NTS Informatica s.r.l.
S.S. Rimini-S. Marino, 142 - 47900 Rimini - Italy
tel. +39 0541 906611 - fax. +39 0541 906610
www.ntsinformatica.it

Cap. Soc. € 61.973,00 i.v.
C.F. e Reg. Imp. 02019510409
C.C.I.A.A. Rimini 234990
P.IVA 02019510409

Personalizzazioni e Scripting in Bus.net

Circolare CT-0907-0027 del 09/12/2009.

Indice

Sommario

1. Script in Bus.net

Eccetto la valutazione della 4° unità di misura (la formula) che usa msscript, gli script in .net sono dll compilate e caricate al volo di tipo *'Option strict OFF'*.

Nella gestione degli alert le procedure possono essere scritte sia in msscritp (vbscript) compatibili con VB6, l'unica differenza è che non viene passata la form del chiamante allo scritp, sia con il nuovo motore di script in VB.NET (in seguito vedremo come).

Al momento, quando si lancia un child, be__menu lo istanzia, ne fa l'init, poi verifica se nella dir Script del server (una dir allo stesso livello della Rpt dir) è presente un file con lo stesso nome del child da lanciare, ma con estensione .NTS.

Se lo trova cerca di compilarlo e se tutto ok passa questo script al child di tipo 'UI' senza bisogno che, da codice, venga scritta alcuna riga particolare.

In fase di compilazione dello script il compilatore interno passa sempre i seguenti oggetti:

```
mscorlib.dll
system.dll
system.windows.forms.dll
Microsoft.VisualBasic.dll
System.Data.dll
System.XML.dll
System.Drawing.dll
System.Runtime.Remoting.dll
```

BN__STD.dll	preso dalla dir locale di bus.net
BE__FRWK.dll	preso dalla dir locale di bus.net
BE__MENU.dll	preso dalla dir locale di bus.net
BN__CHIL.dll	preso dalla dir locale di bus.net
BE__CZOO.dll	preso dalla dir locale di bus.net
BE__BASE.dll	preso dalla dir locale di bus.net
BE__BASN.dll	preso dalla dir locale di bus.net
BELBMENU.dll	preso dalla dir locale di bus.net
BE__TABE.dll	preso dalla dir locale di bus.net
BN__TABE.dll	preso dalla dir locale di bus.net
DevExpress*.dll	presi dalla dir locale di bus.net

per aggiungere altri riferimenti nel file di script basta aggiungere nel file, prima del codice, la direttiva

<reference assembly="BD__BASE.dll"/>

Se il nome dell'assembly inizia per BN, BE, BD viene preso dalla dir locale di bus.net, diversamente il sistema lo cerca nel GAC.

Nel codice presente nel file deve essere contenuta una classe con lo stesso nome del child, con in aggiunta "VBS" che implementa l'interfaccia INT__SCRIPT, inoltre deve contenere una funzione pubblica come nell'esempio che segue (presupponendo che il child lanciato è BN__PAGA):

```
Imports Microsoft.VisualBasic
Imports System
Imports NTSInformatica

Public Class BN__PAGAVBS
    Implements INT__SCRIPT
  Public Function Exec(ByVal strCommand As String, ByRef oApp As Object, ByRef
oIn As Object, ByRef oParam As Object) as Object Implements INT__SCRIPT.Exec
    Try
     MsgBox("PROVA")

    Catch ex As Exception
      MsgBox(ex.Message)
    End Try
  End Function
End Class
```

Se il file non verrà compilato verrà scatenato un errore, ma il child standard verrà lanciato/eseguito normalmente (ovviamente le funzioni contenute nello script non verranno eseguite …).

Una volta istanziato lo script, l'UI lo passa all'entity (questo in fase di creazione/attivazione entità).

Tra gli Imports ricordarsi di inserire almeno:

```
Imports Microsoft.VisualBasic
Imports System
Imports NTSInformatica.CLN__STD
Imports NTSInformatica
```

1.1. Esempio di file di script completo

(le scritte in blu sono fisse e devono essere sempre presenti):

```
<nts>
<use_ assembly="BN__SCTE.DLL"/>

<reference assembly="BD__BASE.dll" />
<reference assembly="BN__PAGA.dll" />

<scriptCode><![CDATA[

Imports Microsoft.VisualBasic
Imports System
Imports NTSInformatica.CLN__STD
Imports NTSInformatica
Imports NTSInformatica.FRM__PAGA

Public Class BN__PAGAVBS
    Implements INT__SCRIPT

  dim frmParent as FRM__PAGA

  Public Function Exec(ByVal strCommand As String, byref oApp as Object, ByRef
oIn As Object, ByRef oParam As Object) as Object Implements INT__SCRIPT.Exec
    Try

      if strCommand = "InitControls" then InitControls(oIn, oApp)
      if strCommand = "edTb_concassp_Validated" then return
edTb_concassp_Validated(oIn, oApp)
      if strCommand = "edtb_speinca_Bold" then return edtb_speinca_Bold(oIn,
oApp)
      return nothing

    Catch ex As Exception
      MsgBox(ex.Message)
      return nothing
    End Try
  End Function

  Sub InitControls(ByRef oIn As Object, ByRef oApp As Object)
    Try
      '-------------------------------------------------
      'inserire gli NTSSetParam per i controlli di tipo Source Extender


    Catch ex As Exception
      Dim strErr As String = CLN__STD.GestError(ex, nothing, "", oApp.InfoError,
oApp.ErrorLogFile, True)
    End Try
  End Sub

  Function edTb_concassp_Validated(ByRef oIn As Object, ByRef oApp As Object) as
Object
     if oIn.Text <= 1000 Then
        oApp.MsgBoxErr("Inserire solo codici > 1000")
        oIn.Text = 1038
        oIn.focus()
      end if
  End Function
```

```
  Function edtb_speinca_Bold(ByRef oIn As Object, ByRef oApp As Object) as
Object
     if oApp.User.Nome = "xxx" then
        return -1
     else
        return 0
     end if
  End Function

End Class
]]></scriptCode>
</nts>
```

NOTE

Da BUSINES NET 2009 la creazione del file .NTS può essere fatta in modo automatico entrando nella form interessata, quindi premere CTRL+SHIFT+ALT+F11

una volta modificato il file di script, per renderlo attivo basta uscire e rientrare dal programma (UI) che lo chiama, non è necessario uscire e rientrare da Bus.net.

Quando da un child occorre lanciare uno script basta utilizzare la funzione pubblica NTSScriptExec che chiama l'oggetto script solo se è diverso da nothing.

Da un entity prima di chiamare lo script bisogna accertarsi che non sia nothing! (anche se in un'entità lo script dovrebbe essere una cosa molto limitata).

A livello di 'source extender' per quanto riguarda l'UI, i seguenti eventi richiamano automaticamente una NTSScriptExec, senza bisogno che da codice si faccia nulla. La funzione chiamata si chiamerà come il controllo + '_' + evento

`Form.GestisciEventiEntity`	GestisciEventiEntity
`Form.InitControls`	InitControls
`Form.ActivatedFirst`	Nome_form_ActivatedFirst
`Form.FormClosed`	Nome_form_FormClosed
`TextBoxNum.Validated`	Controllo'_Validated'
`TextBoxStr.Validated`	Controllo'_Validated'
`TextBoxData.Validated`	Controllo'_Validated'
`ComboBox. SelectedValueChanged`	Controllo'_Changed'
`ListBox.SelectedValueChanged`	Controllo'_Changed'
`CheckBox. CheckedChanged`	Controllo'_CheckedChanged'
`RadioButton.CheckedChanged`	Controllo'_CheckedChanged'
`Button.Click`	Controllo'_Click'
`MenuItem.Click`	Controllo'_Click'
`ToolBarButton.Click`	Controllo'_Click'
`TabSTripItem. SelectedPageChanged`	Controllo'_SelectedPageChanged'
`DataGridview.Enter`	Controllo'_Enter'
`DataGridview.Leave`	Controllo'_Leave'
`DataGridview.NTSRowColChange`	Controllo'_NTSRowColChange'
`DataGridColNum.Validating`	Controllo_NomeColonna'_Validating'
`DataGridColStr.Validating`	Controllo_NomeColonna'_Validating'
`DataGridColData.Validating`	Controllo_NomeColonna'_Validating'
`DataGridColCombo.Validating`	Controllo_NomeColonna'_Validating'
`DataGridColCheck.Validating`	Controllo_NomeColonna'_Validating'

Per i BindingSource presenti nella form, al collegamento dei controlli non griglia a databinding tramite la funzione 'NTSFormAddDataBinding' (per le griglie quando viene settata la proprietà griglia.Datasoruce = …) ed al cambio di posizione nel databinding viene scatenato l'evento "TABELLA_NTSDataBindingPositionChanged " e come parametro viene passato il numero identificante la posizione del record.

Esempio per controllo non griglia o per l'oggetto griglia:

```
NTSScriptExec(Me.Name.ToString + "_Leave", oApp, Me, Nothing)
```

Esempio per colonna griglia:

```
NTSScriptExec(Me.View.Name.ToString + "_" +
dtvTmp.Columns(e.ColumnIndex).Name + "_Validating", oApp, Me,
CType(dtvTmp.Columns(e.ColumnIndex).Name, Object))
```

NB: il dato da validate sarà contenuto in: **frmParent.*nome_gridview*.EditingValue**

Significato dei parametri:

nome_del_controllo '_leave', oppure per le colonne della griglia, nome_della_griglia '_' nome_della_colonna '_leave', oggetto oApp, oggetto da cui esco (quindi, a seconda, textbox, commandbutton, griglia, …). Come quarto parametro, solo per la griglia, viene passato il nome della colonna che ho appena validato.

Nome_della_griglia '_NTSRowColChange', come 'oParam' viene passata una stringa contenente nome_colonna_lasciata + ';' + numero_riga_lasciata.

Sempre a livello di child UI, viene chiamata una NTSScriptExec alla fine della Initcontrols per poter caricare gli NTSSetParam dei controlli caricati con source extender (e questo va fatto a mano in ogni programma!!!!), il framework non può farlo in automatico.

Alla luce di quanto sopra, per eseguire un test personalizzato all'uscita da un controllo basta creare la routine specifica all'interno del file di script, quindi modificare il contenuto della routine Exec per testare se strParam = nome_funzione_.... allora chiama la routine personalizzata ….

1.2. Esempio su Gestione ordini (BNORGSOR):

dopo aver inserito in testord la colonna td_hhprova di tipo 'int' default 0 ed in movord la colonna mo_hhprova2 di tipo varchar(18) nullabile ed aver agganciato tramite source extender la colonna di testord ad in NTSTextBoxNum di nome edEt_hhprova e la colonna di movord alla griglia con nome colonna ec_hhprova2:

```
Public Function Exec(ByVal strCommand As String, byref oApp as Object, ByRef oIn
As Object, ByRef oParam As Object) as Object Implements INT__SCRIPT.Exec
    Try
      select case strCommand
          case "edEt_hhprova_Validated": return edEt_hhprova_Validated(oIn,
oApp)
          case "grvRighe_ec_hhprova2_Validating": return
ec_hhprova2_Validating(oIn, oApp, oParam)
      end select
      return nothing

    Catch ex As Exception
      MsgBox(ex.Message)
      return nothing
    End Try
  End Function


Function edEt_hhprova_Validated(ByRef oIn As Object, ByRef oApp As Object) as
Object
    dim dttTmp as new datatable
    Try
     if oIn.text <> 0 then
       frmParent.oClegsor.ocldgsor.valcodicedb(oin.text,
frmParent.DittaCorrente, "ANAGRA", "N", "", dttTmp)
       if dttTmp.rows.count = 0 then
         msgbox ("Conto inesistente")
         oIn.Text = "0"
       else
         if dtttmp.rows(0)!an_tipo.tostring <> "C" then
           msgbox ("Conto non cliente")
           oIn.Text = "0"
         else
           msgbox (dttTmp.rows(0)!an_descr1.tostring)
         end if
       end if
     end if

     return nothing
    Catch ex As Exception
      Dim strErr As String = CLN__STD.GestError(ex, nothing, "", oApp.InfoError,
oApp.ErrorLogFile, True)
    End Try
  End Function


Function ec_hhprova2_Validating(ByRef oIn As Object, ByRef oApp As Object, ByRef
oParam As Object) as Object
    dim dttTmp as new datatable
    Try
      if frmParent.grvRighe.EditingValue.tostring.trim <> "" then


frmParent.oClegsor.ocldgsor.valcodicedb(frmParent.grvRighe.EditingValue.tostring
, frmParent.DittaCorrente, "ARTICO", "S", "", dttTmp)
```

```
        if dttTmp.rows.count = 0 then
          msgbox ("articolo inesistente")
          frmParent.grvRighe.SetFocusedValue("D")
        else
            msgbox (dttTmp.rows(0)!ar_descr.tostring)
        end if
      end if
      return nothing
    Catch ex As Exception
      Dim strErr As String = CLN__STD.GestError(ex, nothing, "", oApp.InfoError,
oApp.ErrorLogFile, True)
    End Try
  End Function
```

Lo stesso risultato (validazione dei controlli) si sarebbe ottenuto ereditando BEORGSOR (es BFORGSOR) ed all'interno della nuova classe inserire le seguenti funzioni (riga da aggiungere in DLLMAP.INI: BNORGSOR|BEORGSOR|BFORGSOR|NtsInformatica.CLFORGSOR):

```
Public Class CLFORGSOR
  Inherits CLEORGSOR
  Public Overridable Function BeforeColUpdate_TESTA_et_hhprova(ByVal sender As
Object, ByVal e As DataColumnChangeEventArgs) As Boolean
    Dim dttTmp As New DataTable
    Try
      If NTSCInt(e.ProposedValue) <> 0 Then
        oCldGsor.ValCodiceDb(NTSCInt(e.ProposedValue).ToString,
strDittaCorrente, "ANAGRA", "N", "", dttTmp)
        If dttTmp.Rows.Count = 0 Then
          ThrowRemoteEvent(New NTSEventArgs(CLN__STD.ThMsg.MSG_INFO, "conto
inesistente"))
          e.ProposedValue = e.Row!et_hhprova
        Else
          If dttTmp.Rows(0)!an_tipo.ToString <> "C" Then
            ThrowRemoteEvent(New NTSEventArgs("", oApp.Tr(Me,
128776348314402809, "Conto non cliente")))
            e.ProposedValue = e.Row!et_hhprova
          Else
            ThrowRemoteEvent(New NTSEventArgs("", oApp.Tr(Me,
128776348430934809, dttTmp.Rows(0)!an_descr1.ToString)))
          End If
        End If
      End If
      Return True
    Catch ex As Exception
      '-------------------------------------------------------------
      If GestErrorCallThrow() Then
        Throw New NTSException(GestError(ex, Me, "", oApp.InfoError, "", False))
      Else
        ThrowRemoteEvent(New NTSEventArgs("", GestError(ex, Me, "",
oApp.InfoError, "", False)))
      End If
      '-------------------------------------------------------------
    End Try
  End Function

  Public Overridable Function BeforeColUpdate_CORPO_ec_hhprova2(ByVal sender As
Object, ByVal e As DataColumnChangeEventArgs) As Boolean
    Dim dttTmp As New DataTable
    Try
      If NTSCStr(e.ProposedValue).Trim <> "" Then
```

```
        oCldGsor.ValCodiceDb(NTSCStr(e.ProposedValue), strDittaCorrente,
"ARTICO", "S", "", dttTmp)
        If dttTmp.Rows.Count = 0 Then
          ThrowRemoteEvent(New NTSEventArgs(CLN__STD.ThMsg.MSG_INFO, "articolo
inesistente"))
          e.ProposedValue = e.Row!ec_hhprova2
        Else
          ThrowRemoteEvent(New NTSEventArgs("", oApp.Tr(Me, 128776348430934809,
dttTmp.Rows(0)!ar_descr.ToString)))
        End If
      End If
      Return True
    Catch ex As Exception
      '-------------------------------------------------------------
      If GestErrorCallThrow() Then
        Throw New NTSException(GestError(ex, Me, "", oApp.InfoError, "", False))
      Else
        ThrowRemoteEvent(New NTSEventArgs("", GestError(ex, Me, "",
oApp.InfoError, "", False)))
      End If
      '-------------------------------------------------------------
    End Try
  End Function
End Class
```

A livello di programmazione, nelle UI/ENTITY tutti gli oggetti grafici e le variabili di form devono essere dichiarate come PUBLIC, diversamente lo script non può accedervi!

Per utilizzare lo script nella configurazione delle maschere (GCTL - uiconf) bisogna impostare sulla griglia 'usa script', quindi indicare nell'apposito spazio in nome della routine che deve essere eseguita all'interno dello script. La routine va scritta nel file .NTS che ha lo stesso nome del child. Ovviamente la funzione deve restituire un valore esattamente come se lo mettessi fisso (o/-1, S/N, per il testo fisso un testo/numero …)

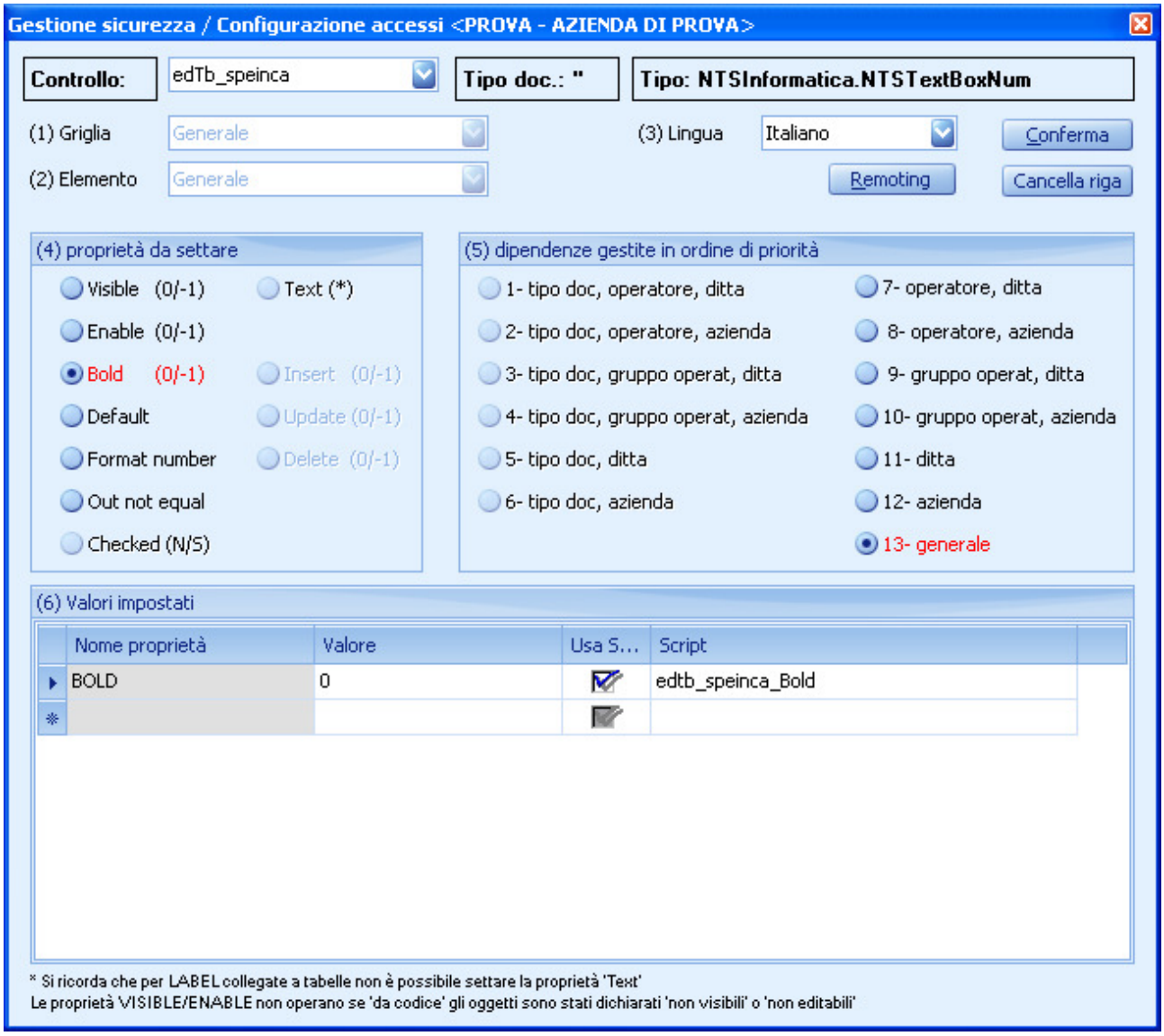
Esempio: (se c'è lo script verrà ignorato il contenuto del campo 'valore')

Particolarità:

Se nel file .NTS è presente la riga

```
<use assembly="BD__PIPPO.DLL"/>
```

il compilatore della DLL non compila il testo contenuto nel file.NTS (che potrebbe essere anche assente) ma istanzia semplicemente la dll pippo (che deve sempre implementare la INT__SCRIPT) contenuta nella stringa e la passa ai vari UI/entity; come dire: il codice personalizzato non è in un file ma in una dll precompilata. L'importante è che la pippo.dll sia compilata con opzione 'OPTION STRICT OFF' e deve implementare il namespace 'NTSInformatica'.


Gestione sicurezza / Configurazione accessi <PROVA - AZIENDA DI PROVA>
Controllo:
edTb_speinca
Tipo doc.: ''
Tipo: NTSInformatica.NTSTextBoxNum
(1) Griglia
Generale
(2) Elemento
Generale
(3) Lingua
Italiano
Conferma
Remoting
Cancella riga
(4) proprietà da settare
Visible (0/-1)
Text (*)
Enable (0/-1)
Bold (0/-1)
Insert (0/-1)
Default
Update (0/-1)
Format number
Delete (0/-1)
Out not equal
Checked (N/S)
(5) dipendenze gestite in ordine di priorità
1- tipo doc, operatore, ditta
2- tipo doc, operatore, azienda
3- tipo doc, gruppo operat, ditta
4- tipo doc, gruppo operat, azienda
5- tipo doc, ditta
6- tipo doc, azienda
7- operatore, ditta
8- operatore, azienda
9- gruppo operat, ditta
10- gruppo operat, azienda
11- ditta
12- azienda
13- generale
(6) Valori impostati
Nome proprietà
Valore
Usa S...
Script
BOLD
0
edtb_speinca_Bold
* Si ricorda che per LABEL collegate a tabelle non è possibile settare la proprietà 'Text'
Le proprietà VISIBLE/ENABLE non operano se 'da codice' gli oggetti sono stati dichiarati 'non visibili' o 'non editabili'

2. Esempi Script complessi in Bus.net

2.1. Esempio campo aggiuntivo (in anagrafica clienti) e relativa decodifica

```
<nts>
<use_ assembly="BN__SCTE.DLL"/>

<reference assembly="BD__BASE.dll" />
<reference assembly="BE__MENU.dll" />
<reference assembly="BN__CLIE.dll" />

<scriptCode><![CDATA[

Imports Microsoft.VisualBasic
Imports System
Imports System.Data
Imports NTSInformatica.CLN__STD
Imports NTSInformatica
Imports NTSInformatica.FRM__CLIE

Imports NTSInformatica.CLD__BASE
Imports System.Data.SqlClient

Public Class BN__CLIEVBS
    Implements INT__SCRIPT

  dim frmParent as FRM__CLIE
  dim desc as string = " "

  Public Function Exec(ByVal strCommand As String, byref oApp as Object, ByRef
oIn As Object, ByRef oParam As Object) as Object Implements INT__SCRIPT.Exec
    Try
      'elenco eventi intercettati da script
      if strCommand = "InitControls" then InitControls(oIn, oApp)
        if strCommand = "edAn_vett3_Validated" then return
edAn_vett3_Validated(oIn, oApp)
      if strCommand = "tlbApri_Click" then return tlbApri_Click(oIn, oApp)
      if strCommand = "tlbPrimo_Click" then return tlbPrimo_Click(oIn, oApp)
      if strCommand = "tlbPrecedente_Click" then return tlbPrecedente_Click(oIn,
oApp)
      if strCommand = "tlbSuccessivo_Click" then return tlbSuccessivo_Click(oIn,
oApp)
      if strCommand = "tlbUltimo_Click" then return tlbUltimo_Click(oIn, oApp)
      return nothing

    Catch ex As Exception
      MsgBox(ex.Message)
      return nothing
    End Try
  End Function

  Sub InitControls(ByRef oIn As Object, ByRef oApp As Object)
    Try
      '-------------------------------------------------
      'inserire gli NTSSetParam per i controlli di tipo Source Extender

      frmParent = oIn

    Catch ex As Exception
      Dim strErr As String = CLN__STD.GestError(ex, nothing, "", oApp.InfoError,
oApp.ErrorLogFile, True)
    End Try
```

```
  End Sub

  Function edAn_vett3_Validated(ByRef oIn As Object, ByRef oApp As Object) as
Object
    Try
      '-------------------------------------------------
      'prova accesso a database da SCRIPT
      'accesso senza utilizzo del DAL base
      'posso ottenere la decodifica leggendo direttamente il database

      Dim dbConn As SqlConnection = Nothing
      Dim dbCmd as SqlCommand = nothing
      Dim da as SQlDataadapter = nothing
      Dim dttTmp as new Datatable
      dbConn = new SqlConnection(oApp.Db.Connect)
      dbCmd = New SqlCommand("SELECT tb_desvett FROM tabvett where tb_codvett ="
& oIn.text , dbConn)
      da = New SqlDataAdapter(dbCmd)
      da.Fill(dttTmp)
      if (dttTmp.rows.count > 0) then
            desc = dttTmp.rows(0)!tb_desvett.tostring()
      else
      desc = ""
      end if
      frmParent.pnIndirDx.controls("lbXx_vett3").Text = desc
      dttTmp.clear()
      dbConn.close

    Catch ex As Exception
      Dim strErr As String = CLN__STD.GestError(ex, nothing, "", oApp.InfoError,
oApp.ErrorLogFile, True)
    End Try
  End Function

  Function tlbApri_Click(ByRef oIn As Object, ByRef oApp As Object) as Object
    Try
      DecodificaCampoAggiuntivo(oIn, oApp)

    Catch ex As Exception
      Dim strErr As String = CLN__STD.GestError(ex, nothing, "", oApp.InfoError,
oApp.ErrorLogFile, True)
    End Try
  End Function

  Function tlbPrimo_Click(ByRef oIn As Object, ByRef oApp As Object) as Object
    Try
      DecodificaCampoAggiuntivo(oIn, oApp)

    Catch ex As Exception
      Dim strErr As String = CLN__STD.GestError(ex, nothing, "", oApp.InfoError,
oApp.ErrorLogFile, True)
    End Try
  End Function

  Function tlbPrecedente_Click(ByRef oIn As Object, ByRef oApp As Object) as
Object
    Try
      DecodificaCampoAggiuntivo(oIn, oApp)

    Catch ex As Exception
      Dim strErr As String = CLN__STD.GestError(ex, nothing, "", oApp.InfoError,
oApp.ErrorLogFile, True)
```

```
    End Try
  End Function

  Function tlbSuccessivo_Click(ByRef oIn As Object, ByRef oApp As Object) as
Object
    Try
      DecodificaCampoAggiuntivo(oIn, oApp)

    Catch ex As Exception
      Dim strErr As String = CLN__STD.GestError(ex, nothing, "", oApp.InfoError,
oApp.ErrorLogFile, True)
    End Try
  End Function

  Function tlbUltimo_Click(ByRef oIn As Object, ByRef oApp As Object) as Object
    Try
      DecodificaCampoAggiuntivo(oIn, oApp)

    Catch ex As Exception
      Dim strErr As String = CLN__STD.GestError(ex, nothing, "", oApp.InfoError,
oApp.ErrorLogFile, True)
    End Try
  End Function

  Sub DecodificaCampoAggiuntivo(ByRef oIn As Object, ByRef oApp As Object)
    Try
      '-------------------------------------------------
      'prova accesso a database da SCRIPT
      'accesso senza utilizzo del DAL base

      Dim dbConn As SqlConnection = Nothing
      Dim dbCmd as SqlCommand = nothing
      Dim da as SQlDataadapter = nothing
      Dim dttTmp as new Datatable
      dbConn = new SqlConnection(oApp.Db.Connect)
      dbCmd = New SqlCommand("SELECT tb_desvett FROM tabvett where tb_codvett ="
& frmParent.pnIndirDx.controls("edAn_vett3").Text , dbConn)
      da = New SqlDataAdapter(dbCmd)
      da.Fill(dttTmp)
      if (dttTmp.rows.count > 0) then
            desc = dttTmp.rows(0)!tb_desvett.tostring()
      else
      desc = ""
      end if
      frmParent.pnIndirDx.controls("lbXx_vett3").Text = desc
      dttTmp.clear()
      dbConn.close

    Catch ex As Exception
      Dim strErr As String = CLN__STD.GestError(ex, nothing, "", oApp.InfoError,
oApp.ErrorLogFile, True)
    End Try
  End Sub

End Class

]]></scriptCode>
</nts>
```

