IDs.

2.10.2 Room ID

When one agent calls another agent; conceptually, a room is created and the talkers are placed in the room. Each room is assigned a Room ID.

3. The PAL Control

This section is a full description of the PAL Windows Control Library, including properties, events, and functions.

3.1. Properties

timerInterval - This is the number of milliseconds between polls for line status and voicemail status events. The default setting is 250 milliseconds.

3.2 Events

3.2.1 notify_status(ByVal user As String, ByVal talkerID As Integer)

Raised when a line status event has been detected. User is the extension for which the event was raised. The talkerID corresponds to the user.

3.2.2 notify_vmail(ByVal user As String)

Raised when a line status event has been detected. User is the extension for which the event was raised.

3.3 Functions

This section contains descriptions of all the supported functions.

3.3.1 announce(ByVal user As String, ByVal talkerid As Integer) As String

Description: Allows an announcement to be made to talkers in a conversation. The Announcer is the user given as the first parameter. The announcement will be audible to all talkers in the same room as the talker identified by the talkerid parameter. The announcer is able to announce, but will not be able to listen to the conversation between talkers.

Parameters:

user – extension number of the user making the announcement

talkerid – an integer representing the TalkerID of the user to whom we want to make the announcement.

Returns: A success message or an error message.

Related Functions: barge, monitor

3.3.2 attendedTransfer(ByVal user As String, ByVal talkerid As Integer) As String

Description: Peform an attended transfer from the talker identified by parameter talkerid to the extension identified by the parameter user.

Parameters:
user – the extension to transfer the call to
talkerid – the talkerid of the talker whose conversation will be transferred. Usually this one of the Talker IDs assigned to the CoAct control's User.

Returns: A success message or an error message.

Related Functions: blindTransfer, cancelTransfer

3.3.3 barge(ByVal user As String, ByVal talkerid As Integer) As String

Description: Allows a user to barge into a conversation. The barging user is the extension specified in the user parameter. The barging user will be able to listen and speak to all talkers in the conversation.

Parameters:
user – the extension of the user who will be barging into the conversation
talkerid – an integer representing the TalkerID of one of the users whose conversation will be barged into

Returns: A success message or an error message.

Related Functions: announce, monitor

3.3.4 blindTransfer(ByVal user As String, ByVal talkerid As Integer) As String

Description: Perform a blind transfer from the talker identified by parameter talkerid to the extension identified by the parameter user.

Parameters:
user – the extension to transfer the call to
talkerid – the talkerid of the talker whose conversation will be transferred. Usually this one of the Talker IDs assigned to the CoAct control's User.

Returns: A success message or an error message.

Related Functions: attendedTransfer, cancelTransfer

3.3.5 callEnd(ByVal talkerid As Integer) As String

Description: End a call

Parameters:
talkerid – the talkerid
Returns: A success message or an error message.

3.3.6 callForward(ByVal roomid As String, ByVal userlist As String) As String

Description: Forward a conversation to a list of users

Parameters:
roomid – RoomID assigned to the conversation
userlist – space separated list of user extensions to whom the conversation will be forwarded

Returns: A success message or an error message.

Related Functions: getRoomID

3.3.7 getParkerIDbyCallee(ByVal user As String, ByVal callee As String) As Integer

Description: Returns the parked talkerID for the user. This function is useful if the extension of th other talker is known.

Parameters:
user – the user who parked the call
callee – the user extension of the other talker

Returns: An integer representing a talker ID. On error, this returns a -1.

Related Functions: totalTalkerIDs

3.3.8 getParkerIDs(ByVal user As String) As String

Description: Returns a list of space separated parked talker ids for the parameter user

Parameters:
user – the user for whom we want to retrieve the talker IDs

Returns: A string of space separated talker IDs. When no talker IDs are assigned, this returns the empty string.

Related Functions: totalTalkerIDs

3.3.9 getParkNumber(ByVal user As String, ByVal talkerid As Integer) As String

Description: Returns the number to retrieve a parked call corresponding to the talkerid

Parameters:
User – extension numbe of user who parked the call
talkerid – the talkerid of the parked call

Returns: A string representing the retrieve number for picking up a parked call. When no retrieve numbers are available, this returns an empty string.

