FoxcodePlus - New features for IntelliSense Microsoft Visual FoxPro 9

By Rodrigo D. Bruscain – Version Beta 3.13.2 – Last updated May 26, 2013

FoxcodePlus does not replace VFP's IntelliSense; it enhances VFP's IntelliSense where the default VFP does not adequately help or does absolutely nothing. The idea of FoxcodePlus is to bring a little of the functionality of Visual Studio IntelliSense to VFP. (In others words, to make it faster and avoid making mistakes when writing VFP code.)

See the new features below:

1- Incremental IntelliSense for functions, commands, variables and so on.

*As in Visual Studio, everything that is typed in the coding screen (Edit window), VFP incrementally searches the entire contents of the program and builds the IntelliSense. The incremental search happens upon each a key pressed. In "IntelliSense Manager", you can choose in "**Incrementally search**" combobox how to increment the IntelliSense.*

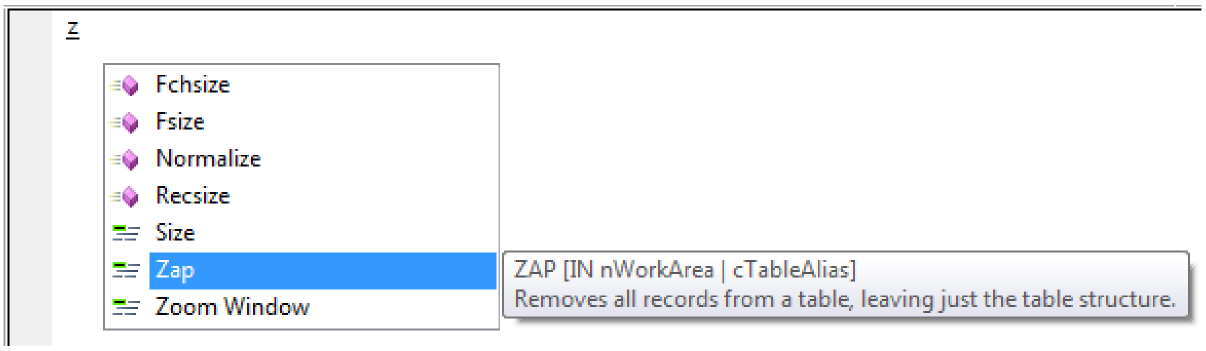

```
z
Fchsize
Fsize
Normalize
Recsize
Size
Zap
Zoom Window
```
ZAP [IN nWorkArea | cTableAlias]
Removes all records from a table, leaving just the table structure.

Image 1.10

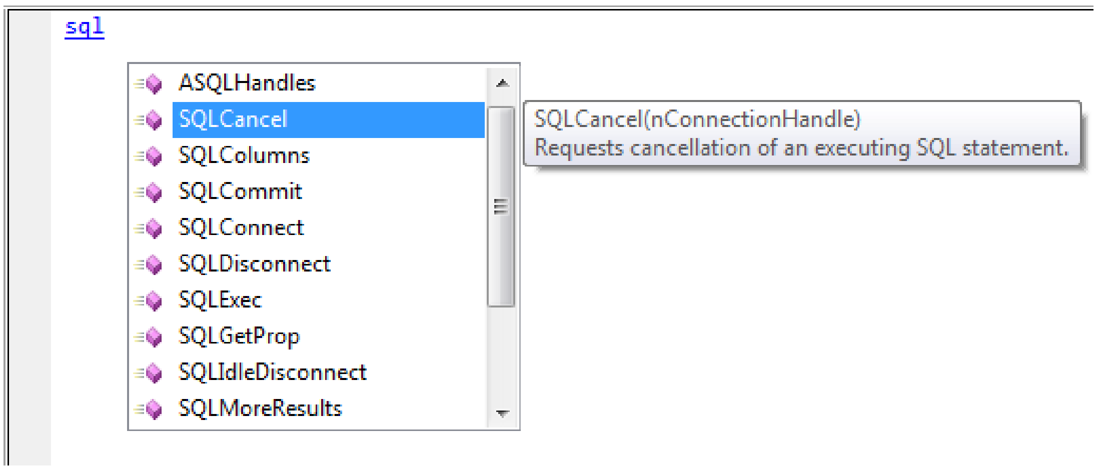

```
sql
ASQLHandles
SQLCancel
SQLColumns
SQLCommit
SQLConnect
SQLDisconnect
SQLExec
SQLGetProp
SQLIdleDisconnect
SQLMoreResults
```
SQLCancel(nConnectionHandle)
Requests cancellation of an executing SQL statement.

Image 1.11

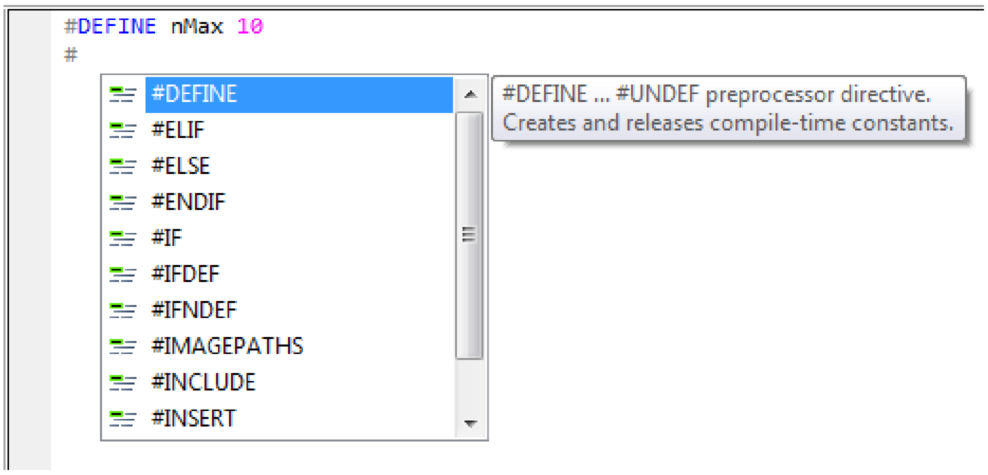
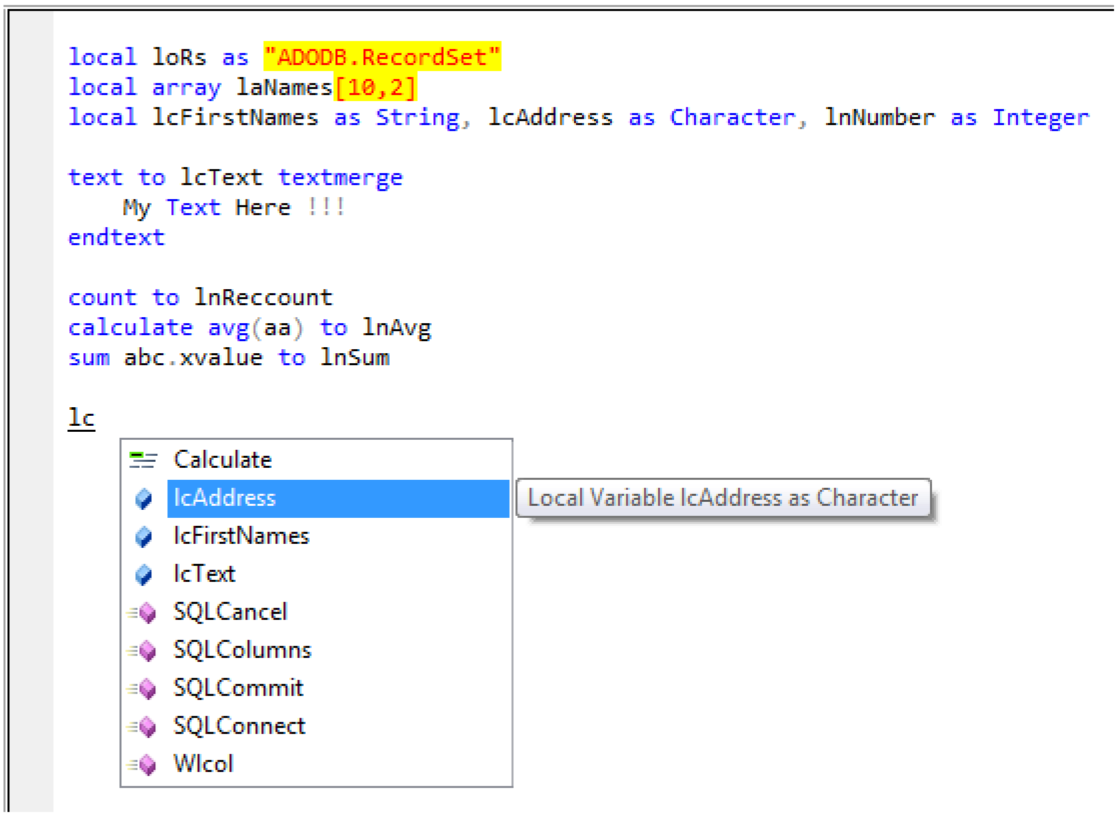

```
#DEFINE nMax 10
#
#DEFINE
#ELIF
#ELSE
#ENDIF
#IF
#IFDEF
#IFNDEF
#IMAGEPATHS
#INCLUDE
#INSERT
```
#DEFINE ... #UNDEF preprocessor directive.
Creates and releases compile-time constants.

Image 1.12

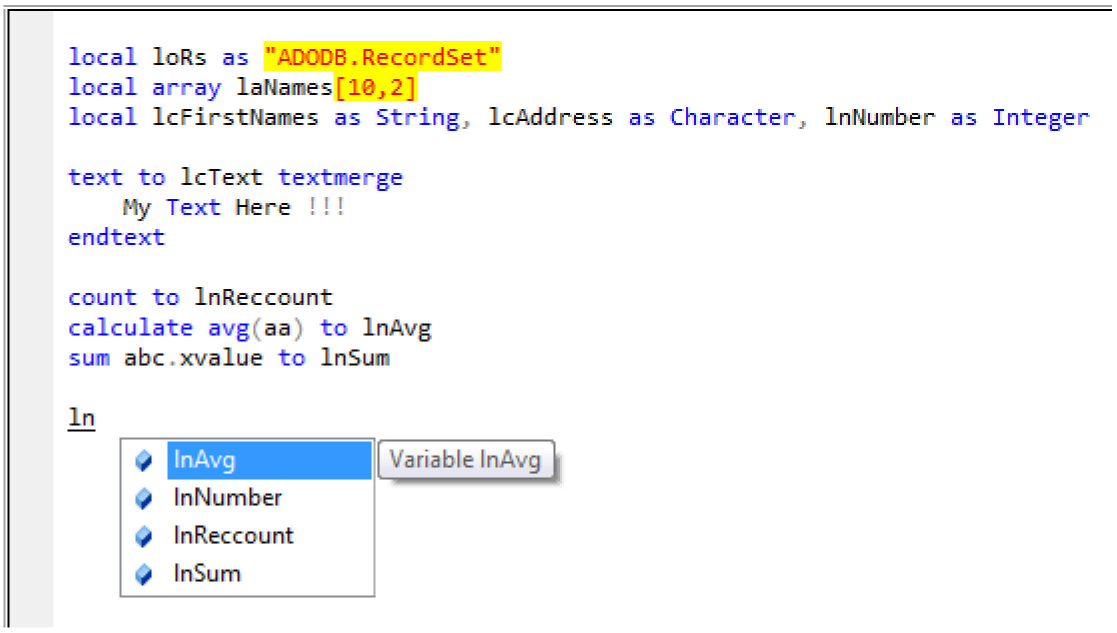
2- Variables at write-time

Working with variables is now easier; IntelliSense captures all kinds of variable declarations. The tooltip indicates the type of the variable if the type has been specified.



```
local loRs as "ADODB.RecordSet"
local array laNames[10,2]
local lcFirstNames as String, lcAddress as Character, lnNumber as Integer

text to lcText textmerge
    My Text Here !!!
endtext

count to lnReccount
calculate avg(aa) to lnAvg
sum abc.xvalue to lnSum

lc
```

Image 2.10

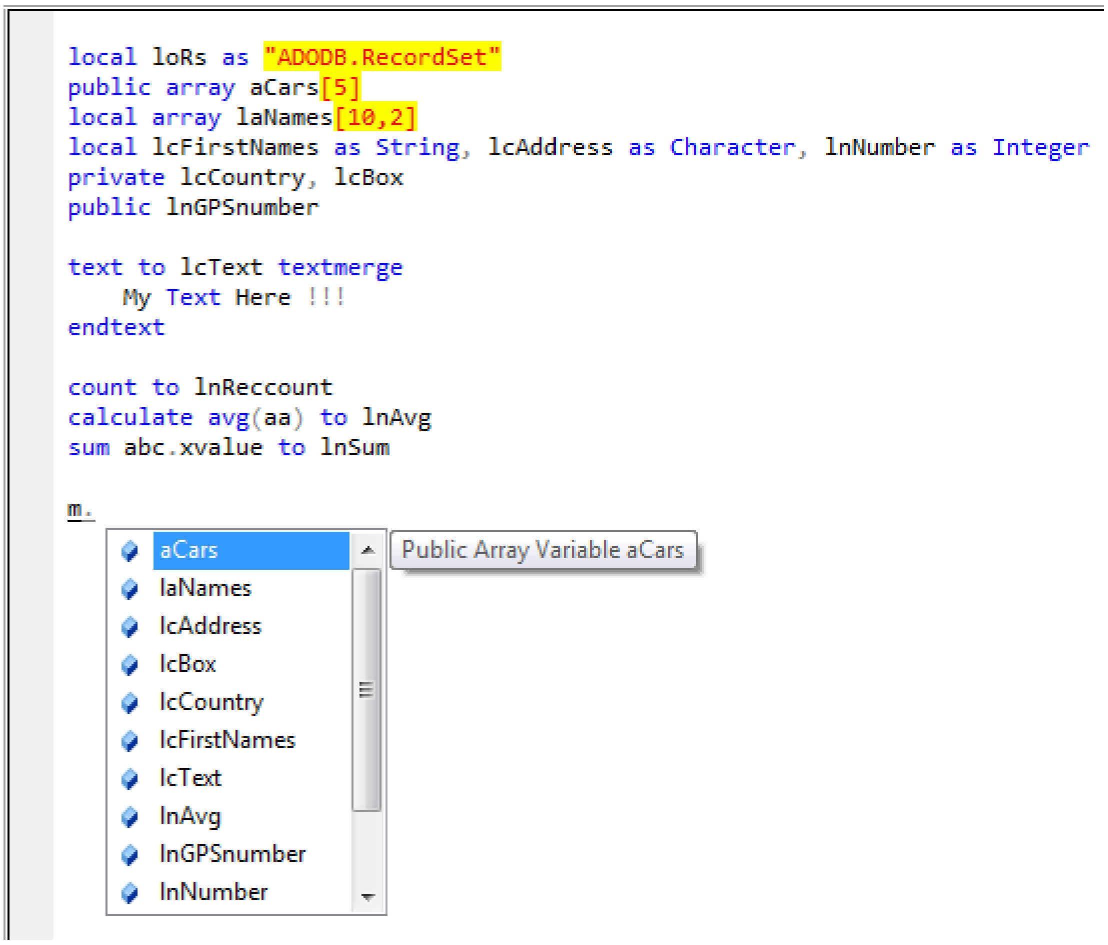


```
local loRs as "ADODB.RecordSet"
local array laNames[10,2]
local lcFirstNames as String, lcAddress as Character, lnNumber as Integer

text to lcText textmerge
    My Text Here !!!
endtext

count to lnReccount
calculate avg(aa) to lnAvg
sum abc.xvalue to lnSum

ln
```

Image 2.11

3- Accessing the list of variables at write-time.

If you want to know all the variables created up until where the cursor is, type "m.", and the IntelliSense will open in a "non-incremental" mode.



```
local loRs as "ADODB.RecordSet"
public array aCars[5]
local array laNames[10,2]
local lcFirstNames as String, lcAddress as Character, lnNumber as Integer
private lcCountry, lcBox
public lnGPSnumber

text to lcText textmerge
    My Text Here !!!
endtext

count to lnReccount
calculate avg(aa) to lnAvg
sum abc.xvalue to lnSum

m.
```

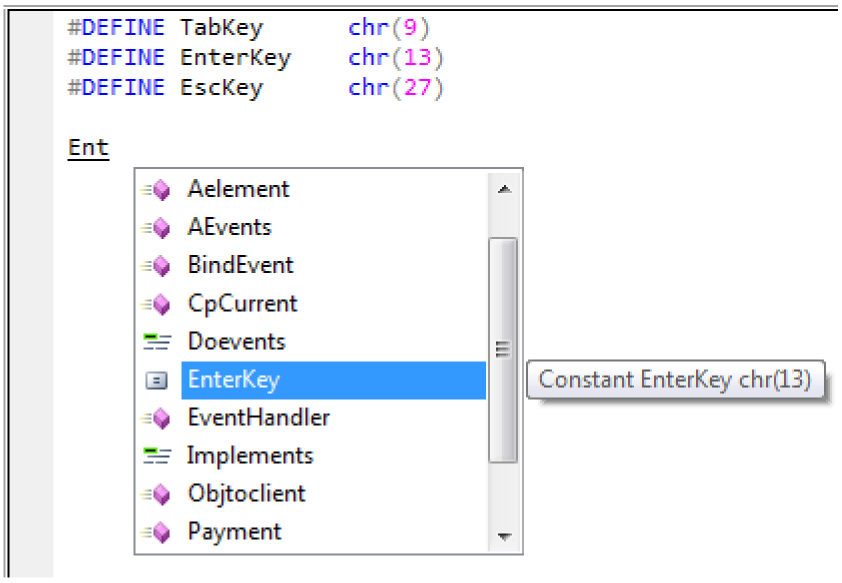
```
aCars
laNames
lcAddress
lcBox
lcCountry
lcFirstNames
lcText
lnAvg
lnGPSnumber
lnNumber
```

Public Array Variable aCars

Image 3.10

4- Constants at write-time

Constants created can also be accessed by IntelliSense. The tooltip displays the value of the constant.



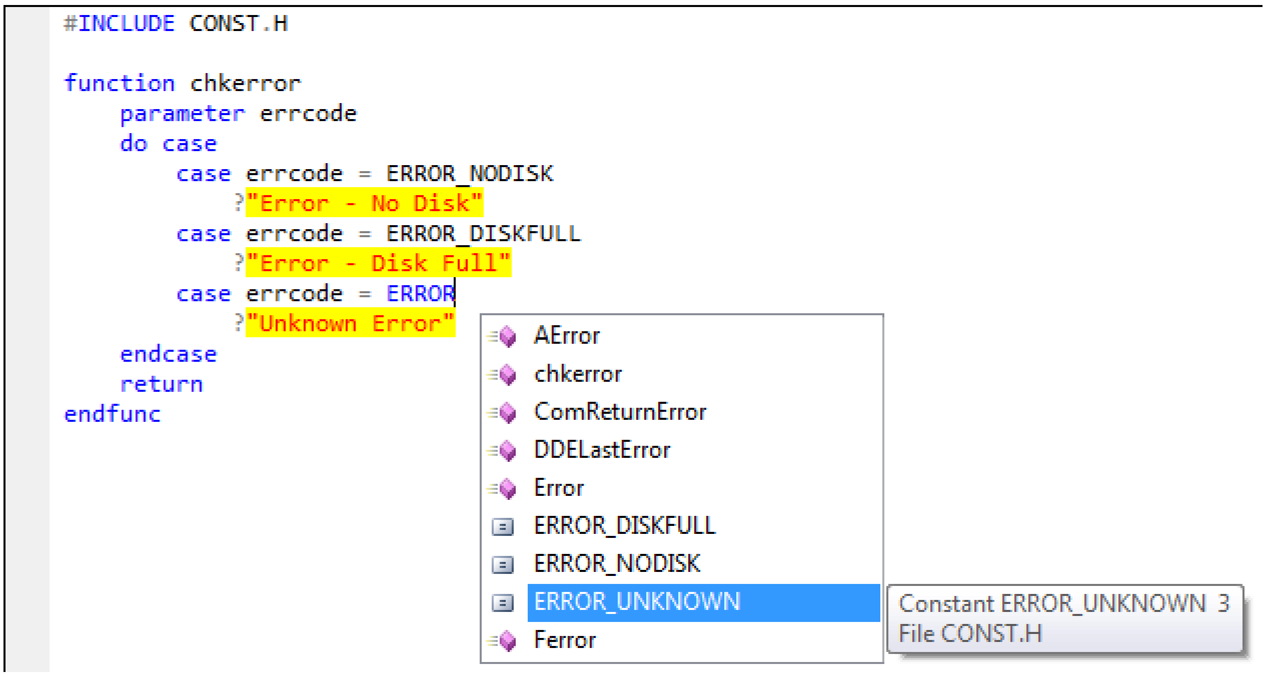
```
#DEFINE TabKey      chr(9)
#DEFINE EnterKey    chr(13)
#DEFINE EscKey      chr(27)

Ent
```

```
Aelement
AEvents
BindEvent
CpCurrent
Doevents
EnterKey
EventHandler
Implements
Objtoclient
Payment
```

Constant EnterKey chr(13)

Image 4.10

Constants in file.H included with command #INCLUDE can be accessed by IntelliSense. The image below shows us the constants inside CONST.H file and tooltip show us the constant name, value and file name.

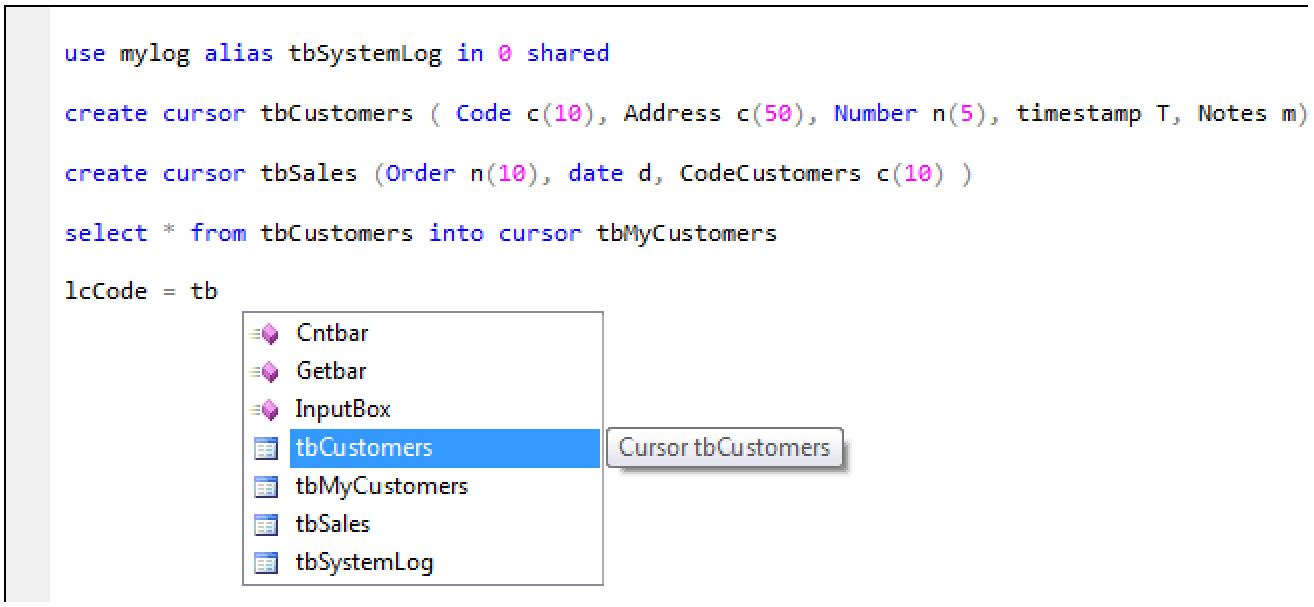


```
#INCLUDE CONST.H

function chkerror
    parameter errcode
    do case
        case errcode = ERROR_NODISK
            ?"Error - No Disk"
        case errcode = ERROR_DISKFULL
            ?"Error - Disk Full"
        case errcode = ERROR
            ?"Unknown Error"
    endcase
    return
endfunc
```

