

Streamlining Development

VS Code and .NET for PxPlus 2026

DireXions 2026 - Developing the Future





Goal

To help you streamline and modernize your development using the VS Code editor with the PxPlus extension and integrating .NET into your PxPlus applications

Agenda

Timeline Overview

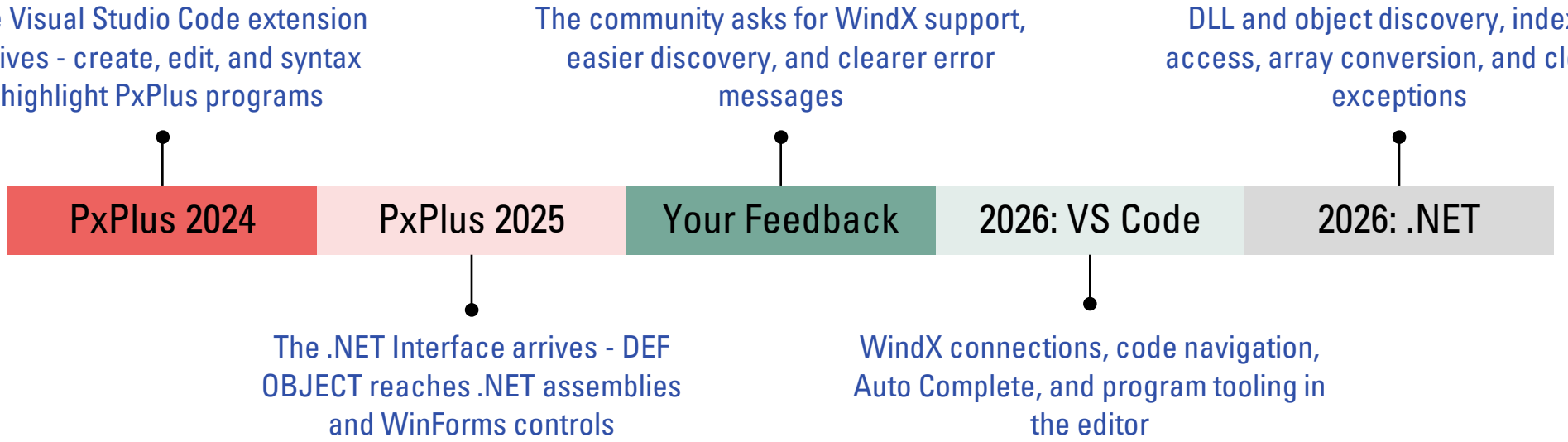
Visual Studio Code Extension Enhancements

