ITWS-5021 CFE/AFE Number Kill

Introduction

To streamline the control structure for deleting phone numbers and running dialler and SMS, kill-list processes.

1 Search for 'DeletePhoneNumber' calls and function definitions; entire system.

Searched 'C:\source' and 'C:\lowell\source\fredpay'.

Debtors-Net only

C:\source\Net2010\DebtorsNet\Controllers\Communication.vb(561): **DeletePhoneNumber**(existingNumbersForAccount.Find(Function(phoneData) phoneData.Deleteddate.HasValue = False AndAlso phoneData.Phonetype = phoneTypeToUse), pConfirmedToDeleteIfActiveOnLinked, pConfirmedToDeleteAllLinked, pIsNonInteractiveMode, opResultCode, opResultMessage) : in AddPhoneNumber() calls Overload 2.

C:\source\Net2010\DebtorsNet\Controllers\Communication.vb(695): Public **Function DeletePhoneNumber**(pOurref As Integer, pPhoneNumber As String, pConfirmedToDeleteIfActiveOnLinked As Boolean, pConfirmedToDeleteAllLinked As Byte, pIsNonInteractiveMode As Boolean, ByRef opResultCode As SavePhoneResultCodes, ByRef opResultMessage As String) As Boolean : Overload 1, calls Overload 2

C:\source\Net2010\DebtorsNet\Controllers\Communication.vb(701): Return **DeletePhoneNumber**(existingNumbersForAccount.Find(Function(phoneData) phoneData.PhoneNo.Replace(" ", "").Trim = pPhoneNumber), pConfirmedToDeleteIfActiveOnLinked, pConfirmedToDeleteAllLinked, pIsNonInteractiveMode, opResultCode, opResultMessage)

C:\source\Net2010\DebtorsNet\Controllers\Communication.vb(727): Public **Function DeletePhoneNumberByType**(pOurref As Integer, pPhoneType As String, pConfirmedToDeleteIfActiveOnLinked As Boolean, pConfirmedToDeleteAllLinked As Byte, pIsNonInteractiveMode As Boolean, ByRef opResultCode As SavePhoneResultCodes, ByRef opResultMessage As String) As Boolean : calls Overload 2

C:\source\Net2010\DebtorsNet\Controllers\Communication.vb(733): Return **DeletePhoneNumber**(existingNumbersForAccount.Find(Function(phoneData) phoneData.Phonetype = pPhoneType), pConfirmedToDeleteIfActiveOnLinked, pConfirmedToDeleteAllLinked, pIsNonInteractiveMode, opResultCode, opResultMessage)

C:\source\Net2010\DebtorsNet\Controllers\Communication.vb(759): Public **Function DeletePhoneNumber**(pAccountPhoneDataDTO As DTO.AccountPhoneData, pConfirmedToDeleteIfActiveOnLinked As Boolean, pConfirmedToDeleteAllLinked As Byte, pIsNonInteractiveMode As Boolean, ByRef opResultCode As SavePhoneResultCodes, ByRef opResultMessage As String) As Boolean : Overload 2, calls Overload 3

C:\source\Net2010\DebtorsNet\Controllers\Communication.vb(763): Return **DeletePhoneNumber**(entPhoneData, pConfirmedToDeleteIfActiveOnLinked, pConfirmedToDeleteAllLinked, pIsNonInteractiveMode, opResultCode, opResultMessage)

C:\source\Net2010\DebtorsNet\Controllers\Communication.vb(783): Friend **Function DeletePhoneNumber**(pAccountPhoneData As Entities.AccountPhoneData, pConfirmedToDeleteIfActiveOnLinked As Boolean, pConfirmedToDeleteAllLinked As Byte, pIsNonInteractiveMode As Boolean, ByRef opResultCode As SavePhoneResultCodes, ByRef opResultMessage As String) As Boolean : Overload 3, Base code

C:\source\Net2010\DebtorsNet\Controllers\CustomerFair.vb(2059): commController.**DeletePhoneNumber**(objentity, True, 1, True, 0, "") : calls Overload 3

C:\source\Net2010\DebtorsNet\DebtorRecords.vb(4135): <Obsolete("Use AddPhoneNumber/DeletePhoneNumber methods instead.", True)>

C:\source\Net2010\DebtorsNet\DebtorRecords.vb(4144): <Obsolete("Use AddPhoneNumber/DeletePhoneNumber methods instead.", True)>

C:\source\Net2010\DebtorsNet\DebtorRecords.vb(4170): <Obsolete("Use AddPhoneNumber/DeletePhoneNumber methods instead.", True)>

C:\source\Net2010\DebtorsNet\DebtorRecords.vb(4662): <Obsolete("Use AddPhoneNumber/DeletePhoneNumber methods instead.", True)>

C:\source\Net2010\DebtorsNet\DebtorRecords.vb(12634): Public **Function DeletePhoneNumberByTypeNonInteractiveMode**(pOurRef As Integer, pPhoneType As String, ByRef opResultCode As Controller.Communication.SavePhoneResultCodes, ByRef opResultMessage As String) As Boolean

C:\source\Net2010\DebtorsNet\DebtorRecords.vb(12636): DeletePhoneNumberByTypeNonInteractiveMode = contComm.**DeletePhoneNumberByType**(pOurRef, pPhoneType, True, True, True, opResultCode, opResultMessage)

C:\source\Net2010\DebtorsNet\DebtorRecords.vb(12637): If DeletePhoneNumberByTypeNonInteractiveMode = True Then

C:\source\Net2010\DebtorsNet\DebtorRecords.vb(12653): Public **Function DeletePhoneNumberByTypeFrontEnd**(pOurRef As Integer, pPhoneType As String, pConfirmedToDeleteIfActiveOnLinked As Boolean, pConfirmedToDeleteAllLinked As Byte, ByRef opResultCode As Controller.Communication.SavePhoneResultCodes, ByRef opResultMessage As String) As Boolean

C:\source\Net2010\DebtorsNet\DebtorRecords.vb(12655): DeletePhoneNumberByTypeFrontEnd = contComm.**DeletePhoneNumberByType**(pOurRef, pPhoneType, pConfirmedToDeleteIfActiveOnLinked, pConfirmedToDeleteAllLinked, False, opResultCode, opResultMessage)

C:\source\Net2010\DebtorsNet\DebtorRecords.vb(12656): If DeletePhoneNumberByTypeFrontEnd = True Then

DeletePhoneNumber() 3 overloads [Communication.vb]