Image 4.11

5- Tables at write-time and at run-time

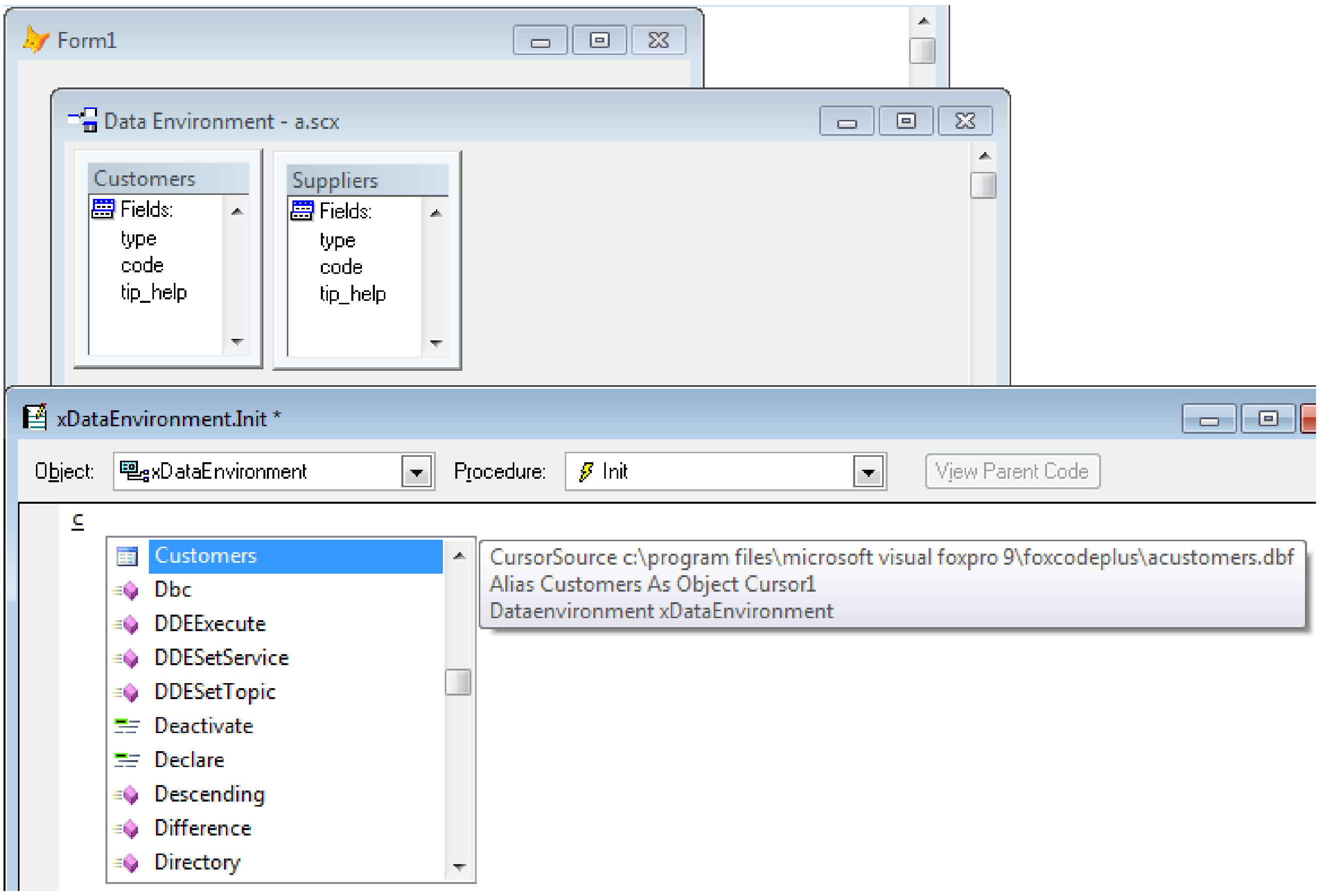
Tables that are created and/or opened, as in the picture below, are also included in the IntelliSense. The tooltip shows the way the table was opened.



```
use mylog alias tbSystemLog in 0 shared

create cursor tbCustomers ( Code c(10), Address c(50), Number n(5), timestamp T, Notes m)

create cursor tbSales (Order n(10), date d, CodeCustomers c(10) )

select * from tbCustomers into cursor tbMyCustomers

lcCode = tb
```

Image 5.10

IntelliSense for tables in DataEnvironment Forms (VFP tables)



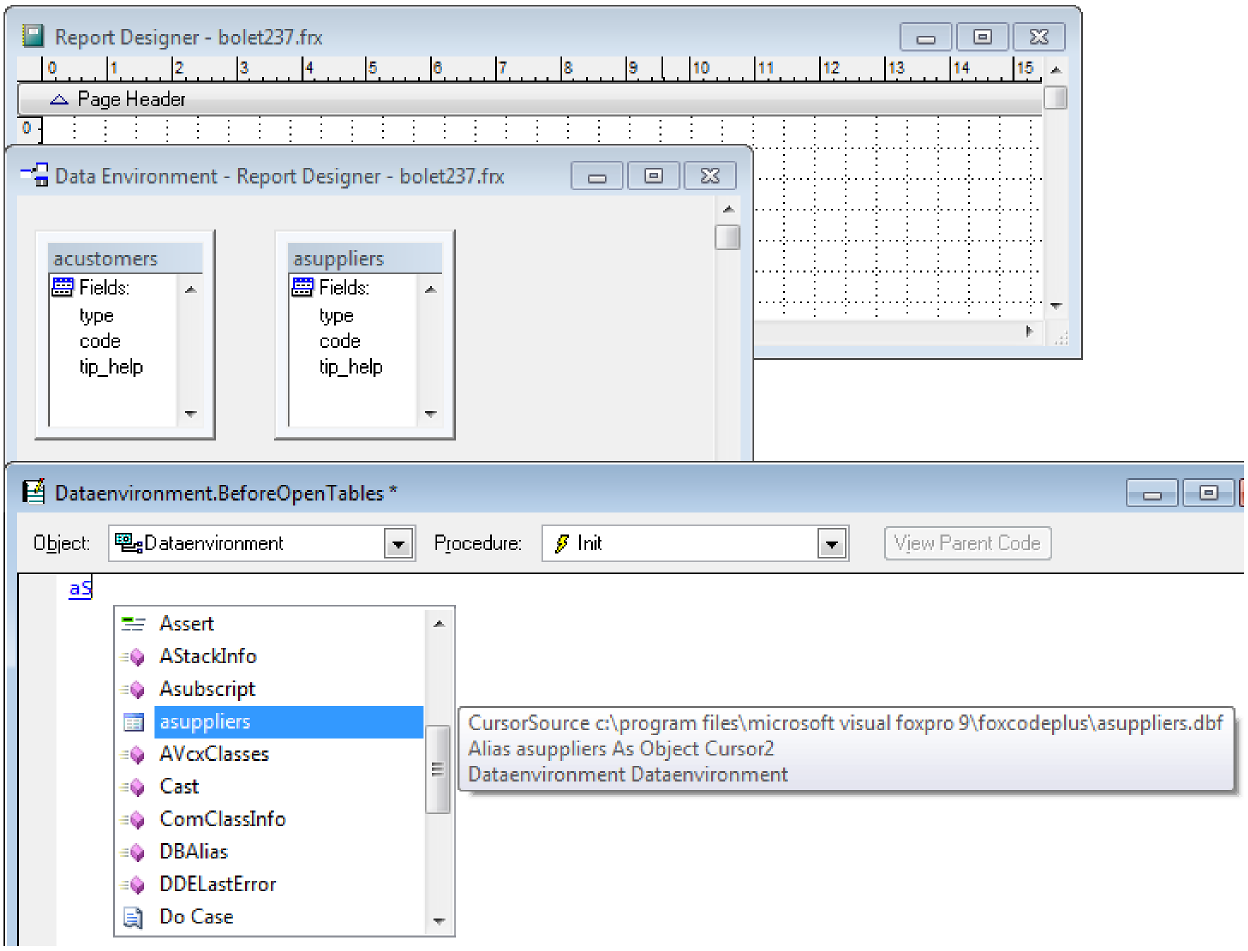
Object: xDataEnvironment Procedure: Init

```
c
```

CursorSource c:\program files\microsoft visual foxpro 9\foxcodeplus\acustomers.dbf
Alias Customers As Object Cursor1
Dataenvironment xDataEnvironment

Image 5.11

IntelliSense for tables in DataEnvironment Reports (VFP tables)



Object: Dataenvironment Procedure: Init

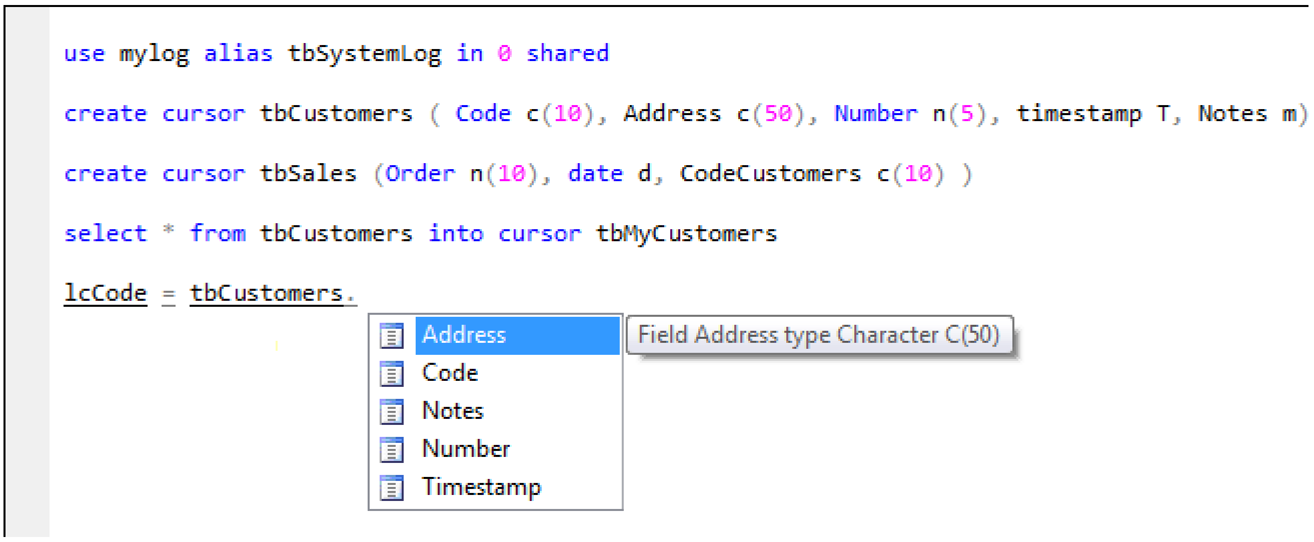
```
aS
```

CursorSource c:\program files\microsoft visual foxpro 9\foxcodeplus\asuppliers.dbf
Alias asuppliers As Object Cursor2
Dataenvironment Dataenvironment

Image 5.12

6- Fields at write-time and at run-time

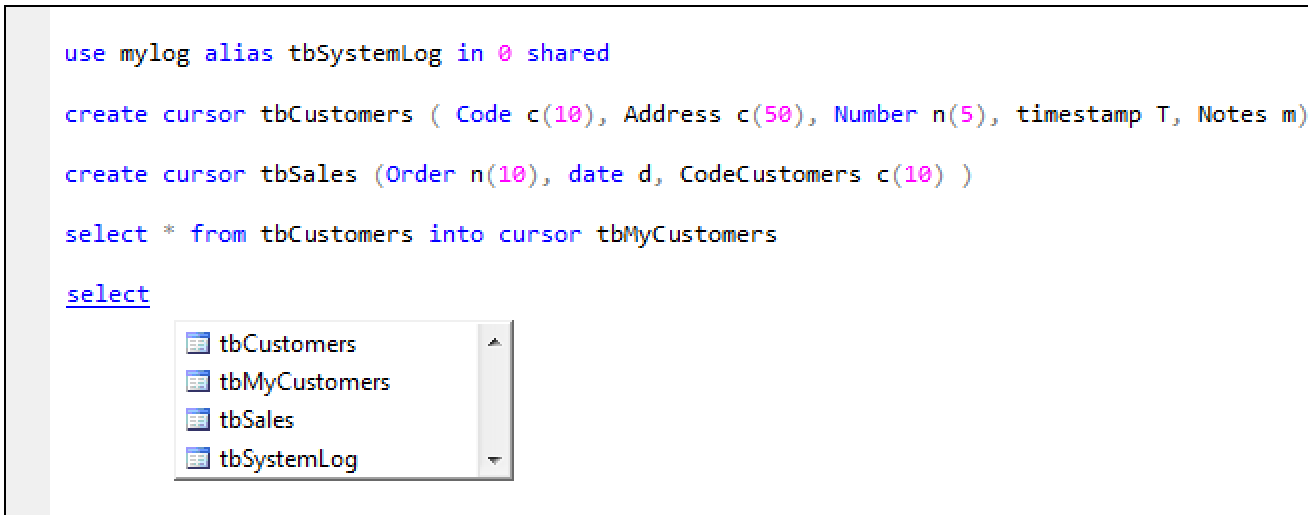
Table fields are also included in the IntelliSense. The type of the field is displayed in the tooltip.
Note: Table fields created by "SELECT - SQL command" are not supported.



```
use mylog alias tbSystemLog in 0 shared

create cursor tbCustomers ( Code c(10), Address c(50), Number n(5), timestamp T, Notes m)

create cursor tbSales (Order n(10), date d, CodeCustomers c(10) )

select * from tbCustomers into cursor tbMyCustomers

lcCode = tbCustomers.
```

Image 6.10

7- Selecting a table with the command "Select" or all commands with the clause "IN"



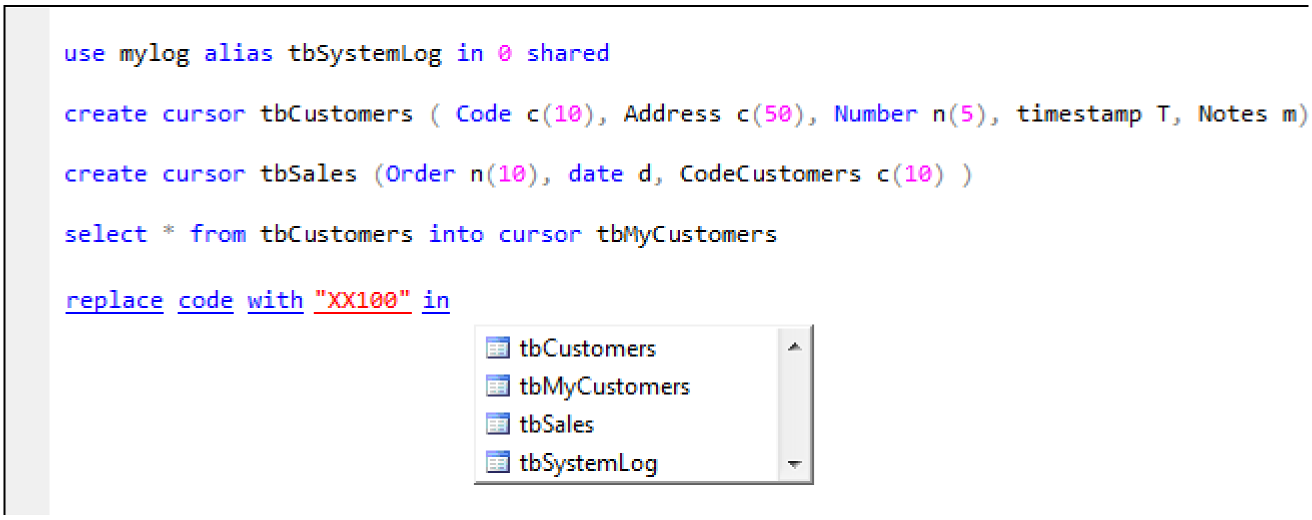
```
use mylog alias tbSystemLog in 0 shared

create cursor tbCustomers ( Code c(10), Address c(50), Number n(5), timestamp T, Notes m)

create cursor tbSales (Order n(10), date d, CodeCustomers c(10) )

select * from tbCustomers into cursor tbMyCustomers

select
```

Image 7.10

All commands for manipulating data in tables that contain the clause "IN" interact directly with IntelliSense, so all the tables created and/ or opened in the program code are presented.

Below is the list of supported commands with the clause "IN"

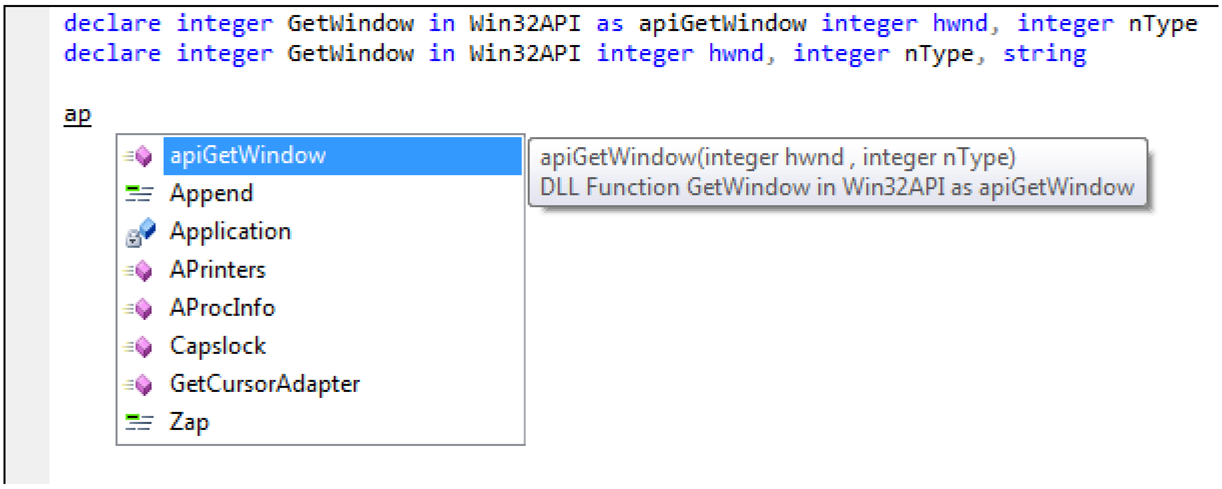
– Append	– Recall
– Replace	– Seek
– Blank	– Select
– Calculate	– Set Filter
– Delete	– Set Order To
– Display	– Set Relation
– Flush	– Skip
– Go | Goto	– Unlock
– List	– Zap



```
use mylog alias tbSystemLog in 0 shared

create cursor tbCustomers ( Code c(10), Address c(50), Number n(5), timestamp T, Notes m)

create cursor tbSales (Order n(10), date d, CodeCustomers c(10) )

select * from tbCustomers into cursor tbMyCustomers

replace code with "XX100" in
```

Image 7.11

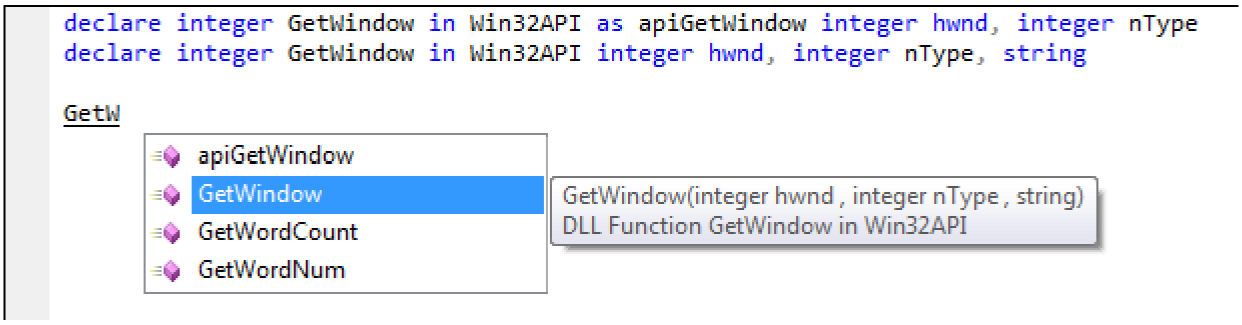
8- APIs at write-time and at run-time



```
declare integer GetWindow in Win32API as apiGetWindow integer hwnd, integer nType
declare integer GetWindow in Win32API integer hwnd, integer nType, string

ap
```

apiGetWindow
Append
Application
APrinters
AProcInfo
Capslock
GetCursorAdapter
Zap
apiGetWindow(integer hwnd , integer nType)
DLL Function GetWindow in Win32API as apiGetWindow

Image 8.10



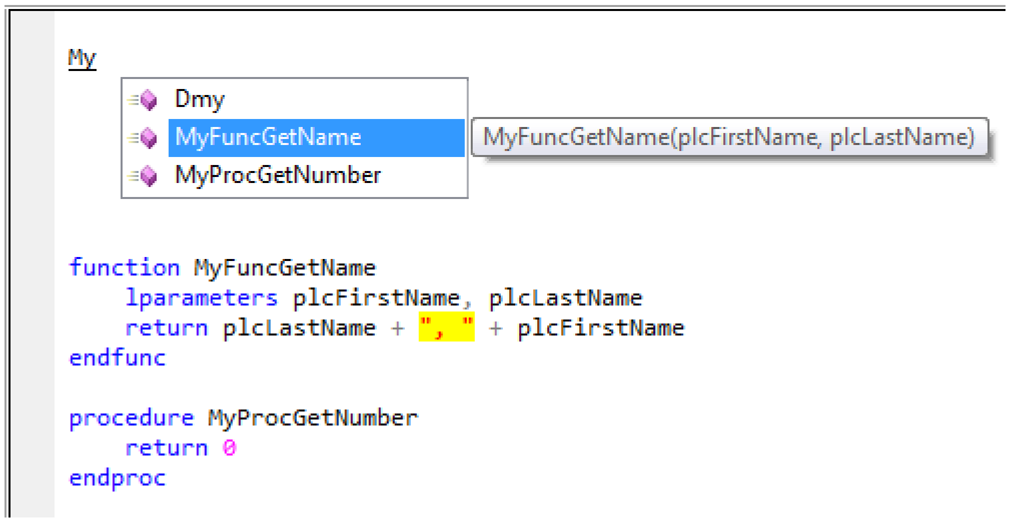
```
declare integer GetWindow in Win32API as apiGetWindow integer hwnd, integer nType
declare integer GetWindow in Win32API integer hwnd, integer nType, string

GetW
```

apiGetWindow
GetWindow
GetWordCount
GetWordNum
GetWindow(integer hwnd , integer nType , string)
DLL Function GetWindow in Win32API

Image 8.11

9- Functions and Procedures at write-time

Functions and Procedures created in the current PRG.



```
My
```

- Dmy
- MyFuncGetName
- MyProcGetNumber

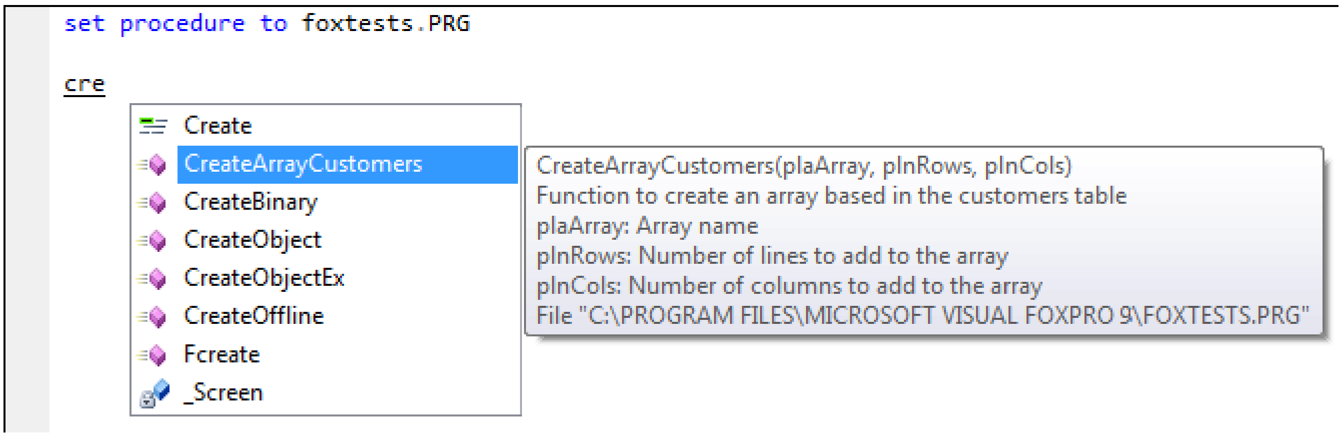
MyFuncGetName(plcFirstName, plcLastName)

```
function MyFuncGetName
    lparameters plcFirstName, plcLastName
    return plcLastName + ", " + plcFirstName
endfunc

procedure MyProcGetNumber
    return 0
endproc
```

Image 9.10

Functions and Procedures created in the others PRGs invocated by SET PROCEDURE TO…

SET PROCEDURE TO … in memory is also considered.



```
set procedure to foxtests.PRG

cre
```

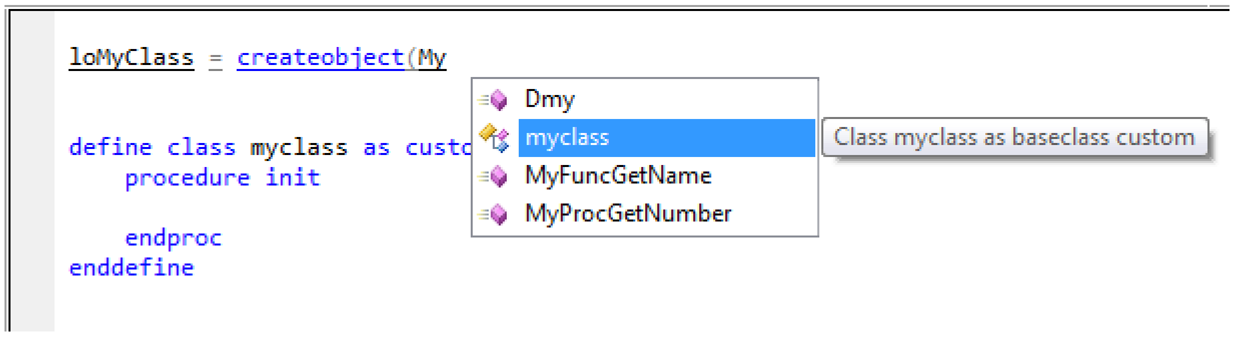
- Create
- CreateArrayCustomers
- CreateBinary
- CreateObject
- CreateObjectEx
- CreateOffline
- Fcreate
- _Screen

CreateArrayCustomers(plaArray, plnRows, plnCols)
Function to create an array based in the customers table
plaArray: Array name
plnRows: Number of lines to add to the array
plnCols: Number of columns to add to the array
File "C:\PROGRAM FILES\MICROSOFT VISUAL FOXPRO 9\FOXTESTS.PRG"

Image 9.11

10- Classes at write-time

Classes created in the current PRG



```
loMyClass = createobject(My


define class myclass as custo
    procedure init

    endproc
enddefine
```

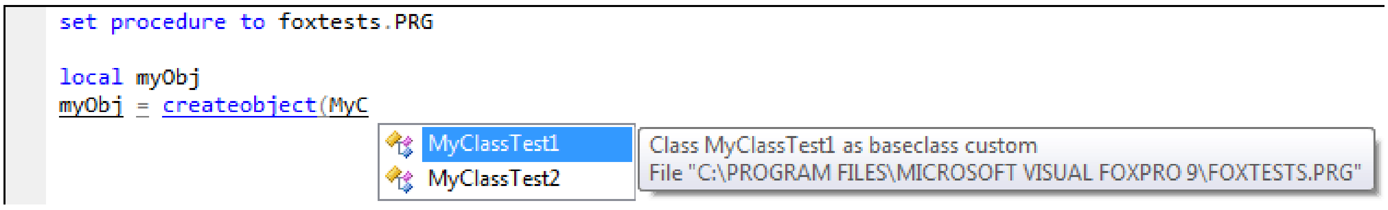
- Dmy
- myclass
- MyFuncGetName
- MyProcGetNumber

Class myclass as baseclass custom

Image 10.10

Classes created in the others PRGs invocated by SET PROCEDURE TO...