2.2. Esempio gestione controlli inseriti in contenitori

```
<nts>
<use_ assembly="BN__SCTE.DLL"/>

<reference assembly="BD__BASE.dll" />
<reference assembly="BE__MENU.dll" />
<reference assembly="BN__CLIE.dll" />

<scriptCode><![CDATA[

Imports Microsoft.VisualBasic
Imports System
Imports System.Data
Imports NTSInformatica.CLN__STD
Imports NTSInformatica
Imports NTSInformatica.FRM__CLIE

Imports NTSInformatica.CLD__BASE
Imports System.Data.SqlClient

Public Class BN__CLIEVBS
  Implements INT__SCRIPT

  Dim frmParent As FRM__CLIE

  Public Function Exec(ByVal strCommand As String, ByRef oApp As Object, ByRef
oIn As Object, ByRef oParam As Object) As Object Implements INT__SCRIPT.Exec
    Try

      If strCommand = "InitControls" Then InitControls(oIn, oApp)
      If strCommand = "edAn_prefiban_Default" Then Return
edAn_prefiban_Default(oIn, oApp)

    Catch ex As Exception
      MsgBox(ex.Message)
    End Try
  End Function

  Sub InitControls(ByRef oIn As Object, ByRef oApp As Object)
    Try
      '-------------------------------------------------
      'inserire gli NTSSetParam per i controlli di tipo Source Extender

      frmParent = oIn

    Catch ex As Exception
      Dim strErr As String = CLN__STD.GestError(ex, Nothing, "", oApp.InfoError,
oApp.ErrorLogFile, True)
    End Try
  End Sub

  Public Function edAn_prefiban_Default(ByRef oIn As Object, ByRef oApp As
Object) As Object
    Try
      'setto il valore di default per i nuovi campi
      'Return 1
      'posso modificare il contenuto dei controlli accedendo
      'direttamente alla form.contenitore.elencocontrollidelcontenitore
      'I contenitori sono Panel Groupbox TabControl
      'posso ricercare il nome di un controllo per leggerne il valore
      'Dim ctrl As Object = CType(oIn.FindForm,
FRM__CHIL).NTSFindControlByName(oIn.FindForm, "edAn_prefiban")
      'if ctrl.Text = "12" Then
```

```
      '     Return 12
      'End If

    Catch ex As Exception
      Dim strErr As String = CLN__STD.GestError(ex, Nothing, "", oApp.InfoError,
oApp.ErrorLogFile, True)
    End Try
  End Function

End Class

]]></scriptCode>
</nts>
```

2.3. Esempio gestione controlli su più form

```
<nts>
<use_ assembly="BN__SCTE.DLL"/>

<reference assembly="BD__BASE.dll" />
<reference assembly="BE__MENU.dll" />
<reference assembly="BN__CLIE.dll" />

<scriptCode><![CDATA[

Imports Microsoft.VisualBasic
Imports System
Imports System.Data
Imports NTSInformatica.CLN__STD
Imports NTSInformatica
Imports NTSInformatica.FRM__CLIE

Imports NTSInformatica.CLD__BASE
Imports System.Data.SqlClient

Public Class BN__CLIEVBS
  Implements INT__SCRIPT

  Dim frmClie As FRM__CLIE
  Dim frmNuov As FRM__NUOV

  Public Function Exec(ByVal strCommand As String, ByRef oApp As Object, ByRef
oIn As Object, ByRef oParam As Object) As Object Implements INT__SCRIPT.Exec
    Try

      If strCommand = "InitControls" Then InitControls(oIn, oApp)
      If strCommand = "edProgr_Validated" Then Return edProgr_Validated(oIn,
oApp)
      Return Nothing

    Catch ex As Exception
      MsgBox(ex.Message)
      Return Nothing
    End Try
  End Function

  Sub InitControls(ByRef oIn As Object, ByRef oApp As Object)
    Try
      '-------------------------------------------------
      'inserire gli NTSSetParam per i controlli di tipo Source Extender
```

```
      'nel caso di presenza di più form all'interno della stessa dll
      'la funzione Initcontrols scatta all'apertura della form principale
      'e delle modali all'interno della stessa dll
      'va testato quale form si sta eseguendo
      'si memorizza un oggetto form per accedere ai controlli grafici della form
      'attualmente in esecuzione
      if oIn.Name = "FRM__CLIE" then
            frmClie = oIn
      elseif In.Name = "FRM__NUOV" then
            frmNuov = oIn
      end if

    Catch ex As Exception
      Dim strErr As String = CLN__STD.GestError(ex, Nothing, "", oApp.InfoError,
oApp.ErrorLogFile, True)
    End Try
  End Sub

  Public Function edProgr_Validated(ByRef oIn As Object, ByRef oApp As Object)
As Object
    Try
      frmNuov.pnNuovo.controls("edProgr").Text = "456"
      Return Nothing

    Catch ex As Exception
      Dim strErr As String = CLN__STD.GestError(ex, Nothing, "", oApp.InfoError,
oApp.ErrorLogFile, True)
      Return Nothing
    End Try
  End Function

End Class

]]></scriptCode>
</nts>
```

2.4. Esempio di lancio zoom con parametri, generare eventi nel entity

```
DESTINAZ DIVE CON PARAMETRO CONTO

Riga sotto da inserire nel entity personalizzato, la chiamata è sincrona
Il codice comune tra entity e script è solitamente di 10 caratteri ad es.
ELABORAZI: dopo tale stringa possono essere passati dei parametri suddivisi da |
      Dim evnt As NTSEventArgs
      evnt = New NTSEventArgs("ELABORAZI:", "")
      ThrowRemoteEvent(evnt) 'chiamata allo script
      If NTSCInt(evnt.RetValue) = 12 Then Return False

<nts>
<use_ assembly="BN__SCTE.DLL"/>

<reference assembly="BD__BASE.dll" />
<reference assembly="BE__MENU.dll" />
<reference assembly="BN__PAGA.dll" />

<scriptCode><![CDATA[

Imports Microsoft.VisualBasic
Imports System
Imports System.Data
Imports NTSInformatica.CLN__STD
```

```
Imports NTSInformatica
Imports NTSInformatica.FRM__PAGA

Imports NTSInformatica.CLD__BASE
Imports System.Data.SqlClient

Public Class BN__PAGAVBS
    Implements INT__SCRIPT

  dim frmParent as FRM__PAGA
  dim desc as string = " "

  Public Function Exec(ByVal strCommand As String, byref oApp as Object, ByRef
oIn As Object, ByRef oParam As Object) as Object Implements INT__SCRIPT.Exec
    Try

      if strCommand = "InitControls" then InitControls(oIn, oApp)
if strCommand = "GestisciEventiEntity" then GestisciEventiEntity(oIn, oApp,
oParam)
      if strCommand = "tlbZoom_Click" then return tlbZoom_Click(oIn, oApp)
      return nothing

    Catch ex As Exception
      MsgBox(ex.Message)
      return nothing
    End Try
  End Function

  Sub InitControls(ByRef oIn As Object, ByRef oApp As Object)
    Try
      '-------------------------------------------------
      'inserire gli NTSSetParam per i controlli di tipo Source Extender

      frmParent = oIn

    Catch ex As Exception
      Dim strErr As String = CLN__STD.GestError(ex, nothing, "", oApp.InfoError,
oApp.ErrorLogFile, True)
    End Try
  End Sub

  Sub GestisciEventiEntity(ByRef oIn As Object, ByRef oApp As Object, ByRef
oParam as Object)
    Try
      'può essere intercettato durante un elaborazione del entity BE per
      'generare un evento verso lo script
      'ad esempio per visualizzare una modale di scelta durante un elaborazione
      'del entry BE

      dim evnt as NTSEventArgs = oParam
      If evnt.Message.Substring(0,10) = "ELABORAZI:" Then
            evnt.RetValue = "CCC" 'posso restituire un valore al entity BE
            'oApp.MsgBoxErr("GestisciEventiEntity personalizzato: " &
evnt.Message & " - " & evnt.TipoEvento &  " - Restituito valore 'ccc' - Codice
passato " & evnt.Message.Substring(11))
      End If

    Catch ex As Exception
      Dim strErr As String = CLN__STD.GestError(ex, nothing, "", oApp.InfoError,
oApp.ErrorLogFile, True)
    End Try
  End Sub
```

```
  Function tlbZoom_Click(ByRef oIn As Object, ByRef oApp As Object) as Object
    Dim oParam As New CLE__PATB
    Try
      frmParent.ValidaLastControl()
        frmParent.NTSZOOM.strIn = frmParent.edTb_123.Text
      'parametri
      oParam.lContoCF = 1

        frmParent.NTSZOOM.ZoomStrIn("ZOOMDESTDIV", frmParent.DittaCorrente,
oParam)
        If frmParent.NTSZOOM.strIn <> frmParent.edTb_123.Text Then
frmParent.edTb_123.NTSTextDB = frmParent.NTSZOOM.strIn

    Catch ex As Exception
      Dim strErr As String = CLN__STD.GestError(ex, nothing, "", oApp.InfoError,
oApp.ErrorLogFile, True)
    End Try
  End Function

End Class

]]></scriptCode>
</nts>
```

2.5. Esempio di un pulsante toolbar che richiama form su altra dll

```
'Tramite abilita/disabilita editing si può inserire un ToolBar Item (voce di
toolbar) trascinando sulla toolbar e settando il nome ad es: tlbPulsanteModale
'per poter richiamare una form vb6/net presente su una dll personalizzata

<nts>
<use_ assembly="BN__SCTE.DLL"/>

<reference assembly="BD__BASE.dll" />
<reference assembly="BE__MENU.dll" />
<reference assembly="BN__PAGA.dll" />

<scriptCode><![CDATA[

Imports Microsoft.VisualBasic
Imports System
Imports System.Data
Imports NTSInformatica.CLN__STD
Imports NTSInformatica
Imports NTSInformatica.FRM__PAGA
Imports NTSInformatica.CLE__CLDP

Imports NTSInformatica.CLD__BASE
Imports System.Data.SqlClient

Public Class BN__PAGAVBS
    Implements INT__SCRIPT

  dim frmParent as FRM__PAGA
  dim desc as string = " "

  Public Function Exec(ByVal strCommand As String, byref oApp as Object, ByRef
oIn As Object, ByRef oParam As Object) as Object Implements INT__SCRIPT.Exec
    Try

      if strCommand = "InitControls" then InitControls(oIn, oApp)
```

```
      if strCommand = "tlbPulsanteModale_Click" then return
tlbPulsanteModale_Click(oIn, oApp)
      return nothing

    Catch ex As Exception
      MsgBox(ex.Message)
      return nothing
    End Try
  End Function

  Sub InitControls(ByRef oIn As Object, ByRef oApp As Object)
    Try
      '-------------------------------------------------
      'inserire gli NTSSetParam per i controlli di tipo Source Extender

      frmParent = oIn

    Catch ex As Exception
      Dim strErr As String = CLN__STD.GestError(ex, nothing, "", oApp.InfoError,
oApp.ErrorLogFile, True)
    End Try
  End Sub

  Function tlbPulsanteModale_Click(ByRef oIn As Object, ByRef oApp As Object) as
Object
    'Dim oPar As CLE__CLDP = Nothing
    Try
      'lettura/scrittura di una variabile esistente o nuova del entity
      'frmParent.oClePaga.lCodPaga = 2

      'lancio programma vb6 personalizzato
      'frmParent.oMenu.RunChild("BSHHPERS", "CLSHHPERS", "Programma
personalizzato", frmParent.DittaCorrente, "", "", Nothing, "", True, True)

      'lancio programma net personalizzato
      'oPar = New CLE__CLDP
      'oPar.Ditta = frmParent.DittaCorrente
      'oPar.strNomProg = "BN__PAGA" 'programma chiamante
      'frmParent.oMenu.RunChild("NTSInformatica", "FRMHHPERS", "",
frmParent.DittaCorrente, "", "BNHHPERS", oPar, "", True, True)

    Catch ex As Exception
      Dim strErr As String = CLN__STD.GestError(ex, nothing, "", oApp.InfoError,
oApp.ErrorLogFile, True)
    End Try
  End Function

End Class

]]></scriptCode>
</nts>
```

2.6. Esempio di un lancio di una dll compilata di script

Se si vuole inglobare il codice di scripting in una dll chiusa compilata
Si può prendere spunto dalla dll BN__SCTE il nome BN__SCTE è modificabile

```
<nts>
<use assembly="BN__SCTE.DLL"/>
</nts>
```

2.7. Esempio di aggiunta di un nuovo evento

Il frame work di business non passa allo script tutti gli eventi che avvengono a livello di interfaccia. Se si vuole far gestire allo script uno evento non disponibile è possibile farlo aggiungendo nell'InitControllo l'Handler dell'evento ad un metodo dello script.

```
Sub InitControls(ByRef oIn As Object, ByRef oApp As Object)
  Try
      frmParent = ctype(oIn, FRM__CHIL)

      AddHandler frmParent.grvRighe.CustomDrawCell, AddressOf
      grvRighe_CustomDrawCell

  Catch ex As Exception
            Dim strErr As String = CLN__STD.GestError(ex, nothing, "",
      oApp.InfoError, oApp.ErrorLogFile, True)
  End Try
End Sub
```

Il metodo associato dovrà accettare i parametri corretti passati dall'evento.
In questo caso:

```
Sub grvRighe_CustomDrawCell(ByVal sender As System.Object, ByVal e As
DevExpress.XtraGrid.Views.Base.RowCellCustomDrawEventArgs)
```

2.8. Esempio di corretto collegamento di un DataSet ad una griglia

Quando si cambia setta il datatable al bindingsource che successivamente dovrà essere agganciato al datasource della griglia, prima bisogna SEMRE scollegare il bindingsource dalla griglia stessa, diversamente possono verificarsi degli errori tipo:

errore in apertura distinta base 'La colonna 'dd_riga' non appartiene alla tabella CICLI. (error type: ArgumentException)', l'errore si verificava nella routine 'grvCicli_NTSFocusedRowChanged'

ad esempio:
in BNDBDIBA la funzione apri legge il datatable dsCicli, lo collega al bindingsource dcCicli e successivamente il dcCicli viene collegato alla griglia grCicli. Alla prima apertura di distinta base tutto opera correttamente, visto che quando viene settato il dcCicli quest'ultimo non è ancora collegato alla griglia.

A questo punto mi posiziono nella riga 4 della griglia dei cicli
Senza uscire dal programma, utilizzo ancora la funzione apri per aprire un'altra DB che ha solo 1 lavorazione:

nel momento in cui si esegue il comando
dcCicli.DataSource = dsCicli.Tables("CICLI"), se precedentemente non è stato settato il comando grCicli.DataSource = Nothing viene scatenato un errore

la sintassi corretta è:
grCicli.DataSource = Nothing
dcCicli.DataSource = dsCicli.Tables("CICLI")
grCicli.DataSource = dcCicli

3. Scripting nelle stampe

Rispetto alle funzioni di scripting (ReportSelectCond, ReportNameCond, PrinterNameCond, CopiesCond e ScriptPrint) di Business Classic Start è tutto cambiato, tanto che le vecchie funzioni di scripting non sono più utilizzate. Del vecchio sistema **rimane solo l'opzione nella ReportsX 'Scripting = 0/-1' che viene utilizzata per attivare o meno il motore di script in fase di stampa**.

Il motore di script viene caricato nella ReportPeInit, quindi ogni volta che viene lanciata una stampa.

Tale motore usa il file BE__CRPE.NTS che deve essere presente nella script del server (esattamente come per gli altri file di script). Al motore di script viene passato un DAL, la form che ha lanciato la stampa (sempre che la CE__CRPE non sia stata istanziata da un entità, in questo caso la form è 'nothing') ed i parametri canonici 'nome del programma', 'nome della reportsx' in uso, ecc…

UN ESEMPIO DETTAGLIATO DI COSA SI PUO' FARE E' DISPONIBILE APRENDO IL FILE BE__CRPE.NTS. L'esempio è stato predisposto per BN__PAGA, ma vale per tutti i child.

Nel file BE__CRPE.NTS (un file per tutti i childs) sarà presente la routine 'EXEC' che rimanderà a tante funzioni specifiche DA SCRIVERE a seconda del programma e del report chiamante.

Se l'opzione 'Scripting' per un programma è stata attivata, per le funzioni ReportName, PrinterName, Copies, ReportSelected e **PaperSize**, viene prima cercato nel file BE__CRPE se presente l'apposita funzione condizionata:

```
ReportNameCond
PrinterNameCond
CopiesCond
ReportSelectedCond
PaperSizeCond      (aggiunta in Business NET)
```

se è presente e restituisce un valore diverso da '' viene preso tale valore, diversamente viene eseguita la funzione standard, come se lo script non fosse stato attivato.

Come in Business in VB6, è disponibile una funzione 'ScriptPrint' che viene richiamata prima di far partire la stampa.

Esempio di file di script BE__CRPE.NTS:

```
<nts>
<reference assembly="BD__BASE.dll" />

<scriptCode><![CDATA[

Imports Microsoft.VisualBasic
Imports System
Imports System.Collections.Generic
Imports System.Data
Imports NTSInformatica.CLN__STD
Imports NTSInformatica
Imports NTSInformatica.CLD__BASE

Public Class BE__CRPEVBS
    Implements INT__SCRIPT

  Dim oCldBase As CLD__BASE

  Public Function Exec(ByVal strCommand As String, byref oApp as Object, ByRef
oIn As Object, ByRef oParam As Object) as Object Implements INT__SCRIPT.Exec
    '-----------------------------------
    'PARAMETRI PASSATI DAL CHIAMANTE
    Dim arParam As new List(Of Object)
    'arParam.Add(oFrmParent)       'potrebbe valore nothing se la crpe è stata
                                  'chiamata da un entity, diversamente contiene
                                  'la form che ha lanciato il report
    'arParam.Add(strkey1)          'nome del programma 'BS--PAGA' (occhio:
                                  'bs--paga, non bn--paga!!!)
```

```
    'arParam.Add(strkey2)           'Reports1, Reports2, ...
    'arParam.Add(nIndex)            'Numero del Rep: 1, 2, 3, ... 10
    'arParam.Add(strTipoDoc)        'Tipo documento
    'arParam.Add(bDefault)          'valore di default della proprietà (può
                                    'essere un numero, una stringa o un valore
                                    'boolean): non disponibile per 'ScriptPrint'
    '-------------------------------------
    Try

      oCldBase = New CLD__BASE
      oCldBase.Init(oApp)

      arParam = oIn

      'esempio di come ottenere un dato presente in form al momento del lancio
della stampa
      'oApp.MsgBoxInfo("Sono stato chiamato da '" & cType(arParam(0),
FRM__CHIL).Text & "' ed il codice pagamento attuale è '" &
arParam(0).edTb_Codpaga.Text & "'")

      'esempio di come reperire un dato sul database
      'Dim dttTmp as new DataTable
      'dttTmp = oCldBase.OpenRecordset("Select * FROM tabpaga Where tb_codpaga =
" & arParam(0).edTb_Codpaga.Text, CLE__APP.DBTIPO.DBAZI)
      'if dttTmp.Rows.Count > 0 then
      '   oApp.MsgBoxInfo("Descrizione pagamento: '" &
dttTmp.Rows(0)!tb_despaga.Tostring & "'")
      'end if
      'dttTmp.clear()

      Select Case strCommand.ToUpper()
            case "BS--PAGA_REPORTS1_REP1_REPORTSELECTEDCOND": return
BS__PAGA_REPORTS1_REP1_ReportSelectedCond(CType(arParam(4), String), oApp)
            case "BS--PAGA_REPORTS1_REP1_REPORTNAMECOND": return
BS__PAGA_REPORTS1_REP1_ReportNameCond(CType(arParam(4), String), oApp)
            case "BS--PAGA_REPORTS1_REP1_PRINTERNAMECOND": return
BS__PAGA_REPORTS1_REP1_PrinterNameCond(CType(arParam(4), String), oApp)
            case "BS--PAGA_REPORTS1_REP1_COPIESCOND": return
BS__PAGA_REPORTS1_REP1_CopiesCond(CType(arParam(4), String), oApp)
            case "BS--PAGA_REPORTS1_REP1_PAPERSIZECOND": return
BS__PAGA_REPORTS1_REP1_PaperSizeCond(CType(arParam(4), String), oApp)
            case "BS--PAGA_REPORTS1_REP1_SCRIPTPRINT": return
BS__PAGA_REPORTS1_REP1_ScriptPrint(CType(arParam(4), String), oApp)
      End Select

      return nothing

    Catch ex As Exception
      MsgBox(ex.Message)
      return nothing
    End Try
  End Function

'----------------------------------------
  private function BS__PAGA_REPORTS1_REP1_ReportSelectedCond(ByVal strTipoDoc As
string, ByRef oApp As Object) as string
    Try
      'oApp.MsgBoxInfo("BS--PAGA_REPORTS1_REP1_ReportSelectedCond - '" &
strTipoDoc & "'")
      return "0"

    Catch ex As Exception
```

```
      Dim strErr As String = CLN__STD.GestError(ex, nothing, "", oApp.InfoError,
oApp.ErrorLogFile, True)
    End Try
  end function
'-----------------------------------------
  private function BS__PAGA_REPORTS1_REP1_ReportNameCond(ByVal strTipoDoc As
string, ByRef oApp As Object) as string
    Try
      'non imposto il report: verrà preso quelli in base all'opzione di registro
ReportName se impostato, diversamente quello standard
      return ""

      return "bs--paga.rpt"

    Catch ex As Exception
      Dim strErr As String = CLN__STD.GestError(ex, nothing, "", oApp.InfoError,
oApp.ErrorLogFile, True)
    End Try
  end function

'-----------------------------------------
  private function BS__PAGA_REPORTS1_REP1_PrinterNameCond(ByVal strTipoDoc As
string, ByRef oApp As Object) as string
    Try
      'non imposto la stampante: verrà presa quella predefinita
      'return ""

      'imposto una stampante
      return "HP Laserjet III"

    Catch ex As Exception
      Dim strErr As String = CLN__STD.GestError(ex, nothing, "", oApp.InfoError,
oApp.ErrorLogFile, True)
    End Try
  end function

'-----------------------------------------
  private function BS__PAGA_REPORTS1_REP1_CopiesCond(ByVal strTipoDoc As string,
ByRef oApp As Object) as string
    Try
      return "3"

    Catch ex As Exception
      Dim strErr As String = CLN__STD.GestError(ex, nothing, "", oApp.InfoError,
oApp.ErrorLogFile, True)
    End Try
  end function

'-----------------------------------------
  private function BS__PAGA_REPORTS1_REP1_PaperSizeCond(ByVal strTipoDoc As
string, ByRef oApp As Object) as string
    Try
      'Esempio dei formati (per un elenco completo consultare l'help in linea
alla proprietà PaperSize)
      '8  =  A3 297 x 420 mm
      '9  =  A4 210 x 297 mm (Predefinito)
      '10  = A4 Small 210 x 297 mm
      '11  =  A5 148 x 210 mm
      '12  =  B4 250 x 354 mm
      '13  =  B5 182 x 257 mm
      return "8"
```

```
    Catch ex As Exception
      Dim strErr As String = CLN__STD.GestError(ex, nothing, "", oApp.InfoError,
oApp.ErrorLogFile, True)
    End Try
  end function

'-----------------------------------------
  private function BS__PAGA_REPORTS1_REP1_ScriptPrint(ByVal strTipoDoc As
string, ByRef oApp As Object) as boolean
    Try
      'deve restituire True o False a seconda si voglia o meno
      'annullare il processo di stampa corrente
      return True

    Catch ex As Exception
      Dim strErr As String = CLN__STD.GestError(ex, nothing, "", oApp.InfoError,
oApp.ErrorLogFile, True)
    End Try
  end function

End Class

]]></scriptCode>
</nts>
```

4. Script negli alert

Configurazione

Con Business NET le procedure degli alerts possono essere scritte sia con il 'vecchio' msscript (ovviamente non bisogna essersi loggati in Business NET con l'utente 'xxx' e pwd 'xxx', visto che per funzionare viene richiesto che il motore di Business VB6 sia istanziato) che con il nuovo motore di scripting sopra esposto. In questo modo è garantita la quasi completa compatibilità degli script precedentemente realizzati con Business fino alla versione 13.

In pratica nel momento in cui si realizza/si modifica una procedura per gli alerts, se questa deve utilizzare il nuovo motore di scritp in vb.net è OBBLIGATORIO che il nome della procedura inizi per 'VBNET.' e nel testo della procedura siano presenti le righe 'FILE:' e 'FUNCION:'.