.NET Interface Enhancements

PxPlus 2026 Update 1



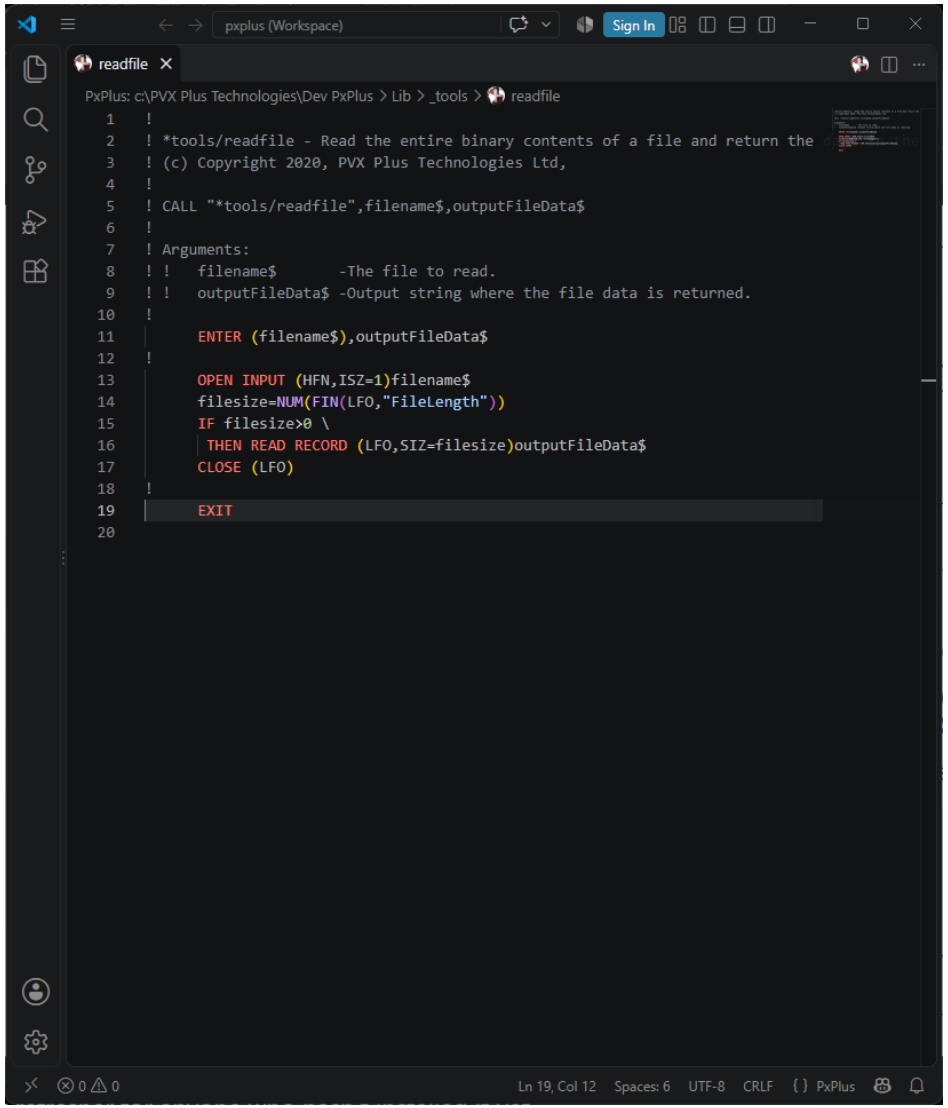
Timeline Overview





Visual Studio Code Extension Enhancements

Fewer licence hurdles, real navigation, and program tooling in the editor your team already uses



The Extension So Far

Create and edit both binary and text-based PxPlus programs

Syntax highlighting and program error diagnostics

Current document based autocomplete

Language preferences: lowercase directives, lowercase and mixed-case variables, Suppress LET

Added in PxPlus 2024, and shaped by user feedback ever since

WindX Plug-in Connections

No full PxPlus licence locally

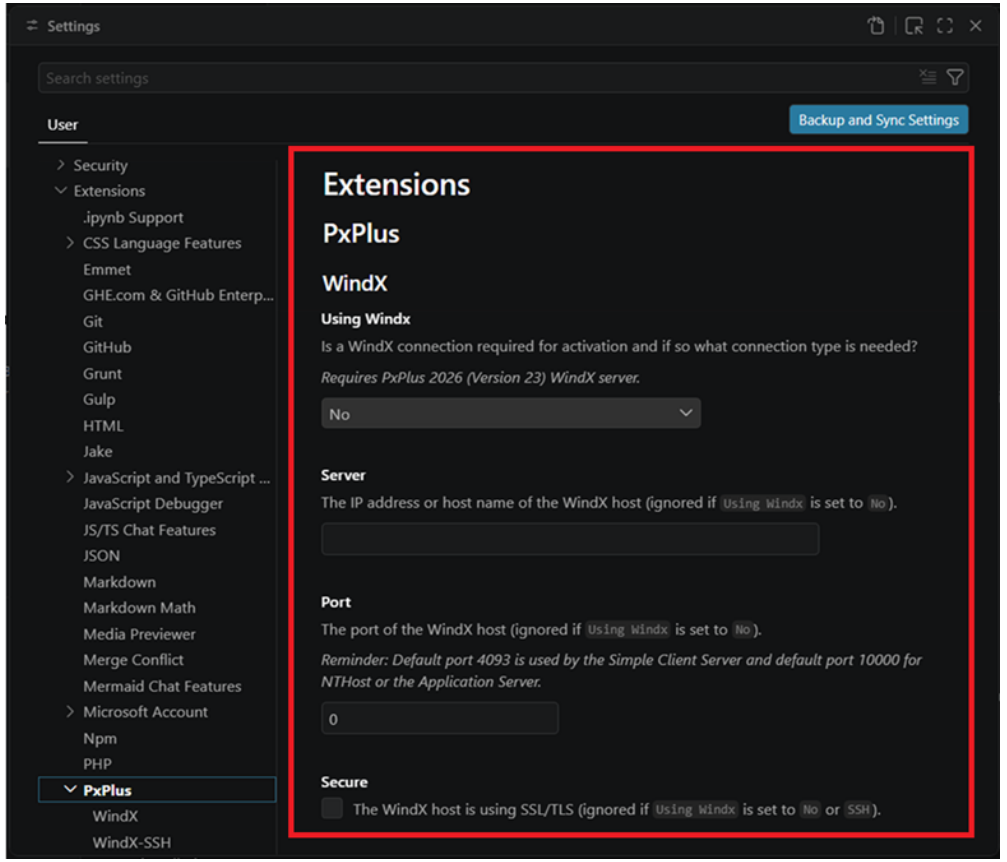
- You still install locally - the WindX 2026+ Plug-in, not a fully licensed PxPlus
- The activated PxPlus comes from the server where your application lives (Must be at least PxPlus 2026)
- Already a common setup: the application on a server, developers working through the plug-in
- Your files stay where they are - VS Code opens what this machine can reach
- Improved support for running VS Code itself on Linux and macOS