Related Functions: park, parkPickup, parkCancel, getParkNumbers

3.3.10 getParkNumbers(ByVal user As String) As String

Description: Returns all retrieve numbers that are assigned to the user

Parameters:
User – extension number of user who parked the call

Returns: A string representing the retrieve numbers for picking up a parked call. When no retrieve numbers are available, this returns an empty string.

Related Functions: park, parkPickup, parkCancel, getParkNumber

3.3.11 getRoomID (ByVal talkerID As Integer) As Integer

Description: Returns the ID number assigned to the conversation.

Parameters:
talkerID – the talker ID of one of the talkers in the room

Returns: An integer representing a room ID. On error, this returns a -1.

Related Functions: callforward

3.3.12 getRoomIDbyCallee(ByVal user As String, ByVal callee As String) As Integer

Description: Returns the ID number assigned to the conversation. This function is useful if the extension number of the other talker is known.

Parameters:
user – the extension number of one of the talkers in the room
callee – the extension number of the other talker

Returns: An integer representing a room ID. On error, this returns a -1.

Related Functions: callforward

3.3.13 getTalkerIDbyCallee(ByVal user As String, ByVal callee As String) As Integer

Description: Returns the talkerid. This function is useful if the extension number of the other talker is known.

Parameters:
user – the user
callee – the extension number of the other talker

Returns: An integer representing a talker ID. On error, this returns a -1.

Related Functions: getTalkerIDs, totalTalkerIDs

3.3.14 getTalkerIDs(ByVal user As String) As String

Description: Returns a list of space separated talker ids for the parameter user

Parameters:
user – the user for whom we want to retrieve the talker IDs

Returns: A space separated string of talker IDs. If no talker IDs are assigned, this returns an empty string.

Related Functions: getTalkerID, totalTalkerIDs

3.3.15 initialize( ) As String

Description: Connects the control to the Brekeke PBX prior to other methods and function calls
Parameters: None.

Returns: Success or Error String.

Related Functions: None.

3.3.16 monitor(ByVal user As String, ByVal talkerid As Integer) As String

Description: Allows a user to listen in on a conversation. The listener is the user given as the first parameter. The listener is able to hear, but will not be able to speak to the talkers.

Parameters:
user – extension number of the user who will monitor the call
talkerid – the talkerid of one of the talkers in the conversation

Returns: A success message or an errror message.

Related Functions: barge, announce

3.3.17 park(ByVal talkerid) As String

Description: park a call.

Parameters:
talkerid – talkerId whose call is to be parked

Returns: A string representing the retrieve number or an error message.

Related Functions: parkPickup, parkCancel, getParkNumber, parkEx

3.3.18 parkCancel(ByVal talkerid) As String

Description: Cancel parking a call. This only works before the parking phone is placed back onhook. Once the phone is back onhook, the park is complete and cannot be cancelled.

Parameters:
talkerid – talkerid whose parked call is to be cancelled

Returns: A success message or an error message.

Related Functions: park, parkPickup, getParkNumber, parkEx

3.3.19 parkPickup(ByVal user As String, ByVal parkNumber As String) As String

Description: pickup a parked call

Parameters:
user – the extension number where you want to pickup the parked call
parkNumber – the retrieve number

Returns: A success message or an error message.

Related Functions: park, parkPickup, parkCancel, getParkNumber, parkEx

3.3.20 parkEx(ByVal talkerid As Integer, ByVal retriever As String) As String

Description: park a call.

Parameters:
talkerid – talkerId whose call is to be parked
retriever – the number that can be used to unpark or retrieve a parked call

Returns: A string representing the retrieve number or an error message.

Related Functions: parkPickup, parkCancel, getParkNumber, park

3.3.21 recordingStart(ByVal talkerid) As String

Description: record a conversation

Parameters:
talkerid – talkerid whose conversation will be recorded. The recording will go into this talker's voice mailbox.