Ad esempio una procedura potrebbe essere la seguente:

codice procedura 999999

nome procedura VBNET.ProvaAlert

testo procedura:

FILE:BECGDCST.NTS

FUNCTION:ProvaAlert

'Deve avere le due dichiarazioni 'FILE:' e 'FUNCTION:'

Nel momento in cui il programma degli script (BE__SIAL.DLL) deve interpretare la procedura compilerà al volo il file becgdcst.nts e, successivamente, cercherà di chiamare tramite 'exec' la routine 'ProvaAlert', passandogli come parametro il codice ditta e il datatable che dovrà essere alimentato con il messaggio da inviare (la struttura del datatable è la stessa di quello già esistente in VB6)

Un esempio del file becgdcst.nts è il seguente:

```
<nts>
<reference assembly="BD__BASE.dll" />

<scriptCode><![CDATA[

Imports Microsoft.VisualBasic
Imports System
Imports System.Collections.Generic
Imports System.Data
Imports NTSInformatica.CLN__STD
Imports NTSInformatica
Imports NTSInformatica.CLD__BASE

Public Class BECGDCSTVBS
    Implements INT__SCRIPT

  '------------------------------------------------------
  'Esempio di procedura VBNET per esecuzione di un alert
  'per poterla richiamare in 'Definizione procedure di Import/Export è stata
creata la procedura 99998
  'VBNET.ProvaAlert (deve iniziare per forza per 'VBNET.') con all'interno il
seguente testo:
  'FILE:BECGDCST.NTS
  'FUNCTION:ProvaAlert
  'ed in impostazione alerts è stato indicato come codice procedura da eseguire
99998
  'il motore degli alert (BE__SIAL) riconosce che la procedura inizia per VBNET.
e richiama questo script
  '(in caso contrario richiala il menu nascosto VB6 per eseguire lo script come
in BUS 13 - compatibilità all'indietro)
```

```
'---------------------------------
  Public Function Exec(ByVal strCommand As String, byref oApp as Object, ByRef
oIn As Object, ByRef oParam As Object) as Object Implements INT__SCRIPT.Exec
    Try
      if strCommand.ToUpper = "PROVAALERT" then return ProvaAlert(ctype(oApp,
CLE__APP), ctype(oIn, string), ctype(oParam, DataTable))
      return nothing

    Catch ex As Exception
      MsgBox(ex.Message)
      return nothing
    End Try
  End Function
'---------------------------------
  Function ProvaAlert(byref oApp as CLE__APP, byval strCodditt as string, byref
dttMsgOutParam as DataTable) as boolean
    Try

     Dim strSQL as string = ""
     Dim dttTmp As New DataTable
     Dim oCldBase as New CLD__BASE
     Dim strMsg as string = ""
     Dim i as integer = 0

     strSQL = "SELECT * FROM artico WHERE ar_codart LIKE 'PF9%' AND codditt = '"
& strCodditt & "'"
     'MsgBox(strSQL)
     oCldBase.Init(oApp)
     dttTmp = oCldBase.OpenRecordset(strSQL,
NTSInformatica.CLE__APP.DBTIPO.DBAZI)
     if dttTmp.Rows.Count > 0 then
       for i = 0 to dttTmp.Rows.count - 1
          strMsg += dttTmp.Rows(i)!ar_codart.tostring + " - " +
dttTmp.Rows(i)!ar_descr.tostring + " trovato " + vbcrlf
       next
       dttTmp.Clear()
     end if
     if strMsg <> "" then
      dttMsgOutParam.rows.add(dttMsgOutParam.newrow())
        dttMsgOutParam.rows(dttMsgOutParam.rows.count - 1)!codditt = strCodditt
        dttMsgOutParam.rows(dttMsgOutParam.rows.count - 1)!strMsg = strMsg
        dttMsgOutParam.AcceptChanges
     end if
     'MsgBox(strMsg)

     return true

    Catch ex As Exception
      MsgBox(ex.Message)
    End Try
  End Function


End Class

]]></scriptCode>
</nts>
```

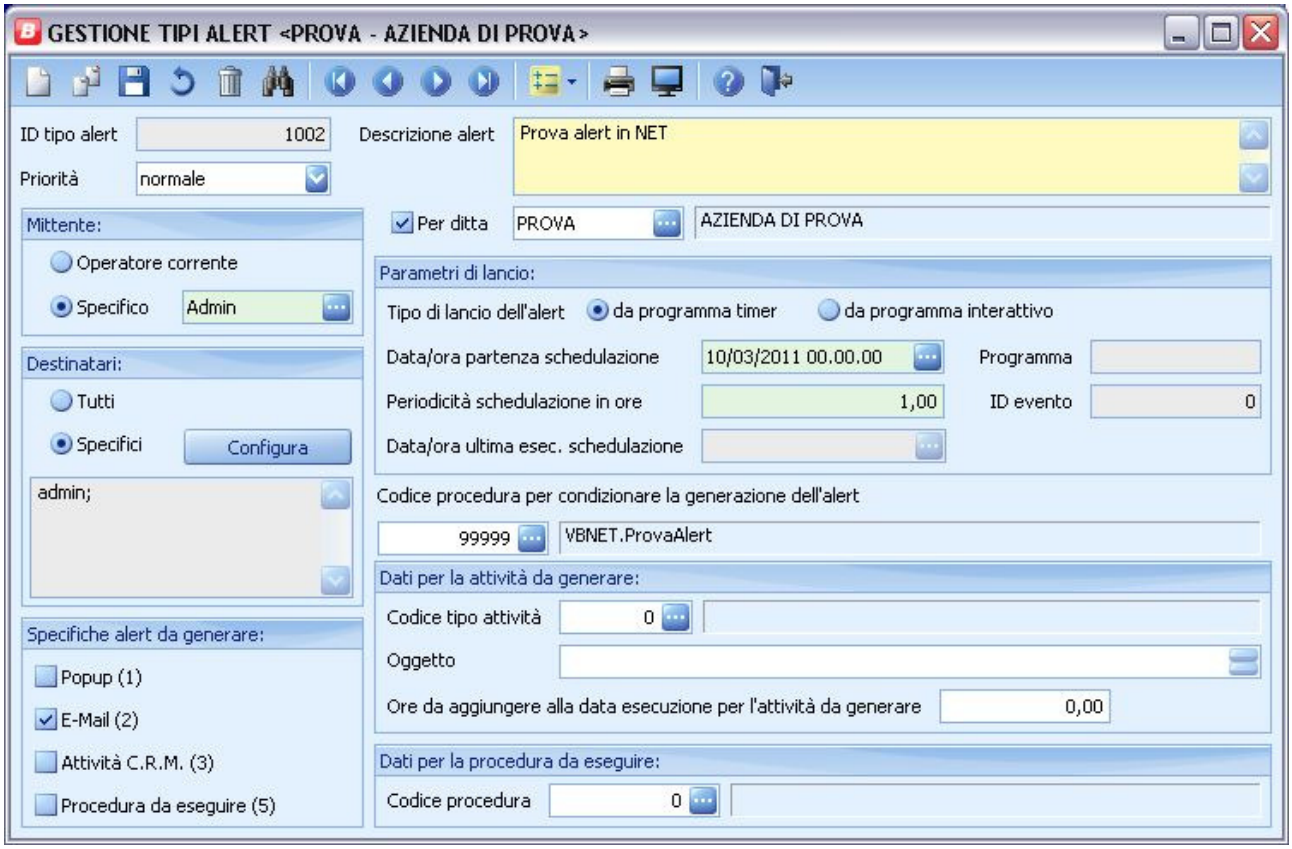
Da Business NET 2010 le procedure da eseguire scritte in .net possono essere anche all'interno delle procedure di import/export, come erano le procedure in vbscript di classic start. l'importante è che abbiano la spunta su 'usa il motore vb.net'. la sintassi da utilizzare è la stessa di quella delle procedure per

import/export in vb.net mentre la funzione che deve essere eseguita deve avere i parametri come le funzioni che erano presenti nel file esterno. Esempio:
Function ProvaAlert(byref oApp as CLE__APP, byval strCodditt as string, byref dttMsgOutParam as DataTable) as boolean

Esempi

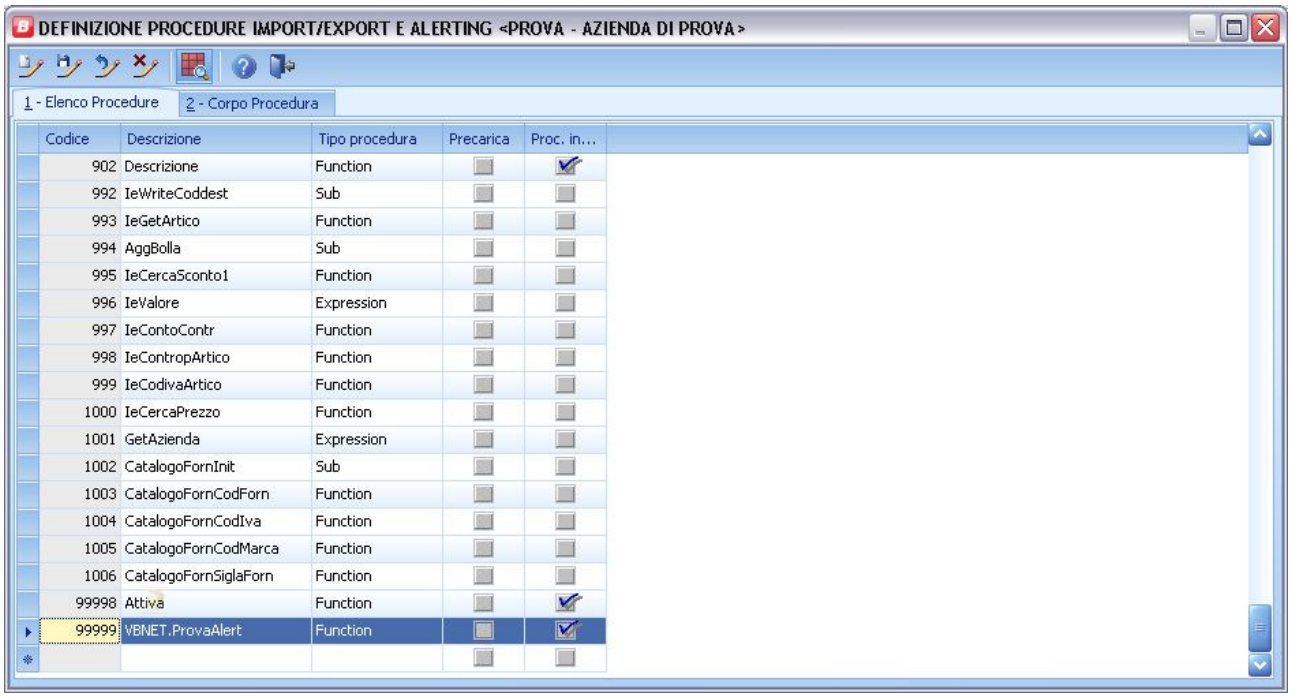
ESEMPIO A: impostare un nuovo alert in .NET con procedura che aggancia un file .NTS nella Script

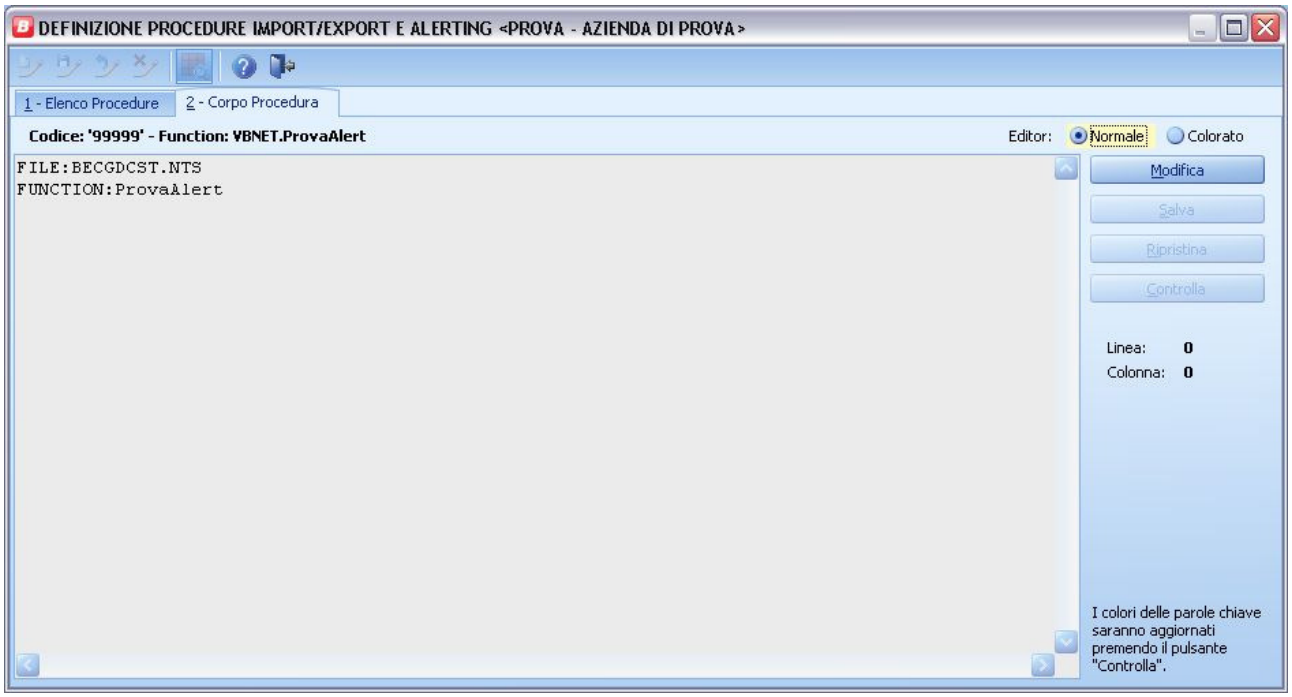
1) creare il nuovo alert



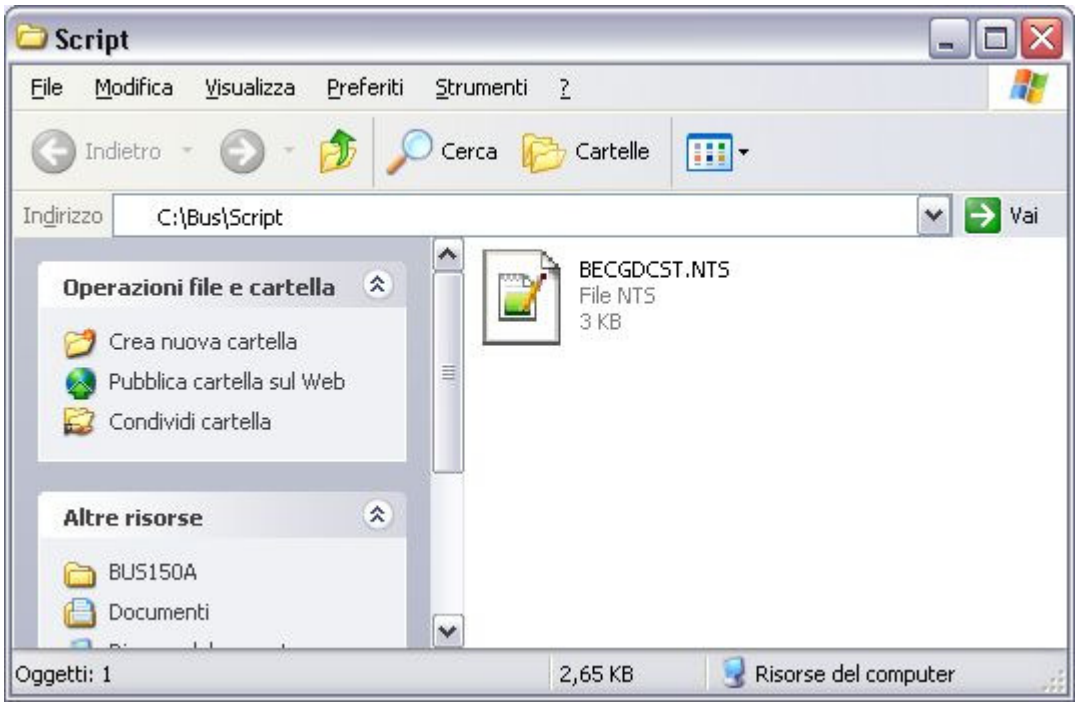
2) compilare l'organizzazione in modo corretto

3) creare la procedura 99999



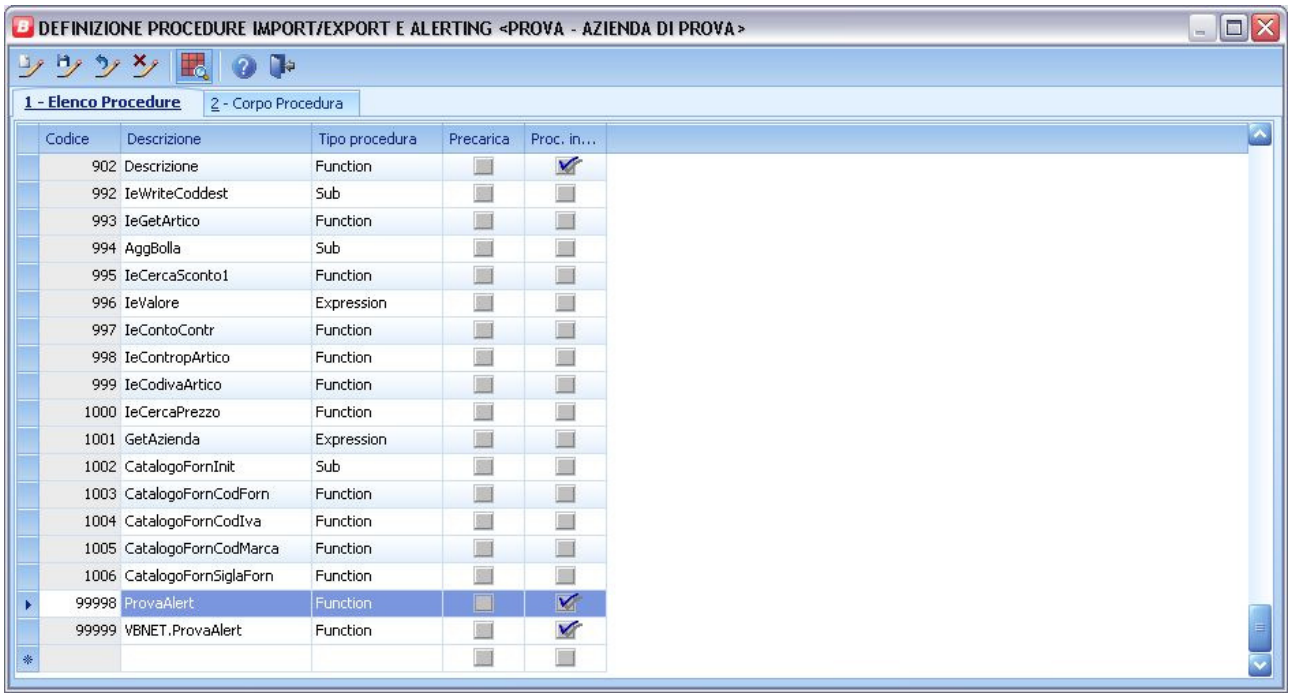


4) copiare il file di script

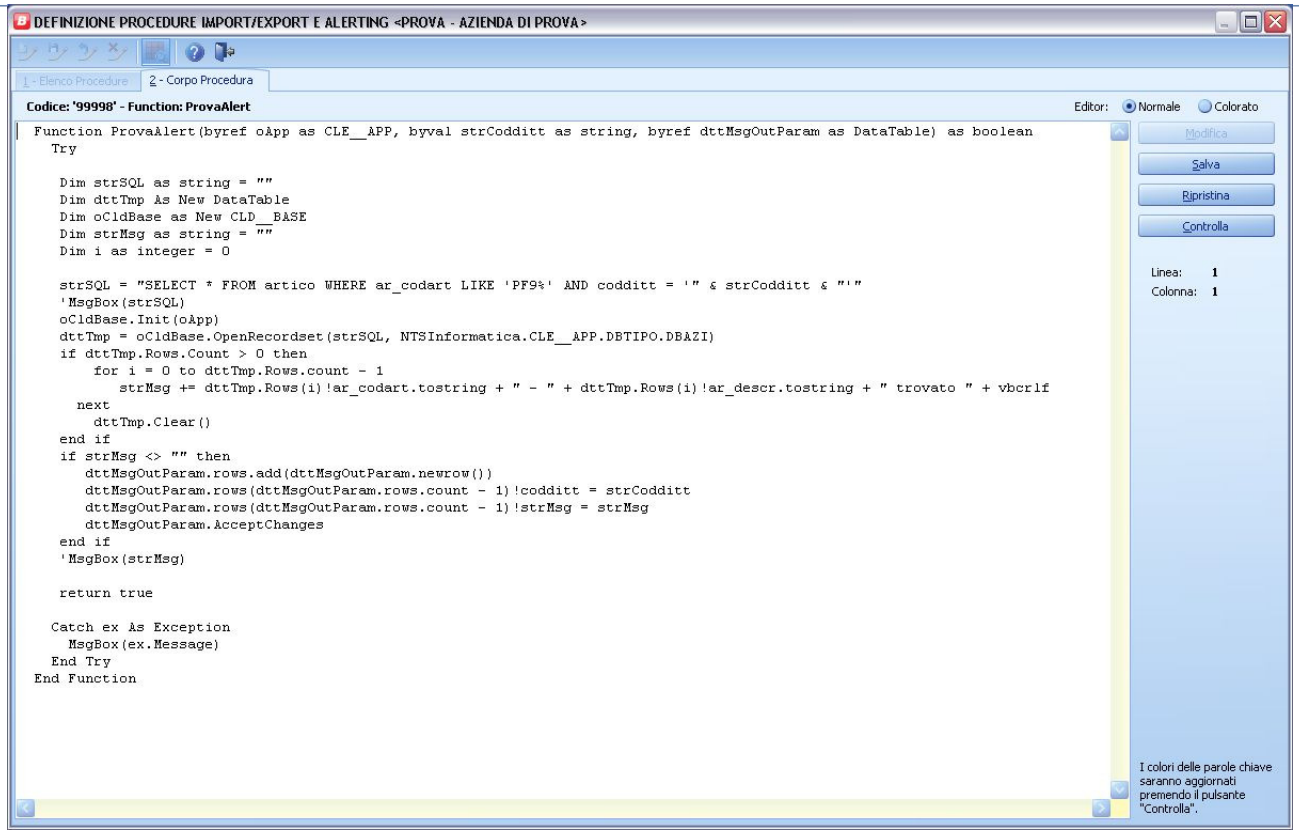


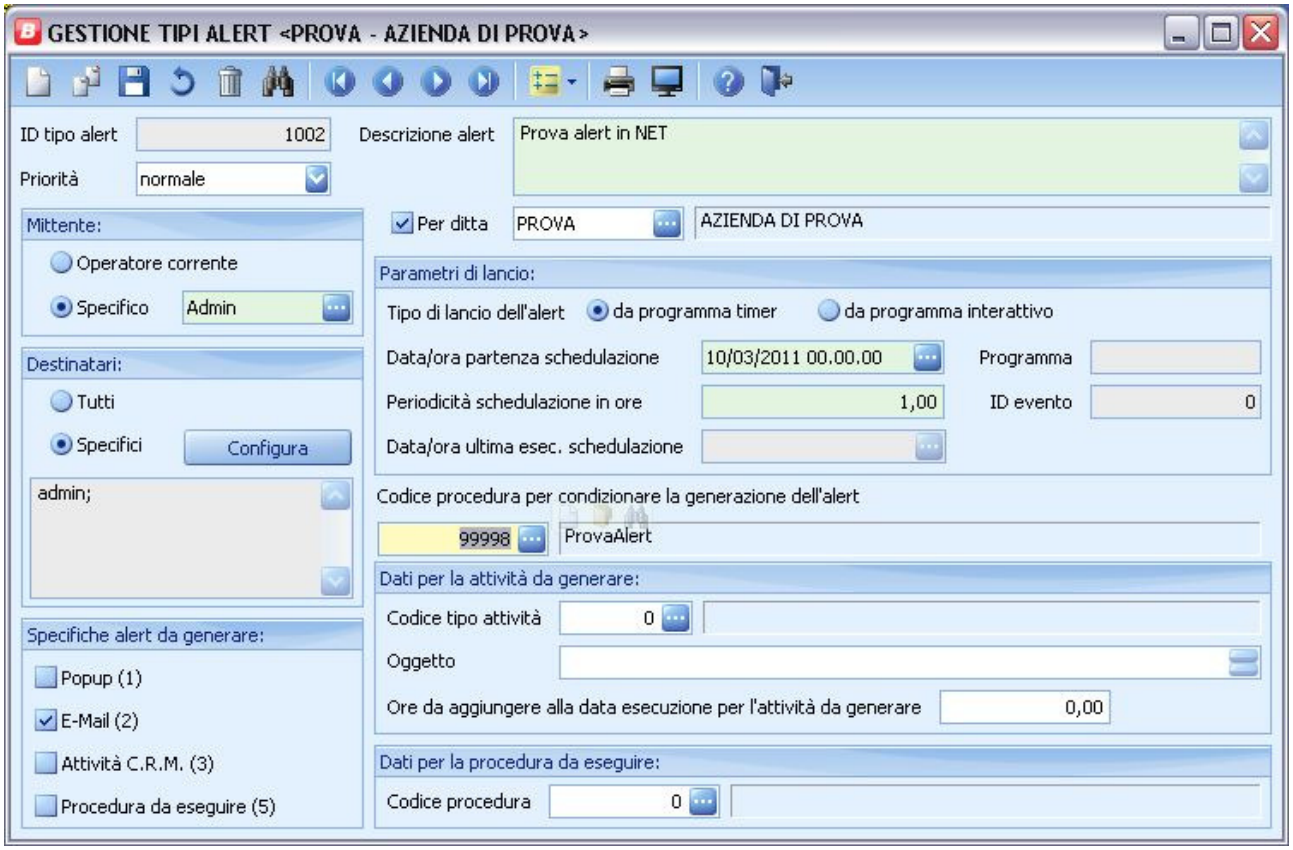
ESEMPIO B: impostare un nuovo alert in .NET con procedura scritta diretttamente nel corpo della procedura

1) procedere come sopra ma creare la procedura nel seguente modo NB: in questo caso il nome procedura non deve iniziare con “VBNET.”