1: Overload by Phone-Number, calls Overload 2 (DTO.AccountPhoneData)

```
    ''' <summary>
    ''' Deletes a phone number for an account by PhoneNumber
    ''' </summary>
    ''' <param name="pOurref"></param>
    ''' <param name="pPhoneNumber"></param>
    ''' <param name="pConfirmedToDeleteIfActiveOnLinked">Only used for interative mode. When False, returns error for user confirmation when the number to be deleted is active on other linked accounts.</param>
    ''' <param name="pConfirmedToDeleteAllLinked">Only used for interative mode. 0=raise error for user confirmation, 1=User confirmed to delete all linked numbers, 2=User confirmed NOT to delete all linked numbers</param>
    ''' <param name="pIsNonInteractiveMode"></param>
    ''' <param name="opResultCode"></param>
    ''' <param name="opResultMessage"></param>
    ''' <returns></returns>
    ''' <remarks></remarks>
    Public Function DeletePhoneNumber(pOurref As Integer, pPhoneNumber As String, pConfirmedToDeleteIfActiveOnLinked As Boolean, pConfirmedToDeleteAllLinked As Byte, pIsNonInteractiveMode As Boolean, ByRef opResultCode As SavePhoneResultCodes,
ByRef opResultMessage As String) As Boolean
        Try
            Dim existingNumbersForAccount As List(Of DTO.AccountPhoneData) = GetAccountPhoneDataByOurref(pOurref, False)
            If existingNumbersForAccount.Exists(Function(phoneData) phoneData.PhoneNo.Replace(" ", "").Trim = pPhoneNumber) Then
                Return DeletePhoneNumber(existingNumbersForAccount.Find(Function(phoneData) phoneData.PhoneNo.Replace(" ", "").Trim = pPhoneNumber), pConfirmedToDeleteIfActiveOnLinked, pConfirmedToDeleteAllLinked, pIsNonInteractiveMode,
opResultCode, opResultMessage)
            Else
                opResultCode = SavePhoneResultCodes.Error_UnableToFindThePhoneRecord
                opResultMessage = "Unable to find the number " & pPhoneNumber & " to delete"
                Return False
            End If
        Catch ex As Exception
            AddError(ex)
            opResultCode = SavePhoneResultCodes.Error_UnexpectedError
            opResultMessage = ex.ToString
        End Try
    End Function
```

2: Overload by DTO.AccountPhoneData, calls Base Overload 3 (Entities.AccountPhoneData)

```
    ''' <summary>
    ''' Deletes a phone number for an account using given DTO.AccountPhoneData
    ''' </summary>
    ''' <param name="pAccountPhoneDataDTO"></param>
    ''' <param name="pConfirmedToDeleteIfActiveOnLinked">Only used for interative mode. When False, returns error for user confirmation when the number to be deleted is active on other linked accounts.</param>
    ''' <param name="pConfirmedToDeleteAllLinked">Only used for interative mode. 0=raise error for user confirmation, 1=User confirmed to delete all linked numbers, 2=User confirmed NOT to delete all linked numbers</param>
    ''' <param name="pIsNonInteractiveMode"></param>
    ''' <param name="opResultCode"></param>
    ''' <param name="opResultMessage"></param>
    ''' <returns></returns>
    ''' <remarks></remarks>
    <ComVisible(False)>
    Public Function DeletePhoneNumber(pAccountPhoneDataDTO As DTO.AccountPhoneData, pConfirmedToDeleteIfActiveOnLinked As Boolean, pConfirmedToDeleteAllLinked As Byte, pIsNonInteractiveMode As Boolean, ByRef opResultCode As
SavePhoneResultCodes, ByRef opResultMessage As String) As Boolean
        Try
            Dim entPhoneData As New Entities.AccountPhoneData(Me.Environment)
            entPhoneData.LoadFromDTO(pAccountPhoneDataDTO)
            Return DeletePhoneNumber(entPhoneData, pConfirmedToDeleteIfActiveOnLinked, pConfirmedToDeleteAllLinked, pIsNonInteractiveMode, opResultCode, opResultMessage)
```

```
        Catch ex As Exception
            AddError(ex)
            opResultCode = SavePhoneResultCodes.Error_UnexpectedError
            opResultMessage = ex.ToString
        End Try
    End Function
```