Returns: A success message or an error message.

Related Functions: recordingStop

3.3.22 recordingStop(ByVal talkerid) As String

Description: stop recording

Parameters:

talkerid – talkerid of user whose recording is to be stopped

Returns: A success message or an error message.

Related Functions: recordingStart

3.3.23 showStatus(ByVal user As String) As String

Description: Show the line status for the user.

Parameters:

user – the extension number whose status is being checked

Returns: A string showing the line status for a user. On error, an error message is returned.

Possible status states are:

- Available – the line is onhook and the user is available
- Connecting – the dialed call is progressing
- Ringing – the callee's line is ringing
- Talking – the call was a success and the callee answered

The Callee is also returned. For example: Ringing 4001

Related Functions: subscribeStatus

3.3.24 subscribeStatus(ByVal user As String, ByVal user_array() As String, ByVal expireSeconds As Integer) As String

Description: Subscribe to receive notification about line status

Parameters:

user – extension where the notifications are to be sent

user_array – an array of extension numbers for which we want to receive status notifcations

expireSeconds – how long before the subscription expires in seconds

Returns: A success message or an error message.

Related Functions: showStatus, subscribeVmail

3.3.25 subscribeVmail(ByVal user As String, ByVal user_array() As String, ByVal expireSeconds As Integer) As String

Description: subscribe to receive notification about voicemail status

Parameters:

user – extension where the notifications are to be sent

user_array – an array of extension numbers for which we want to receive voicmail notifications

expireSeconds – how long before the subscription expires in seconds

Returns: A success message or an error message.

Related Functions: showStatus, subscribeStatus, vmailCount

3.3.26 threePCC(ByVal caller As String, ByVal callee As String) As String

Description: Place a call between caller and callee. The caller's line will ring first. Once the caller pick's up, then the callee's phone will ring.

Parameters:

caller – the caller's extension or phone number

callee – the callee's extension or phone number

Returns: A success message or an error message.

Related Functions: None.

3.3.27 totalParkerIDs(ByVal user As String) As Integer

Description: Returns the total number of Parked Talker IDs that are assigned to the user

Parameters:

user – the user's extension number

Returns: Number of talker IDs that are parked for the user. Returns 0 if there are none.

Related Functions: getParkerID, getParkerIDs

3.3.28 totalTalkerIDs(ByVal user As String) As Integer

Description: Returns the total number of Talker IDs that are assigned to the user

Parameters:
user – the user's extension number

Returns: Number of talker IDs assigned to a user. Returns 0 if there are none..

Related Functions: getTalkerID, getTalkerIDs

3.3.29 transferCancel(ByVal talkerid As Integer) As String

Description: Cancel an attended transfer

Parameters:
talkerid – talkerid which is in process of being transferred

Returns: A success message or an error message.

Related Functions: attendedTransfer

3.3.30 vmailCount(ByVal user As String) As Integer

Description: Get the number of voice messages

Parameters:
user – the user whose voice mail is to be checked

Returns: Number of new voice mail messages. Returns 0 when there are no messages.

Related Functions: subscribeVmail

4 Sample Program

Figure 1. Sample Program Form Layout.