SET PROCEDURE TO ... in memory is also considered.

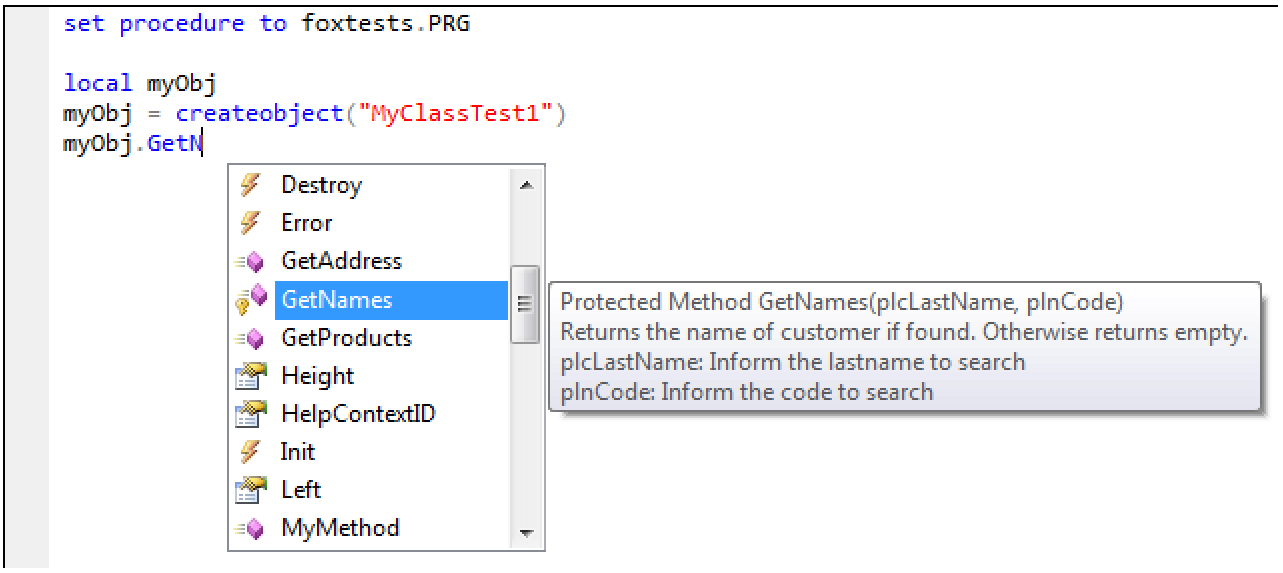


```
set procedure to foxtests.PRG

local myObj
myObj = createobject(MyC
```

MyClassTest1
MyClassTest2

Class MyClassTest1 as baseclass custom
File "C:\PROGRAM FILES\MICROSOFT VISUAL FOXPRO 9\FOXTESTS.PRG"

Image 10.11



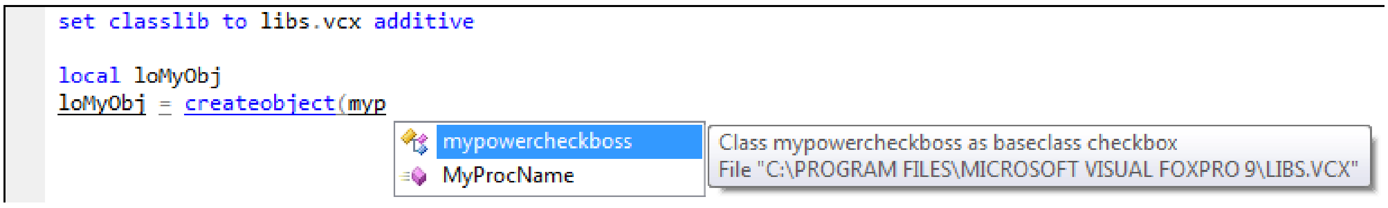
```
set procedure to foxtests.PRG

local myObj
myObj = createobject("MyClassTest1")
myObj.GetN
```

Destroy
Error
GetAddress
GetNames
GetProducts
Height
HelpContextID
Init
Left
MyMethod

Protected Method GetNames(plcLastName, plnCode)
Returns the name of customer if found. Otherwise returns empty.
plcLastName: Inform the lastname to search
plnCode: Inform the code to search

Image 10.12

Classes created in VCX files invocated by SET CLASSLIB TO...

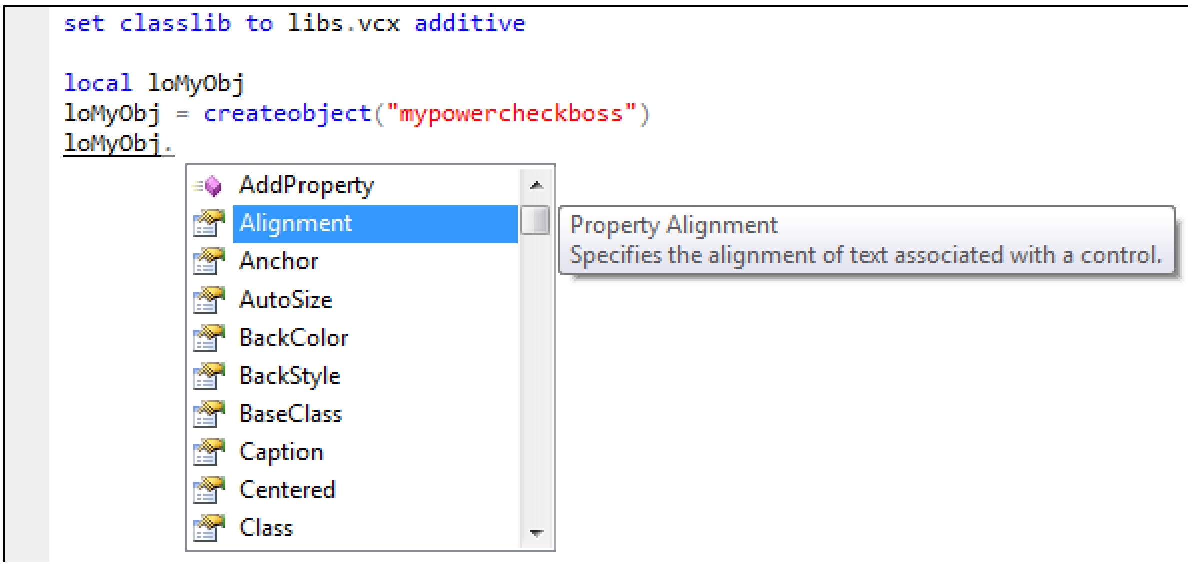


```
set classlib to libs.vcx additive

local loMyObj
loMyObj = createobject(myp
```

mypowercheckboss
MyProcName

Class mypowercheckboss as baseclass checkbox
File "C:\PROGRAM FILES\MICROSOFT VISUAL FOXPRO 9\LIBS.VCX"

Image 10.13



```
set classlib to libs.vcx additive

local loMyObj
loMyObj = createobject("mypowercheckboss")
loMyObj.
```

AddProperty
Alignment
Anchor
AutoSize
BackColor
BackStyle
BaseClass
Caption
Centered
Class

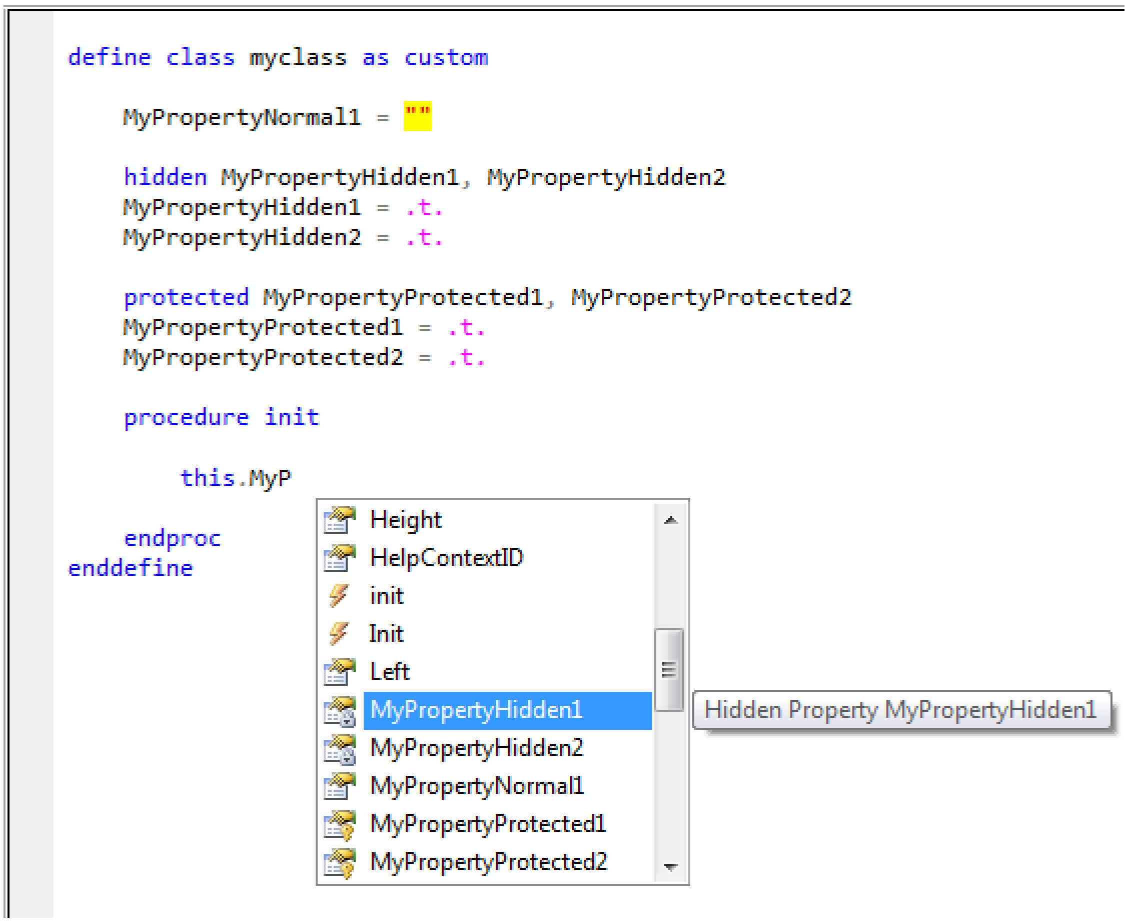
Property Alignment
Specifies the alignment of text associated with a control.

Image 10.14

11- Properties at write-time

Beside the properties of the inherited class, the properties included in the class are also included in the IntelliSense. To access the properties, we use "this." In this case, IntelliSense opens in non-incremental mode.

NOTE: If the class is not inherited from a class or a standard, registered VFP ActiveX, only the properties and methods added to the class will be presented. If the inherited class is in another program, the inherited properties and methods are not included IntelliSense (for now).



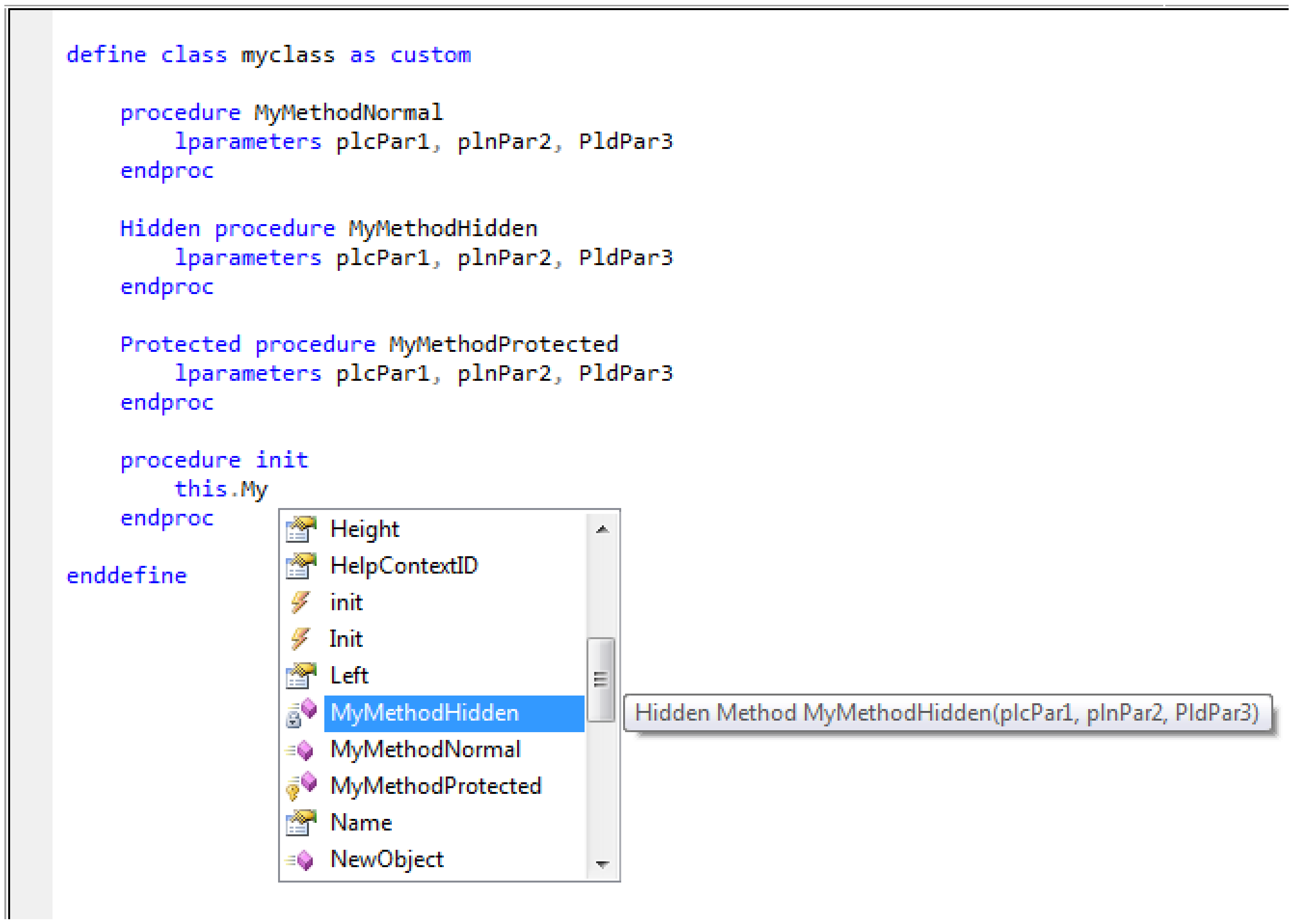
```
define class myclass as custom

    MyPropertyNormal1 = ""

    hidden MyPropertyHidden1, MyPropertyHidden2
    MyPropertyHidden1 = .t.
    MyPropertyHidden2 = .t.

    protected MyPropertyProtected1, MyPropertyProtected2
    MyPropertyProtected1 = .t.
    MyPropertyProtected2 = .t.

    procedure init

        this.MyP

    endproc
enddefine
```

Image 11.10

12- Methods and Events at write-time

Besides the methods and events of the inherited default class, the new methods in the class are also included in the IntelliSense. To access the methods and events, use "this."

NOTE: If the class is not inherited from a class or a standard registered VFP ActiveX, only the methods and events and methods added to the class will be presented. If the inherited class is in another program, the inherited properties and methods are not included in the IntelliSense. (for now).



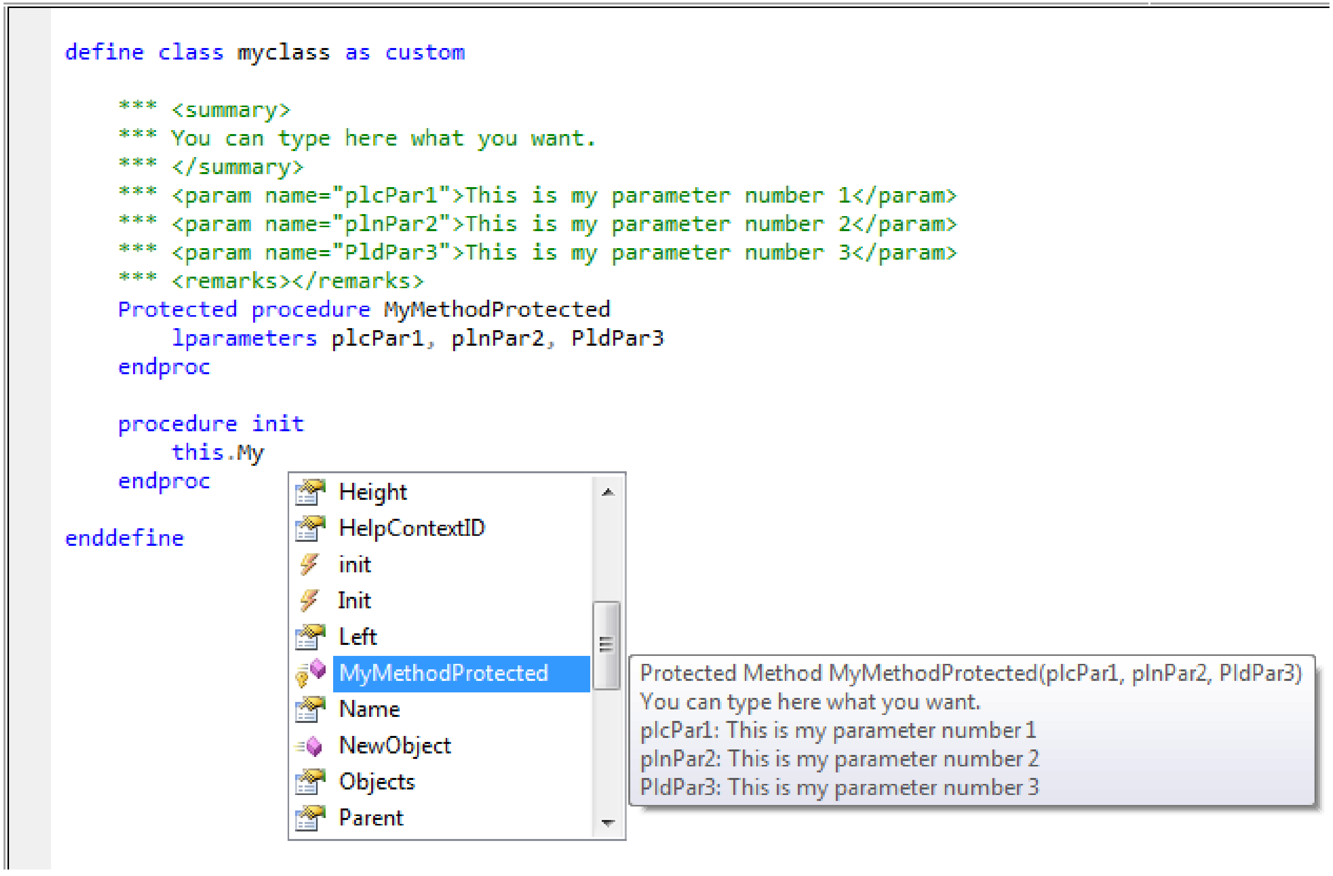
```
define class myclass as custom

    procedure MyMethodNormal
        lparameters plcPar1, plnPar2, PldPar3
    endproc

    Hidden procedure MyMethodHidden
        lparameters plcPar1, plnPar2, PldPar3
    endproc

    Protected procedure MyMethodProtected
        lparameters plcPar1, plnPar2, PldPar3
    endproc

    procedure init
        this.My
    endproc

enddefine
```

Hidden Method MyMethodHidden(plcPar1, plnPar2, PldPar3)

Image 12.10

13- Summary Tooltip for functions, procedures, methods and events.

*Pressing asterisk three times ("***") in the line above where the function, procedure, method or event was created, automatically inserts a block summary. The summary is a way to document the program in a standardized way; in addition, it provides a custom tooltip.*

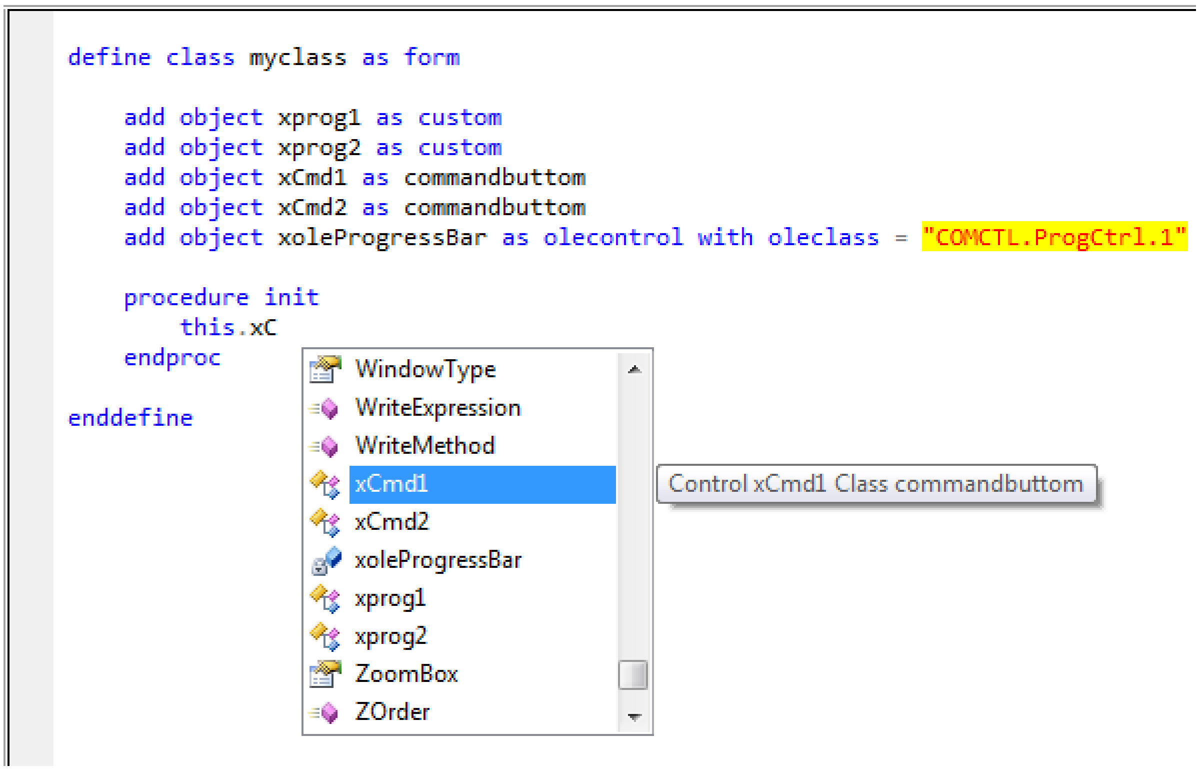


```
define class myclass as custom

    *** <summary>
    *** You can type here what you want.
    *** </summary>
    *** <param name="plcPar1">This is my parameter number 1</param>
    *** <param name="plnPar2">This is my parameter number 2</param>
    *** <param name="PldPar3">This is my parameter number 3</param>
    *** <remarks></remarks>
    Protected procedure MyMethodProtected
        lparameters plcPar1, plnPar2, PldPar3
    endproc

    procedure init
        this.My
    endproc

enddefine
```

```
Protected Method MyMethodProtected(plcPar1, plnPar2, PldPar3)
You can type here what you want.
plcPar1: This is my parameter number 1
plnPar2: This is my parameter number 2
PldPar3: This is my parameter number 3
```

Image 13.10

14- Class objects at write-time

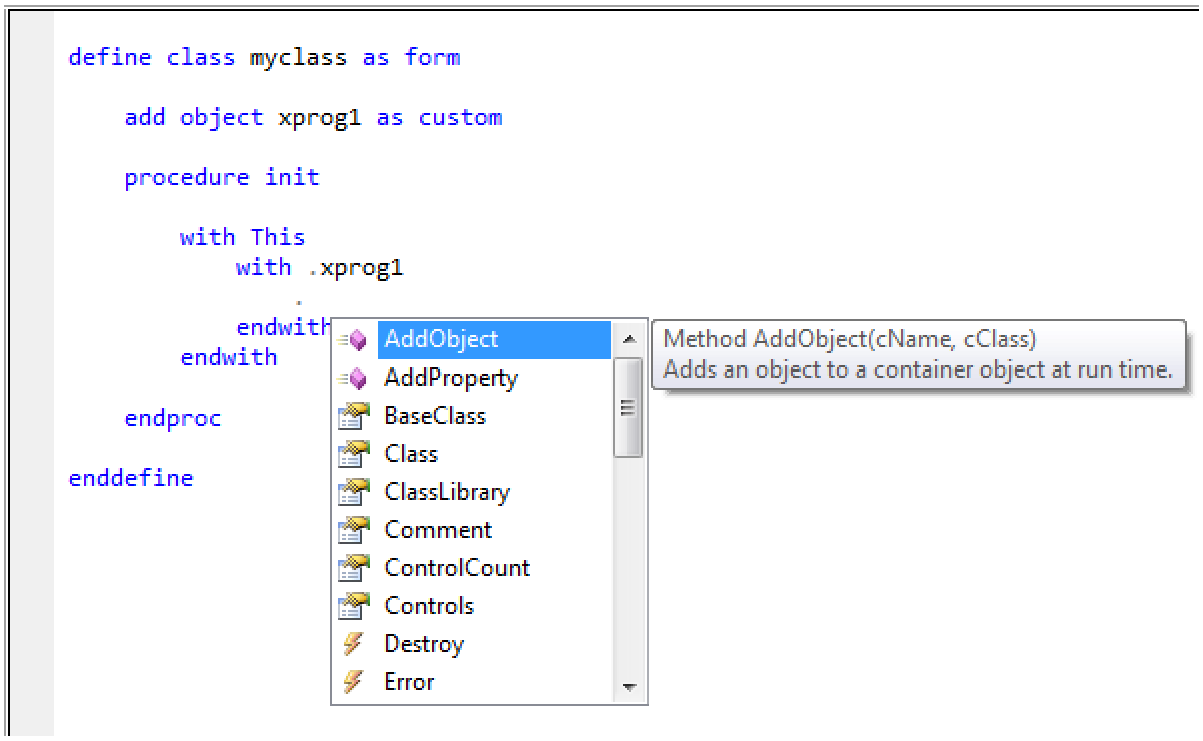


```
define class myclass as form

    add object xprog1 as custom
    add object xprog2 as custom
    add object xCmd1 as commandbuttom
    add object xCmd2 as commandbuttom
    add object xoleProgressBar as olecontrol with oleclass = "COMCTL.ProgCtrl.1"

    procedure init
        this.xC
    endproc

enddefine
```

```
Control xCmd1 Class commandbuttom
```

Image 14.10

15- With ... ENDWITH with nesting infinity for any class or instantiated object at write-time and at run-time.