3 Base Overload 3 by Entities.AccountPhoneData

```
    ''' <summary>
    ''' Deletes a phone number for an account using given Entities.AccountPhoneData
    ''' </summary>
    ''' <param name="pAccountPhoneData"></param>
    ''' <param name="pConfirmedToDeleteIfActiveOnLinked">Only used for interative mode. When False, returns error for user confirmation when the number to be deleted is active on other linked accounts.</param>
    ''' <param name="pConfirmedToDeleteAllLinked">Only used for interative mode. 0=raise error for user confirmation, 1=User confirmed to delete all linked numbers, 2=User confirmed NOT to delete all linked numbers</param>
    ''' <param name="pIsNonInteractiveMode"></param>
    ''' <param name="opResultCode"></param>
    ''' <param name="opResultMessage"></param>
    ''' <returns></returns>
    ''' <remarks></remarks>
    <ComVisible(False)>
    Friend Function DeletePhoneNumber(pAccountPhoneData As Entities.AccountPhoneData, pConfirmedToDeleteIfActiveOnLinked As Boolean, pConfirmedToDeleteAllLinked As Byte, pIsNonInteractiveMode As Boolean, ByRef opResultCode As
SavePhoneResultCodes, ByRef opResultMessage As String) As Boolean
        Try
            Dim linkedNumbers As List(Of DTO.AccountPhoneData) = GetAccountPhoneDataForLinkedAccounts(pAccountPhoneData.Ourref, False)
            If pIsNonInteractiveMode = False Then
                If linkedNumbers.Exists(Function(phoneData) phoneData.PhoneNo.Replace(" ", "").Trim = pAccountPhoneData.PhoneNo.Replace(" ", "").Trim AndAlso phoneData.Deleteddate.HasValue = False) Then
                    If pConfirmedToDeleteIfActiveOnLinked = False Then
                        opResultCode = SavePhoneResultCodes.Confirm_NumberIsActiveOnLinkedAccounts
                        opResultMessage = "This number " & pAccountPhoneData.PhoneNo & " is active on linked accounts. Are you sure to remove this number?"
                        Return False
                    ElseIf pConfirmedToDeleteAllLinked = 0 Then
                        opResultCode = SavePhoneResultCodes.Confirm_DeleteFromAllLinkedAccounts
                        opResultMessage = "Do you want to remove this number " & pAccountPhoneData.PhoneNo & " from all linked account?"
                        Return False
                    End If
                End If
            End If
            pAccountPhoneData.DeleteSoft()
            If pAccountPhoneData.Save Then
                opResultCode = SavePhoneResultCodes.Pass_NumberDeleted
                opResultMessage = "Number " & pAccountPhoneData.PhoneNo & " has been deleted"
                linkedNumbers = linkedNumbers.Where(Function(phoneData) phoneData.PhoneNo.Replace(" ", "").Trim = pAccountPhoneData.PhoneNo.Replace(" ", "").Trim AndAlso phoneData.Deleteddate.HasValue = False).ToList
                If linkedNumbers.Count > 0 AndAlso (pIsNonInteractiveMode OrElse pConfirmedToDeleteAllLinked = 1) Then
                    Dim filteredLinkedNumbers As List(Of DTO.AccountPhoneData) = linkedNumbers
                    If pIsNonInteractiveMode Then
                        filteredLinkedNumbers = linkedNumbers.Where(Function(phoneData) phoneData.Confirmdate.HasValue = False).ToList
                    End If
                    If filteredLinkedNumbers.Count > 0 Then
                        Dim linkedOurrefs As New Text.StringBuilder
                        For Each phoneData As DTO.AccountPhoneData In filteredLinkedNumbers
                            Dim entPhoneData As New Entities.AccountPhoneData(Me.Environment)
                            entPhoneData.LoadFromDTO(phoneData)
                            entPhoneData.DeleteSoft()
                            If entPhoneData.Save Then
                                linkedOurrefs.Append(entPhoneData.Ourref)
```

```
                        linkedOurrefs.Append(",")
                        Dim daAccount As New DataAccess.Account(Me.Environment)
                        daAccount.AddNote(entPhoneData.Ourref, 120, If(entPhoneData.Phonetype = "H", "Home",
                                                    If(entPhoneData.Phonetype = "W", "Work",
                                                      If(entPhoneData.Phonetype = "M", "Mobile",
                                                        "Other"))) &
                                                " phone :" & entPhoneData.PhoneNo & " : REMOVED", Now, mUserWho, "", "")
                    End If
                Next
                If linkedOurrefs.Length > 0 Then
                    opResultCode = SavePhoneResultCodes.Pass_NumberDeletedWithLinkedAccounts
                    opResultMessage = linkedOurrefs.ToString.Trim(","c)
                End If
            End If
        End If
        Return True
    Else
        opResultCode = SavePhoneResultCodes.Error_UnableToSave
        opResultMessage = "Unable to delete the number " & pAccountPhoneData.PhoneNo & ""
        Return False
    End If
Catch ex As Exception
    AddError(ex)
    opResultCode = SavePhoneResultCodes.Error_UnexpectedError
    opResultMessage = ex.ToString
End Try
End Function
```

DeletePhoneNumberByType() [Communication.vb]

By Phone-Type, Calls DeletePhoneNumber overload 2 (DTO.AccountPhoneData)

```
''' <summary>
''' Deletes a phone number for an account by PhoneType
''' </summary>
''' <param name="pOurref"></param>
''' <param name="pPhoneType"></param>
''' <param name="pConfirmedToDeleteIfActiveOnLinked">Only used for interative mode. When False, returns error for user confirmation when the number to be deleted is active on other linked accounts.</param>
''' <param name="pConfirmedToDeleteAllLinked">Only used for interative mode. 0=raise error for user confirmation, 1=User confirmed to delete all linked numbers, 2=User confirmed NOT to delete all linked numbers</param>
''' <param name="pIsNonInteractiveMode"></param>
''' <param name="opResultCode"></param>
''' <param name="opResultMessage"></param>
''' <returns></returns>
''' <remarks></remarks>
Public Function DeletePhoneNumberByType(pOurref As Integer, pPhoneType As String, pConfirmedToDeleteIfActiveOnLinked As Boolean, pConfirmedToDeleteAllLinked As Byte, pIsNonInteractiveMode As Boolean, ByRef opResultCode As
SavePhoneResultCodes, ByRef opResultMessage As String) As Boolean
    Try
        Dim existingNumbersForAccount As List(Of DTO.AccountPhoneData) = GetAccountPhoneDataByOurref(pOurref, False)
        If existingNumbersForAccount.Exists(Function(phoneData) phoneData.Phonetype = pPhoneType) Then
            Return DeletePhoneNumber(existingNumbersForAccount.Find(Function(phoneData) phoneData.Phonetype = pPhoneType), pConfirmedToDeleteIfActiveOnLinked, pConfirmedToDeleteAllLinked, pIsNonInteractiveMode, opResultCode, opResultMessage)
        Else
            opResultCode = SavePhoneResultCodes.Error_UnableToFindThePhoneRecord
            opResultMessage = "Unable to find the phone type " & pPhoneType & " to delete"
            Return False
        End If
    Catch ex As Exception
```

```
            AddError(ex)
            opResultCode = SavePhoneResultCodes.Error_UnexpectedError
            opResultMessage = ex.ToString
        End Try
    End Function
```

DeletePhoneNumberByTypeNonInteractiveMode() [DebtorRecords.vb]

```
    Public Function DeletePhoneNumberByTypeNonInteractiveMode(pOurRef As Integer, pPhoneType As String, ByRef opResultCode As Controller.Communication.SavePhoneResultCodes, ByRef opResultMessage As String) As Boolean
        Dim contComm As New Controller.Communication(mUserWho)
        DeletePhoneNumberByTypeNonInteractiveMode = contComm.DeletePhoneNumberByType(pOurRef, pPhoneType, True, True, True, opResultCode, opResultMessage)
        If DeletePhoneNumberByTypeNonInteractiveMode = True Then
            objsvrAccounts.RefreshAccountPhoneData()
        End If
    End Function
```

DeletePhoneNumberByTypeFrontEnd() [DebtorRecords.vb]

```
    Public Function DeletePhoneNumberByTypeFrontEnd(pOurRef As Integer, pPhoneType As String, pConfirmedToDeleteIfActiveOnLinked As Boolean, pConfirmedToDeleteAllLinked As Byte, ByRef opResultCode As Controller.Communication.SavePhoneResultCodes,
ByRef opResultMessage As String) As Boolean
        Dim contComm As New Controller.Communication(mUserWho)
        DeletePhoneNumberByTypeFrontEnd = contComm.DeletePhoneNumberByType(pOurRef, pPhoneType, pConfirmedToDeleteIfActiveOnLinked, pConfirmedToDeleteAllLinked, False, opResultCode, opResultMessage)
        If DeletePhoneNumberByTypeFrontEnd = True Then
            objsvrAccounts.RefreshAccountPhoneData()
        End If
    End Function
```

Other dotNet Projects

C:/source/Net2010/Operations/WorkQueue/Controls/ItemManagerControls/Widgets/AccountViewer.ascx.vb – more complicated use with DR-Wrapper function.