Four connection types

- No - a local PxPlus install
- Simple CS - default port 4093
- NThost or Application Server - default port 10000
- SSH - username, securely stored password, and a remote PxPlus path

Setting Up WindX Connection

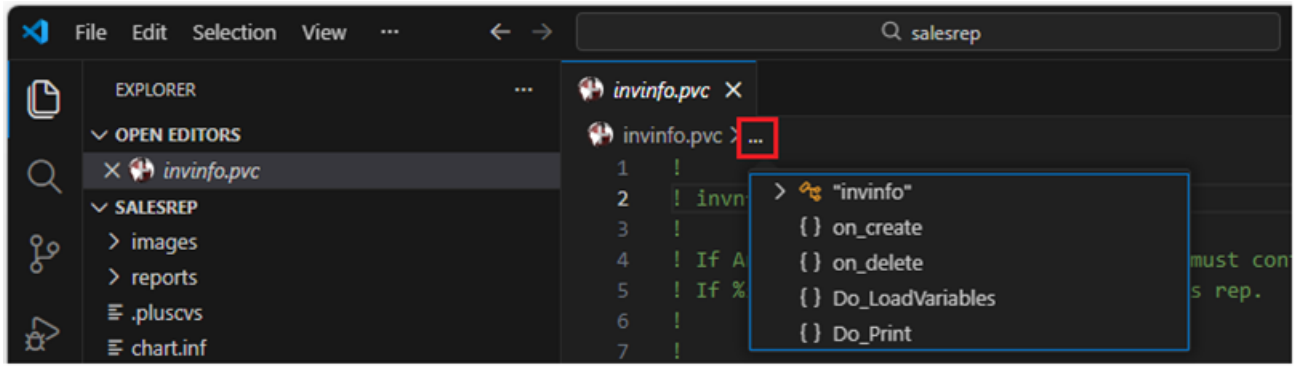
Group	Setting	What to enter
PxPlus	Pxplus Path	Path to the PxPlus or WindX Plug-in directory (required)
WindX	Using Windx	No, Simple CS, NThost, or SSH (default No)
WindX	Server and Port	Host name or IP, plus the port - 4093 or 10000
WindX	Secure	On when the host uses SSL/TLS (ignored for SSH)
WindX-SSH	User & Password	OS login for the host; the password is stored securely
WindX-SSH	Pxplus Path	Full path to the PxPlus executable on the WindX host (ignored unless SSH) Example: /home/user/pxplus2026/pxplus