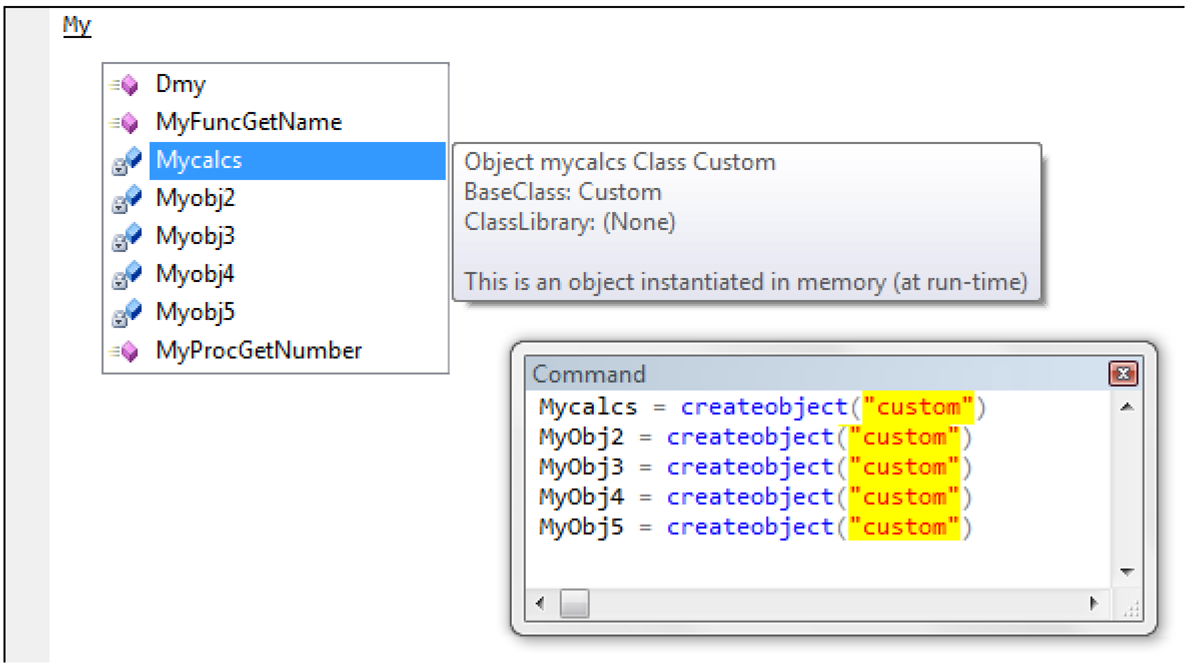


```
define class myclass as form

    add object xprog1 as custom

    procedure init

        with This
            with .xprog1
                .
            endwith
        endwith

    endproc

enddefine
```

Method AddObject(cName, cClass)
Adds an object to a container object at run time.

Image 15.10

16- Objects instantiated in memory



```
My
```

Object mycalcs Class Custom
BaseClass: Custom
ClassLibrary: (None)

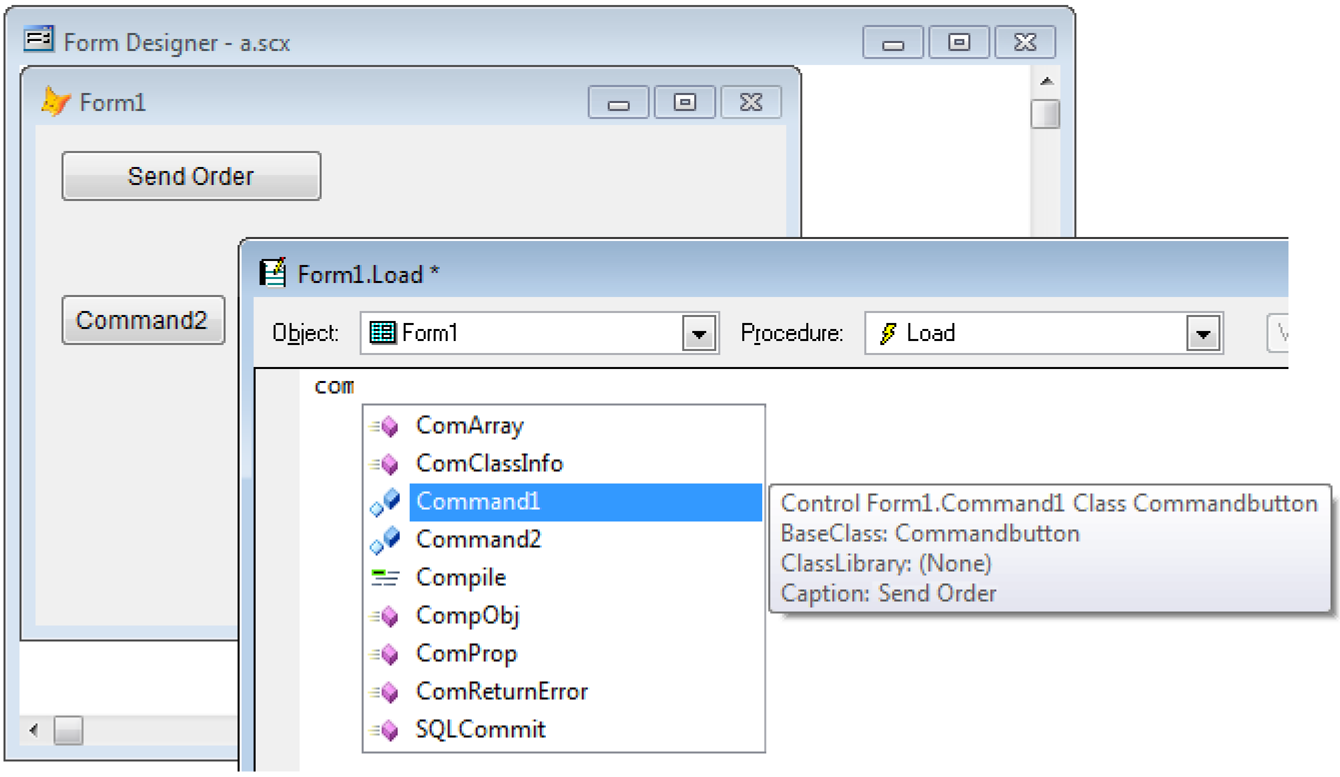
This is an object instantiated in memory (at run-time)

```
Mycalcs = createobject("custom")
MyObj2 = createobject("custom")
MyObj3 = createobject("custom")
MyObj4 = createobject("custom")
MyObj5 = createobject("custom")
```

Image 16.10

17- Incremental Shortcut to controls on the form or class designer.

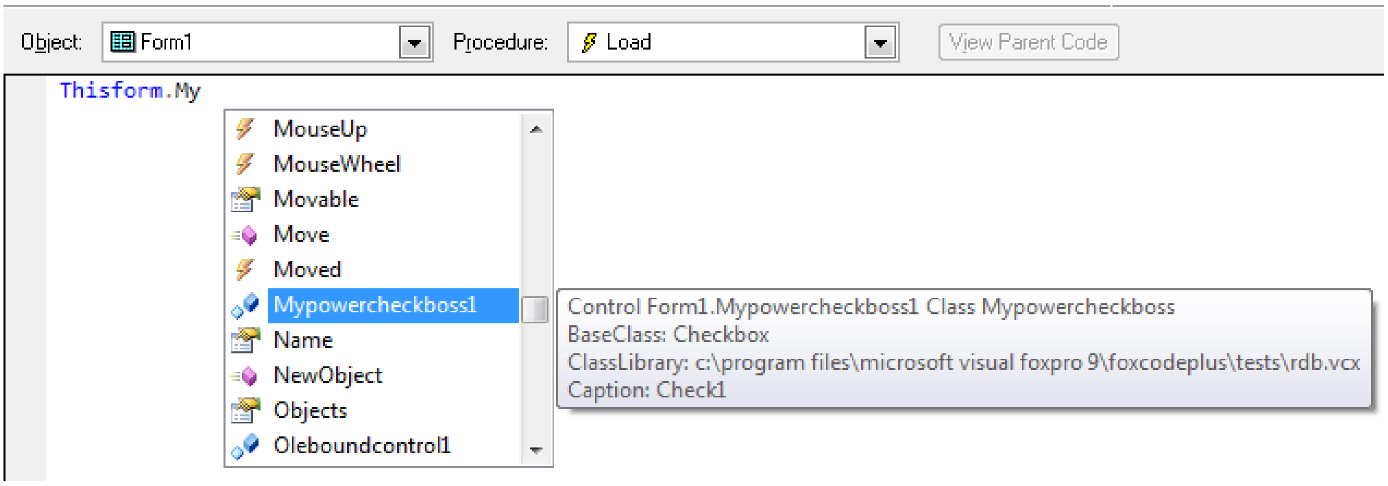
Type just the control name; you don't need to type "this." or "thisform."
When you select the item in IntelliSense, "this." or "thisform." is automatically inserted.
For easy identification, the caption of the control is displayed on the IntelliSense tooltip, such as the propriety's BaseClass and ClassLibrary.



Control Form1.Command1 Class Commandbutton
BaseClass: Commandbutton
ClassLibrary: (None)
Caption: Send Order

Image 17.10

*In the tooltip the name of control is preceded by the parent name object. (e.g. **Form1**.Mypowercheckboss1)*



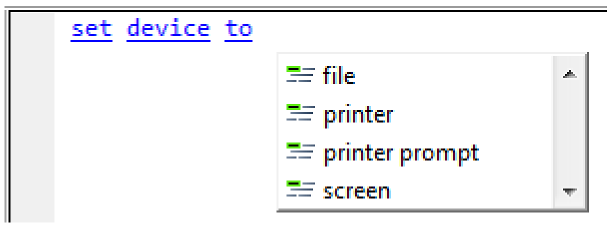
Control Form1.Mypowercheckboss1 Class Mypowercheckboss
BaseClass: Checkbox
ClassLibrary: c:\program files\microsoft visual foxpro 9\foxcodeplus\tests\rdb.vcx
Caption: Check1

Image 17.11

18- Replacement of the native IntelliSense in form and class designer.

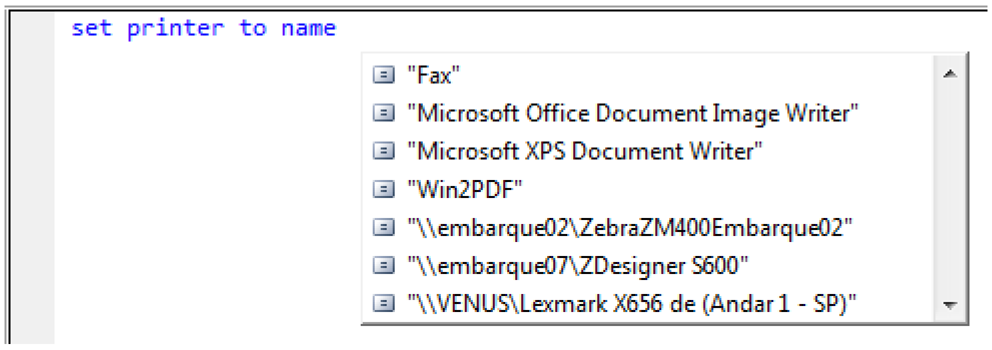
*If the "**Manage Controls at design-time**" option is marked in the "IntelliSense Manager", FoxcodePlus will replace the native IntelliSense in the Form and Class Designer. When replaced, FoxcodePlus can directly interact with VFP and improve the information in tooltip and other features as well.*

19- New IntelliSense for some commands



```
set device to
```

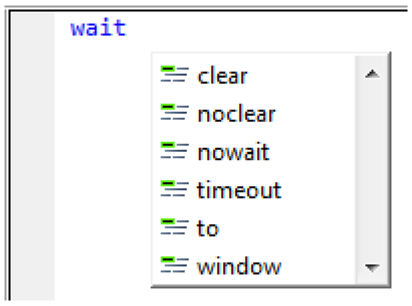
Image 19.10



```
set printer to name
```

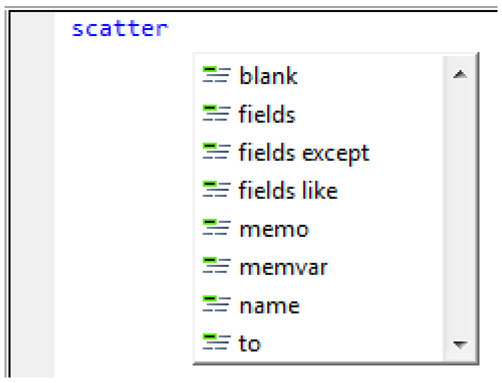
"Fax"
"Microsoft Office Document Image Writer"
"Microsoft XPS Document Writer"
"Win2PDF"
"\\embarque02\ZebraZM400Embarque02"
"\\embarque07\ZDesigner S600"
"\\VENUS\Lexmark X656 de (Andar 1 - SP)"

Image 19.11



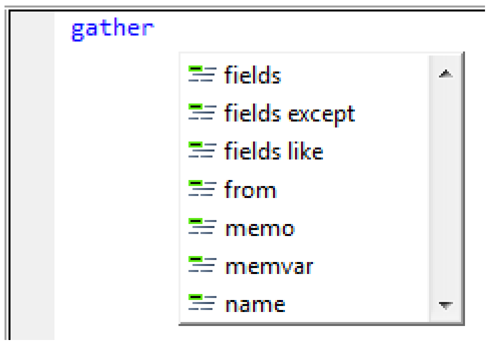
```
wait
```

Image 19.12



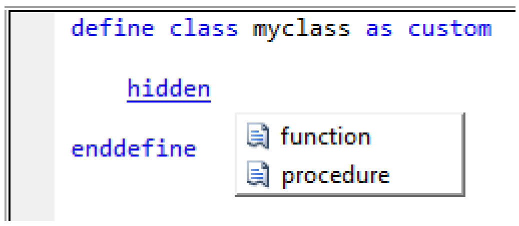
```
scatter
```

Image 19.13



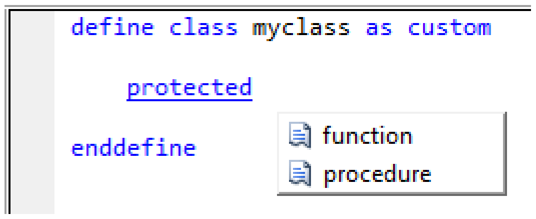
```
gather
```

Image 19.14



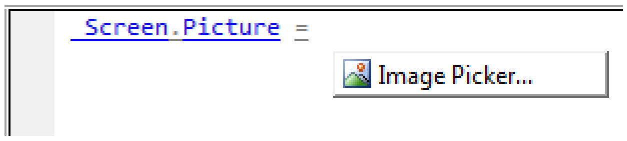
```
define class myclass as custom
    hidden
enddefine
```

Image 19.15



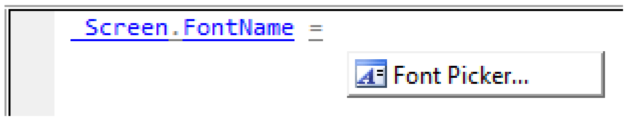
```
define class myclass as custom
    protected
enddefine
```

Image 19.16



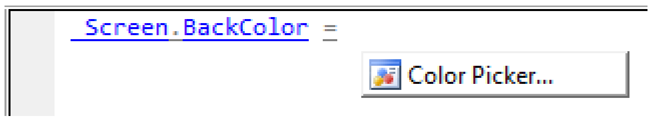
```
_Screen.Picture =
```

Image 19.17



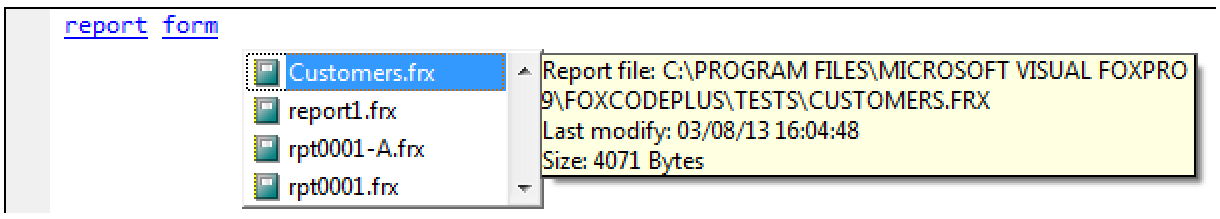
```
_Screen.FontName =
```

Image 19.18



```
_Screen.BackColor =
```

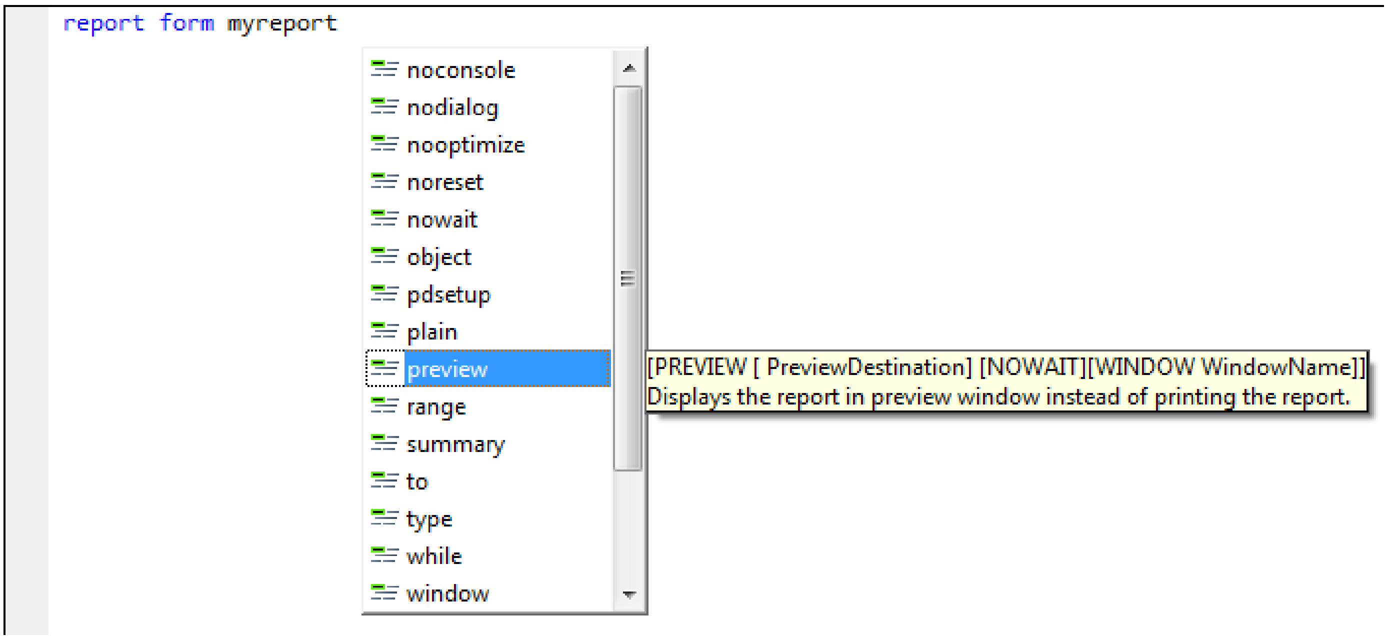
Image 19.19



```
report form
```

Report file: C:\PROGRAM FILES\MICROSOFT VISUAL FOXPRO 9\FOXCODEPLUS\TESTS\CUSTOMERS.FRX
Last modify: 03/08/13 16:04:48
Size: 4071 Bytes

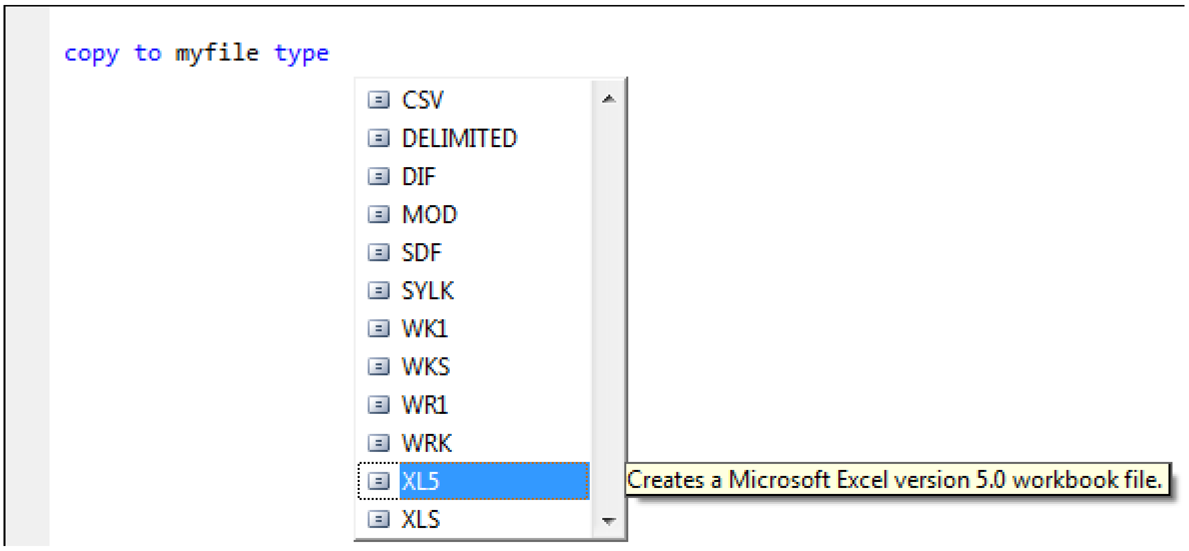
Image 19.20



```
report form myreport
```

[PREVIEW [PreviewDestination] [NOWAIT][WINDOW WindowName]]
Displays the report in preview window instead of printing the report.

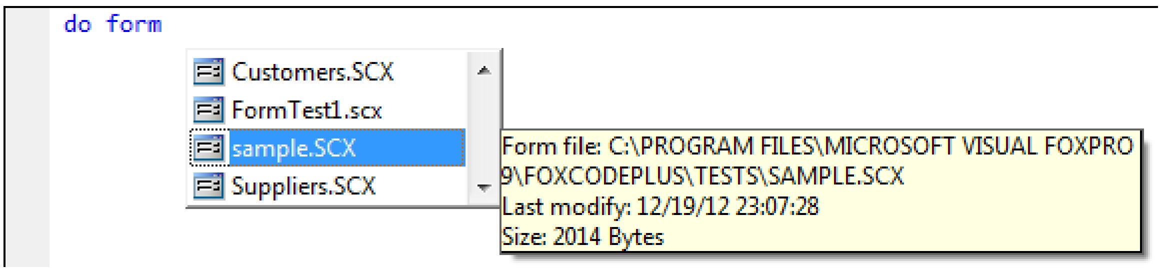
Image 19.21



```
copy to myfile type
```

Creates a Microsoft Excel version 5.0 workbook file.