Defines local function DeletePhoneNumberByType() calls DeletePhoneNumberByTypeFrontEnd() with front-end wrapper.

btnRemove_Click() 4 calls to local DeletePhoneNumberByType() for 4 different types.

btnRemoveAllNumbers_Click() 4 calls to DeletePhoneNumberByTypeNonInteractiveMode() for 4 different types.

btnQuestionMessageNo_Click() calls local DeletePhoneNumberByType().

btnQuestionMessageYes_Click() 2 calls to local DeletePhoneNumberByType().

C:/source/Net2010/HLROps/HLROps/HLROps/DataCleansing.vb – simple DR calls.

UpdatePhoneNumber() 4 calls to DeletePhoneNumberByTypeNonInteractiveMode().

C:/source/Net2010/Account_Processing/Account_Processing/Account_Processing/Clients/Socates/Common/AccountReinstatement.vb – simple DR calls.

ReinstateAccount() 3 calls to DeletePhoneNumberByTypeNonInteractiveMode().

C:/source/Net2010/Account_Processing/Account_Processing/Account_Processing/Clients/TDX/PhoneDeactivationFile/TDXPhoneDeactivationFile.vb – simple DR calls.

RemoveNumber() 4 calls to DeletePhoneNumberByTypeNonInteractiveMode().

VB6 Projects

C:/source/vb6/dll/Debtors1/**DrsAccounts.cls** – simple Debtors1 DR wrappers of Debtors-Net DR

DeletePhoneNumberByTypeNonInteractiveMode() calls Debtors-Net DeletePhoneNumberByType() [same as Debtors-Records routine]

DeletePhoneNumberByTypeFrontEnd() calls Debtors-Net DeletePhoneNumberByType() [same as Debtors-Records routine]

C:/source/vb6/FrontEndSystems/CollectorsFrontEnd-CFEProxt/**Module3.bas** – simple DR wrapper function, using recursion; calls Debtors1 wrapper.

Defines local function DeletePhoneNumberByType() calls Debtors1 DeletePhoneNumberByTypeFrontEnd(); with user-interface recursive calls.

C:/source/vb6/FrontEndSystems/CollectorsFrontEnd-CFEProxt/**AccountCommunications.cls** – simple DR calls to Module3.bas function, thence to Debtors1

netForm_ExecuteSave() 4 calls to local Module3 DeletePhoneNumberByType()

C:/source/vb6/FrontEndSystems/AdminFrontEnd-AFE/**AdminAFE.bas** – simple DR wrapper function, using recursion; calls Debtors1 wrapper

Defines local function DeletePhoneNumberByType() calls Debtors1 DeletePhoneNumberByTypeFrontEnd(); with user-interface recursive calls.

C:/source/vb6/FrontEndSystems/AdminFrontEnd-AFE/**AccountCommunications.cls** – simple DR calls to AdminAFE.bas function, thence to Debtors1

netForm_ExecuteSave() 4 calls to local Module3 DeletePhoneNumberByType()

The CFE and AFE code is identical to each other with respect to 'DeletePhoneNumbe'.

Other 'C:\Source' Projects

None

2 Search for 'SetKill' calls and definitions; entire system.

Debtors-Net only

C:\source\Net2010\DebtorsNet\DebtorRecords.vb(1742): If mApplicationKill And Mid(Status, 2, 2) <> Mid(value, 2, 2) Then SetKill()- Property Status()
C:\source\Net2010\DebtorsNet\DebtorRecords.vb(4807): Public Sub SetKill()
C:\source\Net2010\DebtorsNet\svrClientReferral.vb(536): ' Account.SetKill()- code commented out

SetKill()

```
Public Sub SetKill()
    Dim KillLst As Integer
    Dim Filename As String
    KillSMS()
    Dim accountEmail As New Controller.AccountEmail(Fredrickson.Shared.Utilities.CurrentWindowsUserName)
    accountEmail.AddToKillList(OurReference.ToString(), "")
    Filename = mDrsLoc & "Dialler\Kill\" & mUserWho.Trim & Format(Now, "mmHH")
    If Dir(Filename & ".lst") <> "" Then
        Try
            Rename(Filename & ".lst", Filename & ".$$$")
        Catch ex As Exception
            ' do nothing as it has possibly been taken by the Envox Build program
        End Try
    End If
    KillLst = FreeFile()
resumehere:
    Try
        'changed the file access to shared mode
        FileOpen(KillLst, Filename & ".$$$", OpenMode.Append, OpenAccess.ReadWrite, OpenShare.Shared, Len(OurReference.ToString))
        PrintLine(KillLst, OurReference.ToString)
        FileClose(KillLst)
        Rename(Filename & ".$$$", Filename & ".lst") 'added 16/8/07 mi
    Catch When Err.Number = 70
        If mNoRetry Then
            SystemWait()
        Else
            Dim frmretry As New frmRetry
            frmretry.ShowDialog()
        End If
        GoTo resumehere
    Catch
        Error Err.Number
    End Try
End Sub
```

KillSMS()

```
'KillSMS:
```

```
'Call this function to kill SMS messages queued to be sent to the debtor on the current day.
'This function will call the Kill API (via a webrequest) on the SMS Management web site.
Public Function KillSMS() As Boolean
    Dim sResponse As String = ""
    Dim oRequest As System.Net.HttpWebRequest
    Dim oData As New System.Text.ASCIIEncoding
    Dim oPostData() As Byte
    Dim oResp As System.IO.Stream
    Dim oResponse As System.Net.HttpWebResponse = Nothing
    Dim APIPath As String = ""
    Dim sCallBackParams As String = ""
    Try
        APIPath = GetINIData("SMSAPI", "Path", mDrsLoc & DebtorsIni)
        sCallBackParams = "MsgType=Kill&OurRef=" & OurReference.ToString()
        '*****************************************************************************************
        oRequest = System.Net.WebRequest.Create(APIPath)
        oRequest.Method = "POST"
        oRequest.ContentType = "application/x-www-form-urlencoded"
        oPostData = oData.GetBytes(sCallBackParams)
        oRequest.ContentLength = oPostData.Length
        oResp = oRequest.GetRequestStream
        oResp.Write(oPostData, 0, oPostData.Length)
        oResponse = oRequest.GetResponse()
        Return (oResponse.StatusCode = HttpStatusCode.OK)
        oResponse.Close()
    Catch ex As System.Net.WebException
        If Not oResponse Is Nothing Then
            oResponse.Close()
        End If
        Return False
    Catch ex As Exception
        If Not oResponse Is Nothing Then
            oResponse.Close()
        End If
        Return False
    End Try
End Function
```

C:\source\Net2010\DebtorsNet\DebtorRecords.vb – definition of KillSMS() and single call from SetKill()

KillEmail()

```
Public Sub KillEmail(ByVal ourRef As String, ByVal emailAddress As String)
    Dim cont As New Controller.AccountEmail(Fredrickson.Shared.Utilities.CurrentWindowsUserName)
    cont.AddToKillList(ourRef, emailAddress)
End Sub
```

C:\source\Net2010\DebtorsNet\svrAccounts.vb – definition of KillEmail() but no calls to.