The sample code, written in VB.NET, is shown below:

```
Public Class Form1

    'the demo uses three lines:  the application user and 2 team member
extensions

    'change the next three lines to match your situation
    Dim AppUser As String = "4008"  'this is the application user
    Dim Ext1 As String = "4002"     'this is one of the app user's team member
    Dim Ext2 As String = "4007"     'another team member
    Dim Ext3 As String = "4009"     'another team membe

    Dim UserArray() As String = {AppUser, Ext1, Ext2, Ext3}
    Dim VmailArray() As String = {AppUser}
```

```
    Dim AppUser_tid As Integer = -1
    Dim Ext1_tid As Integer = -1
    Dim Ext2_tid As Integer = -1
    Dim Ext3_tid As Integer = -1

    Dim x As Integer = 60

    'Set up the initial form
    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        OwnerExtension.Text = AppUser   'display the user's extension
        Extension1.Text = Ext1         'display the other extensions
        Extension2.Text = Ext2
        Extension3.Text = Ext3

        'you can still change properties here

        'initialize it
        Pal1.initialize()

        'Subscribe to receive status events for the 4 lines
        Pal1.subscribeStatus(AppUser, UserArray, 120)

        'subscribe to receive voicemail events
        Pal1.subscribeVmail(AppUser, VmailArray, 120)

        'Set the unpark number to blank
        UnparkNum.Text = ""

    End Sub

    'returns a background color to match the line status
    Private Function Status_Color(ByVal status As String) As
System.Drawing.Color
        Dim retColor As System.Drawing.Color
```

```
        'default color indicating the status is Available
        retColor = Color.PaleGreen

        'determine background color
        If (status.StartsWith("Ringing")) Then
            retColor = Color.Yellow
        ElseIf (status.StartsWith("Connecting")) Then
            retColor = Color.Yellow
        ElseIf (status.StartsWith("Talking")) Then
            retColor = Color.Pink
        End If

        Return retColor
    End Function

    'we received a status notification
    Private Sub Pal1_notify_status(ByVal user As String, ByVal talkerID As
Integer) Handles Pal1.notify_status
        If (user = AppUser) Then
            UserStatusLabel.Text = Pal1.showStatus(user)
            'change background color to show status
            UserPanel.BackColor = Status_Color(UserStatusLabel.Text)
            'save the talkerid
            If (UserStatusLabel.Text.StartsWith("Ringing") Or
UserStatusLabel.Text.StartsWith("Connecting")) Then
                AppUser_tid = talkerID
            ElseIf ((AppUser_tid = -1) And
UserStatusLabel.Text.StartsWith("Talking")) Then
                AppUser_tid = talkerID
            ElseIf (UserStatusLabel.Text.StartsWith("Available")) Then
                AppUser_tid = -1
            End If
        End If

        If (user = Ext1) Then
            ExtStatusLabel1.Text = Pal1.showStatus(user)
```

```
            'change background color to show status
            Tel1Panel.BackColor = Status_Color(ExtStatusLabel1.Text)
            'save the talkerid
            If (ExtStatusLabel1.Text.StartsWith("Ringing") Or
ExtStatusLabel1.Text.StartsWith("Connecting")) Then
                Ext1_tid = talkerID
            ElseIf ((Ext1_tid = -1) And
ExtStatusLabel1.Text.StartsWith("Talking")) Then
                Ext1_tid = talkerID
            ElseIf (ExtStatusLabel1.Text.StartsWith("Available")) Then
                Ext1_tid = -1
            End If
        End If

        If (user = Ext2) Then
            ExtStatusLabel2.Text = Pal1.showStatus(user)
            'change background color to show status
            Tel2Panel.BackColor = Status_Color(ExtStatusLabel2.Text)
            'save the talkerid
            If (ExtStatusLabel2.Text.StartsWith("Ringing") Or
ExtStatusLabel2.Text.StartsWith("Connecting")) Then
                Ext2_tid = talkerID
            ElseIf ((Ext2_tid = -1) And
ExtStatusLabel2.Text.StartsWith("Talking")) Then
                Ext2_tid = talkerID
            ElseIf (ExtStatusLabel2.Text.StartsWith("Available")) Then
                Ext2_tid = -1
            End If
        End If

        If (user = Ext3) Then
            ExtStatusLabel3.Text = Pal1.showStatus(user)
            'change background color to show status
            Tel3Panel.BackColor = Status_Color(ExtStatusLabel3.Text)
            'save the talkerid
```

```