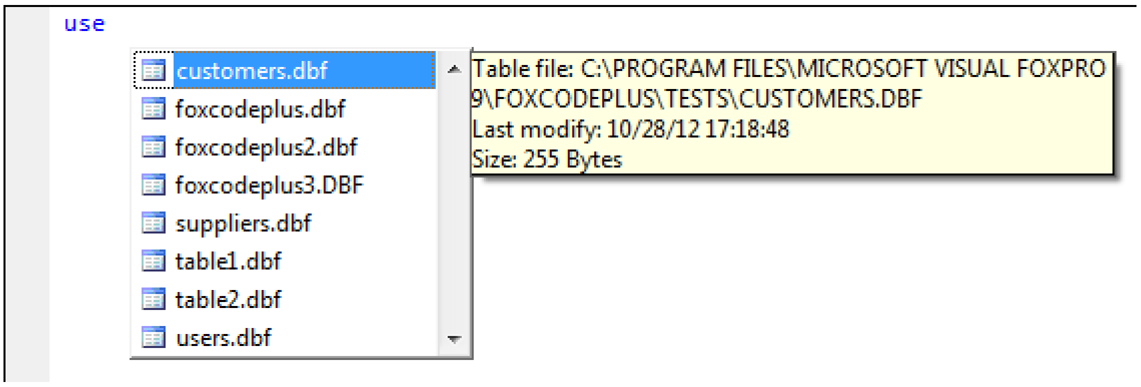
Image 19.22



```
do form
```

Form file: C:\PROGRAM FILES\MICROSOFT VISUAL FOXPRO 9\FOXCODEPLUS\TESTS\SAMPLE.SCX
Last modify: 12/19/12 23:07:28
Size: 2014 Bytes

Image 19.23

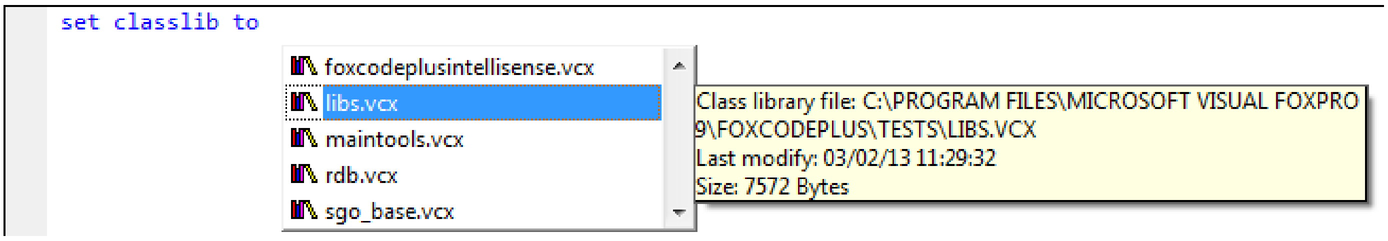


```
use
```

customers.dbf
foxcodeplus.dbf
foxcodeplus2.dbf
foxcodeplus3.DBF
suppliers.dbf
table1.dbf
table2.dbf
users.dbf

Table file: C:\PROGRAM FILES\MICROSOFT VISUAL FOXPRO 9\FOXCODEPLUS\TESTS\CUSTOMERS.DBF
Last modify: 10/28/12 17:18:48
Size: 255 Bytes

Image 19.24

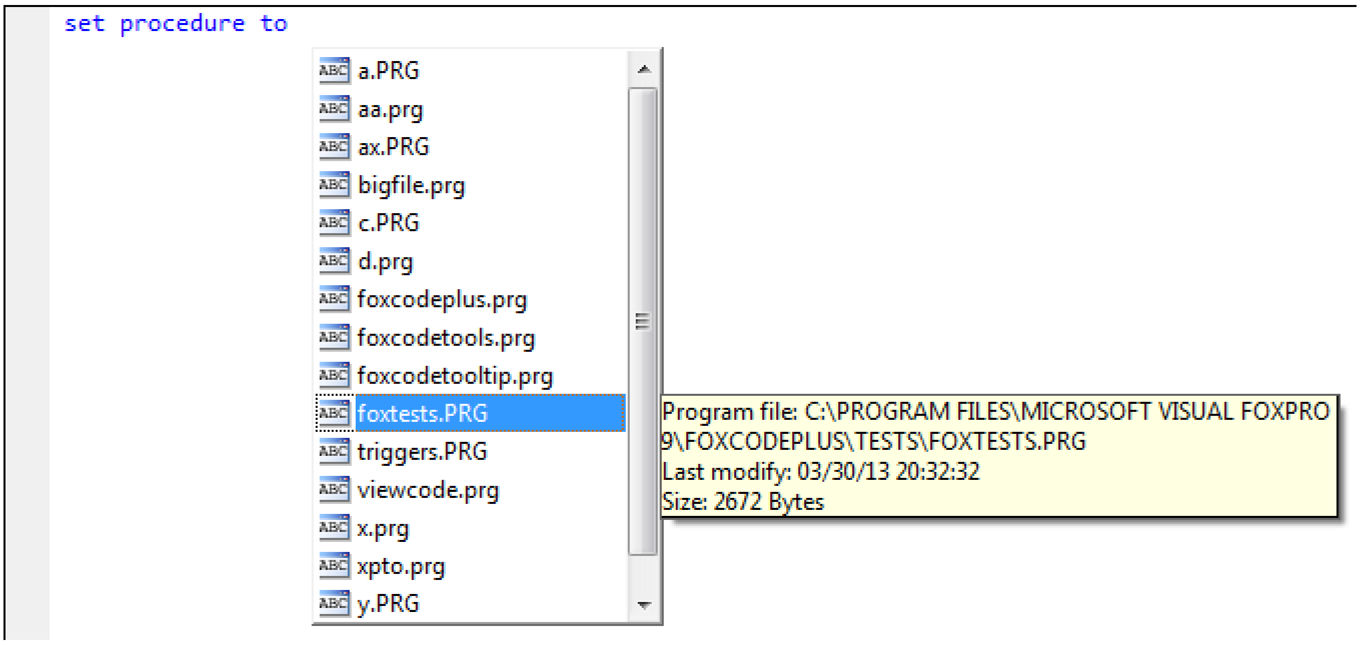


```
set classlib to
```

foxcodeplusintellisense.vcx
libs.vcx
maintools.vcx
rdb.vcx
sgo_base.vcx

Class library file: C:\PROGRAM FILES\MICROSOFT VISUAL FOXPRO 9\FOXCODEPLUS\TESTS\LIBS.VCX
Last modify: 03/02/13 11:29:32
Size: 7572 Bytes

Image 19.25

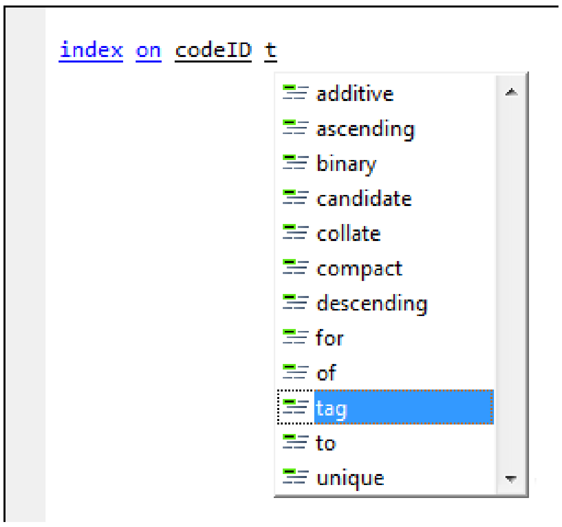


```
set procedure to
```

a.PRG
aa.prg
ax.PRG
bigfile.prg
c.PRG
d.prg
foxcodeplus.prg
foxcodetools.prg
foxcodetooltip.prg
foxtests.PRG
triggers.PRG
viewcode.prg
x.prg
xpto.prg
y.PRG

Program file: C:\PROGRAM FILES\MICROSOFT VISUAL FOXPRO 9\FOXCODEPLUS\TESTS\FOXTESTS.PRG
Last modify: 03/30/13 20:32:32
Size: 2672 Bytes

Image 19.26



```
index on codeID t
```

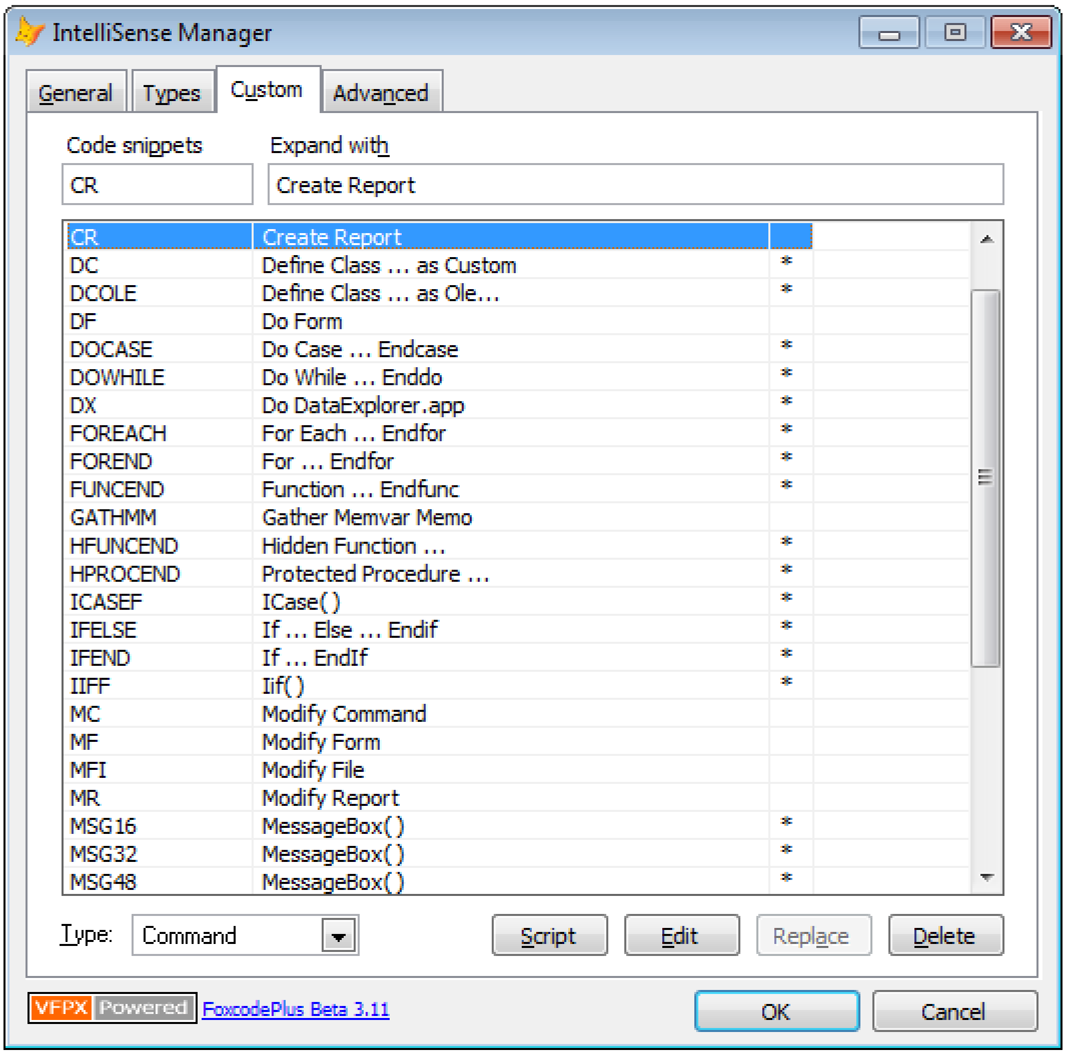
additive
ascending
binary
candidate
collate
compact
descending
for
of
tag
to
unique

Image 19.27

20- Code snippets

*As in Visual Studio, the native and custom Code Snippets are included in the IntelliSense. For that, the "**Show code snippets**" option should be marked in the "IntelliSense Manager".*

The Code Snippet list can be consulted in the "IntelliSense Manager":



Code snippets	Expand with	
CR	Create Report	
DC	Define Class ... as Custom	*
DCOLE	Define Class ... as Ole...	*
DF	Do Form	
DOCASE	Do Case ... Endcase	*
DOWHILE	Do While ... Enddo	*
DX	Do DataExplorer.app	*
FOREACH	For Each ... Endfor	*
FOREND	For ... Endfor	*
FUNCEND	Function ... Endfunc	*
GATHMM	Gather Memvar Memo	
HFUNCEND	Hidden Function ...	*
HPROCEND	Protected Procedure ...	*
ICASEF	ICase( )	*
IFELSE	If ... Else ... Endif	*
IFEND	If ... EndIf	*
IIFF	Iif( )	*
MC	Modify Command	
MF	Modify Form	
MFI	Modify File	
MR	Modify Report	
MSG16	MessageBox( )	*
MSG32	MessageBox( )	*
MSG48	MessageBox( )	*

Code snippets: CR — Expand with: Create Report — Type: Command

Image 20.10

If you select an item in the IntelliSense by pressing the SPACE key or type in the code snippet code and pressing SPACE beside the code, a code block is inserted to reduce typing.

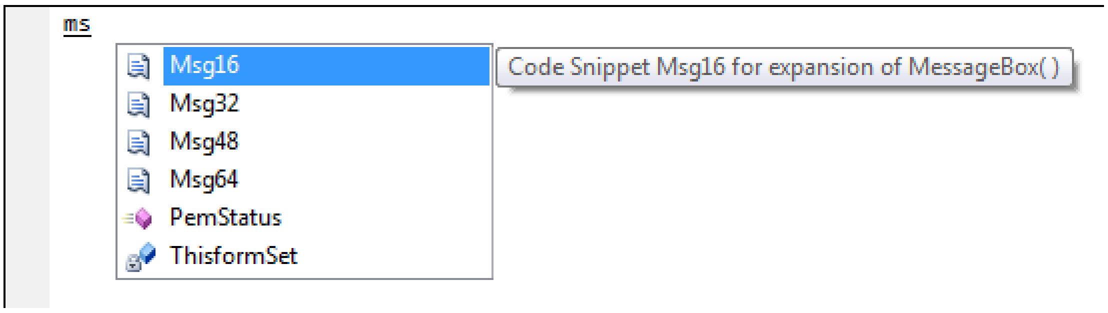
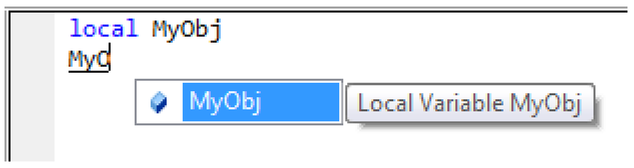


Image 20.11

```
messagebox("mymessage",16,"error")
```

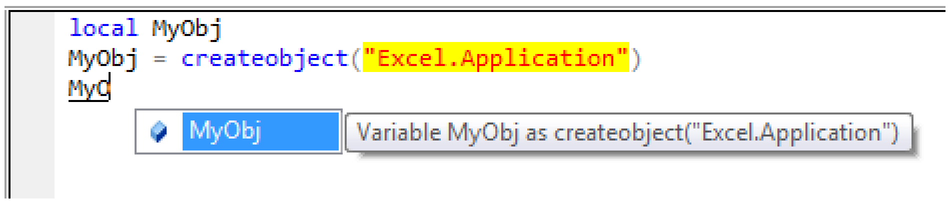
21- IntelliSense at write-time for the objects created with the CreateObject(), CreateObjectEx() and NewObject() functions



```
local MyObj
MyO
```

MyObj | Local Variable MyObj

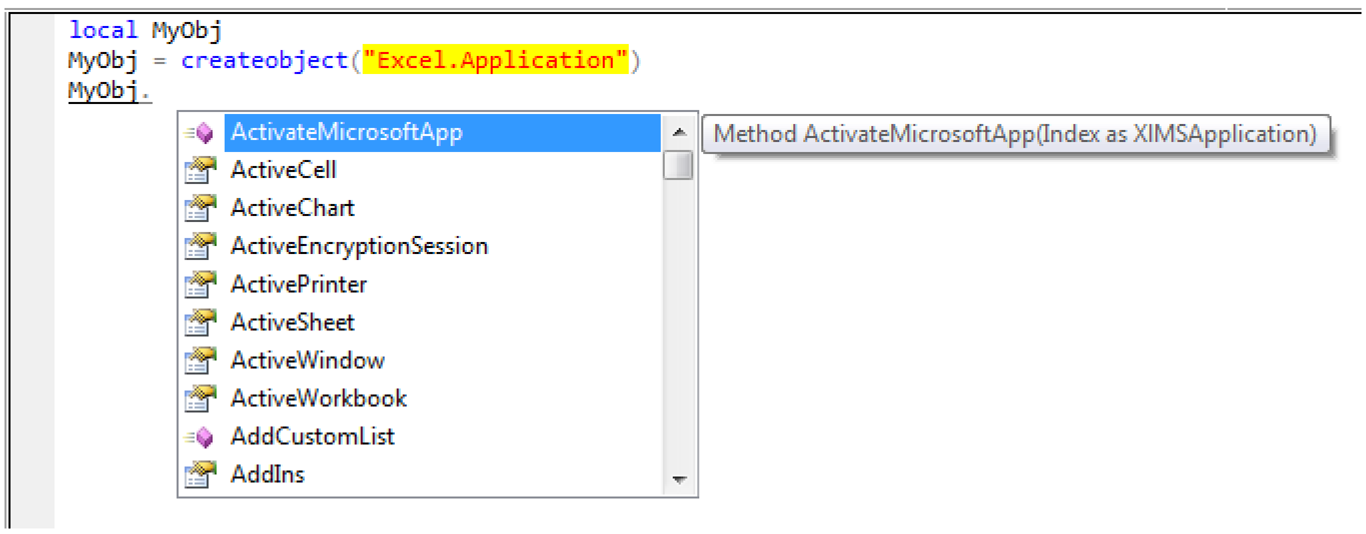
Image 21.10



```
local MyObj
MyObj = createobject("Excel.Application")
MyO
```

MyObj | Variable MyObj as createobject("Excel.Application")

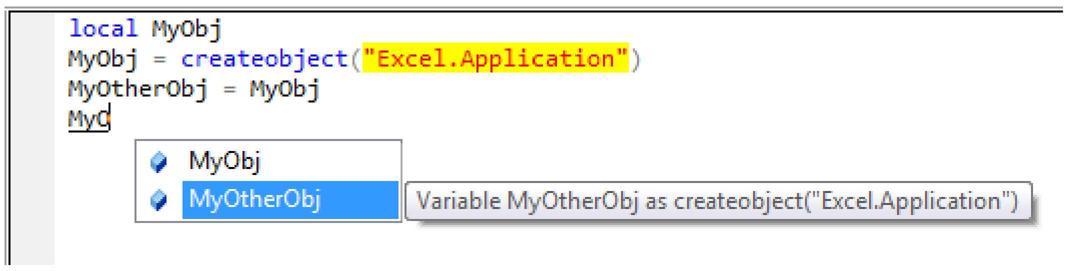
Image 21.11



```
local MyObj
MyObj = createobject("Excel.Application")
MyObj.
```

ActivateMicrosoftApp | Method ActivateMicrosoftApp(Index as XIMSApplication)
ActiveCell
ActiveChart
ActiveEncryptionSession
ActivePrinter
ActiveSheet
ActiveWindow
ActiveWorkbook
AddCustomList
AddIns

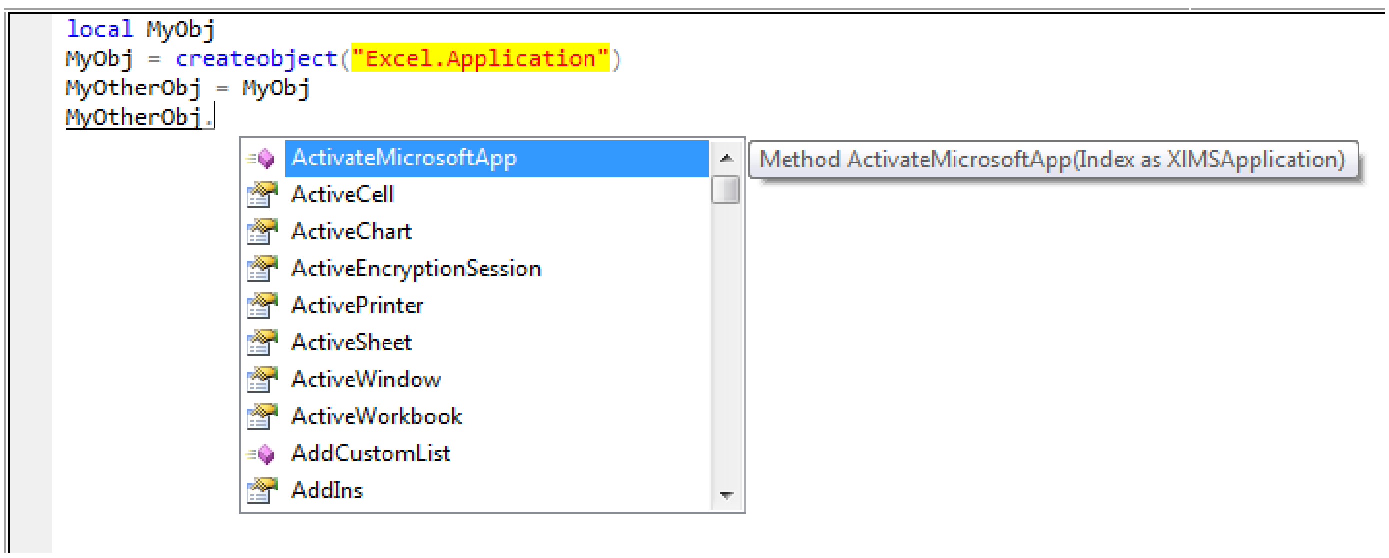
Image 21.12



```
local MyObj
MyObj = createobject("Excel.Application")
MyOtherObj = MyObj
MyO
```

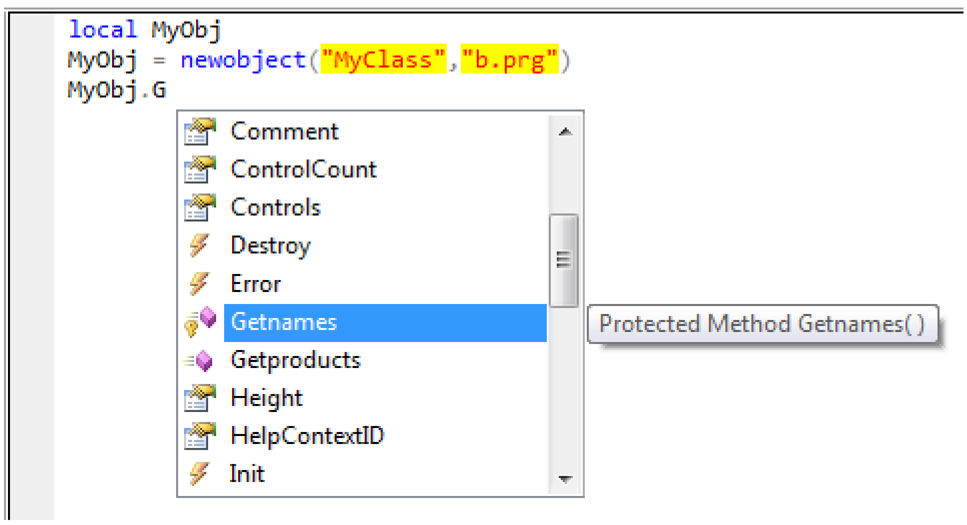
MyObj
MyOtherObj | Variable MyOtherObj as createobject("Excel.Application")

Image 21.13



```
local MyObj
MyObj = createobject("Excel.Application")
MyOtherObj = MyObj
MyOtherObj.
```

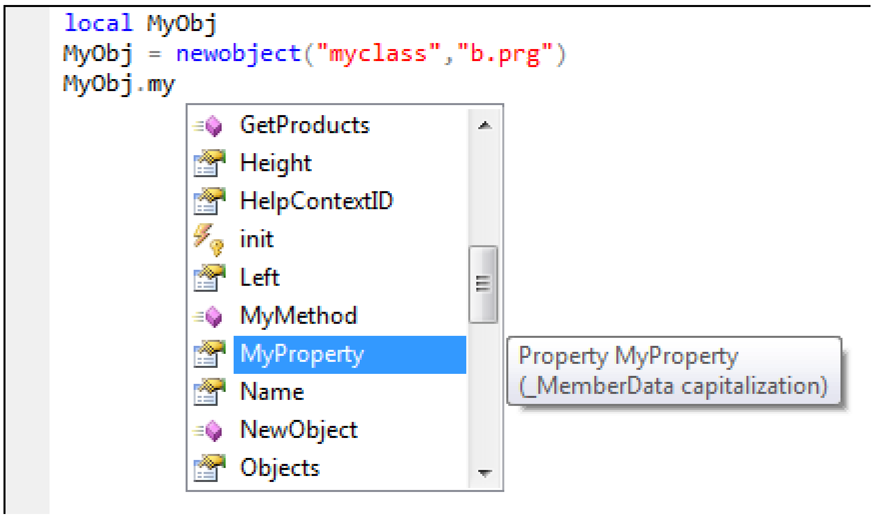
Image 21.14



```
local MyObj
MyObj = newobject("MyClass","b.prg")
MyObj.G
```

Image 21.15

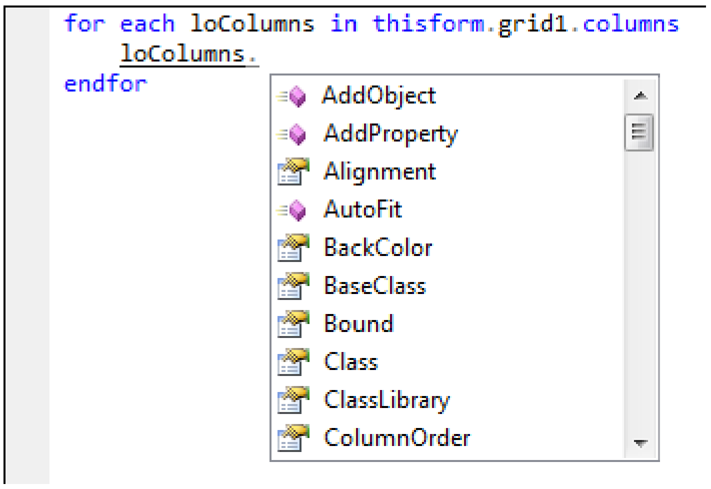
Support for _MemberData property indicating that the property had capitalization.



```
local MyObj
MyObj = newobject("myclass","b.prg")
MyObj.my
```

Image 21.16

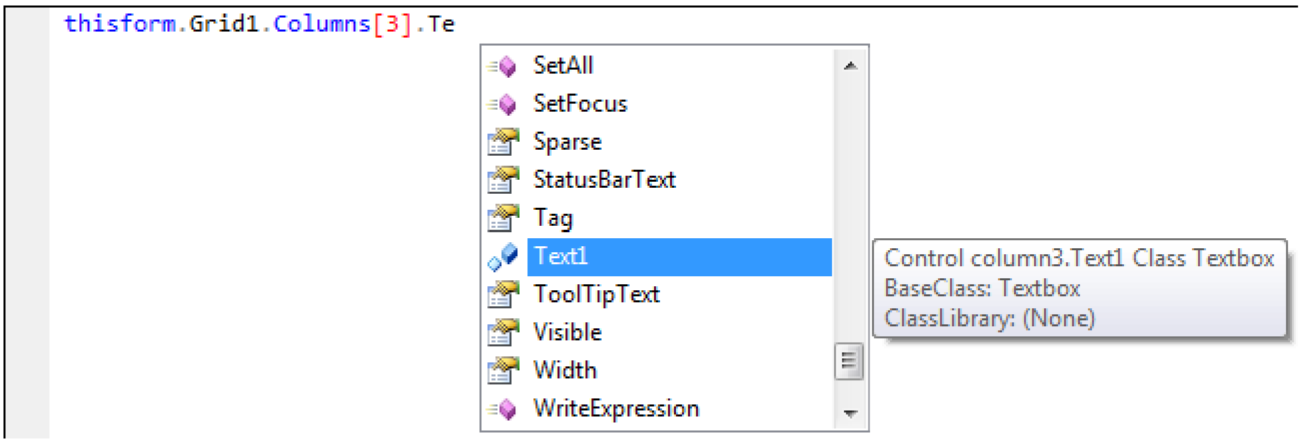
22- FOR EACH for collection objects at run-time and designer-time.



```
for each loColumns in thisform.grid1.columns
    loColumns.
endfor
```

Image 22.10

23- IntelliSense for collection objects at run-time and designer-time.