Codice	Descrizione	Tipo procedura	Precarica	Proc. in...
902	Descrizione	Function		✓
992	IeWriteCoddest	Sub		
993	IeGetArtico	Function		
994	AggBolla	Sub		
995	IeCercaSconto1	Function		
996	IeValore	Expression		
997	IeContoContr	Function		
998	IeContropArtico	Function		
999	IeCodivaArtico	Function		
1000	IeCercaPrezzo	Function		
1001	GetAzienda	Expression		
1002	CatalogoFornInit	Sub		
1003	CatalogoFornCodForn	Function		
1004	CatalogoFornCodIva	Function		
1005	CatalogoFornCodMarca	Function		
1006	CatalogoFornSiglaForn	Function		
99998	ProvaAlert	Function		✓
99999	VBNET.ProvaAlert	Function		✓


DEFINIZIONE PROCEDURE IMPORT/EXPORT E ALERTING <PROVA - AZIENDA DI PROVA>
1 - Elenco Procedure
2 - Corpo Procedura
Codice: '99998' - Function: ProvaAlert
Editor: Normale Colorato
Function ProvaAlert(byref oApp as CLE__APP, byval strCodditt as string, byref dttMsgOutParam as DataTable) as boolean
Try
Dim strSQL as string = ""
Dim dttTmp As New DataTable
Dim oCldBase as New CLD__BASE
Dim strMsg as string = ""
Dim i as integer = 0
strSQL = "SELECT * FROM artico WHERE ar_codart LIKE 'PF9%' AND codditt = '" & strCodditt & "'"
'MsgBox(strSQL)
oCldBase.Init(oApp)
dttTmp = oCldBase.OpenRecordset(strSQL, NTSInformatica.CLE__APP.DBTIPO.DBAZI)
if dttTmp.Rows.Count > 0 then
for i = 0 to dttTmp.Rows.count - 1
strMsg += dttTmp.Rows(i)!ar_codart.tostring + " - " + dttTmp.Rows(i)!ar_descr.tostring + " trovato " + vbcrlf
next
dttTmp.Clear()
end if
if strMsg <> "" then
dttMsgOutParam.rows.add(dttMsgOutParam.newrow())
dttMsgOutParam.rows(dttMsgOutParam.rows.count - 1)!codditt = strCodditt
dttMsgOutParam.rows(dttMsgOutParam.rows.count - 1)!strMsg = strMsg
dttMsgOutParam.AcceptChanges
end if
'MsgBox(strMsg)
return true
Catch ex As Exception
MsgBox(ex.Message)
End Try
End Function
Modifica
Salva
Ripristina
Controlla
Linea: 1
Colonna: 1
I colori delle parole chiave saranno aggiornati premendo il pulsante "Controlla".


GESTIONE TIPI ALERT <PROVA - AZIENDA DI PROVA>
ID tipo alert 1002
Descrizione alert Prova alert in NET
Priorità normale
Mittente:
Operatore corrente
Specifico Admin
Per ditta PROVA AZIENDA DI PROVA
Parametri di lancio:
Tipo di lancio dell'alert da programma timer da programma interattivo
Data/ora partenza schedulazione 10/03/2011 00.00.00
Programma
Periodicità schedulazione in ore 1,00
ID evento 0
Data/ora ultima esec. schedulazione
Destinatari:
Tutti
Specifici Configura
admin;
Codice procedura per condizionare la generazione dell'alert
99998 ProvaAlert
Dati per la attività da generare:
Codice tipo attività 0
Oggetto
Ore da aggiungere alla data esecuzione per l'attività da generare 0,00
Specifiche alert da generare:
Popup (1)
E-Mail (2)
Attività C.R.M. (3)
Procedura da eseguire (5)
Dati per la procedura da eseguire:
Codice procedura 0

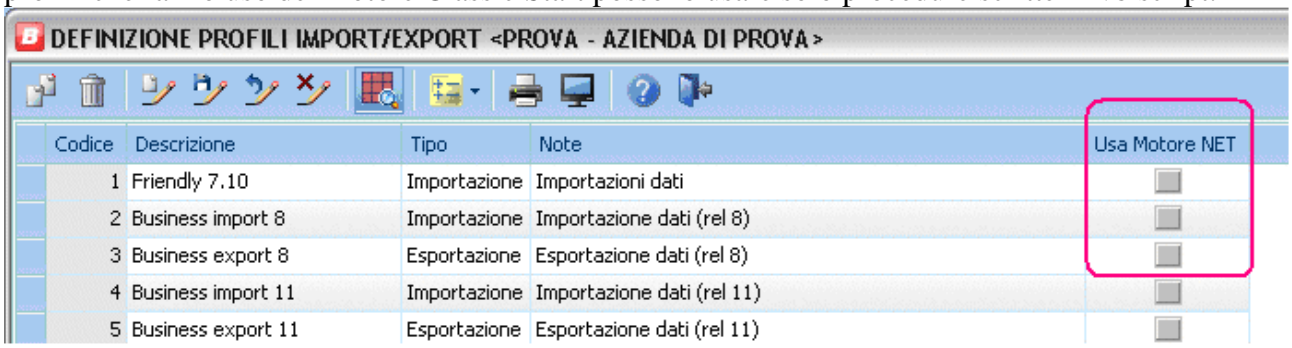
5. Differenze nella definizione di profili di import/export con il motore NET

In questa parte definiremo i passi (differenti) per creare un profilo di imp/exp che fa uso del nuovo motore NET:

- Definizione Profilo
- Definizione dello schema tabelle esterne
- Definizione Procedure

Definizione Profilo

Innanzitutto al momento della definizione del profilo dobbiamo spuntare il flag di **Usa Motore NET**, questa opzione indica se il profilo deve essere elaborato con il motore NET o con il motore Classic Start. Nei profili in export che fanno uso del motore NET i dati prima vengono scritti in datatable in memoria e solo al termine dell'elaborazione vengono riversati nei file definitivo (.mdb/ascii/...), nei profili in import non viene creata la tabella 'ERRORI DI IMPORTAZIONE' poichè potrebbero non esserci le tabelle allegate.
NB: i profili che fanno uso del motore NET possono usare solo procedure scritte in vb NET, i profili che fanno uso del motore Classic Start possono usare solo procedure scritte in vb script.



Codice	Descrizione	Tipo	Note	Usa Motore NET
1	Friendly 7.10	Importazione	Importazioni dati	
2	Business import 8	Importazione	Importazione dati (rel 8)	
3	Business export 8	Esportazione	Esportazione dati (rel 8)	
4	Business import 11	Importazione	Importazione dati (rel 11)	
5	Business export 11	Esportazione	Esportazione dati (rel 11)	

Definizione dello schema tabelle esterne

A questo punto nella definizione dello schema tabelle esterne abbiamo la possibilità di sfruttare delle funzionalità aggiuntive che non sono disponibili in Classic Start:

Tipo tabella – Nei valori a disposizione è possibile scegliere il valore Xml che permette di utilizzare un file XML gestito direttamente dal programma sia in lettura che in scrittura.

Stringa di connessione – Note particolari su Tipo tabella/Stringa di connessione (solo per profili che fanno uso del motore NET):

- Jet o altri Jet ISAM/MSAccess: in export su MDB sono gestiti db di tipo 1997 (default), 2000, 2003 (identico a 2000) e 2007. Per impostare altri tipi di db basta indicare nello schema tabella esterna, nella colonna stringa di connessione il valore 'Access 1997' oppure 'Access 2000' oppure 'Access 2007' (per access 2007 l'estensione del file deve essere .accdb, ma bus non lo controlla). Per ora in export su MDB nelle tabelle create non vengono settate le regole di validazione del campo, il campo richiesto e consenti lunghezza 0.
- Xml: in export su XML ad inizio elaborazione il file .xml viene sempre distrutto. Se l'elaborazione prevede l'export di diverse tabelle e più di una dovrà essere scritta sullo stesso file .xml, i dati verranno accodati. In caso di relazioni tra tabelle in export, verrà realizzato un file master-detail solo se il nome del file .xml di export è lo stesso su entrambi gli schemi

tabelle esterne. Specifiche di file, come ad esempio indicare nell'intestazione file xml il carattere utilizzato (per esempio standalone="yes"), vanno riportare nello schema tabelle esterne nel campo stringa di connessione. In fase di impostazione tabelle esterne, il tracciato per il file .xml deve essere realizzato come se si stesse producendo un export su ASCII. Per assegnare un nome al tag di inizio/fine tabella (il default è come il nome della tabella esterna seguito da '_table') basta scriverlo in schema tabelle esterne nel campo Descrizione (e deve essere diverso dal nome della tabella esterna). **NB:** in import da XML non è possibile utilizzare i campi select, from, where, group by, having, order by personalizzati.

- Jet o altri Jet ISAM/MSExcel: in import è gestita anche l'acquisizione da file di excel di versione 12 (.xlsx). L'impostazione è la stessa che per i file di excel .xls, ma nella stringa di connessione occorre indicare "EXCEL 12.0" invece che "EXCEL 8.0" (oltre ovviamente all'estensione del file da importare .xlsx).

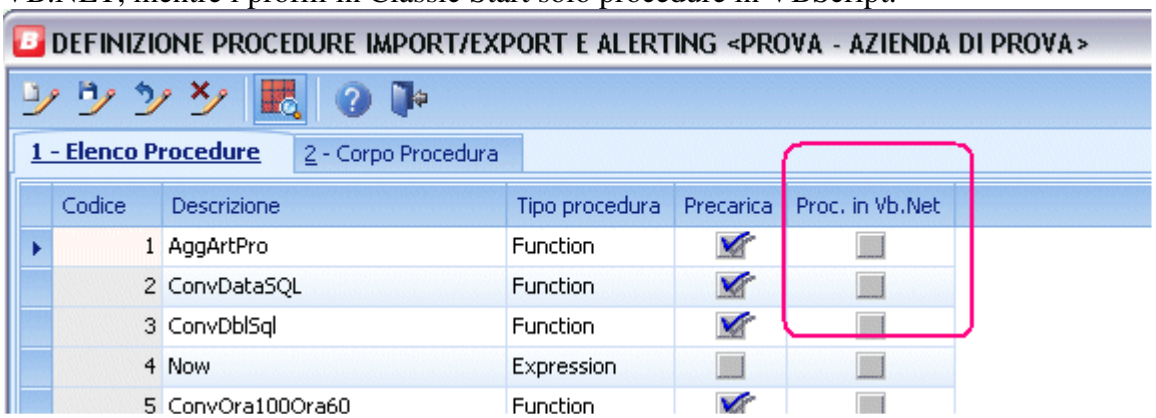
Proprietà format – Note particolari su Tipo tabella/Proprietà Format (solo per profili che fanno uso del motore NET):

- ASCII Text File: ora in export sono gestiti nativamente file ASCII non a lunghezza fissa (come già gestito in import). Per poterli settare basta indicare nello schema tabelle come tipo di file 'ASCII Text file' e come 'Proprietà format' un valore diverso da 'FixedLength'. **NB:** un file di export 'ASCII Text file' di tipo 'CSVDelimited' se salvato con estensione CSV può essere aperto da MSExcel (corrisponde ad un file di testo a lunghezza variabile con separatore ";").

Formato Data/Ora – Rappresenta una stringa che indica il formato della data/ora da importare o esportare, per esempio per interpretare la stringa '15101998153058' (15 Ottobre 1998 ore 15:05:58) sarà necessario indicare la seguente stringa: 'ddMMyyyyhhnnss'. **NB:** in un profilo che usa il motore NET la stringa di formattazione è case sensitive, per esempio il formato yyyymmdd (mese in minuscolo) non è corretto, ma deve essere yyyyMMdd.

Definizione Procedure

Indicare con tale spunta se la procedura è definita in linguaggio VBScript o VB.NET. Tale spunta è molto importante in quanto i profili che utilizzano il motore NET possono usare solo procedure VB.NET, mentre i profili in Classic Start solo procedure in VBScript.



Con questa spunta ora le procedure da eseguire per gli alert scritti in .NET possono essere anche all'interno dell'archivio procedura (anzichè su script), come erano le procedure in VBScript di Classic Start, l'importante è che abbiano la spunta su **Proc. in Vb.Net**. La sintassi da utilizzare è la stessa di quella delle procedure per import/export in VB.NET mentre la funzione che deve essere eseguita deve avere i parametri come le funzioni che erano presenti nel file esterno, per esempio:

```
Function ProvaAlert(byref oApp as CLE__APP, byval strCodditt as string, byref
dttMsgOutParam as DataTable) as boolean
```

Il flag Proc. in Vb.Net è corretto impostarlo solo per procedure di tipo Sub o Function in quanto il profilo se fa uso del motore NET tratta solo queste.

Negli script, per aggiungere degli 'Imports' e/o dei 'Reference' occore utilizzare la seguente sintassi (devono essere scritti fuori dalle function/sub):

```
<reference assembly="BN__STD.dll" />

Imports NTSInformatica.CLN__STD
```

Gli imports passati per default dal framework di busnet sono:

```
Imports Microsoft.visualbasic
Imports System
Imports System.Collections
Imports System.Data
Imports NTSInformatica
Imports NTSInformatica.CLN__STD
```

L'oggetto script di net (oScrIe) ha come variabile globale:

- 'DATA' (di tipo dataset) che contiene i datatable da importare/esportare sia in in che in out
- 'VARS' (di tipo hastable) che contiene le variabili globali (quelle che in bsieimex sono BsIeIn1Profilo, BsIeIn1CancelAll, ...) si accede alle variabili con il comando VARS!BsIeIn1Profilo, VARS!BsIeIn1CancelAll

Gli oggetti/variabili pubblici sempre disponibili in tutti gli script in .NET sono:

```
oApp
oMenu
oCleComm
DATA
VARS
oCleImex
```

Tabella di mappatura fra le variabili BSIEIMEX (Classic Start) e BEIEIMEX (NET)

MsScript	**Script.NET: VARS! (esempio VARS!glProfilo)**
glProfilo	glProfilo
"Ie" & nLivRel & "NomeTabIN"	"Ie" & nLivRel & "NomeTabIN"
"Ie" & nLivRel & "NomeTabOU"	"Ie" & nLivRel & "NomeTabOU"
"Ie" & nLivRel & "SqlWhere"	"Ie" & nLivRel & "SqlWhere"
	"Ie" & nLivRel & "SqlAccessWhere"
"gnLivRel"	"gnLivRel"
"IECancelRel"	"IECancelRel"
"IECancelAll"	"IECancelAll"
"IECancelRow"	"IECancelRow"
"IEExitFalse"	"IEExitFalse"
"Ie" & nLivRel & "NomeFileExt"	"Ie" & nLivRel & "NomeFileExt"

Tabella di mappatura fra le variabili BSIEIMEX (Classic Start) e BEIEIMEX (NET)

MsScript	Script.NET: VARS! (esempio VARS!glProfilo)
In export: "Ie" & nLivRel & "NomeDbExt"	"Ie" & nLivRel & "NomeDbExt"
"Ie" & nLivRel & "NumeroFile" (numero restituito dal comando 'freefile' del file ASCII in export)	"Ie" & nLivRel & "DataTable" (tabella in memoria su cui scrivere i dati)
In import: "Ie" & nLivRel & "DbExt"	Sostituito con "IeDataIn" ovvero datatable corrente contenente i dati da importare
"IeIn" & nLivRel & "nome_campo_da_importare" (elenco dei campi della tabella IN) esempio 'IeIn1ar_descr'	"IeIn" & nLivRel & "Row" (riga della tabella IN) per accedere ai campi esempio: 'VARS!IeIn1Row!ar_descr'
"IeOun" & nLivRel & "nome_campo_da_esportare" (elenco dei campi della tabella OU) esempio 'IeOu1ar_descr'	"IeOu" & nLivRel & "Row" (riga della tabella OU) per accedere ai campi esempio: 'VARS!IeOu1Row!ar_descr'
objImex.WriteFileExport	In export i dati rimangono in un datatable in memoria fino a quando l'operazione di export non è terminata, poi vengono riversati tutti nel file di out. Nel frattempo da ogni funzione/script è possibile aggiungere/togliere/modificare record nel datatable di out
objImex.WriteLog	ocleImex.WriteLog("Messaggio per il log")

Gli oggetti/variabili pubblici disponibili SOLO nella routine TERMINATE in tutti gli script in .NET sono:

- glRigheTot = righe da elaborare considerando anche le relazioni
- glRigheElab = righe elaborateper considerando anche le relazioni
- glRigheIns = righe inserite considerando anche le relazioni
- glRigheUpd = righe aggiornate considerando anche le relazioni
- glRigheSalt = righe saltate considerando anche le relazioni
- glRigheErr = righe con errori considerando anche le relazioni
- lRigheTot = righe da elaborare per singolo livello
- lRigheElab = righe elaborateper singolo livello
- lRigheIns = righe inserite per singolo livello
- lRigheUpd = righe aggiornate per singolo livello
- lRigheSalt = righe saltate per singolo livello
- lRigheErr = righe con errori per singolo livello

6. Source extender

Possibilità di verticalizzare le dll standard di Bus.net

In generale, l'UI quando viene creato, crea ed attiva il proprio entity.

A sua volta, quando l'entity viene creato, crea al proprio interno il dal (uno o più di uno) ed eventualmente altri entity per utilizzare funzionalità standard.

Per il fatto che le nostre DLL sono tutte 'option strict ON', quando viene compilato un entity è necessario che tra i reference ci sia il dal specifico. Stessa cosa quando si compila un UI.

Il problema sorge nel momento in cui si intende personalizzare il dal, aggiungendo delle funzionalità, senza dover modificare l'entity e senza cambiare il riferimento del vecchio dal per farlo puntare a quello nuovo. Stesso discorso vale per la personalizzazione dell'entity senza dover ricompilare l'ui.

La soluzione è la seguente (e vale solo per entity e dal, non per ui); quando segue è riferito alla personalizzazione di un entity, ma lo stesso discorso vale per un qualsiasi dal:

quando l'ui deve creare l'entity, chiama BN__STD passandogli il proprio nome ed il nome del proprio entity standard. A questo punto BN__STD cerca sul server, nella dir Script un file di nome DLLMAP.INI. Se lo trova, al suo interno cerca tra le righe quella con nome dll chiamante e nome dll da chiamare standard uguale a quelli passati in input: se il file DLLMAP.INI non viene trovato, oppure al suo interno non è presente nessuna riga con le caratteristiche passate in input, la funzione del BN__STD creerà ed attiverà la classe standard passata in input. Volendo nel file .ini è possibile indicare come chiamante '*', che è come dire: tutti i programmi che chiamano la dll bemgcomm devono instanziare un'altra dll (es bgmgcomm)

Ad esempio quando la UI di BN__PAGA deve attivare il proprio entity passerà a BN__STD i seguenti parametri 'BN__PAGA, BE__PAGA. Se deve venir caricata la classe entity standard verrà attivata la NTSInformatica.CLE__PAGA, diversamente verrà attivata la classe contenuta nel file specificati nel file DLLMAP.INI.

OVVIAMENTE E' OBBLIGATORIO CHE LA CLASSE CHE DEVE SOSTITUIRE QUELLA STANDARD EREDITI DA QUELLA STANDARD!

Una volta attivata la classe entity, viene restituita all'ui che se la tipizza come la classe standard. In questo modo la ricompilazione dell'ui e l'utilizzo di proprietà/metodi contenuti nella classe standard non generano problemi, così come l'override di funzioni contenuti nella classe standard e rimappate in quella nuova operano correttamente. Il problema si pone nel momento in cui nella nuova classe entità voglio inserire delle funzioni nuove; soluzione: nell'ui la chiamata a queste funzioni 'nuove' avverrà non esplicitamente come per tutte le chiamate alle funzioni della classe standard, ma tramite l'utilizzo del 'CallByName'. Infatti, alla creazione della classe nuova, anche se viene tipizzata come quella vecchia, in realtà contiene anche le proprietà ed i metodi aggiunti, solo che l'ui non li conosce esplicitamente! Tramite la CallByName il compilatore non da errore, anche con OPTION STRICT ON ed in fase di runtime tutto opera correttamente (ovviamente istanziando la dll nuova al posto di quella standard).

Altro pregio già testato: dopo aver realizzato, compilato e consegnato al cliente la nuova classe ereditata dalla standard, possiamo modificare quella base senza bisogno di ricompilare quella ereditata (ovviamente il reference impostato nella classe nuova relativo alla classe standard deve essere ti tipo 'Specific version = false'.

Un esempio pratico di personalizzazione su BE__PAGA.

Si vuole realizzare una nuova funzione in BF__PAGA che eredita da BE__PAGA

- Nel progetto BF__PAGA creo la classe che ha come namespace NTSInformatica, nome CLASSE CLF__PAGA ed eredita da CLE__PAGA, per cui aggiungo tra i refenrece della nuova classe BE__PAGA, BD__BASE, BE__FRWK e BN__STD con 'Specific version = false'
- All'interno della nuova classe inserisco il seguente testo della nuova funzione:

```
Public Class CLF__PAGA
      Inherits CLE__PAGA
      Public Function Pippo(ByVal strIn As String) As Boolean
      MsgBox("La funzione Pippo ha ricevuto in input la stringa '" + strIn
      + "'")
      End Function
End Class
```

- All'interno della nuova classe inserisco il seguente testo della funzione sovrascritta:

```
Public Class CLF__PAGA
```

```
        Inherits CLE__PAGA
        Public Overrides Function Salva() As Boolean
        MsgBox("La funzione Salva personalizzata è stata eseguita.")
        End Function
    End Class
```

- Compilo la nuova classe e la salvo nella dir di bus.net, insieme alle altre dll
- Creo/modifico il file DLLMAP.INI contenuto sul server nella \Script con il seguente testo
  BN__PAGA|BE__PAGA|BF__PAGA|NTSInformatica.CLF__PAGA

Attenzione: per mantenere compatibilità con le future versioni si creano componenti BH, BF e BO personalizzati che ereditano da dll standard BD, BE e BN. Le dll standard non vanno modificate/ricompilate. Se c'e' la necessità di creare un programma simile o con modifiche molto invasive di può ricopiare l'intero progetto e rinominarlo legandosi ad una particolare versione. Ad esempio: standard BN__PAGA personalizzato BNHHPAGA. Si consiglia sempre di utilizzare la tecnica di ereditarietà di dll.

NOTA: tutte le funzioni non private contenute in entity/dal devono essere dichiarata overridable

Per comunicare tra l'entity BE verso l'interfaccia grafica BN si utilizza la funzione TrowRemoteEvent() inviando stringhe di messaggi di errore o codici interpretati dall'interfaccia grafica BN per effetturare opportune funzionalità presenti soltanto in interfaccia grafica vedi funzione GestisciEventiEntity.

In casi particolari se da un entity personalizzato BF si vuole richiamare una funzione pubblica che risiede in uno script .NTS si può richiamare tale funzione nel seguente modo:

```
Me.oScript.Exec("NOME_FUNZIONE_DA_RICHIAMARE_NELLO_SCRIPT", CType(oApp, Object), Nothing, Nothing)
```

Si consiglia di utilizzare appieno la tecnologia a 3 livelli suddividendo il codice tra le varie dll BN BE BD.

Dalla versione Business Net 2009/2010 se da codice personalizzato BF di un componente entity si vuole inserire il posizionamento del focus su una determinata colonna di griglia o su una particolare texbox si possono utilizzare le seguenti istruzioni per ora attivato soltanto su gestione ordini, gestione documenti e gestione offerte.

La GestisciEventiEntity accetta come parametro 'SETFOCUS' e a seguire il nome del controllo (se griglia nome controllo + '.' + colonna della griglia).
In questo modo l'entity può segnalare alla ui che il focus dovrebbe essere assegnato ad un determinato controllo. Sarà campito della ui assegnare il focus se il controllo è presente.
Attenzione ad eventuali loop infiniti di settaggio del focus, attenzione alle maiuscole e minuscole.

```
ThrowRemoteEvent(New NTSEventArgs(CLN__STD.ThMsg.SETFOCUS,
"grrighe.ec_codart"))    'occhio: non grVrighe
ThrowRemoteEvent(New NTSEventArgs(CLN__STD.ThMsg.SETFOCUS, "edet_conto"))
ThrowRemoteEvent(New NTSEventArgs(CLN__STD.ThMsg.SETFOCUS, "edet_codpaga"))
```

Nel caso di creazione un componente personalizzato di interfaccia BO con nome della classe FROORGSOR Inherits FRMORGSOR. Se si sovrascrivono eventi non va indicato l'Handles nella funzione sovrascritta. Nel caso di BN__PAGA funziona, ma nel caso di programmi complessi ORGSOR, VEBOLL da errore.
Impostando l'opzione di registro CHILD_ non da errore in compilazione del componente personalizzato, ma in esecuzione nel caso ORGSOR da errori anomali dal testo poco chiaro del tipo:

Error:
Impossibile stabilire l'associazione alla proprietà o alla colonna xx_vettor di DataSource.
Nome parametro: dataMember (error type: ArgumentException)

Perché inserendo Handles nel componente personalizzato BO l'evento verrebbe eseguito 2 volte.

Errato

```
Public Overrides Sub FRMORGSOR_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
```

Corretto

```
Public Overrides Sub FRMORGSOR_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs)
```

ATTENZIONE

CASO PRATICO: se ho personalizzato la funzione di gestione ordini 'SalvaOrdine' in Net 2010, l'avrò ereditata con la firma uguale a quella della dll standard BDORGSOR, che accetta 8 parametri. Se aggiorno il cliente personalizzato a Net 2011 (dove la funzione non accetta più 8 parametri, ma 10) la vecchia chiamata con 8 parametri è ancora presente, ma l'entity BEORGSOR non chiamerà più la funzione con 8, bensì quella con 10 parametri. Il risultato è che la funzione ereditata contenuta nella personalizzazione non viene più chiamata. Per facilitare il controllo suddetto nel programma 'Configurazione user interface' sono stati aggiunti i comandi 'Esporta personalizzazioni', 'Importa personalizzazioni' e 'Cancella personalizzazioni'. Durante la funzione di import viene eseguito un test per controllare quanto sopra ed eventualmente emette dei messaggi a video.

7. Personalizzazione delle maschere

7.1. Configurazione maschere/griglie in Bus.net

Le routine per la gestione sono in BN__GCTL, sia per il vecchio CTRL+ALT+F4 che per CTRL+Alt+F2

Le routine per applicare le personalizzazioni sono in BN__CHIL.FRM__CHIL e BN__CHIL.BNGRVIEW

Il principio di funzionamento è: quando carico una pagina prima applico il vecchio ctrl+alt+f4, poi applico la personalizzazione delle griglie

Il CTRL+ALT+F4 viene attivato premendo CTRL+SHIFT+CLICK sul controllo da personalizzare

Non viene mai applicato all'avvio della form se l'utente si chiama NTS, oppure se lanciando il child l'utente appartiene al gruppo utenti 'ADMIN' o 'AMMIN' e la descrizione del gruppo è diversa da '' e si tiene premuto il tasto CTRL. Se un controllo è stato reso non visibile, per riabilitarlo basta riaprire la form tenendo premuto il tasto CTRL.