Other dotNet Projects

SetKill()

C:/source/Net2010/Operations/WorkQueue/Controls/ItemManagerControls/Widgets/ClientReferralCreator.ascx.vb – single DR call
C:/source/Net2010/BryanCarter/BryanCarterWS/App_code/clsAccount.vb – single DR call
C:\source\net\SRJ\SRJ_WebService\App_code\clsAccount.vb

KillSMS()

none

VB6 Projects

Uses Debtors1 versions of 'Kill' routines, not Debtors-Net.

SetKill()

Calls KillSMS() and KillEmail()

```
Public Sub SetKill()
    Dim KillLst%
    Dim FileName As String
    KillSMS
    KillEmail (OurReference)
    FileName = mDrsLoc & "Dialler\Kill\" & ApplicationName & UserWho & Format(Now, "nnhh")
    If Dir(FileName & ".lst") <> "" Then
        On Error Resume Next
        Name FileName & ".lst" As FileName & ".$$$"
        On Error GoTo 0
    End If
    KillLst = FreeFile
    On Error GoTo retry
        Open FileName & ".$$$" For Append Shared As KillLst
            Print #KillLst, Format(OurReference)
        Close KillLst
        'Clear for below error trap
        Err.clear
        On Error Resume Next
        Name FileName & ".$$$" As FileName & ".lst"
        If Err.Number <> 0 Then
            Call Err.Raise(Err.Number, Err.Source, "DrsAccounts.SetKill(), Name FileName.$$$ As FileName.lst: " & Err.description & ". FileName = " & FileName, Err.HelpFile, Err.HelpContext)
        End If
    On Error GoTo 0
    Exit Sub
retry:
    If Err.Number <> 70 Then
        Call Err.Raise(Err.Number, Err.Source, "DrsAccounts.SetKill(): " & Err.description & ". FileName = " & FileName, Err.HelpFile, Err.HelpContext)
    End If
    If NoRetry Then SystemWait Else frmRetry.Show vbModal
    Resume
End Sub
```

C:/source/vb6/FrontEndSystems/CollectorsFrontEnd-CFEProxt/W6Col.bas – single call

C:/source/vb6/FrontEndSystems/CollectorsFrontEnd-CFEProxt/Module3.bas – single call

C:/source/vb6/FrontEndSystems/CollectorsFrontEnd-CFEProxt/AccountCommunications.cls – single SetKill() call plus 6 calls to SetKillPhone()

C:/source/vb6/FrontEndSystems/CollectorsFrontEnd-CFEProxt/FFSOnceOff_Low.cls – single call

C:/source/vb6/FrontEndSystems/AdminFrontEnd-AFE/AdminAFE.bas – single call

C:/source/vb6/FrontEndSystems/AdminFrontEnd-AFE/AccountCommunications.cls – single SetKill() call plus 2 calls to SetKillPhone()

C:/source/vb6/dll/Debtors1/DrsAccounts.cls – has local versions of SetKill(), SetKillPhone(), KillSMS() and KillEMail; does not call Debtors-Net routines

C:/source/vb6/dll/Debtors1/DrsAccounts.txt – early version of DrsAccounts.cls

C:/source/vb6/dll/Debtors1EMAIL/DrsAccounts.cls – shorter version of Debtors1/DrsAccounts.cls

C:/source/vb6/dll/Debtors1EMAIL/DrsAccounts.txt – same as Debtors1/DrsAccounts.txt

C:/source/vb6/court/claimproductionv2/W6claim.bas – commented out

C:/source/vb6/court/WarrantProduction/W6warrant.bas – single call

C:/source/vb6/BackOffice/DMC/DMC Automation/ProcessOffers.bas – single call

KillSMS()

```
Public Sub KillSMS(Optional ByVal PhoneNo As String = "")
    Dim DataToSend As String
    Dim oXMLHTTP As New MSXML2.XMLHTTP
    Dim ApiPath As String
    Dim strReponse As String
    On Error GoTo SMSErr
    ApiPath = GetINIData("SMSAPI", "Path", mDrsLoc & "Util\Debtors.ini")
    If ApiPath <> "" Then
        If PhoneNo = "" Or PhoneNo = "*" Then
            DataToSend = "MsgType=Kill&OurRef=" & Format(OurReference)
        Else
            DataToSend = "MsgType=Kill&OurRef=" & Format(OurReference) & "&PhoneNo=" & PhoneNo
        End If
        oXMLHTTP.Open "POST", ApiPath, False
        oXMLHTTP.setRequestHeader "Content-Type", "application/x-www-form-urlencoded"
        oXMLHTTP.send DataToSend
    End If
    On Error GoTo 0
    Exit Sub
SMSErr:
    Call Err.Raise(Err.Number, Err.Source, "DrsAccounts.KillSMS(): " & Err.description, Err.HelpFile, Err.HelpContext)
End Sub
```

C:/source/vb6/dll/Debtors1/DrsAccounts.cls – local version of KillSMS() and calls from local SetKill() and SetKillPhone()

KillEmail()

```
Public Sub KillEmail(Optional ByVal Ourref As String = "")
    Dim DataToSend As String
    Dim oXMLHTTP As New MSXML2.XMLHTTP
    Dim ApiPath As String
    Dim strReponse As String
    On Error GoTo SMSErr
    ApiPath = GetINIData("EMAILKILLAPI", "Path", mDrsLoc & "Util\Debtors.ini")
    If ApiPath <> "" Then
        DataToSend = "?emailAddress=&ourref=" & Format(OurReference)
        oXMLHTTP.Open "GET", ApiPath & DataToSend, False
```

```
      oXMLHTTP.setRequestHeader "Content-Type", "application/x-www-form-urlencoded"
      oXMLHTTP.send
   End If
   On Error GoTo 0
   Exit Sub
SMSErr:
   Call Err.Raise(Err.Number, Err.Source, "DrsAccounts.KillSMS(): " & Err.description, Err.HelpFile, Err.HelpContext)
End Sub
```