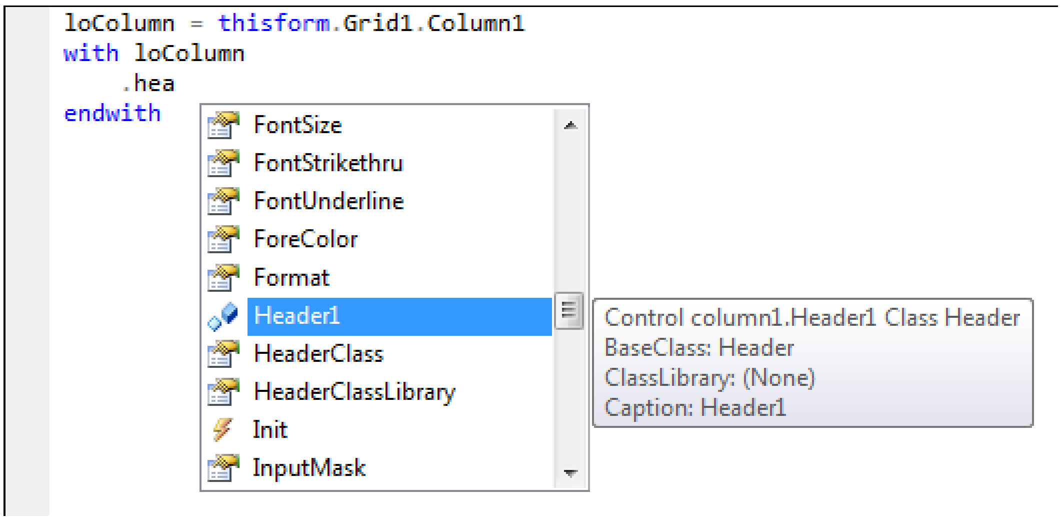


```
thisform.Grid1.Columns[3].Te
```

Control column3.Text1 Class Textbox
BaseClass: Textbox
ClassLibrary: (None)

Image 23.10

24- Referencing an object at run-time and designer-time to a variable at write-time

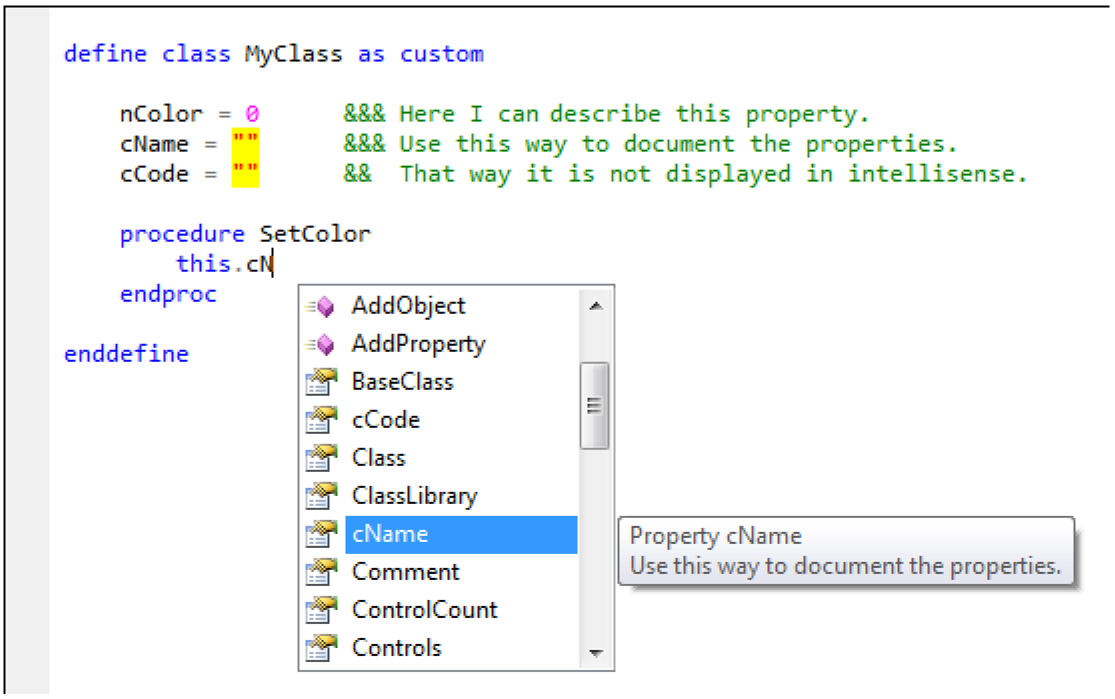


```
loColumn = thisform.Grid1.Column1
with loColumn
    .hea
endwith
```

Control column1.Header1 Class Header
BaseClass: Header
ClassLibrary: (None)
Caption: Header1

Image 24.10

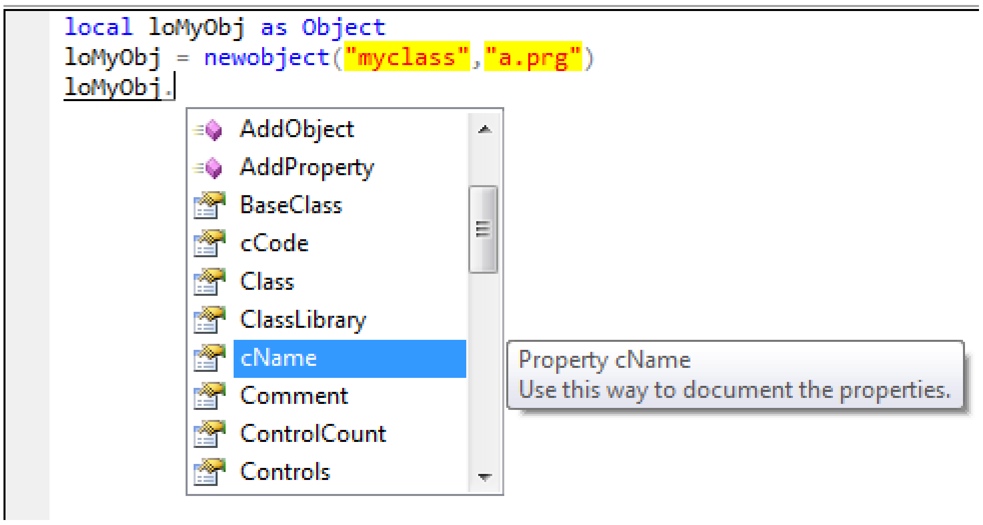
25- Documenting properties with custom tooltip



```
define class MyClass as custom

    nColor = 0       &&& Here I can describe this property.
    cName  = ""      &&& Use this way to document the properties.
    cCode  = ""      &&  That way it is not displayed in intellisense.

    procedure SetColor
        this.cN
    endproc

enddefine
```

Property cName
Use this way to document the properties.

Image 25.10



```
local loMyObj as Object
loMyObj = newobject("myclass","a.prg")
loMyObj.
```

Property cName
Use this way to document the properties.

Image 25.11

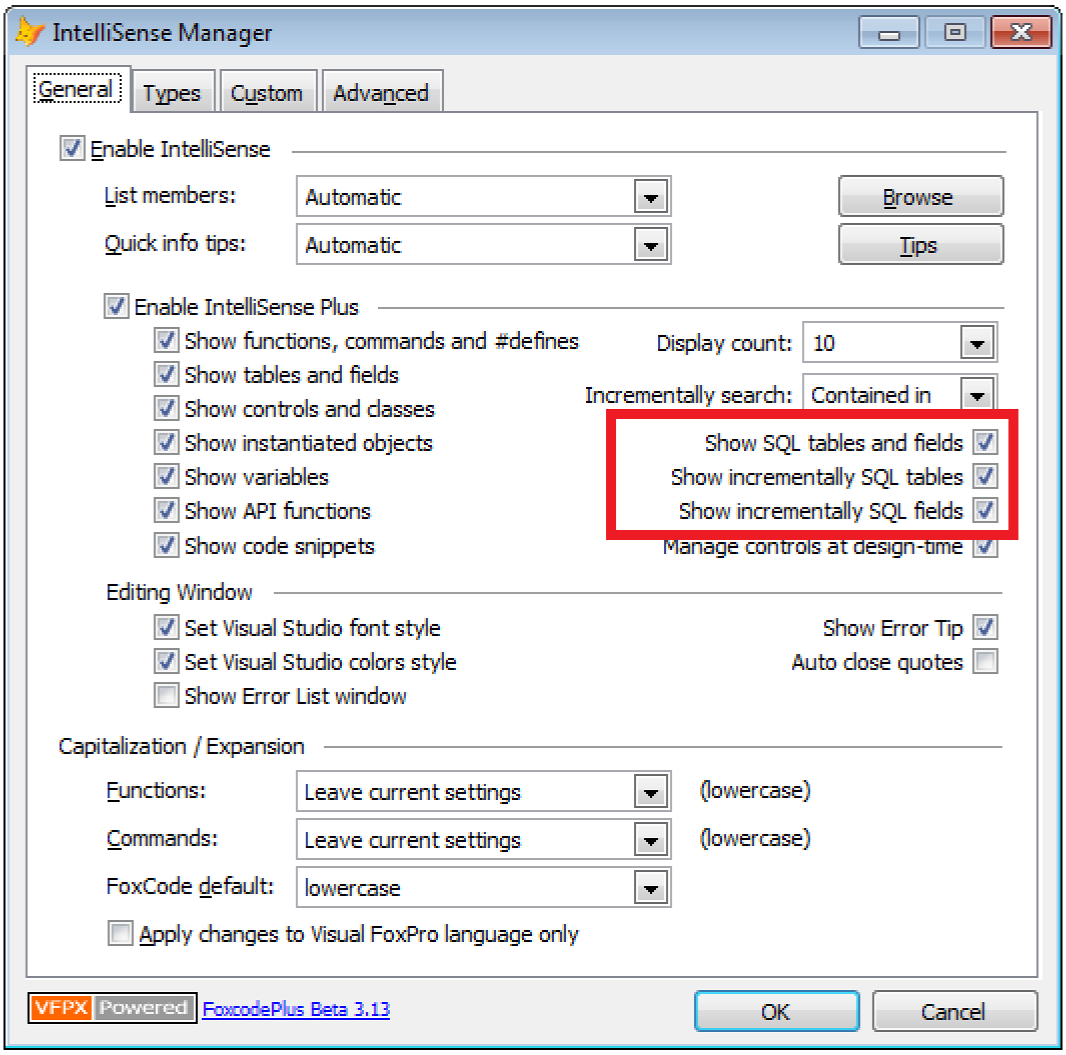
26- Help pressing F1

Pressing F1 to open IntelliSense, VFP will open the help positioned to the command, function, method or event positioned.

27- SELECT, INSERT, UPDATE and DELETE for database connected. (Tested with MS Sql Server)

This feature provides the IntelliSence bringing information from a database connect. It's possible to work disconnected, however the IntelliSence display only the tables (no fields) included in the current sql-command. NOTE: It's necessary you put your SQL-Command in a TEXT...ENDTEXT block.

You can choose the options showed below in order to select the best way that you want to use it.



IntelliSense Manager

General | Types | Custom | Advanced

[✓] Enable IntelliSense

List members: Automatic

Quick info tips: Automatic

Browse

Tips

[✓] Enable IntelliSense Plus

[✓] Show functions, commands and #defines

[✓] Show tables and fields

[✓] Show controls and classes

[✓] Show instantiated objects

[✓] Show variables

[✓] Show API functions

[✓] Show code snippets

Display count: 10

Incrementally search: Contained in

Show SQL tables and fields [✓]

Show incrementally SQL tables [✓]

Show incrementally SQL fields [✓]

Manage controls at design-time [✓]

Editing Window

[✓] Set Visual Studio font style

[✓] Set Visual Studio colors style

[ ] Show Error List window

Show Error Tip [✓]

Auto close quotes [ ]

Capitalization / Expansion

Functions: Leave current settings (lowercase)

Commands: Leave current settings (lowercase)

FoxCode default: lowercase

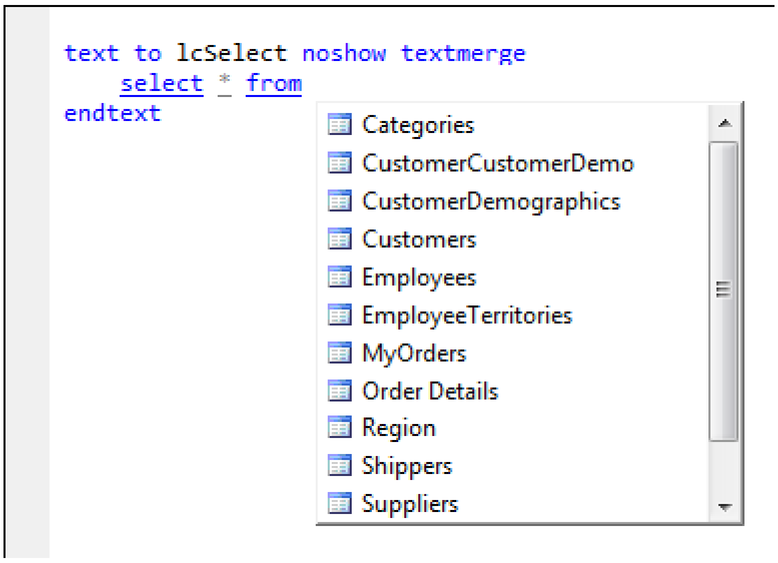
[ ] Apply changes to Visual FoxPro language only

VFPX Powered FoxcodePlus Beta 3.13

OK Cancel

Image 27.10

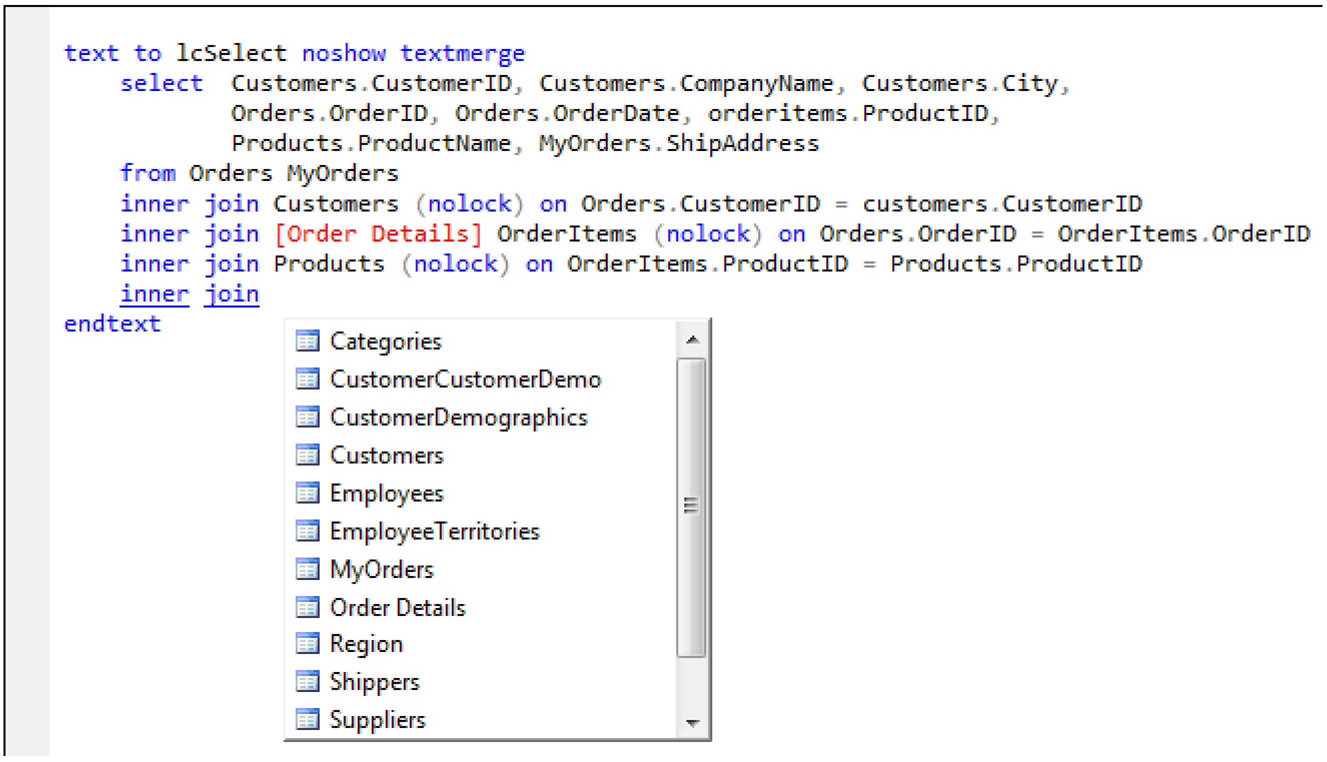
SELECT – After the clause "FROM" and "JOIN" a list of tables and tables' alias from the current database is shown (non-incremental mode)



```
text to lcSelect noshow textmerge
    select * from
endtext
```

Categories
CustomerCustomerDemo
CustomerDemographics
Customers
Employees
EmployeeTerritories
MyOrders
Order Details
Region
Shippers
Suppliers

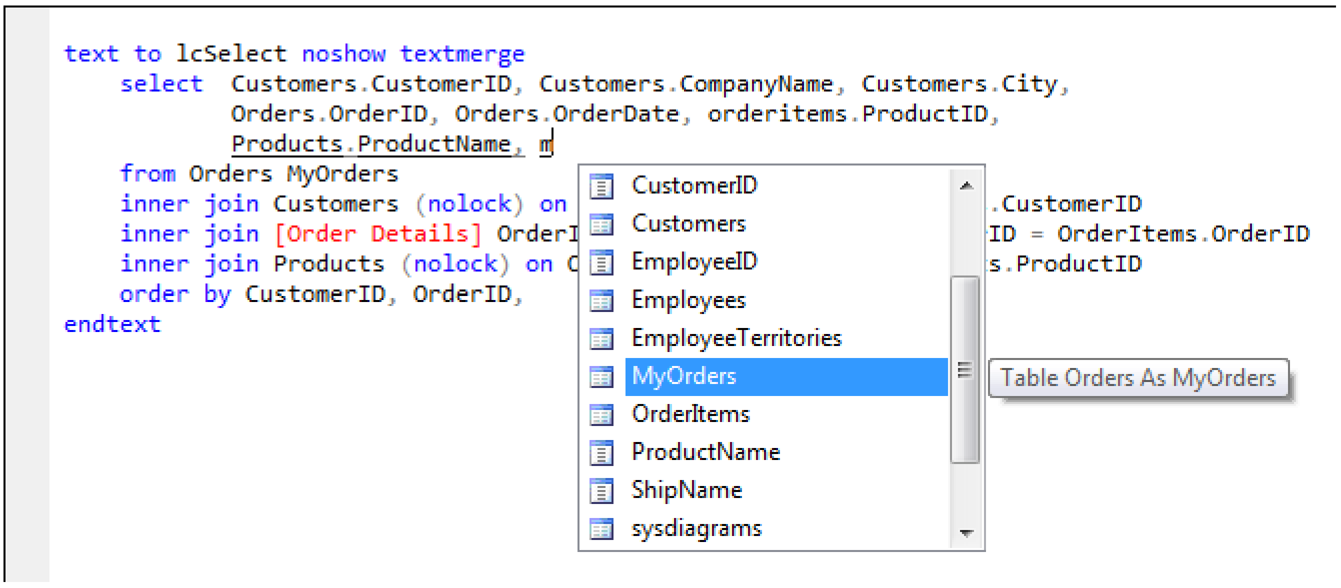
Image 27.11



```
text to lcSelect noshow textmerge
    select  Customers.CustomerID, Customers.CompanyName, Customers.City,
            Orders.OrderID, Orders.OrderDate, orderitems.ProductID,
            Products.ProductName, MyOrders.ShipAddress
    from Orders MyOrders
    inner join Customers (nolock) on Orders.CustomerID = customers.CustomerID
    inner join [Order Details] OrderItems (nolock) on Orders.OrderID = OrderItems.OrderID
    inner join Products (nolock) on OrderItems.ProductID = Products.ProductID
    inner join
endtext
```

Image 27.12

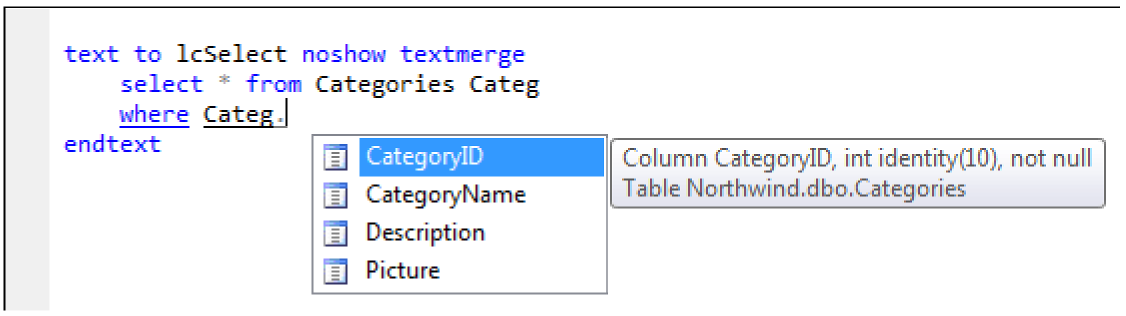
*SELECT – As in **SQL Server Management Studio,** tables and tables' alias are shown in the IntelliSense in incremental mode. All Fields that belong for each table included in SELECT are show as well.*



```
text to lcSelect noshow textmerge
    select  Customers.CustomerID, Customers.CompanyName, Customers.City,
            Orders.OrderID, Orders.OrderDate, orderitems.ProductID,
            Products.ProductName, m
    from Orders MyOrders
    inner join Customers (nolock) on                      .CustomerID
    inner join [Order Details] OrderI                     ID = OrderItems.OrderID
    inner join Products (nolock) on O                     s.ProductID
    order by CustomerID, OrderID,
endtext
```

Image 27.13

INSERT – IntelliSense for fields through the tables' alias

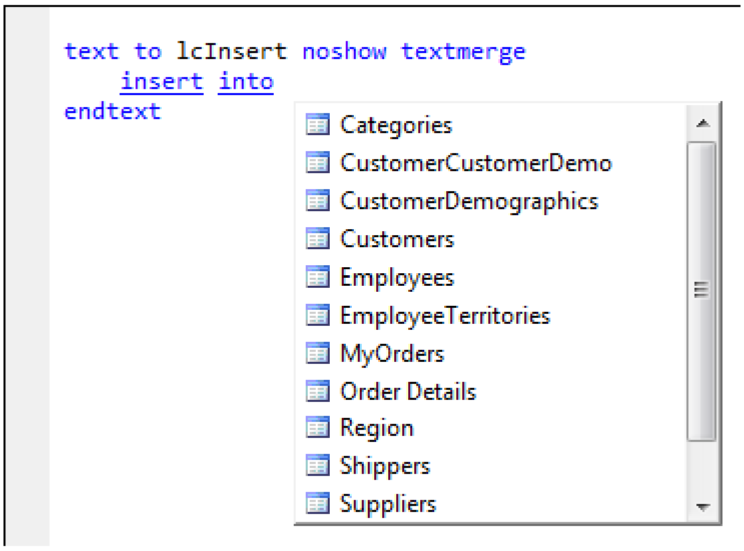


```
text to lcSelect noshow textmerge
    select * from Categories Categ
    where Categ.
endtext
```

CategoryID
CategoryName
Description
Picture
Column CategoryID, int identity(10), not null
Table Northwind.dbo.Categories

Image 27.14

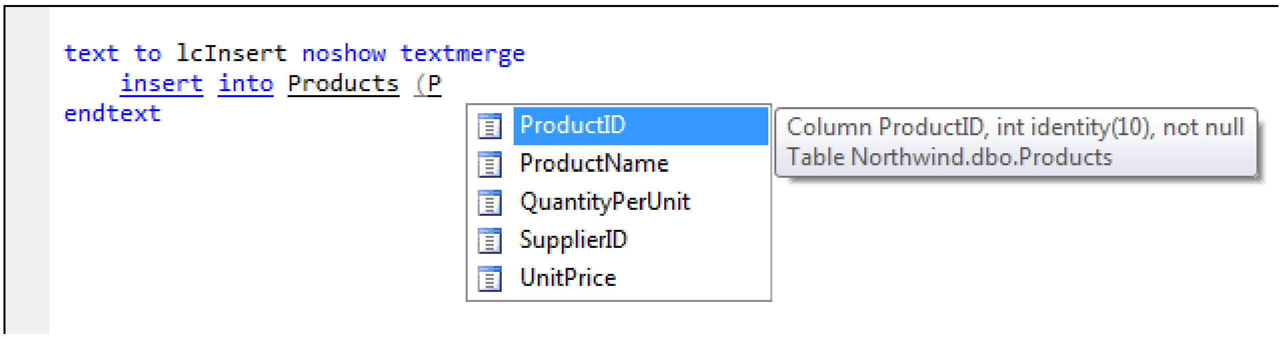
INSERT – After the clause “INTO” a list of tables from the current database is shown (non-incremental mode)



```
text to lcInsert noshow textmerge
    insert into
endtext
```

Image 27.15

In incremental mode the fields that belong to the table defined after the clause “INTO” are shown.