- Non è possibile accedere alla form di gestione se l'utente non appartiene ai gruppi sopra indicati.

La configurazione del CTRL+ALT+F4 ora viene memorizzata tutta su una unica tabella in arcproc (UICONF) e permette di impostare tutta una serie di funzionalità aggiuntive, oltre a poter gestire una priorità di assegnazione

Le proprietà sono:

Visibile	rende il controllo visibile/non visibile. posso solo rendere non visibile: non posso rendere visibile una cosa che non lo è da codice
Enable	rende il controllo editabile/non editabile. In automatico il controllo viene disattivato e impostato con sfondo grigio. posso solo rendere non editabile: non posso rendere visibile una cosa che non lo è da codice
Bold	serve per far risaltare le informazioni che l'utente ritiene più importanti: i textbox vengono evidenziati in verde, le colonne delle griglie vengono evidenziate in verde, per gli altri controlli (label, command button, …) il testo viene portato in grassetto
Default	permette, quando viene inserito un nuovo record, sia che sia in un programma come bn—paga, oppure in un programma con griglia (es bnmgmaga) di preimpostare il default da assegnare ai campi testo e combobox.
Format number	permette di parametrizzare il formato del numero nei campi numerici (occhio, i decimali possono solo essere diminuiti, non aumentati. La formattazione agisce solo a livello grafico ed in fase di accettazione numero (es colonna quantità a 3 decimali in una ditta che non gestisce i decimali può essere portato a numero senza decimali…)
Out not equal	Permette di controllare che al salvataggio di un record, sia che sia contenuto in una griglia, sia che sia un record come in bn—paga, di verificare che il contenuto del campo sia diverso da un valore. Ad esempio se in una ditta, in anagrafica articoli il codice marca deve sempre essere compilato si può impostare che il cod marca sia diversa da 0. Attenzione: nelle griglie il controllo viene fatto solo se il record viene inserito o modificato, non viene eseguito se si passa sopra una riga della griglia senza modificarla. Nel programmi come bn—paga il controllo viene fatto anche se si seleziona di salvare un record non modificato
Checked	permettevi impostare, sui nuovi record, se il controllo checkbox o radiobutton deve essere selezionato per default oppure no. Funziona solo durante l'inserimento di nuovi record
Text	permette di cambiare il testo di form, label, command button, voci di menu, colonne di griglia, … Non è possibile cambiare il testo nelle label collegate a tabella
TextError	permette di modificare il messaggio di errore in caso di errore formale su textbox (esempio quando si inserisce un testo al posto di un numero in un NTSTextBoxNum)

Le proprietà sopra esposte vengono applicate in base ad una gerarchia, per cui è possibile inserire delle impostazioni generali, con delle deroghe per utente o ditta/utente. Ad esempio si può impostare che in bn—paga la contropartita cassa/banca non è mai visibile (quindi imposto una regola generale con VISIBLE=0) ma con la deroga che nella ditta 'PIPPO' il campo deve essere visibile (quindi applico la regola ditta VISIBLE =-1)

Le priorità sono elencate dalla maggiore alla minore nel programma di gestione CTRL+ALT+F4, da codice 1 a 13.

NB: se si imposta la priorità DITTA deve sempre essere indicato anche il DATABASE a cui appartiene la ditta.

Nelle griglie possono essere gestite solo le colonne con intestazione di colonna diverso da '' (in pratica a livello di codice possono essere inserire nelle griglie anche colonne solo ad uso interno, basta che nell'intestazione di colonna non sia indicato del testo.

A livello di programmazione, e solo nei componenti grafici (UI):

- il salvataggio della dimensione della finestra del child, così come il suo ridimensionamento quando il child viene riaperto è automatica, per cui non occorre inserire righe di codice nei nuovi child.

- l'impostazione delle limitazioni per Bus easy e/o TipoInstall A/C/T da opzione di registro (in bus objstd.InitFormDefault) è nativa nel framework per cui non occorre scrivere nulla.

- il salvataggio del vecchio CTRL+ALT+F4 viene fatto dall'appostita form. Il richiamo della form è nativo nel framework, per cui a livello di inserimento nuovo child non occorre scrivere nulla.

- la lettura ed applicazione delle impostazioni del CTRL+ALT+F4 viene fatta nella form_load SEMPRE ALLA FINE DELLA FUNZIONE con il comando

GctlSetRoules()

Se il programma gestisce più tipi documenti (ad esempio bsveboll, bsorgsor, …) prima del comando sopra riportato occorre settare il dipo documento che deve essere gestito. Lo si fa con l'istruzione

GctlTipoDoc = 'B' ('B' è il tipo documento che deve essere settato …)

GctlSetRoules()

Se durante il programma viene cambiato il tipo documento (ad esempio ala creazione di un nuovo ddt, oppure all'apertura di una fattura immediata) oppure viene cambiata la ditta, per applicare le regole reimpostare le direttive sopra esposte. Queste in automatico toglieranno i vincoli precedentemente impostati ed applicheranno i nuovi.

Se nel programma c'è una o più griglie, ai comandi sopra esposti va aggiunto

GctlSetConfigGrid()

Che permette di formattare la griglia in base alle impostazioni precedentemente memorizzate

Anche in questo caso, se il programma gestisce più tipi di documenti prima della direttiva sopra esposta va settato il tipo documento **GctlTipoDoc =**

NB: la gctlSetRoules esegue anche la GcltSetConfigGrid, per cui non serve chiamare direttamente la GctlSetConfigGrid nei vari child se non in casi molto particolari (vedi bncgprin)

Per la griglia occorre tenere in considerazione due situazioni: la visibilità/editabilità/disposizione delle colonne (che viene settata da apposita form): il salvataggio delle impostazioni avviene alla chiusura della form specifica; il salvataggio della larghezza delle colonne e l'altezza della riga della griglia, che deve avvenire sempre alla chiusura del child (o al cambio di tipo documento). In questo secondo caso, nella form_closed di BN__CHIL viene fatto in automatico la **GctlSaveConfigGrid** ma a volte fa fatto manualmente prima del cambio delle impostazioni della griglia (magari per il cambio di un tipo documento …) il comando è sempre **GctlSaveConfigGrid()**

Per far applicare le regole di default/checked nella creazione di nuovi record, nella recordnuovo/filenuovo o nell'apertura di una form di un programma elaborativo scrivere la seguente riga di codice

GctlApplicaDefaultValue()

Per far proporre i valori di default nelle griglie non occorre fare nulla: è tutto nel framework

Dopo aver impostato i valori di default nella nuova riga viene scatenato l'evento NtsGridView.NTSNewRow

Per far controllare, prima del salvataggio, la proprietà OutNotEqual, nella routine dove viene eseguito il test pre-savataggio aggiungere la riga di codice

If GctlControllaOutNotEqual() = False Then Return False

Per fare in modo che I controlli caricati al volo (EXT) vengano collegati al dataset come I controlli standard, non occorre fare nulla. Pensa a tutto la riga sottostante

NTSFormAddDataBinding(dcPaga, Me)

Attenzione: quando si impostano dei Default/OutNotEqual occorre impostare dei valori coerenti con il controllo, per cui non è possibile impostare in 'Default' una stringa se il campo è numerico …. Il programma non controlla e scatena un errore bruttissimo!

Quando si imposta un default da codice, oppure si deve rendere visibile/editabile un campo occorre controllare che l'utente tramite CTRL+ALT+F4 non abbia nascosto/reso non editabile il controllo ...

Attenzione:

RICORDA: quando viene settata nella UI 'DittaCorrente = ….' Nella property viene lanciato in automatico la GCTLSetRoules.

IN PRATICA QUANDO USARE LA GctlSetRoules, GctlSetConfig, GctlApplicaDefaultValue, GctlControllaOutNotEqual

Programmi tabellari (BN__PAGA, BNMGMAGA, BNVEBANC)

Routine	GctlSetRoules	GctlApplicaDefaultValue	GctlControllaOutNotEqual
Form_load	X		
Nuovo		X (non serve se il programma contiene solo una griglia: viene fatto in automatico da NTSGRIDVIEW)	
Salva			X
Cambio tipo doc	X		

Programmi elaborativi (BNCGGNEF, BNCGSTRI, BNCGSTPN)

Routine	GctlSetRoules	GctlApplicaDefaultValue	GctlControllaOutNotEqual
Form_load	X	X	
LeggiDatiDitta		X	
Cambio tipo doc	X	X	

Programmi complessi (BNORGSOR, BNCGPRIN)

Routine	GctlSetRoules	GctlApplicaDefaultValue	GctlControllaOutNotEqual
Form_load	X		
Nuovo		X	
Salva			X
Cambio tipo doc	X		

Programmi ZOOM

Routine	GctlSetRoules	GctlApplicaDefaultValue	GctlControllaOutNotEqual
Form_load	X	X	

Particolarità su visibilità/editabilità dei controlli NTS…

La funzione GctlSetVisenab 'abilita'/'rende visibile' SOLO se non è stata impostata una NON VISIBILITA'/NON EDITABILITA' tramite il vecchio CTRL+ALT+F4 (va a leggere UICONF)

Le impostazioni con CTRL+ALT+F4 di visible/enable possono solo rendere NON VISIBILE/NON EDITABILE, non possono abilitare un controllo bloccato da codice. Nella form di configurazione è stata data la possibilità di rendere visibile con valore '-1' per poter gestire il seguente caso. Il controllo edXXX, che da codice è abilitato, per tutti deve essere disabilitato mentre per l'operatore 'mirto' deve essere abilitato: inserisco una regola di ENABLE = 0 per tutti ed una regola di ENABLE = -1 per l'operatore mirto. In base alle priorità …

Particolarità per le colonne di griglia non visibili per default ma che potrebbero essere rese visibili con CTRL+ALT+F2: la visibilità = true e/o editabilità = true è settabile solo se non sono stati messi dei blocchi con CTRL+ALT+F4 (nella maschera del CTRL+ALT+F2 nella colonna 'SEC'appariranno delle 'E' o 'V' che evidenziano il settaggio di CTRL+ALT+F4). Se non sono stati impostati dei vincoli con CTRL+ALT+F4 le colonne possono essere normalmente rese visibili o meno, mentre l'editabilità può solo essere bloccata, non attivata! (quando si preme CTRL+ALT+F2 nel tag di tutte le colonne bloccate viene forzata una 'E' come se fosse stato impostato da CTRL+ALT+F4)

Le limitazioni di Bus Easy/tipoInstall = A/C/T hanno prevalenza sul CTRL+ALT+F4 e CTRL+ALT+F2 e vengono caricate in automatico con la funzione GctlInitFormDefault chiamata in automatico dentro la GctlSetRoules (le colonne di griglia vengono nascoste ed impostando la 'caption' di colonna = '' non sarà possibile vederle ne con CTRL+ALT+F4 ne con CTRL+ALT+F2; per tutti gli altri controlli viene impostata la proprietà 'Tag' = 'V' che blocca il CTRL+ALT+F4)

Nella form di salvataggio delle impostazioni della griglia (CTRL+ALT+CLICK sulla griglia) è presente la colonna 'Sec': in questa colonna possono essere presenti due lettere: 'V' e 'E': se c'è 'E' vuol dire che con CTRL+ALT+F4 la colonna è stata resa non editabile, per cui non sarà possibile renderla editable (la cella sarà bloccata). Se c'è 'V' il discorso vale per la modifica della visibilità

Il richiamo della form suddetta da dentro la griglia è nativo nel framework, per cui non occorre scrivere righe di codice per farla apparire

7.2. Gestione della posizione / ridimensionamento dei controlli nella form e riordinamento del tabindex

Per attivare l'editing occorre essere utenti di tipo amministratore e selezionare CTRL+ALT+SHIFT + F10. Stessa combinazione di tasti per disattivare l'editing dei controlli

In tutte le form, se ci si posiziona sui controlli in alto a SX (il cursore cambia di icona) è possibile spostare il controllo in un'altra parte dello schermo. Se, sempre con la stessa modalità, si posiziona il cursore in basso a DX (anche in questo caso il cursore cambia di icona) è possibile ridimensionare il controllo in altezza/larghezza). I controlli con la proprietà AUTOSIZE = TRUE, ovviamente, non sono ridimensionabili.

Se invece si preme CTRL poi si clicca con il tasto sx del mouse su un controllo appare una form che permette di gestire le proprietà 'anchor' e 'Dock' del controllo stesso.

Attenzione: l'unica limitazione è che i RADIOBUTTON devono essere lasciati dentro il container in cui sono caricati all'avvio del programma, non è possibile spostarli dal loro container!

Per salvare le impostazioni, in modo che al riavvio vengano mantenute le modifiche, occorre entrare nella form di configurazione del controllo con CTRL+ALT+SHIFT+CLICK e utilizzare la funzione 'Salva posizione'

I controlli come la griglia e tutti quelli dove il cursore non cambia icona non possono essere spostati/ridimensionati.

I controlli con Dock = fill non possono essere ridimensionati a piacimento.

I TextBox con proprietà 'multiline' = false non possono essere ridimensionati verticalmente

Per default le label senza bordo, i checkbox ed i radiobutton hanno la proprietà AutoSize = true, e quindi non possono essere ridimensionate.

A livello di codice non occorre scrivere nulla: è tutto integrato nel framework.

7.3. Registro di configurazione di Business (bs—greg)

Dai vari child è possibile accedere direttamente al registro di configurazione di Business. Basta che l'utente appartenga al gruppo degli amministratori (viene testato che nel gruppo sia presente la parola 'ADMIN' o 'AMMIN' oppure la descrizione del gruppo sia '' – è stata fatta una funzione in BN__STD) la combinazione di tasti è CTRL+ALT+G dentro la form

Ad oggi, le combinazioni di tasti per le funzioni sull'interfaccia sono:

(oltre alle nuove voci di menu della form)

CTRL+CLICK sulla griglia	: appare il menu per la configurazione della griglia
CTRL+ALT+F sulla griglia	: fa apparire la form per eseguire ricerche nella griglia
CTRL+F sulla griglia	: continua la ricerca sulla griglia
CTRL +SHIFT+CLICK sul controllo	: ex CTRL+ALT+F4 specifico del controllo
CLICK tasto SX in alto a sx del controllo	: permette di spostare il controllo
CLICK tasto SX in basso a dx del controllo	: permette di ridimensionare il controllo
CTRL +CLICK tasto SX sul controllo	: permette di gestire Anchor e Dock del controllo
CTRL mentre si avvia un child	: carica il child senza applicare il CTRL+Alt+F4
CTRL+ALT+SHIFT+F10 sulla form	: abilita/disabilita le operazioni grafiche sui controlli
ALT + F1 su una colonna testo in griglia	: apre una form per visualizzare meglio il testo (solo se la colonna può contenere testo per più di 30 caratteri
ALT + F2 su un campo zoomabile	: apre la form di gestione del campo in modalità 'inserimento nuovo record'
ALT + F3 su un campo zoomabile	: apre la form di gestione del campo in modalità 'gestione del record'

I principali comandi sono visibili facendo tasto DX sulla caption di qualsiasi form di Busnet

Piccola chicca in più rispetto a Business per multireport:

se l'utente appartiene al gruppo degli amministratori e si lancia la stampa a video (solo a video) tenendo premuto solo il pulsante SHIFT viene verificato se esiste già una impostazione multireport: se non presente viene creata la struttura impostando la proprietà 'Count = 1' e ReportName = nome del report sulla Rep1.

Per spostarsi da una pagina di TabControntrol ad un'altra, o ci si posiziona sulle linguette delle pagine, quindi freccia a sx/dx, oppure quando si è dentro ad un controllo contenuto in una pagina del tabcontrol, con CTRL+Freccia DX o CTRL+Freccia a SX si cambia pagina

Quando si dovrà settare, in un controllo, la proprietà visibile o enable = true, usare la seguente sintassi:

```
GctlSetVisEnab(cbTipoDoc, "", True)
GctlSetVisEnab(cbTipoDoc, "", False)
GctlSetVisEnab(tlbReportStampaPdf.Name, False)
GctlSetVisEnab(tlbReportStampaWord.Name, False)
GctlSetVisEnab( tlbFirst.Name, False)
GctlSetVisEnab( tlbPrevious.Name, False)
```

Mentre per settare queste proprietà = false usare il sistema standard

ATTENZIONE: nelle form utilizzare solo controlli di tipo NTS!

A livello di controlli aggiunti con source extender, ovvero CTRL+SHIFT+ALT+F10, appare la form per aggiungere i controlli, trascino un controllo sulla form ….

Per salvare il nuovo controllo bisogna fare tasto DX sul nuovo controllo, quindi 'salva impostazioni'. Per cancellarlo (sia fisicamente che dal database) basta fare tasto DX sul controllo + cancella. Sono cancellabili solo i controlli aggiunti con source extender.

A livello di programmazione non occorre fare nulla. Pensa a tutto BN__CHIL. I controlli aggiunti, così come la posizione, la dimensione e tutto quello che riguarda la disposizione dei controlli nella form sono globali di form, non dipendenti da tipo documento/operatore/ditta …

Per quanto riguarda i menu, possono essere aggiunti menu solo nelle form già contenenti una toolbar o un menù. Le nuove voci di menù possono essere accodate a menu già esistenti, oppure possono essere inserite in un nuovo menù 'custom': per aggiungere le voci basta trascinare il controllo 'Menu item' su una voce di menu esistente per accodare, oppure basta trascinare il controllo sul menù dove non sono presenti voci di menu (ad esempio all'estrema destra del menu) per far creare in automatico il menu 'Custom' ed inserire sotto di esso la nuova voce di menu.

Con che ordine le voci di menu vengono caricate nella form_load? In base all'ordine con cui sono state SALVATE/RISALVATE in fase di design. Una volta aggiunta una voce di menù non è possibile spostarla. Occorre cancellarla e reinserirla. Per cambiare l'ordinamento dei sottomenu, salvarli/risalvarli in ordine di come devono essere visualizzati, a partire da quello più in alto.

Per i pulsanti della toolbar, possono essere aggiunti solo in coda ai pulsanti esistenti.

Quando vengono creati/caricati cercano una immagine che abbia lo stesso nome ".jpg" del controllo creato; ad esempio se aggiungo il bottone con nome 'tlbProva' verrà cercata una immagine che si chiama 'tlbprova.jpg" nella dir \Images del pc su cui gira Busnet. Se l'immagine non viene trovata verrà caricato il pulsante con esposto il testo. In fase di load della form questo tipo di controlli verranno caricati in base all'ordine con cui sono stati SALVATI/RISALVATI in fase di design (esattamente come le voci di menu)

Per le voci di TabControl, possono essere aggiunti solo in coda alle linguette esistenti.

In fase di load della form questo tipo di controlli verranno caricati in base all'ordine con cui sono stati SALVATI/RISALVATI in fase di design (esattamente come le voci di menu)

ATTENZIONE: per salvare/cancellare il controllo bisogna posizionarsi SEMPRE sulla linguetta, poi fare TASTO DX, quindi salva/cancella!!!

Per la configurazione globale di griglia ciccare nel quadratino in alto a SX dove si incrocia la riga delle caption delle colonne e la prima colonna di sx (quella dove si evidenzia se la colonna è in editing)

Per le colonne di griglia vale lo stesso discorso delle voci di TabControl.

7.4. Inserimento di clienti/fornitori velocizzato

Nei campi textbox e colonne di griglia dove viene richiesto un codice cliente fornitore (ovvero dove è abilitato lo zoom su anagra (NTSSetparamZoom = "ZOOMANAGRA")) è possibile inserire al posto del sottoconto completo (es 4010001, 11010001) solo il progressivo numerico: una routine interna provvederà ad aggiungere al conto indicato il primo mastro clienti o fornitori indicato in tabella mastri.
Per distinguere se il progressivo è relativo ad un cliente o un fornitore viene guardato il segno del numero imputato: se positivo è un cliente, se negativo è un fornitore
L'automatismo viene abilitato solo se il numero inserito è minore di 7 caratteri!

7.5. Come impostare campi zoommabili personalizzati in Bus.NET

Di seguito i passi per creare un campo personalizzato in una maschera di Bus.Net i cui valori possono essere alimentati da uno zoom su di una tabella personalizzata.

Esempio:

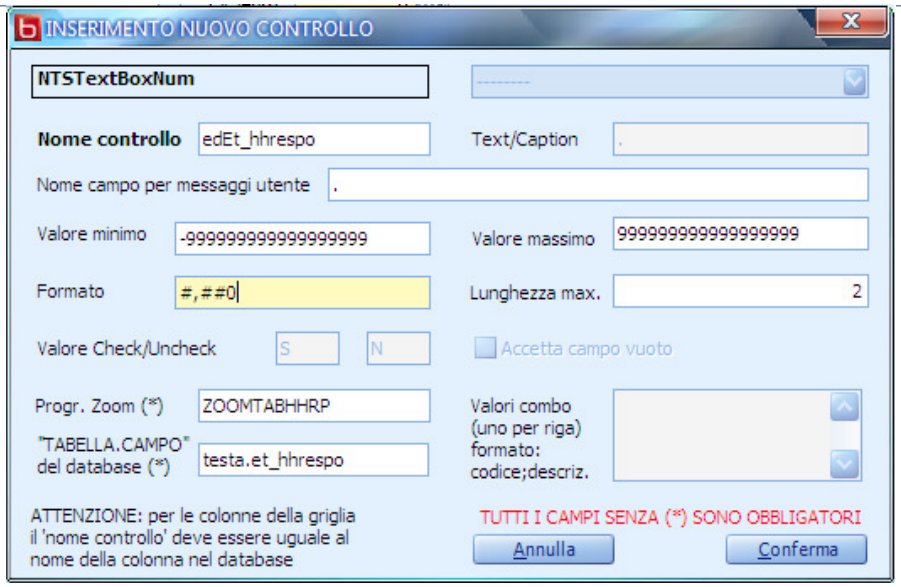
Aggiungere il campo “Responsabile documento” a BSVEBOLL (Gestione documenti) i cui valori devono poter essere letti da una tabella contenente l’elenco dei possibili responsabili.

Operazioni da eseguire:

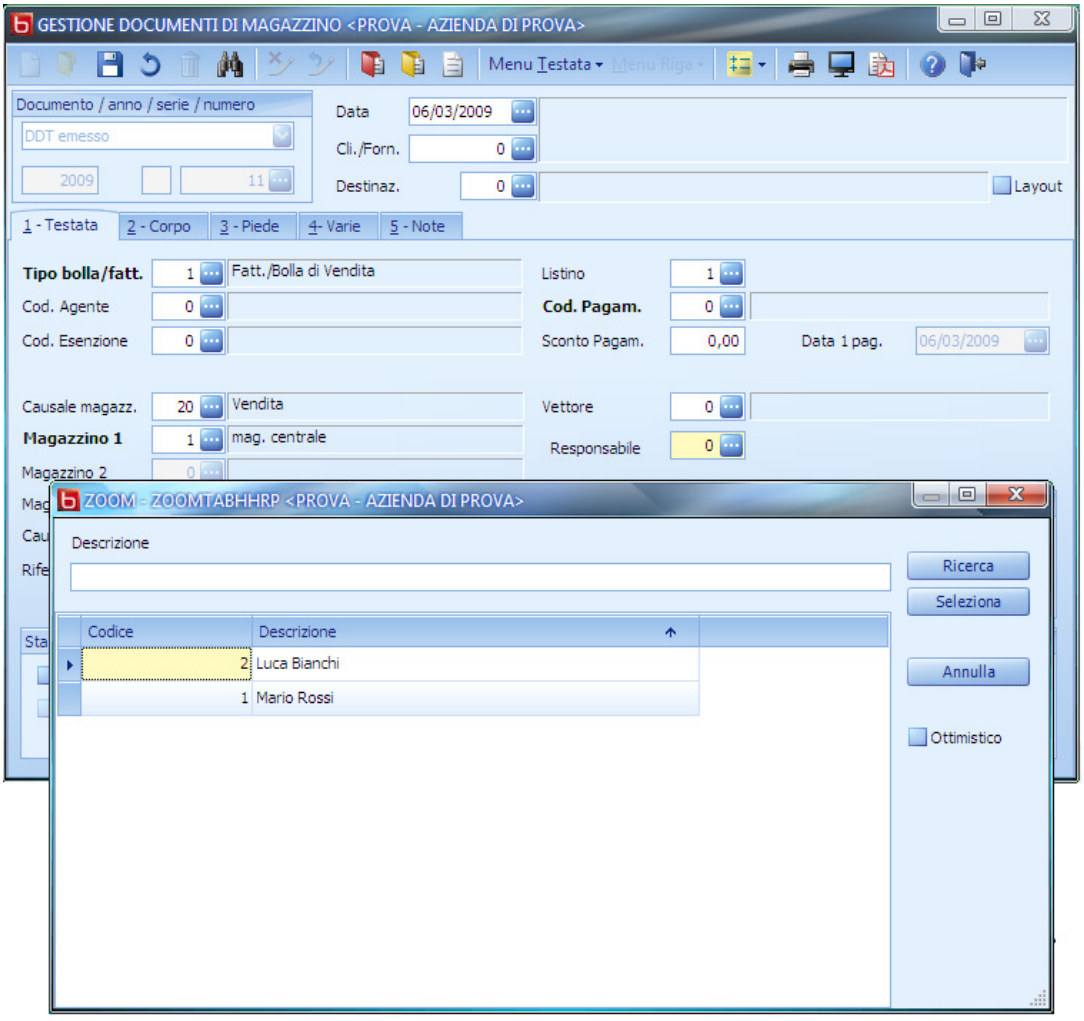
1. Aggiungere il campo personalizzato TM_HHRESPO alla tabella TESTMAG (nel nostro caso di tipo numerico) nel database PROVA.MOD. Il prefisso HH è indispensabile per evitare che le utility BUSCONV / BUSCONVS sovrascrivano il campo in fase di aggiornamento di release di business. In alternativa ad HH si può usare il prefisso QQ. In alternativa i campi potrebbero iniziare per quei suffissi (ad esempio HH_RESPO)
2. Aggiungere la tabella personalizzata TABHHRP nel database PROVA.MOD contenente i campi:
   - TB_CODHHRP
   - TB_DESHHRP

   In alternativa ai caratteri HH si possono utilizzare i caratteri QQ. Il nome della tabella personalizzata deve essere lungo esattamente 7 caratteri ed iniziare per TABHH oppure TABQQ (per poter essere utilizzata in uno zoom standard) per cui rimangono solo 2 caratteri per identificare tale tabella.