C:/source/vb6/dll/Debtors1/DrsAccounts.cls – definition of KillEmail() and single call from local SetKill()

SetKillPhone()

Calls KillSMS

```
Public Sub SetKillPhone(Optional ByVal PhoneNo As String = "")
   Dim KillLst%, PhNo$
   If HasPhone = 0 Then Exit Sub
   KillLst = FreeFile
   Open mDrsLoc & "Dialler\Kill\" & UserWho & Format(Now, "nnhh") & ".lst" For Append As KillLst
      If PhoneNo = "*" Then
         PhNo = OnlyDigits(Home_Phone)
         GoSub PrintKill
         PhNo = OnlyDigits(Work_Phone)
         GoSub PrintKill
         PhNo = OnlyDigits(Mobile_Phone)
         GoSub PrintKill
         PhNo = OnlyDigits(Other_Phone)
         GoSub PrintKill
      Else
         PhNo = PhoneNo
         GoSub PrintKill
      End If
   Close KillLst
   If PhoneNo = "" Or PhoneNo = "*" Then
      KillSMS
   Else
      KillSMS PhoneNo
   End If
   Exit Sub
PrintKill:
   If Len(PhNo) > 0 Then Print #KillLst, Format(OurReference) & "," & PhNo
   Return
retry:
   If Err.Number <> 70 Then Error Err.Number
   If NoRetry Then SystemWait Else frmRetry.Show vbModal
   Resume
End Sub
```

C:\source\vb6\dll\Debtors1\DrsAccounts.cls – local definition of SetKillPhone() no calls

C:/source/vb6/FrontEndSystems/CollectorsFrontEnd-CFEProxt/AccountCommunications.cls

netForm_DDProcessKillNumbersFromDialler() – 4 calls to SetKillPhone()

netForm_ExecuteSave() – 2 calls to SetKillPhone()

C:/source/vb6/FrontEndSystems/AdminFrontEnd-AFE/AccountCommunications.cls

netForm_ExecuteSave() – 2 calls to SetKillPhone()

Conclusions

11 Jan 2018

Adding SetKill() to DeletePhoneNumber() versions.

'DeletePhoneNumber': All related routines end up calling the base overload, giving us a single point of change. However some of the VB6 projects use recursion over a number of phone-numbers, and repeat calls to handle user-interaction; therefore careful placement required.

'SetKill': There are two main versions of this function, one for Debtors-Net and one for Debtors1.

Debtors-Net | SetKill: Calls KillSMS() and Controller.AccountEmail.AddToKillList().

Debtors1 | SetKill: Calls KillSMS() and KillEmail().

Debtors1 | SetKillPhone: Calls KillSMS().

Code Changes

Move SetKill() and KillSMS() from 'DebtorRecords' module to 'Controllers-Communication' module, leaving behind stubs to call base methods.

Add call to 'SetKill()' to root 'DeletePhoneNumber()' overload.

16 Jan 2018

Re-think.

Code Changes

Move SetKill(), SetKillPhone(), KillEmail() and KillSMS() from 'DebtorRecords' module to 'Controllers-Communication' module as 'SetKill()', leaving behind stubs to call base methods.

Add call to 'SetKill()' to root 'DeletePhoneNumber()' overload.

Add parameter to DebtorsNet and Debtors1 'DeletePhoneNumber()' overloads; and DebtorsNet 'SetKill()' and 'SetKillPhone()' to modify behaviour.

Re-Run of Searches using 'FileSeek' application

DeletePhoneNumber…() VB.net

C:/source/Net2010/Account_Processing/Account_Processing/Account_Processing/Clients/Socates/Common/**AccountReinstatement.vb** – simple DR calls.
ReinstateAccount() 3 calls to DeletePhoneNumberByTypeNonInteractiveMode().
C:/source/Net2010/Account_Processing/Account_Processing/Account_Processing/Clients/TDX/PhoneDeactivationFile/**TDXPhoneDeactivationFile.vb** – simple DR calls.
RemoveNumber() 4 calls to DeletePhoneNumberByTypeNonInteractiveMode().
DeletePhoneNumberByType() [Communication.vb]

By Phone-Type, Calls DeletePhoneNumber overload 2 (DTO.AccountPhoneData)

DeletePhoneNumberByTypeNonInteractiveMode() [DebtorRecords.vb]

DeletePhoneNumberByTypeFrontEnd() [DebtorRecords.vb]

DeletePhoneNumber() 3 overloads [Communication.vb]

1: Overload by Phone-Number, calls Overload 2 (DTO.AccountPhoneData)

2: Overload by DTO.AccountPhoneData, calls Base Overload 3 (Entities.AccountPhoneData)

3 Base Overload 3 by Entities.AccountPhoneData

C:\source\Net2010\DebtorsNet\Controllers**Communication.vb**(561): **DeletePhoneNumber**(existingNumbersForAccount.Find(Function(phoneData) phoneData.Deleteddate.HasValue = False AndAlso phoneData.Phonetype = phoneTypeToUse), pConfirmedToDeleteIfActiveOnLinked, pConfirmedToDeleteAllLinked, pIsNonInteractiveMode, opResultCode, opResultMessage) : in AddPhoneNumber() calls Overload 2.

C:\source\Net2010\DebtorsNet\Controllers**CustomerFair.vb**(2059): commController.**DeletePhoneNumber**(objentity, True, 1, True, 0, "") : calls Overload 3

C:/source/Net2010/HLROps/HLROps/HLROps/**DataCleansing.vb** – simple DR calls.
UpdatePhoneNumber() 4 calls to DeletePhoneNumberByTypeNonInteractiveMode().
C:/source/Net2010/Operations/WorkQueue/Controls/ItemManagerControls/Widgets/**AccountViewer.ascx.vb** – more complicated use with DR-Wrapper function.
Defines local function DeletePhoneNumberByType() calls DeletePhoneNumberByTypeFrontEnd() with front-end wrapper.
btnRemove_Click() 4 calls to local DeletePhoneNumberByType() for 4 different types.
btnRemoveAllNumbers_Click() 4 calls to DeletePhoneNumberByTypeNonInteractiveMode() for 4 different types.
btnQuestionMessageNo_Click() calls local DeletePhoneNumberByType().
btnQuestionMessageYes_Click() 2 calls to local DeletePhoneNumberByType().