            If (ExtStatusLabel3.Text.StartsWith("Ringing") Or
ExtStatusLabel3.Text.StartsWith("Connecting")) Then
                Ext3_tid = talkerID
            ElseIf ((Ext3_tid = -1) And
ExtStatusLabel3.Text.StartsWith("Talking")) Then
                Ext3_tid = talkerID
            ElseIf (ExtStatusLabel3.Text.StartsWith("Available")) Then
                Ext3_tid = -1
            End If
        End If

        DebugMessage.Text = Pal1.showNotify()
    End Sub

    'we received a voicemail notification
    Private Sub Pal1_notify_vmail(ByVal user As String) Handles
Pal1.notify_vmail
        If (user = AppUser) Then
            vmailCountLabel.Text = Pal1.vmailCount(user)
        End If
        'DebugMessage.Text = Pal1.showNotify()
    End Sub

    'extension events
    Private Sub CallButton1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles CallButton1.Click
        DebugLabel1.Text = Pal1.threePCC(AppUser, Ext1)
    End Sub

    Private Sub MonitorButton1_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles MonitorButton1.Click
        If (Pal1.totalTalkerIDs(Ext1) > 0) Then
            DebugLabel1.Text = Pal1.monitor(AppUser, Ext1_tid)
        End If
    End Sub
```

```
    Private Sub CallButton2_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles CallButton2.Click
        DebugLabel2.Text = Pal1.threePCC(AppUser, Ext2)
    End Sub

    Private Sub MonitorButton2_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles MonitorButton2.Click
        If (Pal1.totalTalkerIDs(Ext2) > 0) Then
            DebugLabel2.Text = Pal1.monitor(AppUser, Ext2_tid)
        End If
    End Sub

    Private Sub CallButton3_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles CallButton3.Click
        DebugLabel3.Text = Pal1.threePCC(AppUser, Ext3)
    End Sub

    Private Sub MonitorButton3_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles MonitorButton3.Click
        If (Pal1.totalTalkerIDs(Ext3) > 0) Then
            DebugLabel3.Text = Pal1.monitor(AppUser, Ext3_tid)
        End If
    End Sub

    Private Sub RecButton_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles RecButton.Click
        If (Pal1.totalTalkerIDs(AppUser) > 0) Then
            Pal1.recordingStart(AppUser_tid)
        End If
    End Sub

    Private Sub StopButton_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles StopButton.Click
        If (Pal1.totalTalkerIDs(AppUser) > 0) Then
            Pal1.recordingStop(AppUser_tid)
        End If
```

```
    End Sub

    Private Sub UnparkNum_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles UnparkNum.Click
        Pal1.parkPickup(AppUser, "33*" + UnparkNum.Text)
        UnparkNum.Text = ""
    End Sub

    Private Sub ParkButton_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles ParkButton.Click
        If (Pal1.totalTalkerIDs(AppUser) > 0) Then

            'If (x = 62) Then
            'x = 60
            'Else
            'x = x + 1 'increment the retrieve number
            'End If

            UnparkNum.Text = Pal1.parkEx(AppUser_tid, UnParkBox.Text)
            'UnparkNum.Text = Pal1.parkEx(AppUser_tid, x.ToString)
            'UnparkNum.Text = Pal1.park(AppUser_tid)

        End If
    End Sub

    Private Sub TransferButton_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles TransferButton.Click
        If (Pal1.totalTalkerIDs(AppUser) > 0) Then
            Pal1.blindTransfer(TransferNum.Text, AppUser_tid)
        End If
    End Sub

    Private Sub FwdButton_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles FwdButton.Click
        Dim result As String
        If (Pal1.totalTalkerIDs(AppUser) > 0) Then
```

```
            result = Pal1.callForward(Pal1.getRoomID(AppUser_tid),
FwdTextBox.Text)
        End If
    End Sub

    Private Sub EndButton1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles EndButton1.Click
        If (Pal1.totalTalkerIDs(Ext1) > 0) Then
            DebugLabel1.Text = Pal1.callEnd(Ext1_tid)
        End If
    End Sub

    Private Sub EndButton2_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles EndButton2.Click
        If (Pal1.totalTalkerIDs(Ext2) > 0) Then
            DebugLabel2.Text = Pal1.callEnd(Ext2_tid)
        End If
    End Sub

    Private Sub EndButton3_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles EndButton3.Click
        If (Pal1.totalTalkerIDs(Ext3) > 0) Then
            DebugLabel3.Text = Pal1.callEnd(Ext3_tid)
        End If
    End Sub

End Class
```