   I nomi del campo codice e descrizione sono automaticamente definiti dal nome tabella. Il campo TB_CODHHRP deve obbligatoriamente essere richiesto = SI e chiave primaria della tabella.
3. Aggiungere un record alla tabella ORDERTBL nel database PROVA.MOD. Il record deve contenere il nome della tabella appena creata TABHHRP. Prestare attenzione al flag ORDERTBL.OT_PERDITTA che deve essere settato ad N se la tabella personalizzata NON contiene il campo CODDITT.
4. Una volta preparato il file PROVA.MOD lanciare l’utlity BUSCONV (Access) o BUSCONVS (Sql Server) per aggiornare la struttura del database azienda da personalizzare.
5. Popolare la tabella TABHHRP nel database azienda.
6. Aprire “Gestione documenti” (BSVEBOLL) , creare un nuovo documento ed abilitare l’editing della maschera (tasto destro del mouse sulla barra del programma Abilita/Disabilita Editing)
7. Trascina il controllo di interesse nel punto di interesse sulla maschera, nell’esempio un campo TEXTBOX (dello stesso tipo del campo TABHHRP.TB_CODHHRP) ed impostarlo come indicato in figura:
   - Nome Controllo: edEt_hhrespo
   - Progr. Zoom: ZOOMTABHHRP
   - TABELLA.CAMPO: testa.et_hhrespo (Vedi note 1).



8. Confermare le impostazioni e poi disabilitare l'editing della maschera (tasto destro del mouse sulla barra del programma Abilita/Disabilita Editing)
9. Di seguito il risultato:

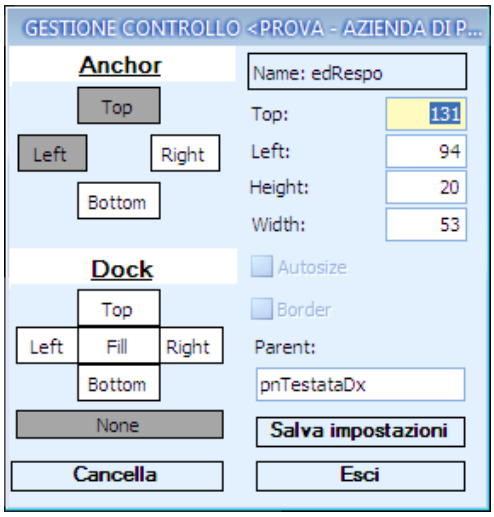


7.6. Come impostare campi personalizzati in Bus.NET

Se si fosse voluto solamente aggiungere un campo a "Gestione documenti" non gestito con uno zoom con valori liberi sarebbe stato sufficiente seguire i punti 1, 4, 6, 7.a, 7.c, 8, 9.

Per modificare/cancellare un controllo personalizzato aggiunto ad una maschera occorre:

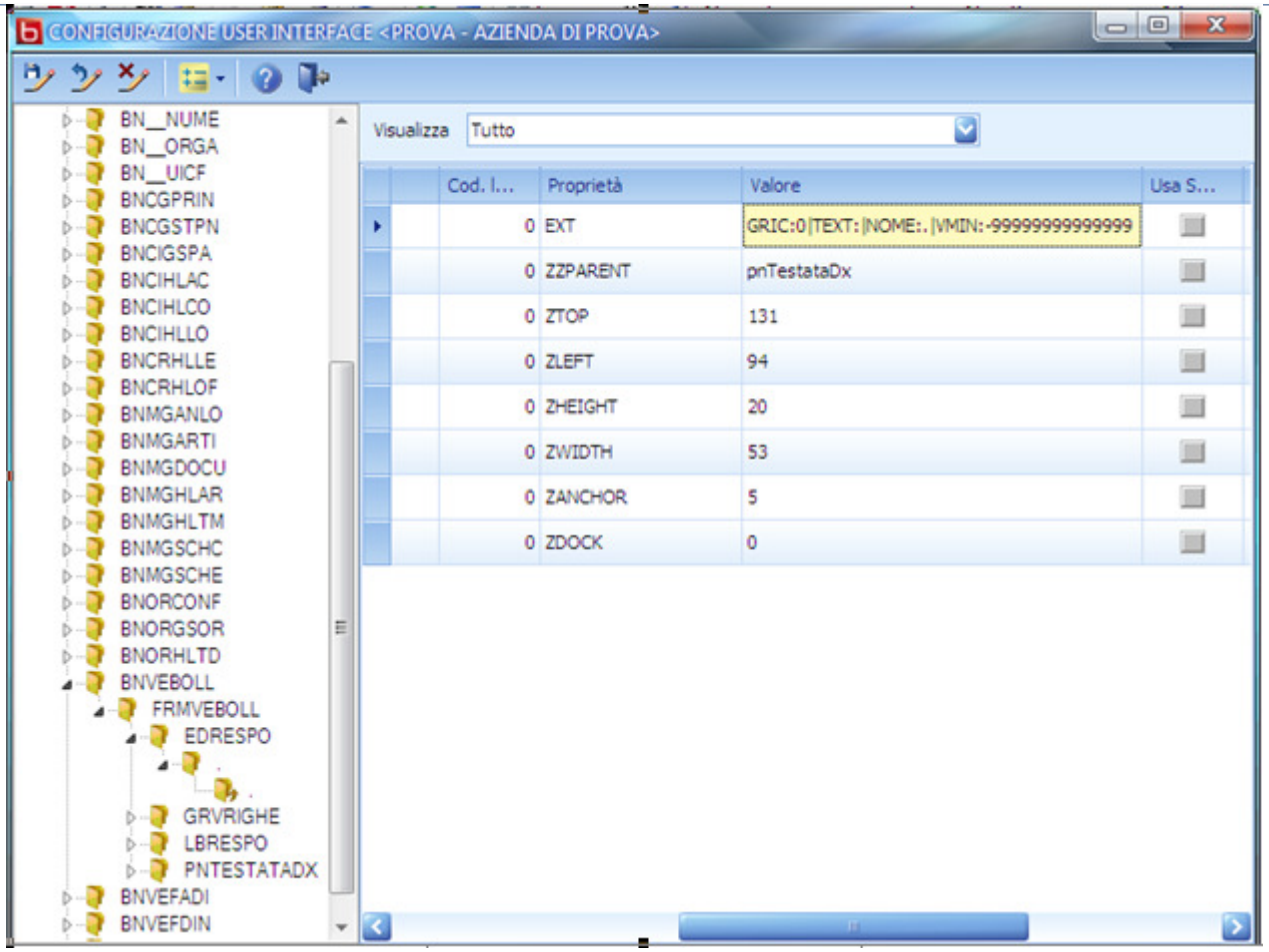
1. Aprire la maschera ed abilitarne l'editing (tasto destro del mouse sulla barra del programma Abilita/Disabilita Editing)
2. Eseguire ctrl + click sinistro sul controllo
3. Eseguire le modifiche o cancellare



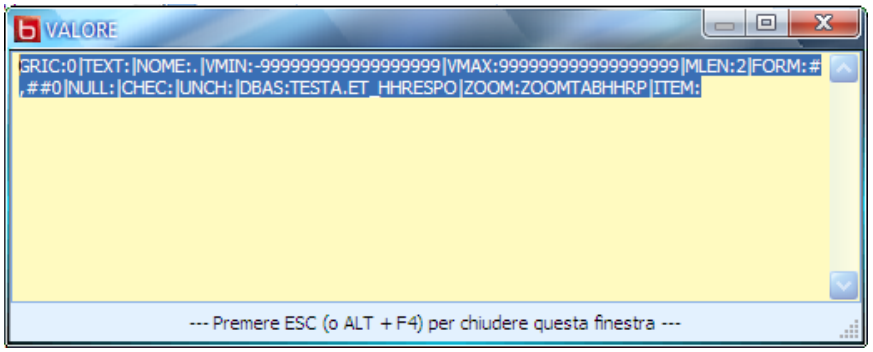
4. Disabilitare l'editing (tasto destro del mouse sulla barra del programma Abilita/Disabilita Editing)

Siccome non è possibile, una volta salvato, modificare le impostazioni salvate al punto 7, l'unico modo per intervenire è eliminare e ricreare il controllo oppure intervenire manualmente da "Configuratore User Interface" richiamabile:

1. dalla maschera modificata (tasto destro del mouse sulla barra del programma e lanciare "Configuratore User Interface")
2. e poi dal menu 1.O.B di Business

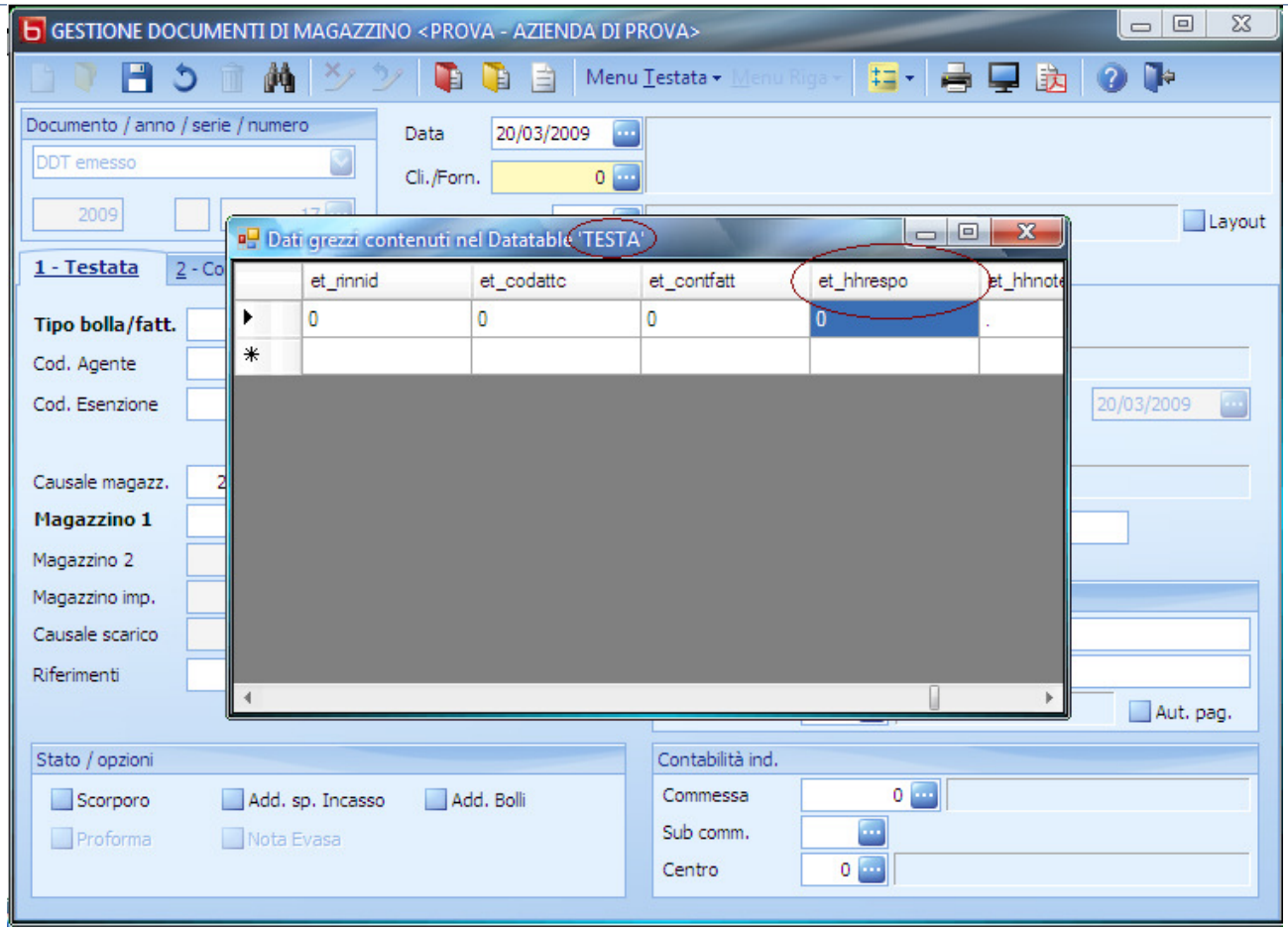


e poi Alt + F1 sul campo VALORE della proprietà EXT



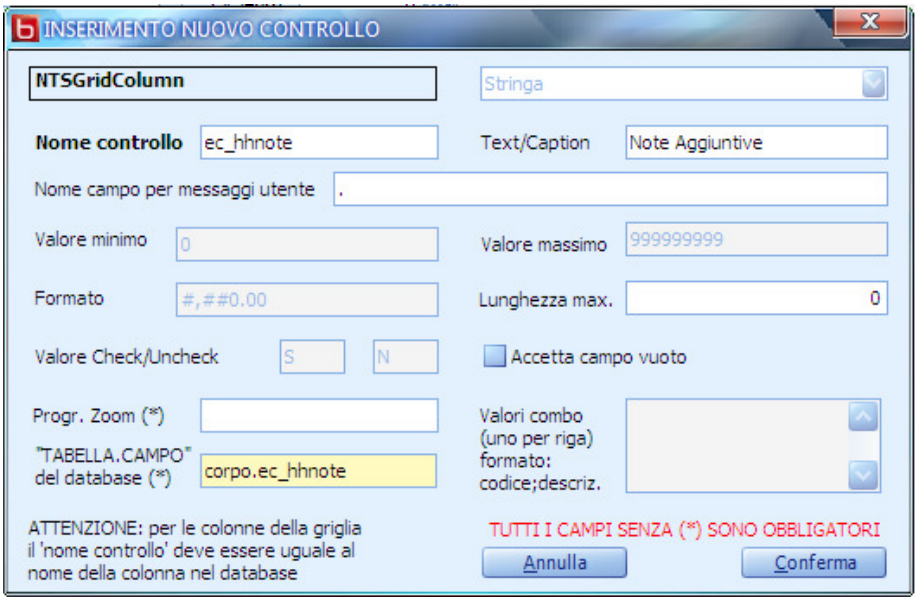
Note

Il nome da indicare nel campo TABELLA.CAMPO del database(*) in alcuni casi potrebbe non coincidere con il reale nome del campo sul database. Nell'esempio sarebbe venuto naturale indicare TESTMAG.TM_HHRESPO ma abbiamo indicato TESTA.ET_HHRESPO. Ad esempio nelle maschere di Gestione Documenti e Gestione Ordini che contengono delle routine comuni, per evitare di duplicare codice, si sono ri-mappati i nomi dei campi del database, per cui la configurazione delle maschere necessita del nome ri-mappato. Per visualizzare, in ogni maschera, quali sono i nomi reali da indicare, è sufficiente utilizzare la voce di menu "Visualizza Datatable", facendo tasto destro sulla barra del titolo del programma.


GESTIONE DOCUMENTI DI MAGAZZINO <PROVA - AZIENDA DI PROVA>
Menu Testata
Menu Riga
Documento / anno / serie / numero
DDT emesso
2009
Data
20/03/2009
Cli./Forn.
0
Layout
1 - Testata
2 - Co
Tipo bolla/fatt.
Cod. Agente
Cod. Esenzione
Causale magazz.
Magazzino 1
Magazzino 2
Magazzino imp.
Causale scarico
Riferimenti
20/03/2009
Aut. pag.
Dati grezzi contenuti nel Datatable 'TESTA'
et_rinnid
et_codattc
et_contfatt
et_hhrespo
et_hhnote
0
0
0
0
Stato / opzioni
Scorporo
Add. sp. Incasso
Add. Bolli
Proforma
Nota Evasa
Contabilità ind.
Commessa
0
Sub comm.
Centro
0

7.7. Come impostare campi di griglia personalizzati in Bus.NET

Per aggiungere un campo personalizzato ad una griglia, ad esempio al corpo dei documenti occorre eseguire i seguenti punti:

1. Aggiungere il campo personalizzato MM_HHNOTE alla tabella MOVMAG (nel nostro caso di tipo stringa) nel database PROVA.MOD. Il prefisso HH è indispensabile per evitare che le utility BUSCONV / BUSCONVS sovrascrivano il campo in fase di aggiornamento di release di business. In alternativa ad HH si può usare il prefisso QQ. In alternativa i campi potrebbero iniziare per quei suffissi (ad esempio HH_NOTE).
2. Una volta preparato il file PROVA.MOD lanciare l'utlity BUSCONV (Access) o BUSCONVS (Sql Server) per aggiornare la struttura del database azienda da personalizzare.
3. Aprire "Gestione documenti" (BSVEBOLL), creare un nuovo documento, posizionarsi sul corpo del documento, ed abilitare l'editing della maschera (tasto destro del mouse sulla barra del programma Abilita/Disabilita Editing).
4. Trascinare il controllo di interesse (Colonna di Griglia) nel punto di interesse sulla griglia ed impostarlo come indicato in figura:
   - Nome Controllo: ec_hhnote
   - TABELLA.CAMPO: corpo.ec_hhnote
   - 

5. Confermare le impostazioni e poi disabilitare l'editing della maschera (tasto destro del mouse sulla barra del programma Abilita/Disabilita Editing).

Note

Il nome da indicare nel campo TABELLA.CAMPO del database(*) in alcuni casi potrebbe non coincidere con il reale nome del campo sul database.

Nell'esempio sarebbe venuto naturale indicare MOVMAG.TM_HHNOTE ma abbiamo indicato CORPO.EC_HHNOTE.

Ad esempio nelle maschere di Gestione Documenti e Gestione Ordini che contengono delle routine comuni, per evitare di duplicare codice, si sono ri-mappati i nomi dei campi del database, per cui la configurazione delle maschere necessita del nome ri-mappato.

Per visualizzare, in ogni maschera, quali sono i nomi reali da indicare, è sufficiente utilizzare la voce di menu "Visualizza Datatable", facendo tasto destro sulla barra del titolo del programma.

7.8. Come gestire una nuova tabella utilizzando DALCONF.INI in Bus.NET

Per poter gestire una nuova tabella molto semplice di nome TABxxxx con codice tb_codxxxx tb_desxxxx

Rimando al documento BN__TABE.pdf

Per poter gestire qualsiasi nuova tabella anche molto complessa, con chiavi multiple su database azienda o arcproc.

Si può configurare il file DALCONF.INI per poter utilizzare le funzioni del framework di Bus.Net.

Senza preoccuparsi di scrivere query di select, insert into, update, delete su tale nuova tabella per leggere e salvare i dati.

Le funzioni del framework di Bus.Net. per gestire le tabelle sono:

LeggiTabellaSemplice	per ottenere tutti i dati di una tabella filtrati per codice ditta.
ScriviTabellaSemplice	per salvare su database i dati della tabella aggiunti, modificati, cancellati.
ValCodiceDB	per controllare se il codice è presente nella tabella per ottenere il record relativo a quel codice. Questa funzione è equivalente della LegArti, LegAnag di vb6 però utilizzabile su qualsiasi tabella.

Per tabelle esistenti come anagra, artico, ecc tali definizioni sono già inserite a livello di framework.

Il file DALCONF.INI serve per inserire le definizioni per tabelle nuove personalizzate.

Esempi di chiamate a tali funzioni di Framework:

```
oCldArti.LeggiTabellaSemplice(strDittaCorrente, "HHARTI2", dsShared)
```

- **strDittaCorrente** è la ditta su cui si è posizionati equivalente della gstrExt di vb6.
- **“HHARTI2”** è la tabella personalizzata tale nome deve coincidere con il campo dl_nome indicato nel file di configurazione.
- **dsShared** è il dataset contenente i dati riempiti dalla query di select effettuata dal frame work se la tabella è per ditta viene filtrato il codice dittta.

```
oCldArti.ScriviTabellaSemplice(strDittaCorrente, "HHARTI2",
                               dsShared.Tables("HHARTI2"))
```

- **strDittaCorrente** è la ditta su cui si è posizionati equivalente della gstrExt di vb6.
- **“HHARTI2”** è la tabella personalizzata tale nome deve coincidere con il campo dl_nome indicato nel file di configurazione.
  dsShared.Tables(“HHARTI2”) è il datatable contenente i dati passati al framework per effettuare i salvataggi tramite query di insert into, update, delete.

```
If oCldArti.ValCodiceDb(strCodart, strDitta, "HHARTI2", "S", strTmp, dttTmp)
                               Then
```

- **strCodart** è una variabile stringa per passargli il valore del codice della chiave primaria
- **strDitta** è la ditta su cui si è posizionati.
- **“HHARTI2”** è la tabella personalizzata tale nome deve coincidere con il campo dl_nome indicato nel file di configurazione.
- **“S”** se il codice della chiave primaria è di tipo stringa “N” numerico “D” data.
- **strTmp** è il valore restituito relativo alla decodifica della tabella se il codice è stato trovato verrà restituito il valore del campo relativo a quel codice il campo di decodifica è impostato nel dl_descampo indicato del file di configurazione.
- **dttTmp** (opzionale) è il datatable contenente i dati relativi al record del codice passato alla funzione.

La funzione restituisce True se il codice passato è esistente False se inesistente.

Per utilizzare DALCONF.INI bisogna configurarlo in questo modo:

Il file è presente nel seguente percorso:

Cartella di installazione \Script\DALCONF.INI

```
#----------------------------------------------------------------------------------------------------
#dl_nome|dl_nometab|dl_perditta|dl_prefisso|dl_campichiave|dl_tipocampichiave|dl_descampo|dl_db
#esempio:
TABCIVA|tabciva|N|tb_|tb_codciva|N|tb_desciva|D
ARTICO|artico|S|ar_|ar_codart|S|ar_descr|D
ARTDEFX|artdefx|S|adx_|adx_codart;adx_fase|S;N|adx_codart|D
AZIENDE|aziende|N|az|azcodaz|S|AzDescr|A
```

dl_nome	il nome della costante utilizzata nelle chiamate di framework (in maiuscolo)
dl_nometab	il nome della tabella fisica del database (in minuscolo)
dl_perditta	N se la tabella non è per ditta S se la tabella è per ditta (se c'e' il campo codditt in tale tabella)
dl_prefisso	il nome di prefisso comune dei campi della tabella compreso l'underscore _
dl_campichiave	elenco campi chiave separati da ; (senza indicare il campo codditt)
dl_tipocampichiave	elenco dei tipi dei campi chiave rispettando l'ordine indicato nella precedente proprietà N numerico S stringa D data
dl_descampo	campo di decodifica di tale tabella.
dl_db	la nuova tabella è presente D nel database azienda A per database arcproc.

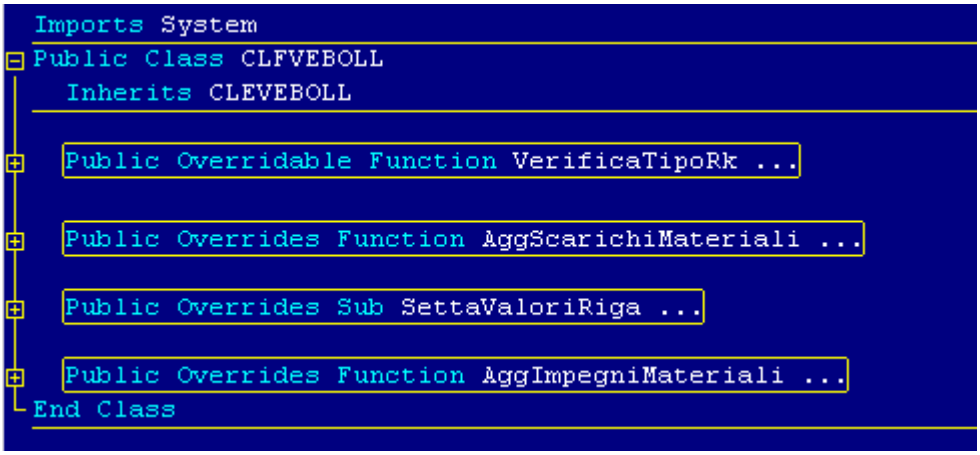
Attenzione non mettere spazi all'inizio, durante e fine della stringa di configurazione nel file DALCONF.INI

Non lasciare caratteri di interlinea prima e dopo tale stringa di configurazione.

7.9. Come modificare il valore di riga e il calcolo dei totali nei documenti di magazzino

Per poter modificare il calcolo del valore di riga (senza considerare la valuta) nei programmi come Gestione Ordini (BNORGSOR) e Gestione Documenti (BNVEVOLL) si può procedere creando una classe BFMGDOCU che eredita dal componente BEMGDOCU.
Se invece la modifica deve essere fatta solo BEVEBOLL o BEORGSOR si può creare un componete che eredita direttamente dal componente da modificare che a sua volta eredita da BEMGDOCU.
Nell'esempio è stato creato un componente BFVEBOLL per far si che le modifiche siano apportate solo a BEVEBOLL dato che non interessa modificare anche BEORGSOR.

I metodi da ereditare e modificare sono:
- AggScarichiMateriali
- SettaValoriRiga
- AggImpegniMateriali



Il metodo VerificaTipoRk è stato aggiunto per poter filtrare solo in quali documenti deve essere effettuata la modifica.

All'interno dei 3 metodi ereditati di questo esempio si può copiare/incollare il codice originale.

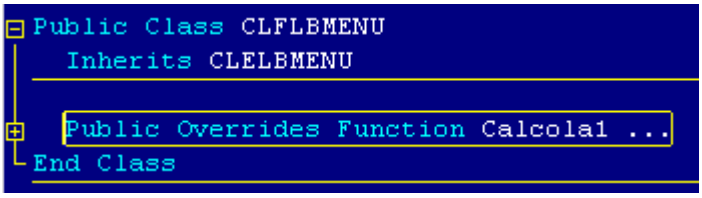
In molti casi è consigliato utilizzare la tecnica del MyBase.NomeFunzione descritta in modo completa durante le giornate dei corsi di programmazione net, per mantenere una completa compatibilità con le successive versioni di Business Net. Per modifiche critiche alle logiche di calcolo valutare la modifica con un esperto contabile/fiscalista.
Ora all'interno del codice cercare "!ec_valore =" e dove è presente il vecchio calcolo aggiornarlo con quello nuovo.
Calcolo originale:

```
                dtrT(i)!ec_valore = ArrDbl(NTSCDec(dtrT(i)!ec_prezzo) *
NTSCDec(dtrT(i)!ec_quant) / NTSCDec(dtrT(i)!ec_perqta), oCldDocu.TrovaNdec(0))
```

Calcolo modificato (per alcuni tipork aggiunge ec_misura2 al prezzo):

```
If VerificaTipoRk() Then
                dtrT(i)!ec_valore = ArrDbl((NTSCDec(dtrT(i)!ec_prezzo) +
ArrDbl(NTSCDec(dtrT(i)!ec_misura2), oCldDocu.TrovaNdecSuPrzUn(0))) *
NTSCDec(dtrT(i)!ec_quant) / NTSCDec(dtrT(i)!ec_perqta), oCldDocu.TrovaNdec(0))
              Else
                dtrT(i)!ec_valore = ArrDbl(NTSCDec(dtrT(i)!ec_prezzo) *
NTSCDec(dtrT(i)!ec_quant) / NTSCDec(dtrT(i)!ec_perqta), oCldDocu.TrovaNdec(0))
              End If
```

Dopo aver modificato il calcolo del valore di riga è necessario modificare anche il calcolo dei totali, che non tiene conto del valore di riga impostato ma viene ricalcolato.
I programmi che calcolano il totale di un documento (come BNVEBOLL, BSORGSOR, BNVEFDIN, ecc…) lo fanno in BELBMENU. Quindi creando una nuova classe BFLBMENU ed ereditando dal componente orginale è possibile modificare il calcolo dei totali su tutti i documenti.
All'interno di BELBMENU il calcolo dei totali viene fatto nella Calcola1



All'interno di questo metodo vi è ad un certo punto un controllo che verifica il programma chiamente:

```
For i = 0 To dttCorpo.Rows.Count - 1
        With dttCorpo.Rows(i)
          Select Case oPar.strNomProg
            Case "GENERAL"
              dMMquant = NTSCDec(!QUANT)
              dMMprezzo = NTSCDec(!PREZZO)
              dMMprezvalc = NTSCDec(!PREZVALC)
              dMMpreziva = NTSCDec(!PREZIVA)
              dMMscont1 = NTSCDec(!SCONT1)
              dMMscont2 = NTSCDec(!SCONT2)
```

All'interno di questo metodo vengono impostati i dati con cui verrà effettuato il calcolo, in base al programma chiamante. Di conseguenza si può modificare la parte del case relativo al programma che stiamo modificando per essere sicuri che il calcolo non verrà modificato per gli altri componenti.