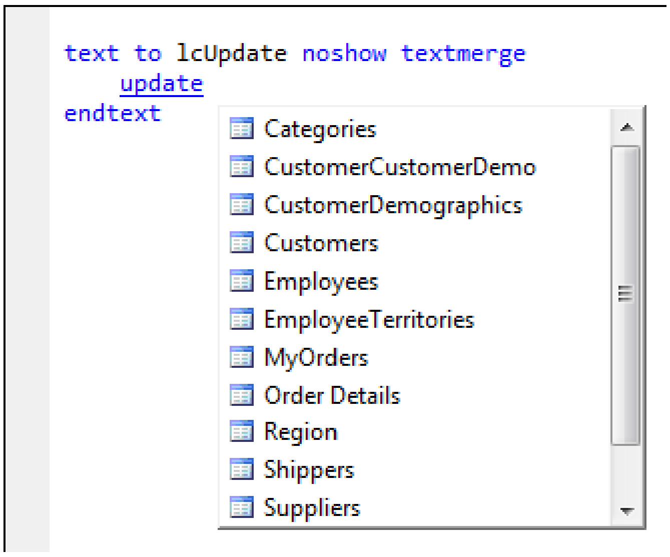


```
text to lcInsert noshow textmerge
    insert into Products (P
endtext
```

Column ProductID, int identity(10), not null
Table Northwind.dbo.Products

Image 27.16

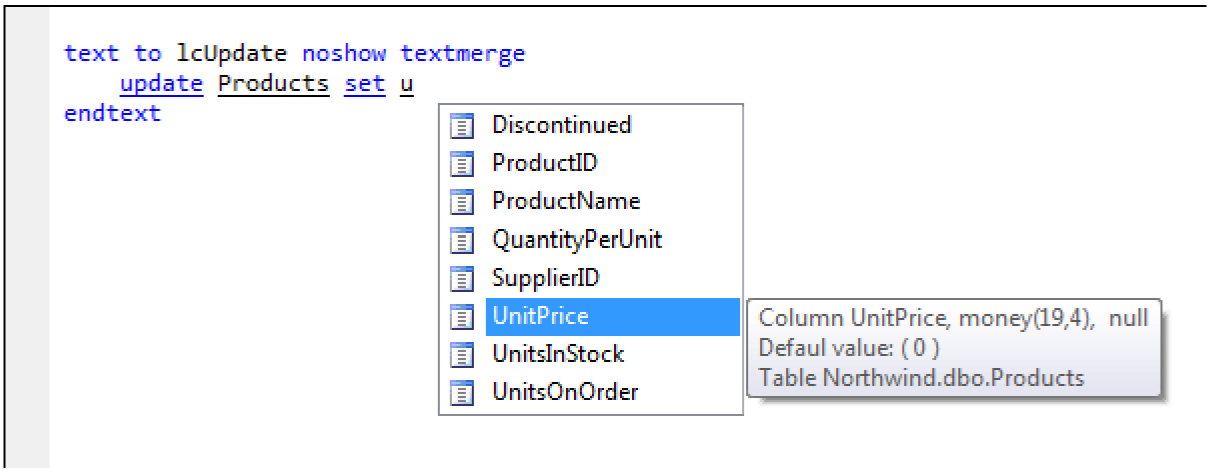
UPDATE – After the command “UPDATE”, a list of tables from the current database is shown (non-incremental mode)



```
text to lcUpdate noshow textmerge
    update
endtext
```

Image 27.17

In incremental mode the fields that belong to the table defined after the command "UPDATE" are shown.

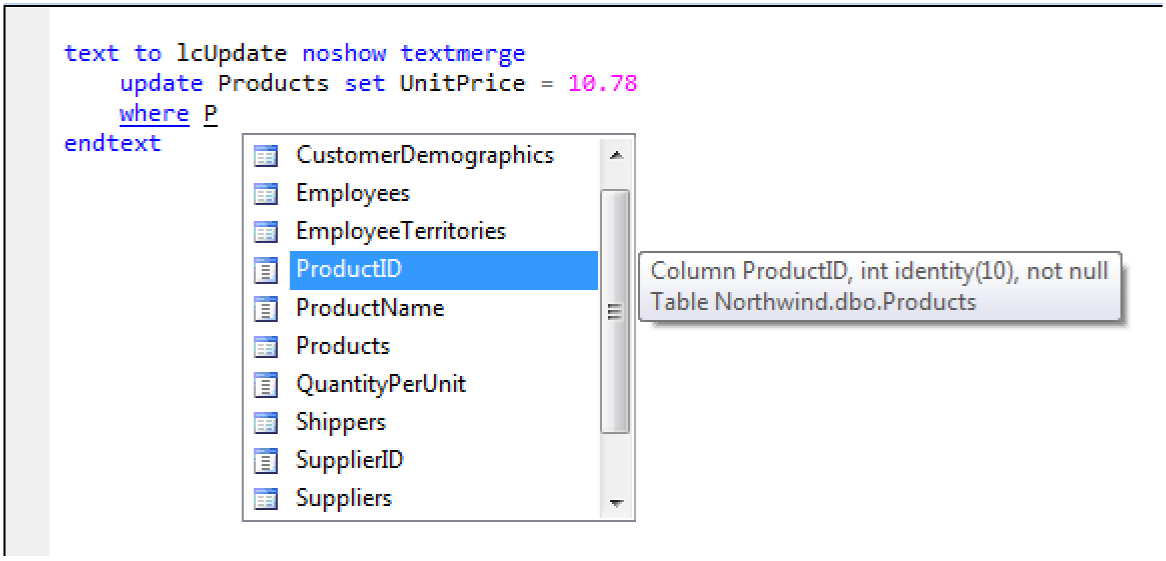


```
text to lcUpdate noshow textmerge
    update Products set u
endtext
```

Column UnitPrice, money(19,4), null
Defaul value: (0)
Table Northwind.dbo.Products

Image 27.18

After de clause "WHERE", in incremental mode the tables from the current database and fields that belong to the table defined after the command "UPDATE" are shown.

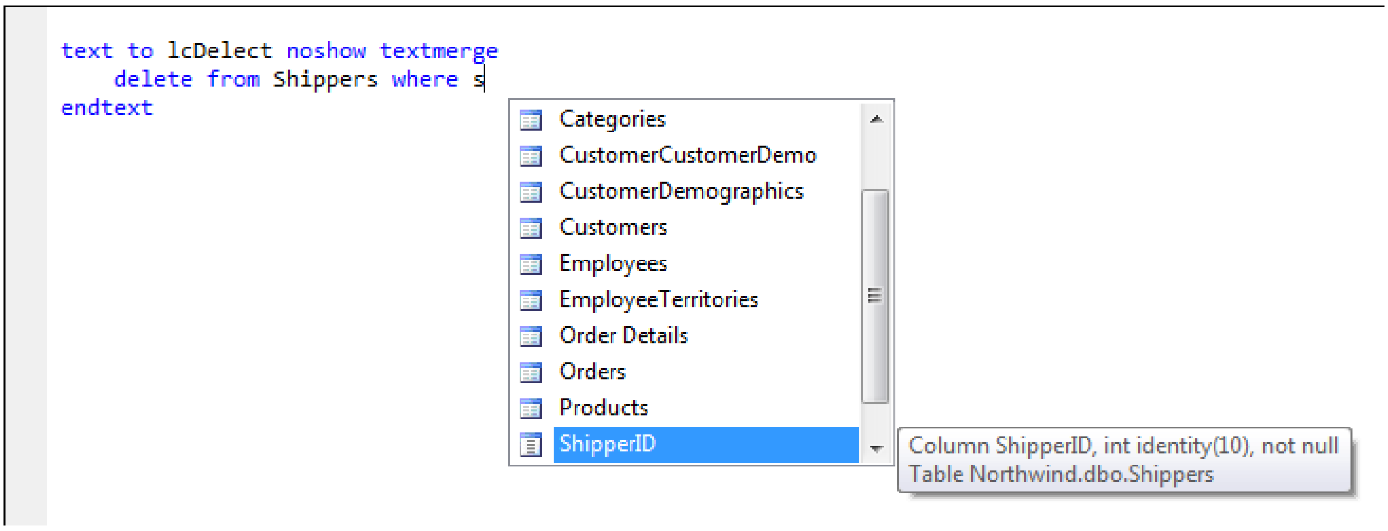


```
text to lcUpdate noshow textmerge
    update Products set UnitPrice = 10.78
    where P
endtext
```

Column ProductID, int identity(10), not null
Table Northwind.dbo.Products

Image 27.19

DELETE – After de clause "WHERE", in incremental mode the tables from the current database and fields that belong to the table defined after the clause "FROM" are shown.



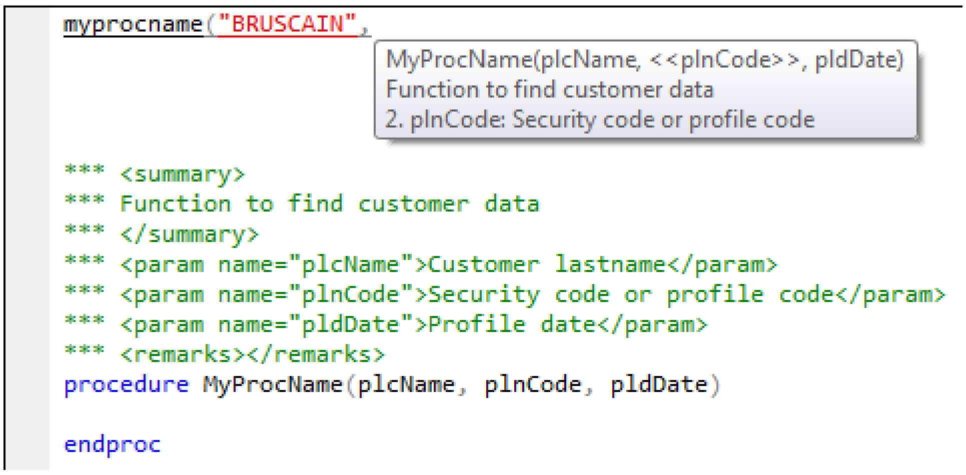
```
text to lcDelect noshow textmerge
    delete from Shippers where s
endtext
```

Column ShipperID, int identity(10), not null
Table Northwind.dbo.Shippers

Image 27.20

28- Signature of custom procedures, functions, methods and events.

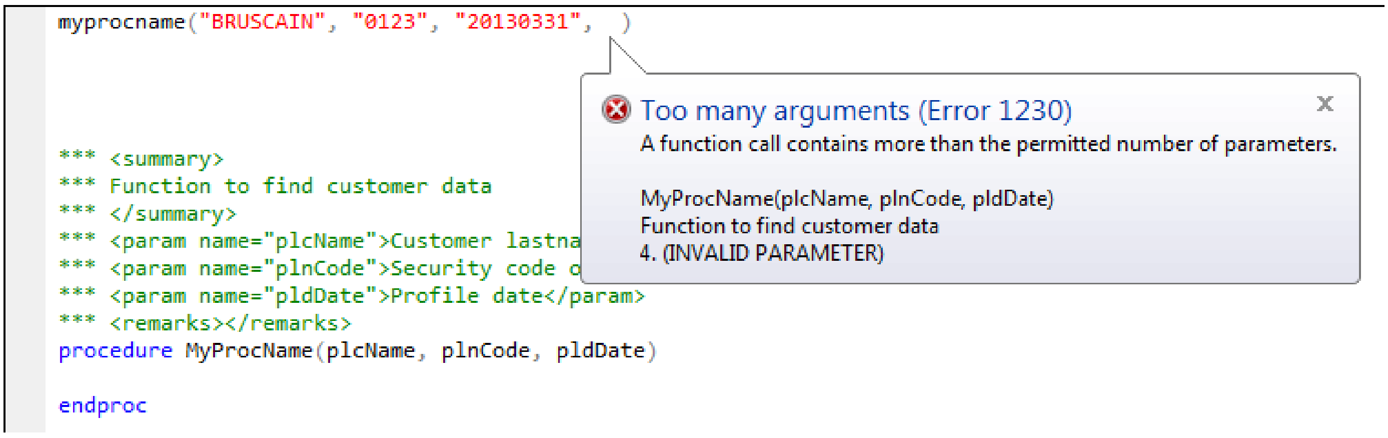
Procedures and functions created in the current PRG or invoked by SET PROCEDURE TO... now can show a tooltip with the signature; in addition the summary is supported. That is, according to the positioned parameter, the tooltip respectively can show the number of the parameter and the information defined in the summary. NOTE: No support for native functions, nor for methods and events outside of a "Define Class ...". (By now)



```
myprocname("BRUSCAIN",
                        MyProcName(plcName, <<plnCode>>, pldDate)
                        Function to find customer data
                        2. plnCode: Security code or profile code

*** <summary>
*** Function to find customer data
*** </summary>
*** <param name="plcName">Customer lastname</param>
*** <param name="plnCode">Security code or profile code</param>
*** <param name="pldDate">Profile date</param>
*** <remarks></remarks>
procedure MyProcName(plcName, plnCode, pldDate)

endproc
```

Image 28.10

If the function contains more than the permitted number of parameters, an error will be shown at write-time.



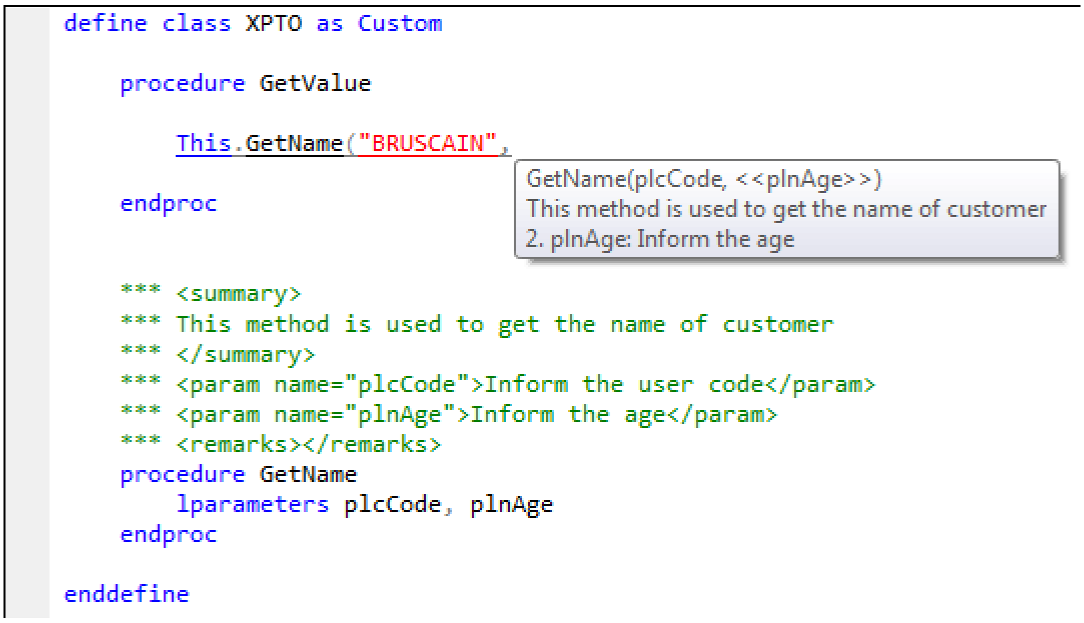
```
myprocname("BRUSCAIN", "0123", "20130331",  )

*** <summary>
*** Function to find customer data
*** </summary>
*** <param name="plcName">Customer lastna
*** <param name="plnCode">Security code o
*** <param name="pldDate">Profile date</param>
*** <remarks></remarks>
procedure MyProcName(plcName, plnCode, pldDate)

endproc
```

Too many arguments (Error 1230)
A function call contains more than the permitted number of parameters.

MyProcName(plcName, plnCode, pldDate)
Function to find customer data
4. (INVALID PARAMETER)

Image 28.11

For methods and events, the functionality is the same.



```
define class XPTO as Custom

    procedure GetValue

        This.GetName("BRUSCAIN",
                                  GetName(plcCode, <<plnAge>>)
    endproc                       This method is used to get the name of customer
                                  2. plnAge: Inform the age

    *** <summary>
    *** This method is used to get the name of customer
    *** </summary>
    *** <param name="plcCode">Inform the user code</param>
    *** <param name="plnAge">Inform the age</param>
    *** <remarks></remarks>
    procedure GetName
        lparameters plcCode, plnAge
    endproc

enddefine
```

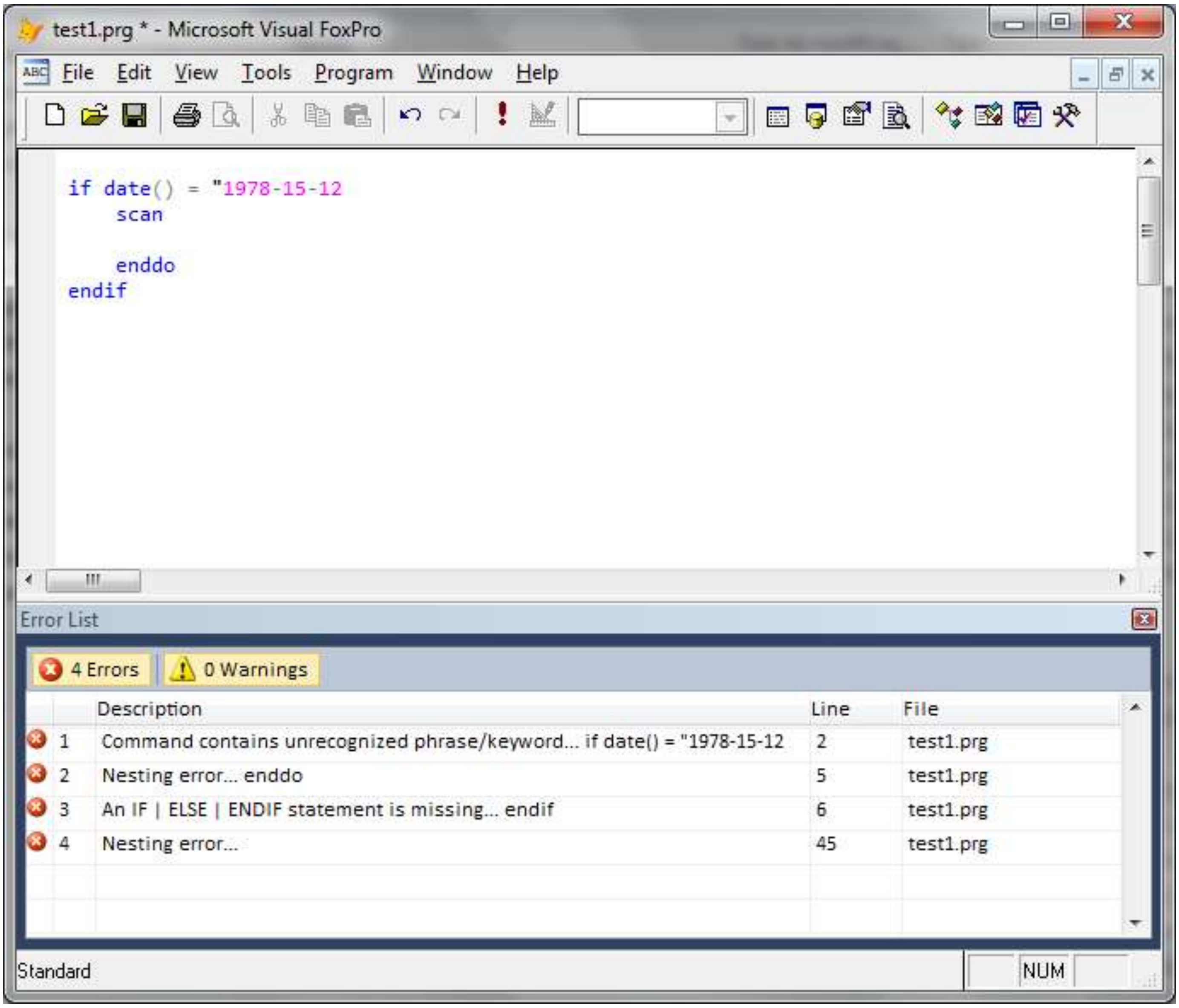
Image 28.12

29- Error list

Display the program errors by compiling at write-time.

When you click on an error in the list, VFP will go to the program line containing the error.

NOTE: This option can slow VFP down, depending on the size of the PRG file.



test1.prg * - Microsoft Visual FoxPro
File Edit View Tools Program Window Help
Error List
4 Errors
0 Warnings
Standard
NUM

```
if date() = "1978-15-12
    scan

    enddo
endif
```

	Description	Line	File
1	Command contains unrecognized phrase/keyword... if date() = "1978-15-12	2	test1.prg
2	Nesting error... enddo	5	test1.prg
3	An IF | ELSE | ENDIF statement is missing... endif	6	test1.prg
4	Nesting error...	45	test1.prg

Image 29.10

*To activate the "Error List window", select "Error List" from the "View" menu. You can also configure VFP to always start with the "Error List" window opened; to do that, mark the "**Show Error List window**" checkbox in the "IntelliSense Manager".*

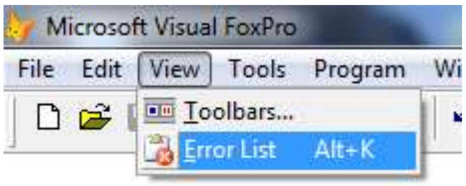
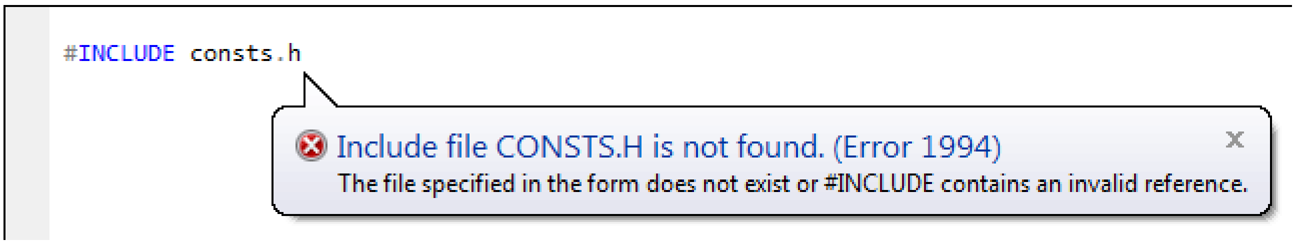


Image 29.11

30- Error Tip

*Show some possible run-time errors in write-time. To display the errors in a tool tip, you need to check the “**Show Error Tip”** checkbox in the "IntelliSense Manager"*

Bellow, the Errors Tips available:

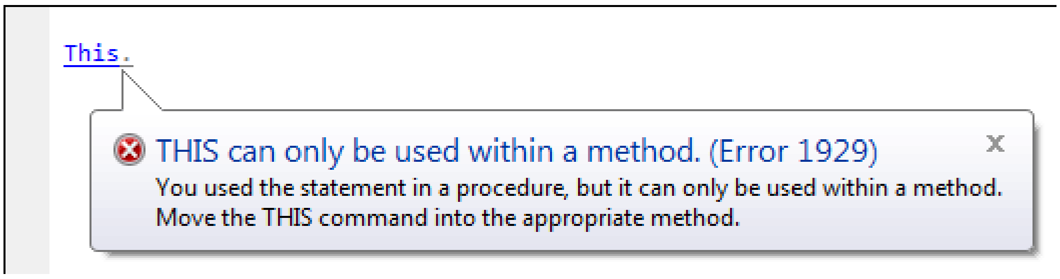


```
#INCLUDE consts.h
```

Include file CONSTS.H is not found. (Error 1994)
The file specified in the form does not exist or #INCLUDE contains an invalid reference.