DeletePhoneNumber…() VB6

C:/source/vb6/dll/Debtors1/**DrsAccounts.cls** – simple Debtors1 DR wrappers of Debtors-Net DR

DeletePhoneNumberByTypeNonInteractiveMode() calls Debtors-Net DeletePhoneNumberByType() [same as Debtors-Records routine]

DeletePhoneNumberByTypeFrontEnd() calls Debtors-Net DeletePhoneNumberByType() [same as Debtors-Records routine]

C:/source/vb6/FrontEndSystems/CollectorsFrontEnd-CFEProxt/**Module3.bas** – simple DR wrapper function, using recursion; calls Debtors1 wrapper.

Defines local function DeletePhoneNumberByType() calls Debtors1 DeletePhoneNumberByTypeFrontEnd(); with user-interface recursive calls.

C:/source/vb6/FrontEndSystems/CollectorsFrontEnd-CFEProxt/**AccountCommunications.cls** – simple DR calls to Module3.bas function, thence to Debtors1

netForm_ExecuteSave() 4 calls to local Module3 DeletePhoneNumberByType()

C:/source/vb6/FrontEndSystems/AdminFrontEnd-AFE/**AdminAFE.bas** – simple DR wrapper function, using recursion; calls Debtors1 wrapper

Defines local function DeletePhoneNumberByType() calls Debtors1 DeletePhoneNumberByTypeFrontEnd(); with user-interface recursive calls.

C:/source/vb6/FrontEndSystems/AdminFrontEnd-AFE/**AccountCommunications.cls** – simple DR calls to AdminAFE.bas function, thence to Debtors1

netForm_ExecuteSave() 4 calls to local Module3 DeletePhoneNumberByType()

Additions

C:\source\vb6\BackOffice\DiallerUpdation**DiallerUpdation.frm**

Call Dd.DeletePhoneNumberByTypeNonInteractiveMode(), x4

C:\source\vb6\Dialler\CallMedia**Upload.frm**

Call DD.DeletePhoneNumberByTypeNonInteractiveMode(), x4

C:\source\vb6\reports\ExcellReporting**MDIMain.frm**

Call Dd.DeletePhoneNumberByTypeNonInteractiveMode(), x12

C:\source\vb6\FrontEndSystems\CollectorsFrontEnd-CFEProtx\goneaway.frm

Call DeletePhoneNumberByType(), x1

SetKill() VB.net

C:\source\Net2010\DebtorsNet\DebtorRecords.vb(1742): If mApplicationKill And Mid(Status, 2, 2) <> Mid(value, 2, 2) Then SetKill()- Property Status()
C:\source\Net2010\DebtorsNet\DebtorRecords.vb(4807): Public Sub SetKill()
C:\source\Net2010\DebtorsNet\svrClientReferral.vb(536): ' Account.SetKill()- code commented out

C:/source/Net2010/Operations/WorkQueue/Controls/ItemManagerControls/Widgets/ClientReferralCreator.ascx.vb – single DR call

C:/source/Net2010/BryanCarter/BryanCarterWS/App_code/clsAccount.vb – single DR call

C:\source\net\SRJ\SRJ_WebService\App_code\clsAccount.vb

SetKill() VB6

C:/source/vb6/dll/Debtors1/DrsAccounts.cls – has local versions of SetKill(), SetKillPhone(), KillSMS() and KillEMail; does not call Debtors-Net routines

C:/source/vb6/FrontEndSystems/CollectorsFrontEnd-CFEProxt/W6Col.bas – single call
C:/source/vb6/FrontEndSystems/CollectorsFrontEnd-CFEProxt/Module3.bas – single call
C:/source/vb6/FrontEndSystems/CollectorsFrontEnd-CFEProxt/AccountCommunications.cls – single SetKill() call plus 6 calls to SetKillPhone()
C:/source/vb6/FrontEndSystems/CollectorsFrontEnd-CFEProxt/FFSOnceOff_Low.cls – single call
C:/source/vb6/FrontEndSystems/AdminFrontEnd-AFE/AdminAFE.bas – single call
C:/source/vb6/FrontEndSystems/AdminFrontEnd-AFE/AccountCommunications.cls – single SetKill() call plus 2 calls to SetKillPhone()
C:/source/vb6/court/WarrantProduction/W6warrant.bas – single call
C:/source/vb6/BackOffice/DMC/DMC Automation/ProcessOffers.bas – single call

Additions

C:\source\vb6\BackOffice\DirectDebit\DDRecon\DDRecon.frm – single call
C:\source\vb6\BackOffice\DMC\DMC Automation\ProcessOffers.bas – single call
C:\source\vb6\BackOffice\MailMerge\Copy of W6MMerge.frm – 2 calls
C:\source\vb6\BackOffice\Payments\W6PdCheq\W5PdCheq.frm – single call
C:\source\vb6\FrontEndSystems\AdminFrontEnd-AFE\FFSOnceOff_Low.frm
C:\source\vb6\FrontEndSystems\AdminFrontEnd-AFE\frmClientReferralNEW.frm
C:\source\vb6\FrontEndSystems\CollectorsFrontEnd-CFEProtx\AbusiveDebtor.frm
C:\source\vb6\FrontEndSystems\CollectorsFrontEnd-CFEProtx\Arrangement.frm
C:\source\vb6\FrontEndSystems\CollectorsFrontEnd-CFEProtx\CannotPay.frm
C:\source\vb6\FrontEndSystems\CollectorsFrontEnd-CFEProtx\ClaimsPaid.frm
C:\source\vb6\FrontEndSystems\CollectorsFrontEnd-CFEProtx\DebtorWants.frm
C:\source\vb6\FrontEndSystems\CollectorsFrontEnd-CFEProtx\Deceased.frm
C:\source\vb6\FrontEndSystems\CollectorsFrontEnd-CFEProtx\delay.frm
C:\source\vb6\FrontEndSystems\CollectorsFrontEnd-CFEProtx\DisputeOrRefuseToPay.frm
C:\source\vb6\FrontEndSystems\CollectorsFrontEnd-CFEProtx\DUDTelephoneNumberUpdate.frm
C:\source\vb6\FrontEndSystems\CollectorsFrontEnd-CFEProtx\FFSOnceOff.frm
C:\source\vb6\FrontEndSystems\CollectorsFrontEnd-CFEProtx\FFSOnceOff_Low.cls
C:\source\vb6\FrontEndSystems\CollectorsFrontEnd-CFEProtx\FFSOnceOff_Low.frm
C:\source\vb6\FrontEndSystems\CollectorsFrontEnd-CFEProtx\fraud.frm
C:\source\vb6\FrontEndSystems\CollectorsFrontEnd-CFEProtx\frmArrangementReConfirm.frm
C:\source\vb6\FrontEndSystems\CollectorsFrontEnd-CFEProtx\frmAwaitingCorre.frm
C:\source\vb6\FrontEndSystems\CollectorsFrontEnd-CFEProtx\frmCap1DebtManagement.frm
C:\source\vb6\FrontEndSystems\CollectorsFrontEnd-CFEProtx\frmCollectionFinancialStatement.frm
C:\source\vb6\FrontEndSystems\CollectorsFrontEnd-CFEProtx\frmCreditCardAmend.frm
C:\source\vb6\FrontEndSystems\CollectorsFrontEnd-CFEProtx\frmCreditCardPayment.frm