Il Case "BSVEBOLL" originale

```
Case "BSVEBOLL"
              dMMquant = NTSCDec(!ec_quant)
              dMMprezzo = NTSCDec(!ec_prezzo)
              dMMprezvalc = NTSCDec(!ec_prezvalc)
              dMMpreziva = NTSCDec(!ec_preziva)
              dMMscont1 = NTSCDec(!ec_scont1)
              dMMscont2 = NTSCDec(!ec_scont2)
              dMMscont3 = NTSCDec(!ec_scont3)
              dMMscont4 = NTSCDec(!ec_scont4)
              dMMscont5 = NTSCDec(!ec_scont5)
              dMMscont6 = NTSCDec(!ec_scont6)
```

```
                dMMcolli = NTSCDec(!ec_colli)
                dMMvprovv = NTSCDec(!ec_vprovv)
                dMMvprovv2 = NTSCDec(!ec_vprovv2)
                nMMcodiva = NTSCInt(!ec_codiva)
                nMMcontrop = NTSCInt(!ec_controp)
                strMMstasino = !ec_stasino.ToString
                strMMcodart = !ec_codart.ToString
                strMMumprz = !ec_umprz.ToString
                dMMperqta = NTSCDec(!ec_perqta)
                lMMacnum = NTSCInt(!ec_acnum)
```

Il Case “BSVEBOLL” modificato con le stesse logiche della modifica del valore di riga:

```
Case "BSVEBOLL"
                dMMquant = NTSCDec(!ec_quant)
                If NTSCStr(!ec_tipork) = "A" Or NTSCStr(!ec_tipork) = "B" Then
                  dMMprezzo = NTSCDec(!ec_prezzo) + ArrDbl(NTSCDec(!ec_misura2),
ocldBase.TrovaNdecSuPrzUn(0))
                Else
                  dMMprezzo = NTSCDec(!ec_prezzo)
                End If
                dMMprezvalc = NTSCDec(!ec_prezvalc)
                dMMpreziva = NTSCDec(!ec_preziva)
                dMMscont1 = NTSCDec(!ec_scont1)
                dMMscont2 = NTSCDec(!ec_scont2)
                dMMscont3 = NTSCDec(!ec_scont3)
                dMMscont4 = NTSCDec(!ec_scont4)
                dMMscont5 = NTSCDec(!ec_scont5)
                dMMscont6 = NTSCDec(!ec_scont6)
                dMMcolli = NTSCDec(!ec_colli)
                dMMvprovv = NTSCDec(!ec_vprovv)
                dMMvprovv2 = NTSCDec(!ec_vprovv2)
                nMMcodiva = NTSCInt(!ec_codiva)
                nMMcontrop = NTSCInt(!ec_controp)
                strMMstasino = !ec_stasino.ToString
                strMMcodart = !ec_codart.ToString
                strMMumprz = !ec_umprz.ToString
                dMMperqta = NTSCDec(!ec_perqta)
                lMMacnum = NTSCInt(!ec_acnum)
```

Mentre se si vuole correggere la formula usata nei calcoli dei totali si può modificare il seguente codice:

```
'Valori riga (lordo sconti testata/piede: wValore)
              If nValuta <> 0 Then
                wValore = ArrDbl(dMMprezvalc * dMMquant / dMMperqta * (100 -
dMMscont1) / 100 * (100 - dMMscont2) / 100 * (100 - dMMscont3) / 100 * (100 -
dMMscont4) / 100 * (100 - dMMscont5) / 100 * (100 - dMMscont6) / 100,
oCldComm.TrovaNdec((nValuta)))
              Else
                wValore = ArrDbl(dMMprezzo * dMMquant / dMMperqta * (100 -
dMMscont1) / 100 * (100 - dMMscont2) / 100 * (100 - dMMscont3) / 100 * (100 -
dMMscont4) / 100 * (100 - dMMscont5) / 100 * (100 - dMMscont6) / 100,
oCldComm.TrovaNdec(0))
              End If
              If bScorp Then
                wValoreciva = ArrDbl(dMMpreziva * dMMquant / dMMperqta * (100 -
dMMscont1) / 100 * (100 - dMMscont2) / 100 * (100 - dMMscont3) / 100 * (100 -
dMMscont4) / 100 * (100 - dMMscont5) / 100 * (100 - dMMscont6) / 100,
oCldComm.TrovaNdec(0))
                wValore = ArrDbl(((dMMpreziva * dMMquant / dMMperqta * (100 -
dMMscont1) / 100 * (100 - dMMscont2) / 100 * (100 - dMMscont3) / 100 * (100 -
dMMscont4) / 100 * (100 - dMMscont5) / 100 * (100 - dMMscont6) / 100) * 100) /
(100 + oCldComm.AliquotaIva(nMMcodiva)), oCldComm.TrovaNdec(0))
               End If
```

7.10. Intervenire prima dell'invio effettivo di e-mail e/o fax (BE__SEND)

Funzioni chiamate prima dell'invio effettivo di e-mail e/o fax che permette di cambiare i dati contenuti nel documento da inviare (oggetto, destinatario, allegati, ...) o di eseguire invii ulteriori oltre a quello elaborato dalla routine standard

```
SEND_InvioFax(ByRef strOggetto As String, ByRef strFileDaInviare As String,
ByRef strFax As String, ByRef strDesconto As String, ByRef dttOrga As DataTable,
ByVal strProgrChiamante As String)
```

```
SEND_InvioMail(ByRef strTo As String, ByRef strToEx As String, ByRef strCC As
String, ByRef strOggetto As String, ByRef strAllegati() As String, ByRef
bConfermaLetturaDaClient As Boolean, ByRef strHtmlEmail As String, ByRef
strTestoEmail As String, ByRef bModaleOutlook As Boolean, ByVal
strProgrChiamante As String)
```

Tutte le variabili (ad esclusione del programma chiamante) sono passate per riferimento e se il metodo ritorna "False" l'invio viene annullato.

Estese le possibilità di personalizzazione all'interno di BE__SEND sfruttando una serie di variabili globali:

`Public SEND_strCome As String` Il tipo di invio che è stato scelto:

- S = e-mail
- X = fax xp/2003
- Y = fax 2000
- Z = zetafax
- H = hylafax
- N = mcrosoft fax mapi
- W = winfax pro symantec
- blank = come indicato in anagrafica

`Public SEND_dttMessages As DataTable` I destinatari dell'e-mail, gli allegati specifici del cliente, i riferimenti al documento di partenza, il corpo dell'e-mail se deve essere personalizzato per ogni cliente

`Public SEND_nMessaggio As Integer` La riga di dttMessages attualmente in elaborazione

`Public SEND_strOggetto As String` L'oggetto dell'e-mail

`Public SEND_dttAllegati As DataTable` Gli allegati comuni a tutti i clienti

`Public SEND_bConfermaLetturaMailDaClient As Boolean` Se chidere la conferma di lettura o meno

`Public SEND_bAncheAgente As Boolean` Se spedire l'e-mail anche all'agente

`Public SEND_bIsDestinatarioLead As Boolean` Se l'invio è stato fatto verso dei lead oppure no.

`Public SEND_strSendErrors As String` Gli errori da ritornare al programma chiamante

`Public SEND_strCorpoEmail As String` Il corpo della mail (può essere sia semplice che HTML)

`Public SEND_strRuoloOrganig As String` Il ruolo di organig al quale fare le spedizioni

`Public SEND_bTestoDaDoc As Boolean` Indica se per il corpo del messaggio deve usare SEND_strCorpoEmail o il contenuto di dttMessages

`Public SEND_bckUsaMailFaxDest As Boolean` Durante l'invio di un fax indica se deve controllare se spedire le e-mail alle destinazione diversa.

`Public SEND_strTestoSempliceSeHtml As String` Se SEND_strCorpoEmail è HTML, contiene il corpo in testo semplice dell'e-mail

`Public SEND_bModaleOutlook As Boolean` Se far apparire o meno la modale di Outlook o di Business e-mail prima di effettuare l'invio

Per uniformare i nome dati alle variabili globali i nomi dei 2 metodi precedenti sono stati cambiati in SEND_InvioMail e SEND_InvioFax.

È stato creato anche un esempio di come personalizzare l'invio delle e-mail.

L'esempio aggiunge in coda all'oggetto dell'e-mail "C.A. Titolo Nome Cognome" presi dall'anagrafica del cliente quando viene inviata una e-mail da gestione documenti, inoltre se è la data del compleanno del cliente aggiunge come allegato una immagine di auguri.

```
Public Overrides Function SEND_InvioMail(ByRef strTo As String, ByRef strToEx As
String, ByRef strCC As String, ByRef strOggetto As String, _
                                         ByRef strAllegati() As String, ByRef
bConfermaLetturaDaClient As Boolean, ByRef strHtmlEmail As String, _
                                         ByRef strTestoEmail As String, ByRef
bModaleOutlook As Boolean, ByVal strProgrChiamante As String) As Boolean
    Dim dttClie As New DataTable
    Dim strAllegatiTmp() As String
    Try

oCldSend.ValCodiceDb(NTSCStr(SEND_dttMessages.Rows(SEND_nMessaggio)!CONTO),
strDittaCorrente, "ANAGRA", "N", , dttClie)

      'Non ha trovato il cliente, termina l'invio
      If dttClie.Rows.Count = 0 Then Return False

      'Se il programma chiamante è Gestione documenti aggiunge al titolo dell'e-
mail "C.A. 'Nome cliente'"
      If strProgrChiamante = "BSVEBOLL" Then
        With dttClie.Rows(0)
          'Se in anagrafica non ho indicato nome e\o cognome del cliente salta
la modifica del titolo dell'oggetto
          If NTSCStr(!an_nome).Trim <> "" Or NTSCStr(!an_cognome).Trim <> ""
Then
            'Aggiunge anche il "Sig" o "Doc" o altro che c'è nel titolo
dell'anagrafica
            strOggetto &= " C.A. " & NTSCStr(IIf(NTSCStr(!an_titolo).Trim = "",
"", NTSCStr(!an_titolo) & " ")) & _
                          NTSCStr(!an_cognome) & " " & NTSCStr(!an_nome).Trim
          End If
        End With
      End If


      'In tutte le e-mail inviate nel giorno del compleanno indicato
nell'anagrafica del cliente aggiunge come allegato una immagine di auguri
      If NTSCDate(dttClie.Rows(0)!an_datnasc).Month = Now.Month And
NTSCDate(dttClie.Rows(0)!an_datnasc).Day = Now.Day Then
        'Adeguo le dimensioni dell'array temporaneo per accettare un campo in +
        ReDim strAllegatiTmp(strAllegati.Length)

        'I dati vengono copiati nell'array temporaneo
        For z As Integer = 0 To strAllegati.Length - 1
          strAllegatiTmp(z) = strAllegati(z)
        Next

        'Sostituisco gli allegati vecchi con quelli nuovi
        strAllegati = strAllegatiTmp

        'Aggiungo l'allegato
        strAllegati(strAllegati.Length - 1) = "c:\auguri.jpg" ' <-- Assicurarsi
di avere il file, altrimenti da errore
      End If

      Return True
    Catch ex As Exception
      '-------------------------------------------------------------
      If GestErrorCallThrow() Then
```

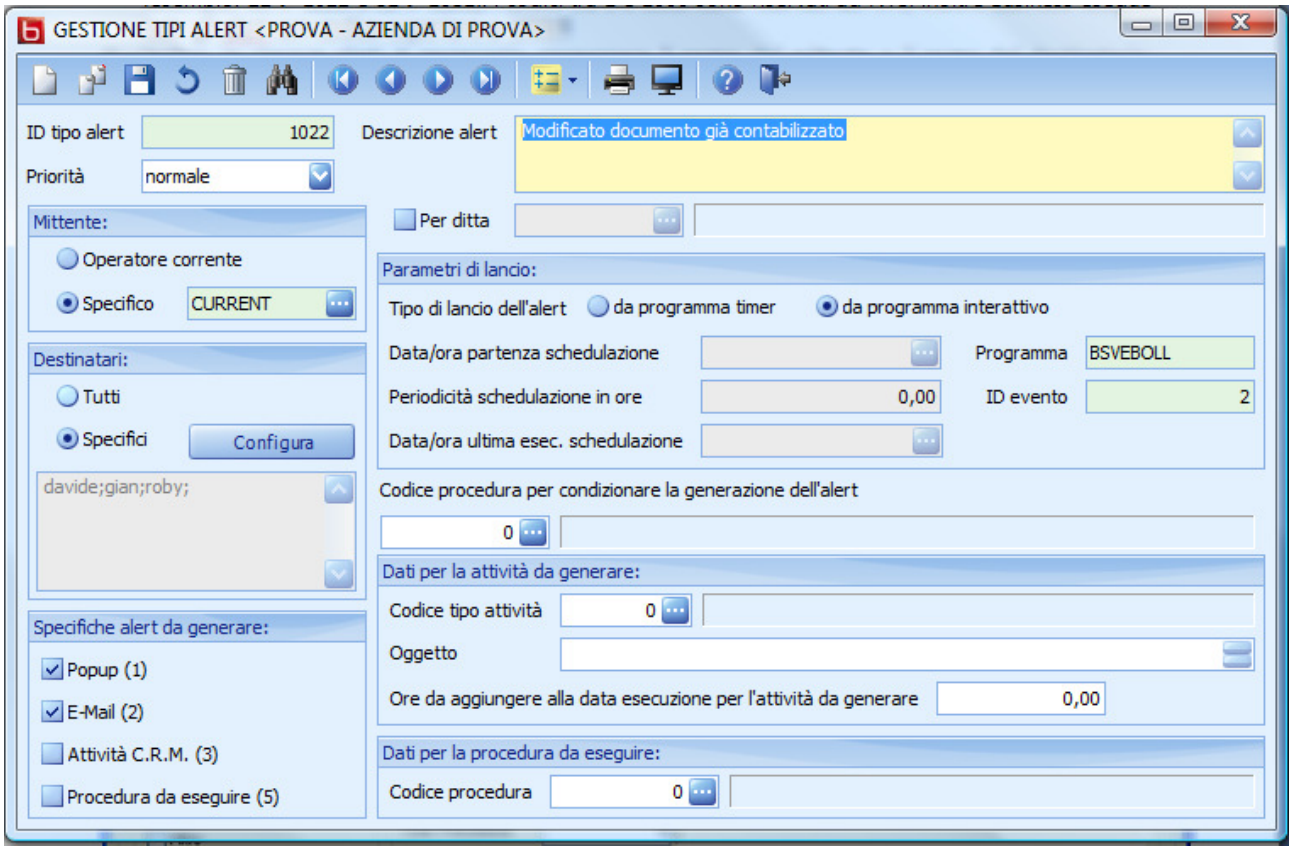
```
        Throw New NTSException(GestError(ex, Me, "", oApp.InfoError, "", False))
      Else
        ThrowRemoteEvent(New NTSEventArgs("", GestError(ex, Me, "",
oApp.InfoError, "", False)))
      End If
      '-------------------------------------------------------------
    End Try
  End Function
```

8. Come si installano gli Alert da programma Interattivo (Classic Start e Busnet)

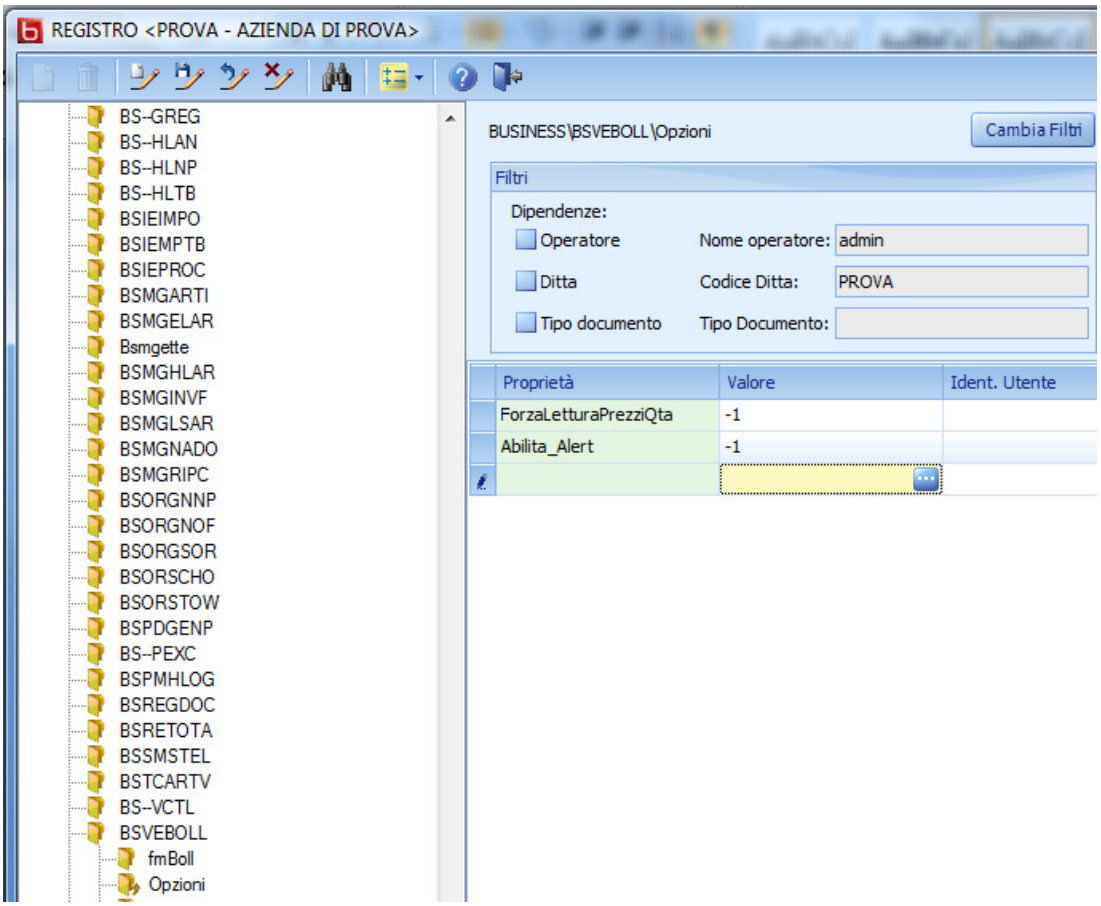
Decalogo di come impostare un alert di tipo “da programma interattivo”.

Questo documento è compilato su due esempi di alert distribuiti come standard (22 e 81) che inviano un pop-up e una email quando un operatore salva un documento già contabilizzato.

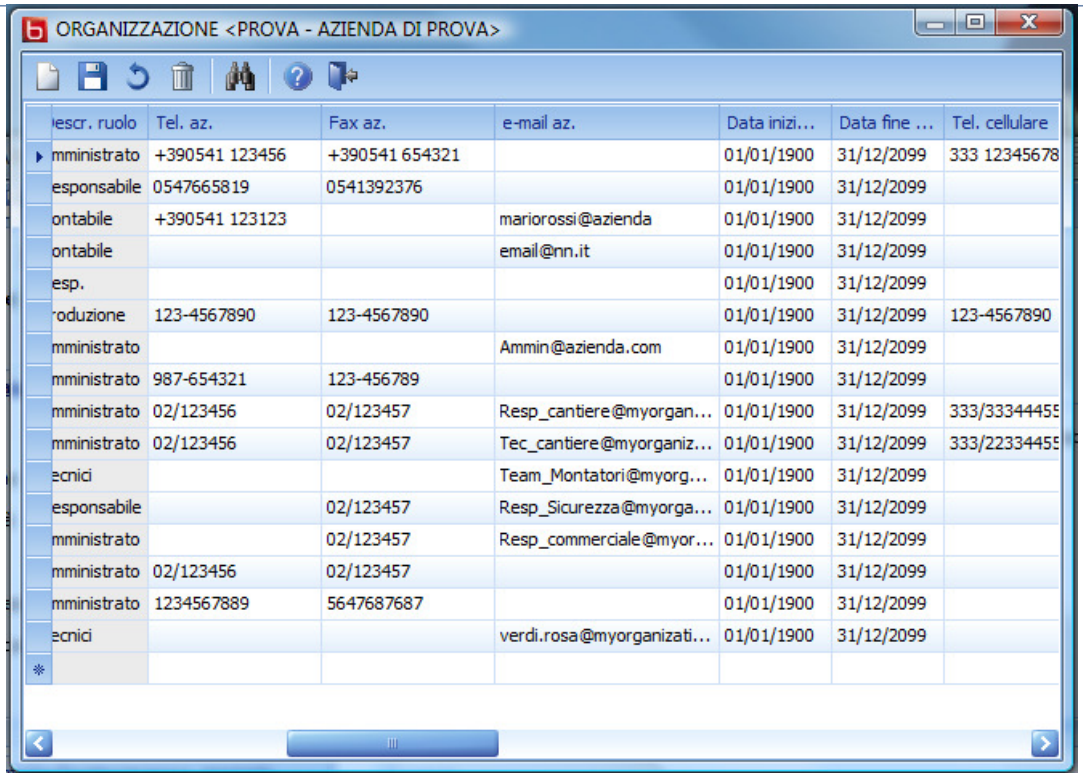
1) Mettere la DLL modificata/personalizzata (e la pdb se si tratta di Busnet) e il prova.mod modificato/personalizzato sul server nella cartella *\Bus* e nella cartella *\Bus\Agg* e aumentare il file Aggnumber.txt di una unità;
2) Fare uscire da Business tutti gli operatori;
3) Lanciare il BUSCONV.EXE con la sola spunta “Solo aggiornamento dati” oppure, se trattasi di installazioni SQL lanciare il BUSCONVS.EXE e riconvertire l’archivio, anche senza ricreare le tabelle allegate;
4) Fare rientrare in Business tutti gli operatori: in questo modo vengono scaricati dalla cartella *\\nomeserver\Bus\Agg* vengono prelevati tutti gli aggiornamenti; se non è abilitata l’opzione di installazione aggiornamenti automatica all’avvio di Business fare lanciare a ciascun operatore l’installazione aggiornamenti dal menu *Start\Programmi\Business*;
5) Entrare in Business, e dal menu 1-O-7 copiare gli alert con codice 22 e 81 in codici superiori a 1000 (esempio: 22→ 1022 e 81→ 1081): i codici tra 1 e 1000 sono riservati ad NTS, inoltre Business esegue solo gli alert con codice superiore a 1000;
6) Inoltre, in ciascun alert, è necessario correggere il nome del mittente e il nome del destinatario, indicando il nome utente (nel senso: utente di Business) corretto ce deve ricevere il pop-up e/o email;



7) Entrare nel registro di Business e abilitare l'opzione **Abilita_Alert** con valore -1 sotto le cartelle interessate (in questo caso BSVEBOLL/OPZIONI e BSVEFDIN/OPZIONI);

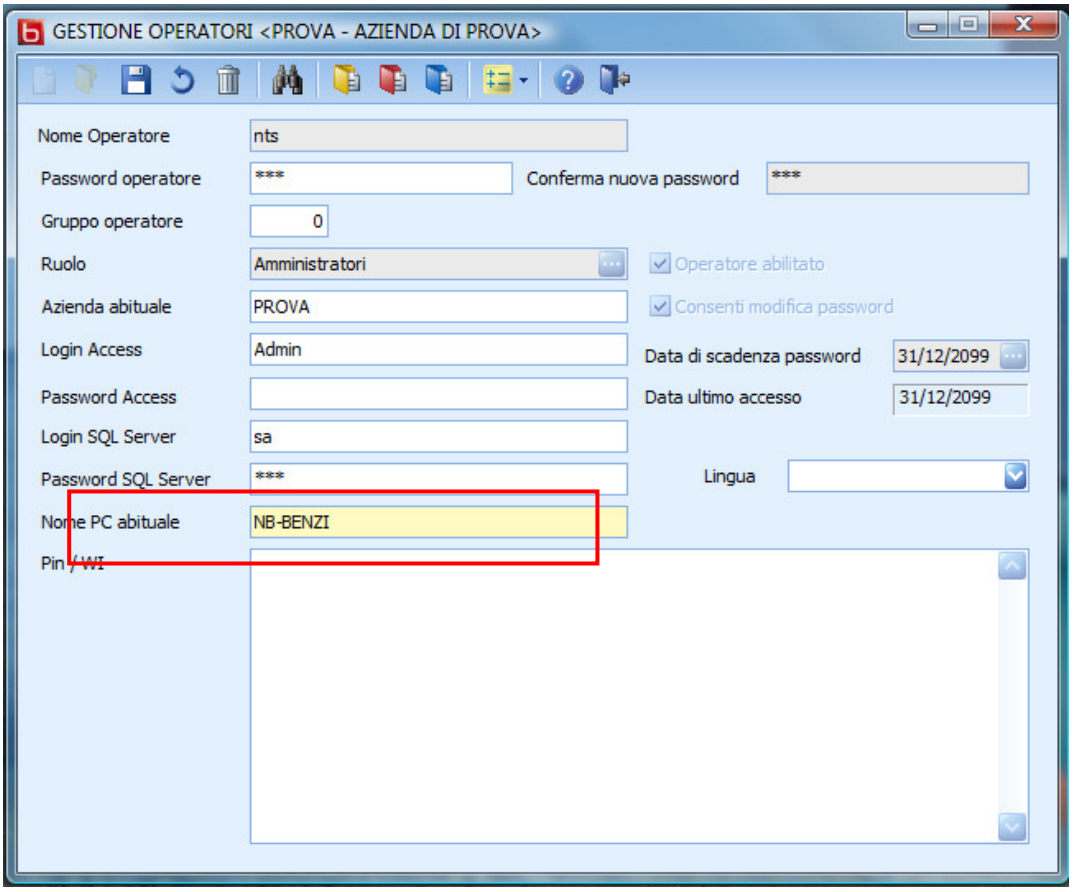


8) Nell'anagrafica ditta, menu 1-C-1, andare nel menu *Record (Pulsante Giallo)/Organizzazione* e indicare in modo particolare: ciascun ruolo, nome operatore e indirizzo email di ciascun operatore presente nell'organigramma aziendale;