Image 30.10

The message bellow can be showed for THIS, THISFORM and THISFORMSET

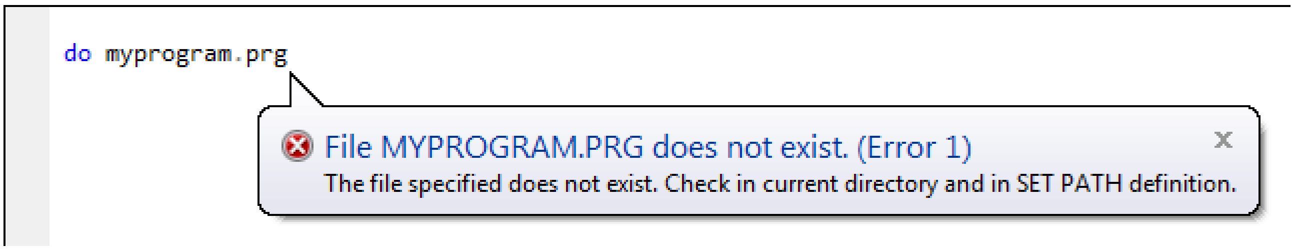


```
This.
```

THIS can only be used within a method. (Error 1929)
You used the statement in a procedure, but it can only be used within a method. Move the THIS command into the appropriate method.

Image 30.11

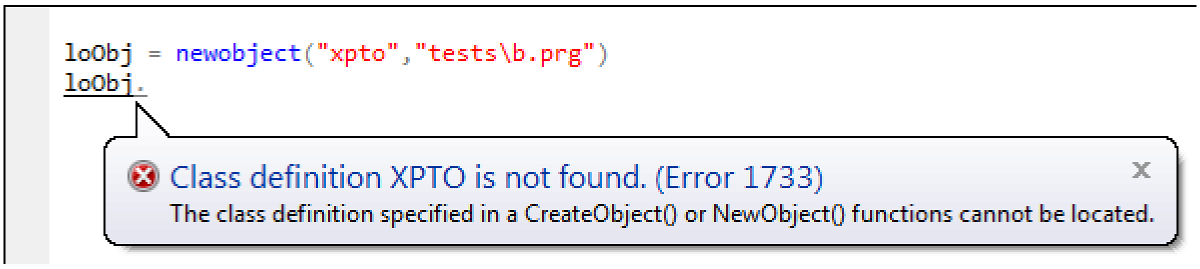
The message bellow can be showed for PRG|MPR|QPR|FXP|APP|EXE



```
do myprogram.prg
```

File MYPROGRAM.PRG does not exist. (Error 1)
The file specified does not exist. Check in current directory and in SET PATH definition.

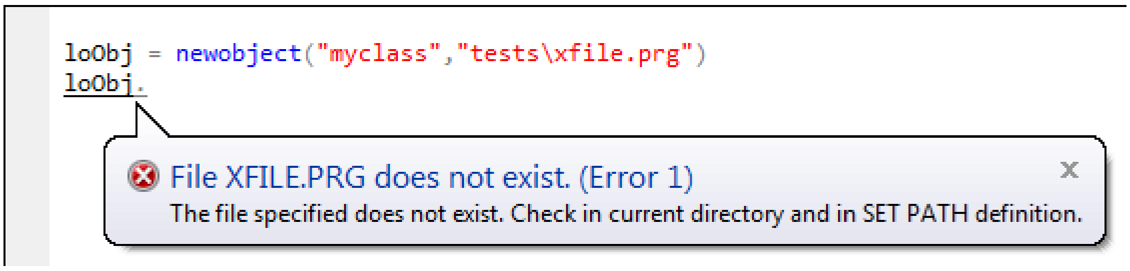
Image 30.12



```
loObj = newobject("xpto","tests\b.prg")
loObj.
```

Class definition XPTO is not found. (Error 1733)
The class definition specified in a CreateObject() or NewObject() functions cannot be located.

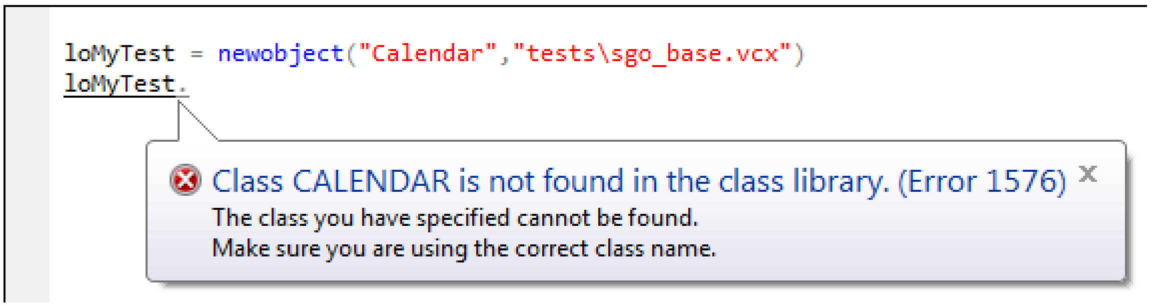
Image 30.13



```
loObj = newobject("myclass","tests\xfile.prg")
loObj.
```

File XFILE.PRG does not exist. (Error 1)
The file specified does not exist. Check in current directory and in SET PATH definition.

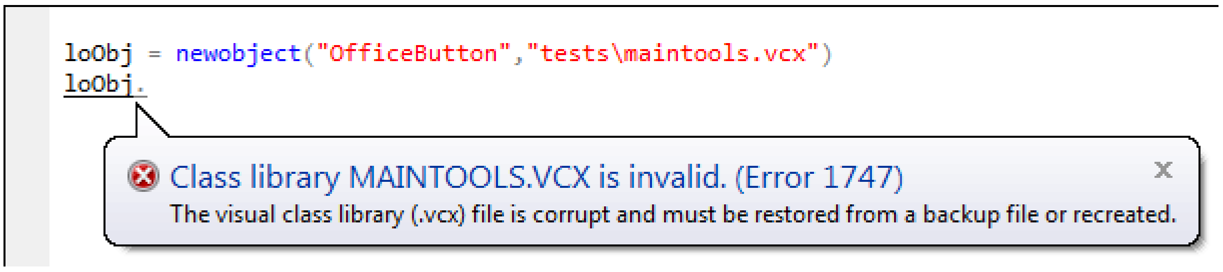
Image 30.14



```
loMyTest = newobject("Calendar","tests\sgo_base.vcx")
loMyTest.
```

Class CALENDAR is not found in the class library. (Error 1576)
The class you have specified cannot be found.
Make sure you are using the correct class name.

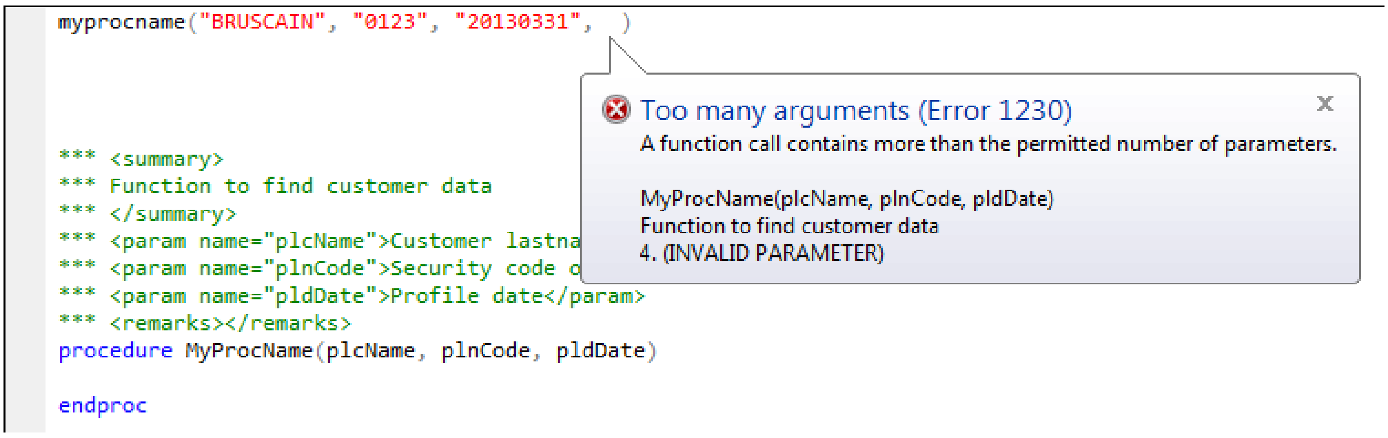
Image 30.15



```
loObj = newobject("OfficeButton","tests\maintools.vcx")
loObj.
```

Class library MAINTOOLS.VCX is invalid. (Error 1747)
The visual class library (.vcx) file is corrupt and must be restored from a backup file or recreated.

Image 30.16

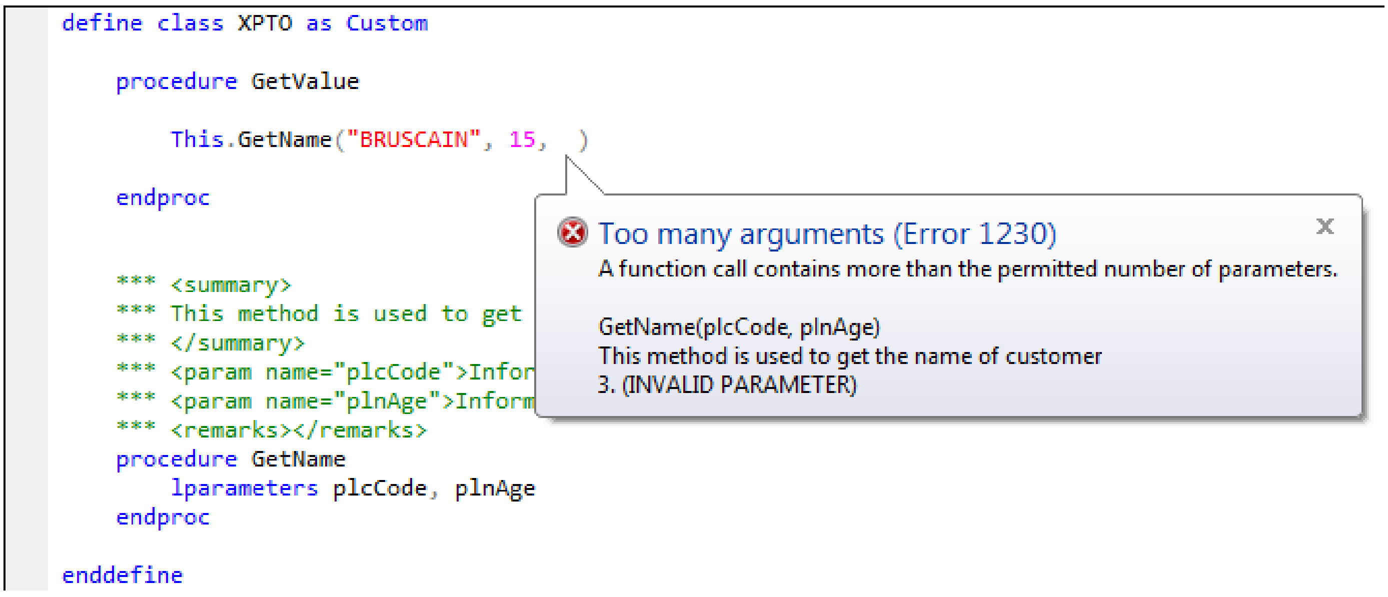


```
myprocname("BRUSCAIN", "0123", "20130331",  )


*** <summary>
*** Function to find customer data
*** </summary>
*** <param name="plcName">Customer lastna
*** <param name="plnCode">Security code o
*** <param name="pldDate">Profile date</param>
*** <remarks></remarks>
procedure MyProcName(plcName, plnCode, pldDate)

endproc
```

Too many arguments (Error 1230)
A function call contains more than the permitted number of parameters.

MyProcName(plcName, plnCode, pldDate)
Function to find customer data
4. (INVALID PARAMETER)

Image 30.17



```
define class XPTO as Custom

    procedure GetValue

        This.GetName("BRUSCAIN", 15,  )

    endproc


    *** <summary>
    *** This method is used to get
    *** </summary>
    *** <param name="plcCode">Infor
    *** <param name="plnAge">Inform
    *** <remarks></remarks>
    procedure GetName
        lparameters plcCode, plnAge
    endproc

enddefine
```

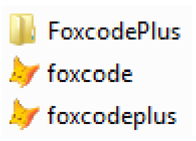
Too many arguments (Error 1230)
A function call contains more than the permitted number of parameters.

GetName(plcCode, plnAge)
This method is used to get the name of customer
3. (INVALID PARAMETER)

Image 30.18

31- <u>Installing FoxcodePlus (Only for Visual FoxPro 9)</u>

Files available:



FoxcodePlus
foxcode
foxcodeplus

If you already have installed FoxcodePlus 3.10 or a previous version, you must have to replace the files <u>foxcode.app</u> , <u>foxcodeplus.app</u> and whole directory <u>...\Foxcodeplus*.*</u>. Also, it's necessary to do the step 7. If you have installed 3.11 or 3.12 you only have to replace <u>foxcode.app</u> and <u>foxcodeplus.app.</u>

Follow the 7 steps below to install for the <u>first time</u>:

1) If VFP is open, close it.
2) Open the folder where Visual FoxPro 9 is installed.
3) Rename files FoxCode.App
4) Copy the new files available to the folder.
5) Open VFP and access the "IntelliSense Manager" as indicated below:

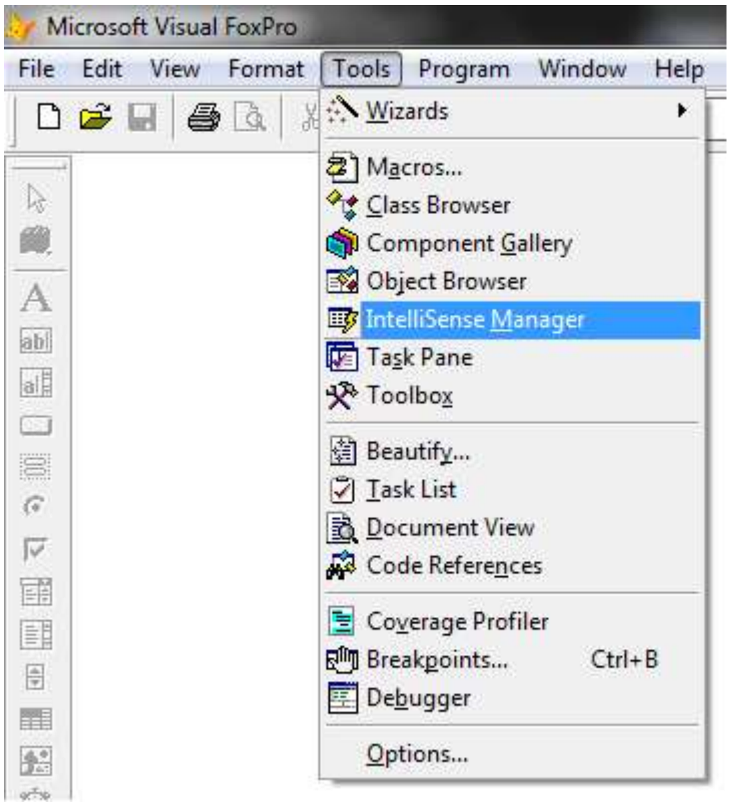


Image 31.10

6) Make the settings as below.

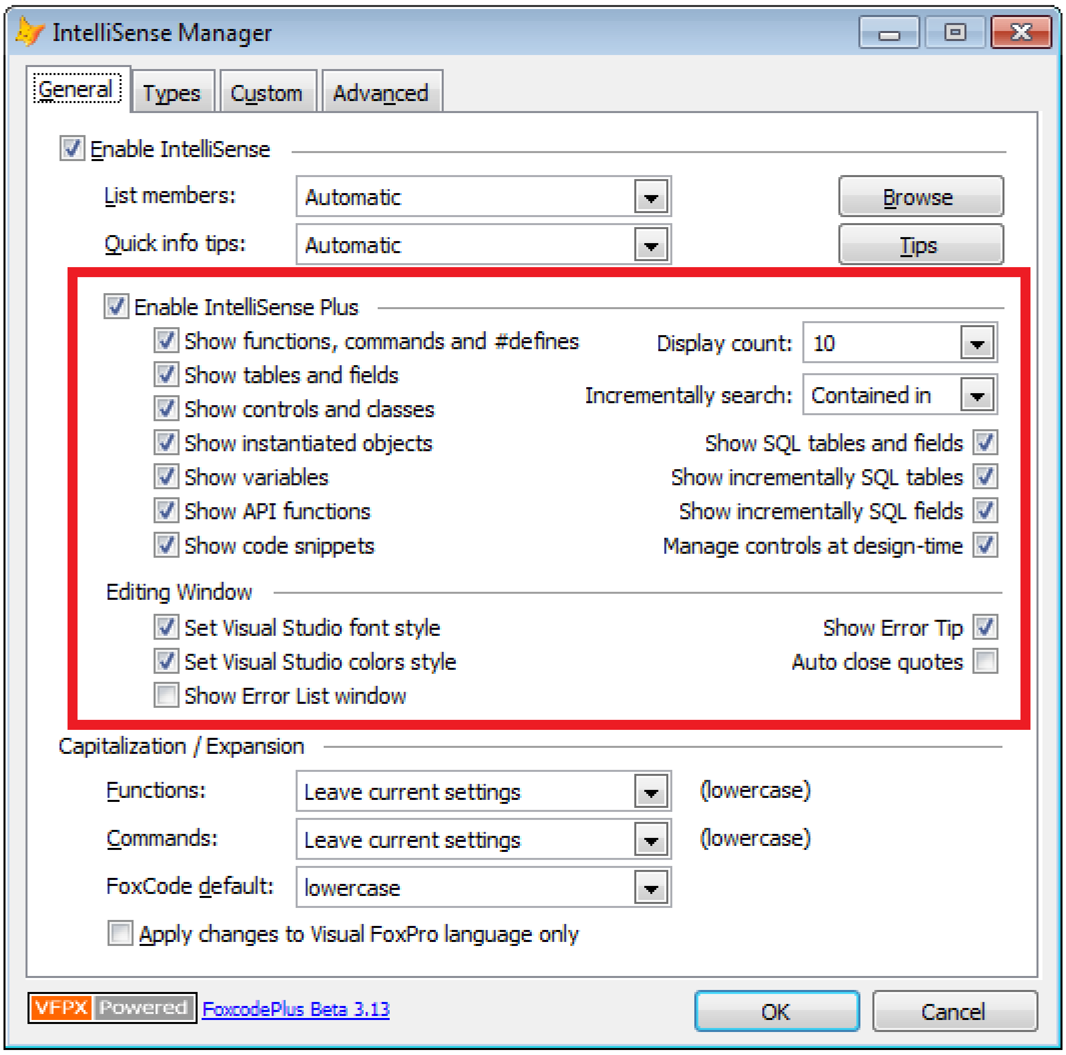
The "IntelliSense Manager" now includes some new options as outlined below in red.
*Activate the "**Enable IntelliSense Plus**" checkbox, select or deselect the options according to your choice.*
*By default, the "**Show Error List Window"** and "**Auto close quotes**" checkboxes are not marked.*

NOTE 1: If your VFP is customized with a program like "STARTUP", it will be removed. To resolve this, you have to create a PRG file to call your program and FoxcodePlus.App

*NOTE 2: The colors established by the "**Set Visual Studio colors style**" checkbox option can be reconfigured by the general editor of the VFP.*

*NOTE 3: When you activate the "**Enable IntelliSense Plu**s" checkbox, the CONFIG.FPW file will be created if it does not already exist, and it will be modified to run the correct FoxcodePlus.App.*

NOTE 4: Not all the new features of FoxcodePlus work in the "Command Window".



[✓] Enable IntelliSense
List members: Automatic
Quick info tips: Automatic

[✓] Enable IntelliSense Plus
[✓] Show functions, commands and #defines
[✓] Show tables and fields
[✓] Show controls and classes
[✓] Show instantiated objects
[✓] Show variables
[✓] Show API functions
[✓] Show code snippets
Display count: 10
Incrementally search: Contained in
Show SQL tables and fields [✓]
Show incrementally SQL tables [✓]
Show incrementally SQL fields [✓]
Manage controls at design-time [✓]

Editing Window
[✓] Set Visual Studio font style
[✓] Set Visual Studio colors style
[ ] Show Error List window
Show Error Tip [✓]
Auto close quotes [ ]

Capitalization / Expansion
Functions: Leave current settings (lowercase)
Commands: Leave current settings (lowercase)
FoxCode default: lowercase
[ ] Apply changes to Visual FoxPro language only

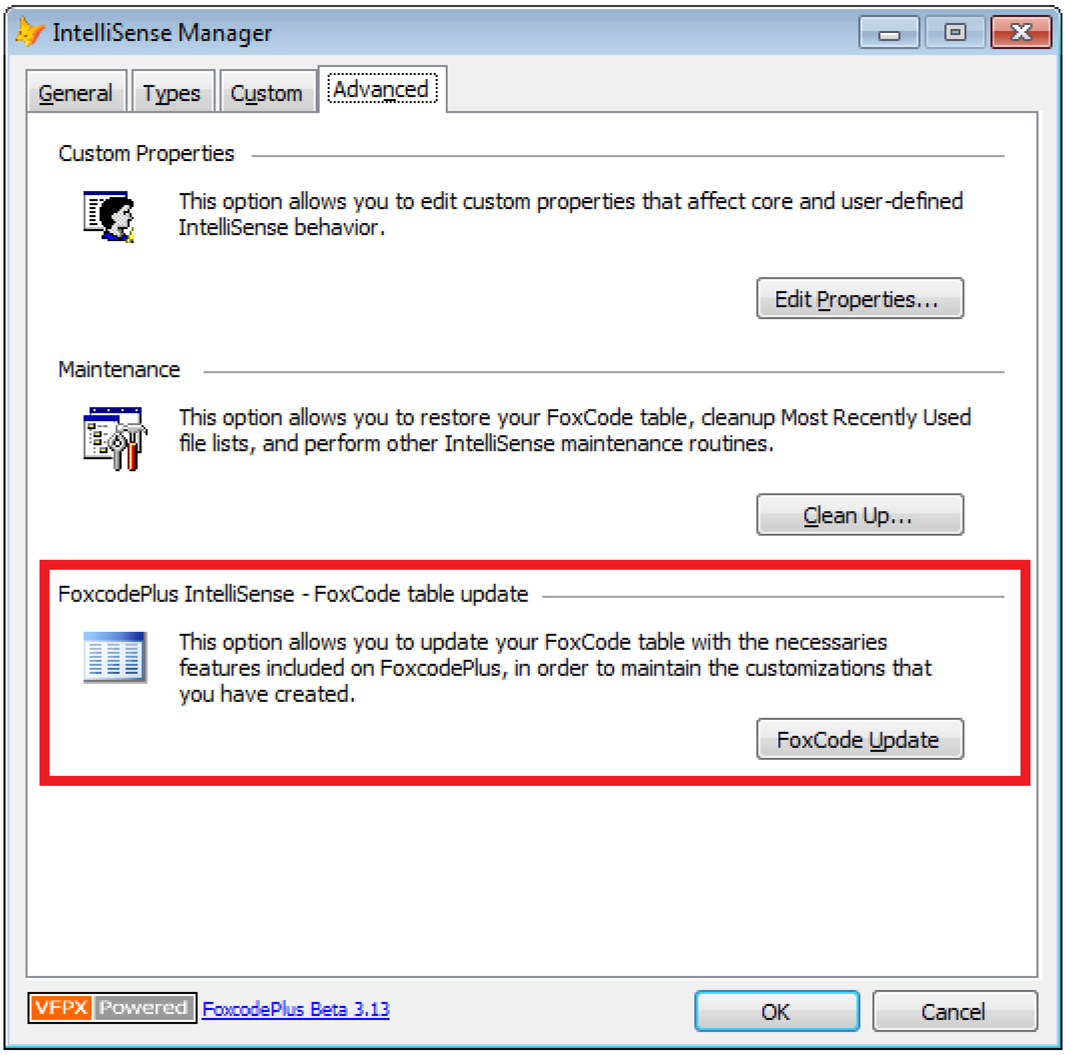
Image 31.11

7) The last step, FoxCode table update.

With this update you will keep your customizations in the FoxCode table. A backup of your FoxCode table will be created automatically and a log file will be generated to indicate all updates and additions in your FoxCode table.

The update will change some existing registers and will include new registers in your FoxCode. However, there is the possibility of the update override some customization if the customization has the same information in the "TYPE", "ABBREV" and "EXPANDED" fields.

If you have no customization in the FoxCode table, perfect. If you have, run the update and check if something that you have created has been overridden. In this case, you must manually adjust your FoxCode table.


IntelliSense Manager

General | Types | Custom | Advanced

Custom Properties

This option allows you to edit custom properties that affect core and user-defined IntelliSense behavior.

Edit Properties...

Maintenance

This option allows you to restore your FoxCode table, cleanup Most Recently Used file lists, and perform other IntelliSense maintenance routines.

Clean Up...

FoxcodePlus IntelliSense - FoxCode table update

This option allows you to update your FoxCode table with the necessaries features included on FoxcodePlus, in order to maintain the customizations that you have created.

FoxCode Update

VFPX Powered FoxcodePlus Beta 3.13

OK Cancel

Image 31.12

Thank you everyone and have a good times.

Rodrigo Duarte Bruscain
Visual Studio | Visual C# | MS SQL Server | Visual FoxPro
https://www.mcpvirtualbusinesscard.com/VBCServer/rodrigobruscain/profile
www.linkedin.com/in/rodrigobruscain