C:\source\vb6\FrontEndSystems\CollectorsFrontEnd-CFEProtx\frmCreditCardPaymentNEW.frm
C:\source\vb6\FrontEndSystems\CollectorsFrontEnd-CFEProtx\frmDataProtectionCheckFailed.frm
C:\source\vb6\FrontEndSystems\CollectorsFrontEnd-CFEProtx\frmDataProtectionXtraEnh.frm
C:\source\vb6\FrontEndSystems\CollectorsFrontEnd-CFEProtx\frmDirectDebitAmend.frm
C:\source\vb6\FrontEndSystems\CollectorsFrontEnd-CFEProtx\frmDispute.frm
C:\source\vb6\FrontEndSystems\CollectorsFrontEnd-CFEProtx\frmFFSOnceOff.frm
C:\source\vb6\FrontEndSystems\CollectorsFrontEnd-CFEProtx\frmFullBalanceRedemption.frm
C:\source\vb6\FrontEndSystems\CollectorsFrontEnd-CFEProtx\frmLatePayment.frm
C:\source\vb6\FrontEndSystems\CollectorsFrontEnd-CFEProtx\frmStatementOfAccountReques.frm
C:\source\vb6\FrontEndSystems\CollectorsFrontEnd-CFEProtx\frmVodafoneLiveArrangement.frm
C:\source\vb6\FrontEndSystems\CollectorsFrontEnd-CFEProtx\goneaway.frm
C:\source\vb6\FrontEndSystems\CollectorsFrontEnd-CFEProtx\InsufficientInstalment.frm
C:\source\vb6\FrontEndSystems\CollectorsFrontEnd-CFEProtx\InsufficientPostLitInstalment.frm
C:\source\vb6\FrontEndSystems\CollectorsFrontEnd-CFEProtx\WriteOff.frm
C:\source\vb6\Overnite\LBNA\FrmLbna.frm
C:\source\vb6\Overnite\SetLegalReplies\SetLegal.frm

SetKillPhone() VB6

C:\source\vb6\dll\Debtors1\DrsAccounts.cls – local definition of SetKillPhone() no local calls

C:/source/vb6/FrontEndSystems/CollectorsFrontEnd-CFEProxt/AccountCommunications.cls
    netForm_DDProcessKillNumbersFromDialler() – 4 calls to SetKillPhone()
    netForm_ExecuteSave() – 2 calls to SetKillPhone()
C:/source/vb6/FrontEndSystems/AdminFrontEnd-AFE/AccountCommunications.cls
    netForm_ExecuteSave() – 2 calls to SetKillPhone()

C:\source\vb6\FrontEndSystems\CollectorsFrontEnd-CFEProtx\DUDTelephoneNumberUpdate.frm – 6 calls to SetKillPhone()

DebtorsNet DLL method ordering

To reduce the need for re-compiling of dependant applications.

Current end-of-file list

…
DeletePhoneNumber() [Overload 1]
DeletePhoneNumberByType() **
DeletePhoneNumber() [Overload 2] **
DeletePhoneNumber() [Overload 3] **
GetAccountPhoneDataPendingByOurref()
DeleteAccountPhoneDataPendingByKeyID()

'**' = Replaced with Stubs to call new methods with extra parameter boolean 'bKillPhone' (if true call KillPhone()); set True.

Added Methods

DeletePhoneNumberByType_Kill() with extra parameter [Overloaded]
DeletePhoneNumber() [Overload 2] with extra parameter [Overload 4]
DeletePhoneNumber() [Overload 3] with extra parameter [Overload 5]
KillSMS() from DebtorRecords
SetKill()from DebtorRecords
KillPhone() from DebtorRecords SetKillPhone(), but single phone no only

Debtors1 DLL method ordering

To reduce the need for re-compiling of dependant applications.

Current end-of-file list

…
DeletePhoneNumberByTypeNonInteractiveMode() change to stub with extra parameter boolean 'bKillPhone' set True
DeletePhoneNumberByTypeFrontEnd() change to stub with extra parameter boolean 'bKillPhone' set True'
…

Added Methods

DeletePhoneNumberByType_NonInteractiveMode_Kill()
DeletePhoneNumberByType_FrontEnd_Kill()

Code Review Comments

After deleting the lines as requested, none of the other points raised line-up.

Parameter Names

I consider the company guidance on the naming policy to be wrong, it is almost counter-productive.

The inability to indicate scope/type/paradigm by naming hinders the ability to maintain the code-base. Having to inspect every textual encounter before understanding is gained greatly impedes all work by a factor of 3 or 4 at least.

We no longer work with 80 column green/black screens or 8 bit processors; obeying rules created by jumped-up typists promoted to a level beyond their competence, is not a good work-plan.

SetKill()

This is a verbatim copy and is therefore not altered in any way.

KillPhone()

Although this may be based on VB6 SetKillPhone() it is different, the different name use is to show this differentiation. If it is required to use the same name then we should also duplicate the functioning of the VB6 code in the VB.net code; extra work is needed to allow references to the underlying customer account.

The renaming of the file is to prevent it being picked-up by the dialler and changed/deleted; this is because locking the file has caused problems in the past.

Debtors.ini

[SMSAPI]

Path=http://dev3.notify.jobs.sms.fredad.net/killnotifications.aspx

DebtorsNet.ini

[SMSAPI]

Path=http://fredpays/sms_management/killnotifications.aspx, does not work.

Path=http://dev3.notify.jobs.sms.fredad.net/killnotifications.aspx, use VB6 version.

Debtors.ini.live

[SMSAPI]

Path=http://notify.jobs.sms.fredad.net/killnotifications.aspx