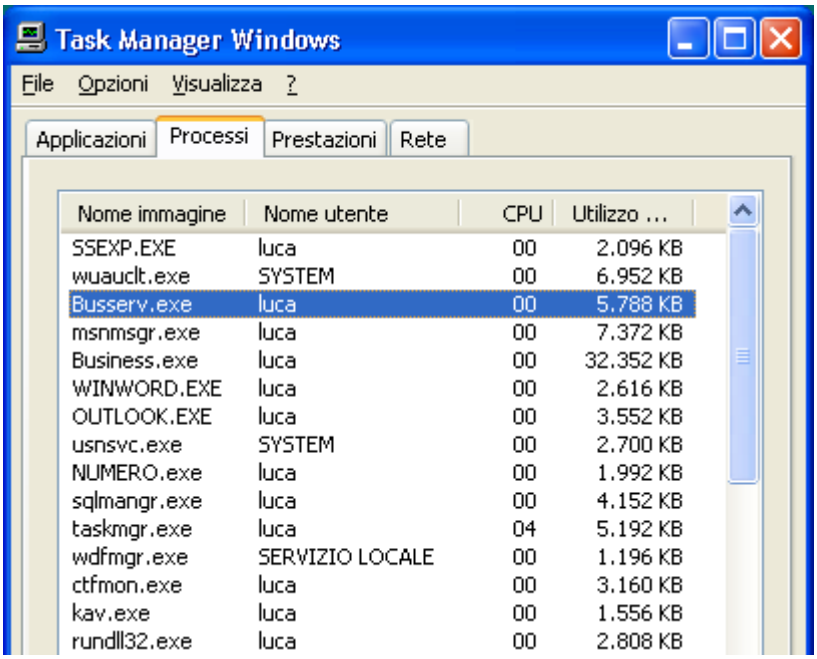


escr. ruolo	Tel. az.	Fax az.	e-mail az.	Data inizi...	Data fine ...	Tel. cellulare
mministrato	+390541 123456	+390541 654321		01/01/1900	31/12/2099	333 12345678
esponsabile	0547665819	0541392376		01/01/1900	31/12/2099	
ontabile	+390541 123123		mariorossi@azienda	01/01/1900	31/12/2099	
ontabile			email@nn.it	01/01/1900	31/12/2099	
esp.				01/01/1900	31/12/2099	
oduzione	123-4567890	123-4567890		01/01/1900	31/12/2099	123-4567890
mministrato			Ammin@azienda.com	01/01/1900	31/12/2099	
mministrato	987-654321	123-456789		01/01/1900	31/12/2099	
mministrato	02/123456	02/123457	Resp_cantiere@myorgan...	01/01/1900	31/12/2099	333/33344455
mministrato	02/123456	02/123457	Tec_cantiere@myorganiz...	01/01/1900	31/12/2099	333/22334455
ecnici			Team_Montatori@myorg...	01/01/1900	31/12/2099	
esponsabile		02/123457	Resp_Sicurezza@myorga...	01/01/1900	31/12/2099	
mministrato		02/123457	Resp_commerciale@myor...	01/01/1900	31/12/2099	
mministrato	02/123456	02/123457		01/01/1900	31/12/2099	
mministrato	1234567889	5647687687		01/01/1900	31/12/2099	
ecnici			verdi.rosa@myorganizati...	01/01/1900	31/12/2099	

9) Nel menu degli operatori 1-O-3 è necessario indicare il nome del pc abituale (almeno per gli operatori che devono ricevere dei Pop-up);



10) Abilitare il programma **BUSSERV.EXE** (magari da mettere in esecuzione automatica) su tutti i pc che devono ricevere dei Pop-up (attenzione: per vedere se è attivo lo vedo solo dal Task Manager di Windows).

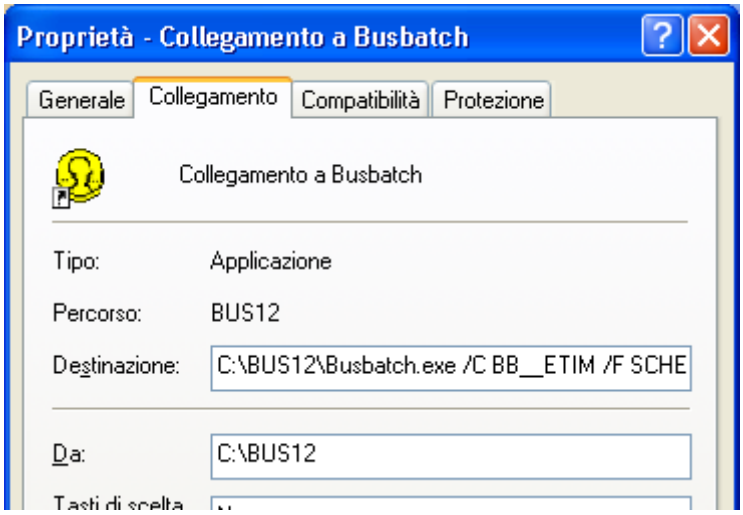


9. Come si installano gli Alert da programma Timer (Solo Classic Start)

Decalogo di come impostare un alert di tipo “da programma timer”.

Questo documento è compilato esempio di un alert personalizzato (1002, richiesto da XYZ) che genera un’attività CRM se il lead non viene contattato da *n* giorni (dove *n* è il numero indicato nella prima estensione-lead numerica).

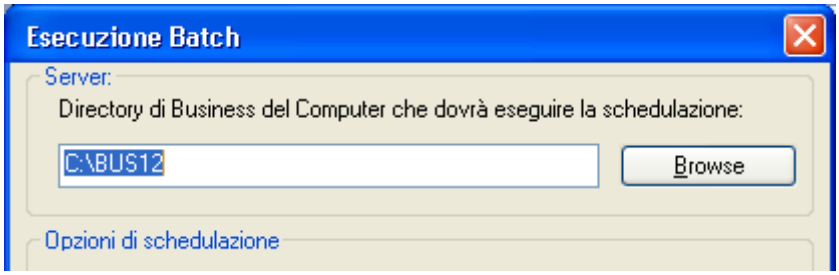
1) Mettere in esecuzione automatica il BUSBATCH.EXE. Attenzione! Questo programma deve possedere alcuni parametri nella riga di comando.



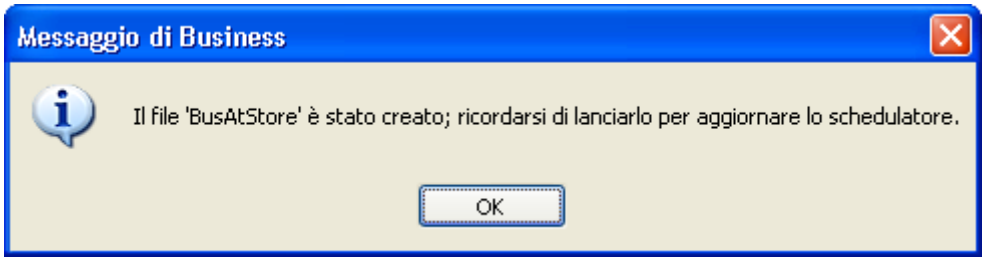
C:\BUS12\Busbatch.exe /C BB__ETIM /F SCHE_BB--ETIM_ADMIN.BUB /U ADMIN /A PROVA12

E’ necessario generare preventivamente il file SCHE_BB—ETIM_ADMIN.BUB: per generarlo si può procedure in questo modo:

- Entrare in Business con il medesimo operatore indicato nella stringa di comando (nell’esempio sopra: ADMIN);
- Andare nel menu V-G “Esecuzione automatica procedure di alerting”;
- Premere la combinazione di tasti CTRL+ALT+F5
- Indicare nella maschera il percorso della cartella di installazione di Business (nell’esempio: C\Bus12)



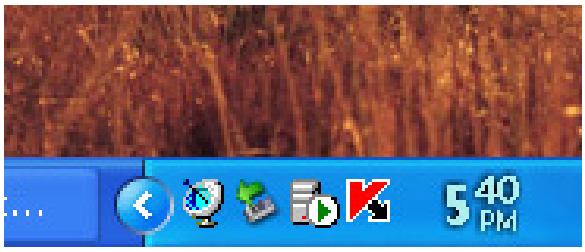
- Al termine premere “Ok”: apparirà questo messaggio.



- Nella cartella C:\Bus12\Asc mi ritrovo già generato il file SCHE_BB—ETIM_ADMIN.BUB

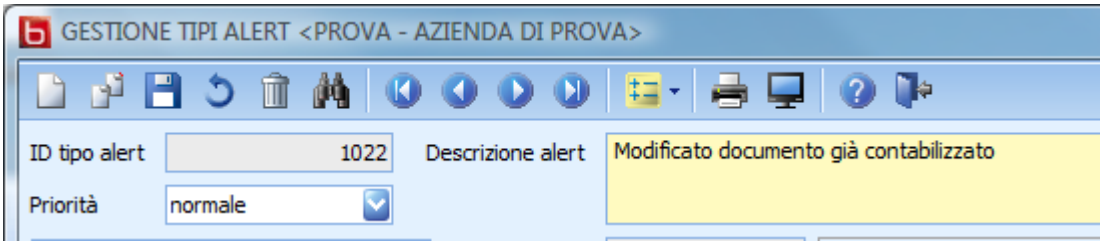
Attenzione!!! Il programma BUSBATCH.EXE opera esclusivamente sul profilo Business.

2) Se il programma è correttamente in esecuzione apparirà accanto all'orologio di sistema una parabola, simbolo dell'attesa del BUSBATCH.EXE

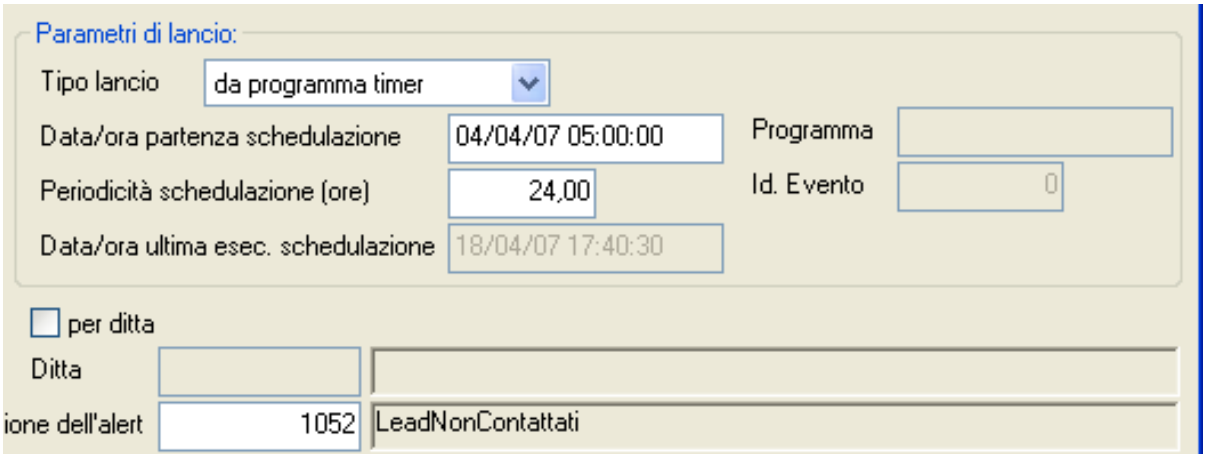


3) Entrare in Business e entrare nel menu 1-O-7 "Gestione Tipi Alert";

4) Creare il nuovo Alert con codice 1002 e completare la descrizione;

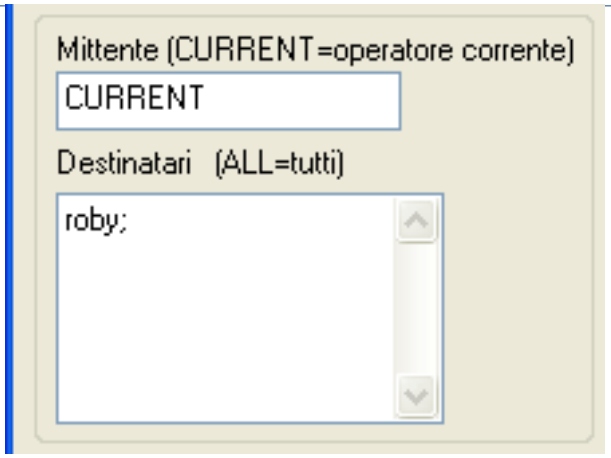


5) Completare il riquadro "Parametri di lancio": tipo di lancio deve essere "da programma timer"; la casella "Data/ora partenza schedulazione" contiene, soprattutto, l'ora di quando deve essere schedulato questo alert (se voglio che parta alle 8 della mattina dovrò indicare 08.00.00); inoltre indicare in ore la periodicità; nell'esempio riportato qui sotto l'alert verrà generato il 19/04/07 alle 17,40 circa poiché viene scatenato ogni 24 ore dall'ultima esecuzione.

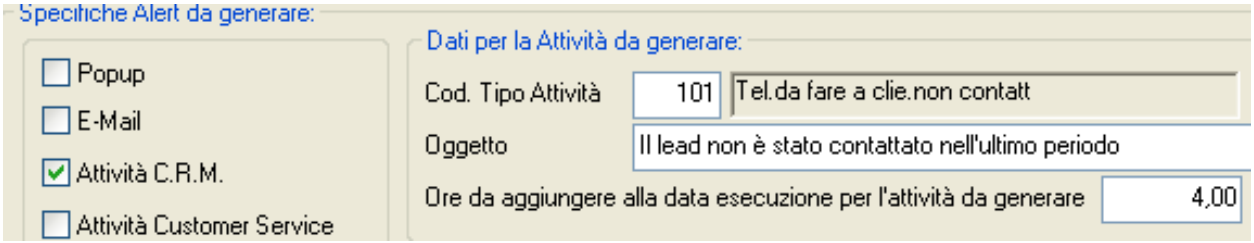


6) Al di sotto del riquadro "Parametri di lancio" troviamo il codice procedura per condizionare la generazione dell'alert: questo codice richiama la tabella PROCED dell'arcproc (che è anche la medesima tabella che contiene le procedutine per l'import/export); in questo esempio andiamo a richiamare la procedura 1052* che contiene tutto il codice VBScript per la generazione dell'alert. *: vedi parte finale per le specifiche sulla procedura 1052.

7) Completare, nella parte di sinistra, la gestione degli operatori, indicando, soprattutto, a quale operatore CRM devono essere assegnate le varie attività: nell'esempio riportato il destinatario dei vari alert sarà l'operatore "Roby";

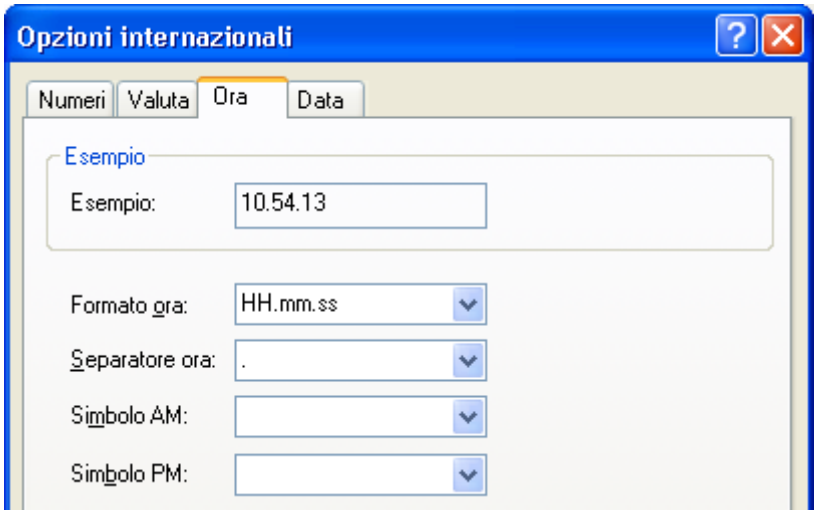


8) In fine, nella sezione in basso, indicare che si vuole generare un'attività CRM e dare le specifiche di questa attività CRM (in questo esempio: verrà generata un'attività con codice 101, con l'oggetto specificato in questo alert e con un orario di esecuzione maggiorato di 4 ore rispetto alla generazione dello stesso);



9) È possibile anche impostare ogni quanto tempo deve partire il programma BB—ETIM: all'interno del registro di Business troviamo la cartella BS—ETIM\OPZIONI con l'opzione Intervallo_Esecuzione_Programma (30 è il default, espresso in minuti): impostando, ad esempio, 5 vuol dire che il BS—ETIM va a vedere se c'è qualche alert da scatenare, con le regole indicate al punto 5.

ATTENZIONE!!! E' cosa buona verificare il settaggio del formato ora nel pannello di controllo: da "Opzioni internazionali e della lingua" verificare che il formato ora abbia come separatore ore il **. (punto)** e non : (doppi punti).



FUNZIONE VBS PER L'ALERT

Vediamo ora, nello specifico, il contenuto della funzione VBS che compila l'alert

```
Function LeadNonContattati(strCodditt, dynMsgOutParam)
  'strCodditt è la ditta da esaminare (= " " significa tutte)
  Dim nGiorni
  Dim strSQL
  Dim snaTmp, snaTmp2, snaTmp3
  Dim strAgg
  If Trim(strCodditt) <> "" Then
    strSQL = " SELECT codditt FROM tabanaz " & _
          " WHERE codditt = " & convstrnullsql(strCodditt)
  Else
    strSQL = " SELECT codditt FROM tabanaz " & _
          " ORDER BY codditt"
  End If
  Set snaTmp = gdb.OpenRecordset(strSQL, dbOpenSnapshot)
  If Not snaTmp.EOF Then
    snatmp.movefirst
    While Not snatmp.eof
      strSQL = " SELECT le_codlead, le_descr1, le_descr2, le_telef, le_conto, le_coddest, ax_num1" & _
            " FROM leads INNER JOIN anaext ON leads.codditt = anaext.codditt" & _
            " AND leads.le_codlead = anaext.ax_codlead" & _
            " WHERE leads.codditt = " & convstrnullsql(snaTmp("codditt")) & _
            " AND ax_tipork = 'L'" & _
            " ORDER BY le_codlead"
      Set snaTmp2 = gdb.openrecordset(strsql, dbopensnapshot)
      If Not snaTmp2.EOF Then
        snatmp2.movefirst
        While Not snaTmp2.eof
          nGiorni = snaTmp2("ax_num1")
          strSQL = " SELECT ca_codlead FROM cract " & _
                " WHERE codditt = " & convstrnullsql(snaTmp("codditt")) & _
                " AND ca_codlead = " & snaTmp2("le_codlead") & _
                " AND ca_status = 'E'" & _
                " AND ca_dataes > " & convdatasql(DateAdd("d", -nGiorni, Date), false, false)
          Set snaTmp3 = gdb.openrecordset(strsql, dbopensnapshot)
          If snaTmp3.EOF Then
            dynMsgoutParam.AddNew
            dynMsgoutParam("codditt") = snaTmp("codditt")
            strAgg = ""
```

```
            If snaTmp2("le_conto") <> 0 Then
              If snaTmp2("le_coddest") <> 0 Then
               strAgg = " (Cliente " & snaTmp2("le_conto") & " Destinazione " & snaTmp2("le_coddest") & ")"
              Else
               strAgg = " (Cliente " & snaTmp2("le_conto") & ")"
              End If
            End If
            If dynMsgoutParam("strMsg") <> "" Then
             dynMsgoutParam("strMsg") = dynMsgoutParam("strMsg") & vbcrlf & _
                                  "Lead " & snaTmp2("le_codlead") & strAgg & _
                                  " - '" & RTrim(snaTmp2("le_descr1")) & _
                                  " " & RTrim(convnullstr(snaTmp2("le_descr2"))) & _
                                  "' telefono '" & convnullstr(snaTmp2("le_telef")) & _
                                  "' non è stato contattato negli ultimi " & nGiorni & " giorni."
            Else
              dynMsgoutParam("strMsg") = "Lead " & snaTmp2("le_codlead") & strAgg & _
                                  " - '" & RTrim(snaTmp2("le_descr1")) & _
                                  " " & RTrim(convnullstr(snaTmp2("le_descr2"))) & _
                                  "' telefono '" & convnullstr(snaTmp2("le_telef")) & _
                                  "' non è stato contattato negli ultimi " & nGiorni & " giorni."
            End If
            dynMsgoutParam("strCodLead") = snaTmp2("le_codlead")
            dynMsgoutParam.Update
          End If
          snaTmp3.close
          Set snaTmp3 = nothing
          snaTmp2.movenext
       Wend
     End If
     snaTmp2.close
     Set snaTmp2 = nothing
     snatmp.movenext
   Wend
End If
snaTmp.Close
Set snaTmp = Nothing
If dynMsgoutparam.recordcount > 0 Then
  LeadNonContattati = True
Else
  LeadNonContattati = False
```

```
End if
```

End Function

I valori che vengono sempre passati sono:

dynMsgoutParam.AddNew apre il recordset per la creazione degli alert timer; **dynMsgoutParam("codditt")** deve essere valorizzato con il codice ditta sulla quale deve essere generato l'alert timer; **dynMsgoutParam("strMsg")** deve essere valorizzato con il testo dell'alert (che nel caso di email sarà il testo dell'email, in caso di attività CRM sarà compilato nel campo note); **dynMsgoutParam("strCodLead")** deve essere valorizzato con il codice lead.

10. Business NET 2011: aggiunta/modifica di 'Formule' per Contabilità analitica duplice contabile

Le formule sono contenute in BECXFORM.DLL. Per poterle modificare, oppure per aggiungere nuove formule è necessario ereditarla dll suddetta, seguendo le tecniche standard (ovvero realizzare una nuova dll che eredita da CLECXFORM e dichiarare il suo utilizzo in DLLMAP.INI)

Le funzioni devono essere scritte con nome funzione TUTTO IN MAIUSCOLO ed i parametri devono essere come nell'esempio sotto riportato:

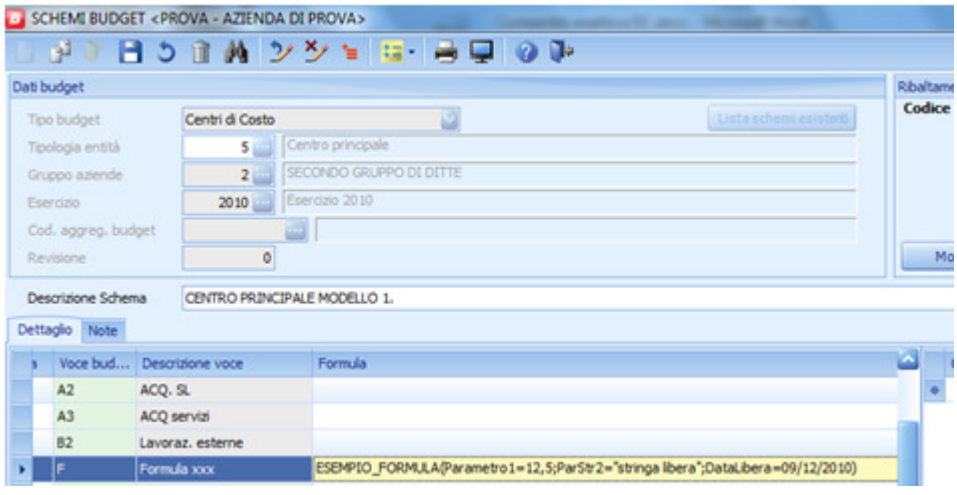
```
Public Overridable Function ESEMPIO_FORMULA(ByVal bPreventivo As Boolean, ByVal strParams As String,
ByRef dtrBud As DataRow) As Boolean
    'esempio di stringa passata alla funzione
    'in schemi budget scrivo ESEMPIO_FORMULA(Parametro1=12,5;ParStr2="stringa
libera";DataLibera=09/12/2010)
    'in dtrBud sono presenti i dati necesari per collegarsi alla riga delbudget, lo schema budget e
quanto altro serve per determinare quantità e valore del risultato della formula, sia per dati a
preventivo che a consuntivo
    'se la funzione restituirà TRUE il dato verrà aggiornato nel budget
    Try
      'ottengo l'elenco dei parametri
      Dim strT() As String = strParams.Split(";"c)

      'modifico i dati da restituire
      If bPreventivo Then
        dtrBud!bu_qta = 10.5 + NTSCDec(strT(0).Substring(11))
        dtrBud!bu_val = 999.33
      Else
        dtrBud!bu_qtacon = 10.5
        dtrBud!bu_valcon = 999.33
      End If
      dtrBud.AcceptChanges()

      'se restituisco FALSE i dati non verranno aggiornati
      Return True

    Catch ex As Exception
      '-----------------------------------------------------------
      If GestErrorCallThrow() Then
        Throw New NTSException(GestError(ex, Me, "", oApp.InfoError, "", False))
      Else
        ThrowRemoteEvent(New NTSEventArgs("", GestError(ex, Me, "", oApp.InfoError, "", False)))
      End If
      '-----------------------------------------------------------
    End Try
  End Function
```

La funzione potrà essere utilizzata nel programma 'Impostazione schemi budget' in questo modo



NOTE TECNICHE

Revisioni

29/07/2009	Versione originale
07/08/2009	Inserito in fondo a source extender la descrizione di come richiamare una funzione di script da entity personalizzato
07/10/2009	Aggiunto “6.9. Come modificare il valore di riga e il calcolo dei totali nei documenti di magazzino”
09/12/2009	Aggiunti/commentati esempi di scripting complessi
09/04/2010	Inserito “Esempio aggiunta di nuovi eventi”
20/07/2010	Inserito descrizione del funzionamento del setfocus nel capitolo 5
06/09/2010	Modificato capitolo 5-Source extender corrette imprecisioni e inserito descrizione errore in funzioni di eventi personalizzati di interfaccia BO.
09/12/2010	Inserito “Business NET 2011: aggiunta/modifica di ‘Formule’ per Contabilità analitica duplice contabile”
28/12/2011	Documentata nuova proprietà nello scripting delle stampe PaperSizeCond

NTS Informatica