Code Navigation

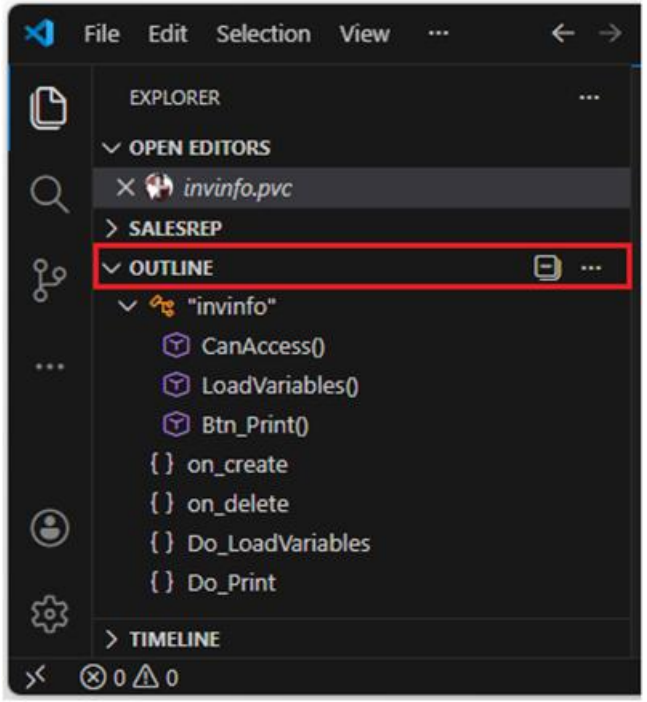
Two ways in

- The breadcrumb menu at the top of the editor tab - click the ellipsis for the list
- The OUTLINE section in the EXPLORER view



What it finds

- Labels, functions, classes, and methods
- Select an entry to jump straight to it in the code
- No more scrolling a thousand-line program to find one label



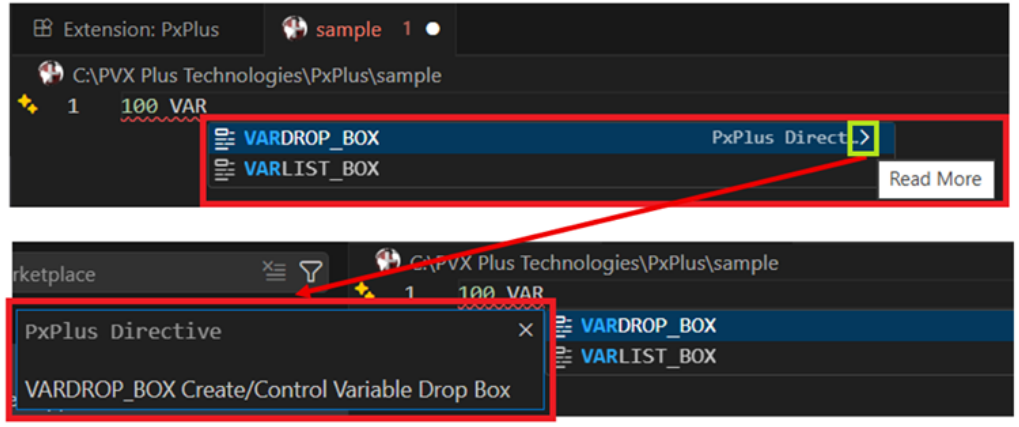
Auto Complete Gets Smarter

Now suggests

- Directives and mnemonics
- System functions, system parameters, and system variables
- Names already in use in the current document

In practice

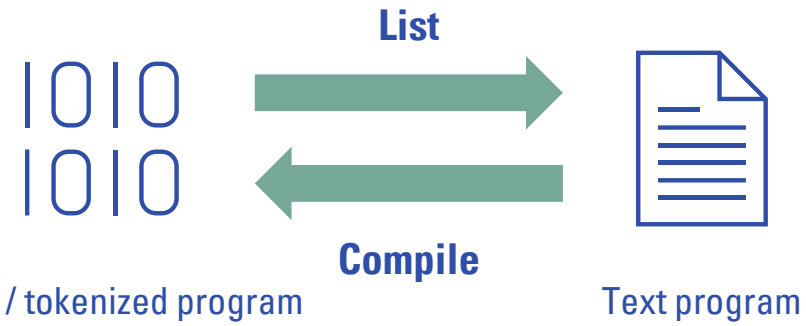
- Click the arrow beside a suggestion to expand its description
- Press Ctrl + Spacebar to trigger it manually
- Auto line numbering for programs that use line numbers



Compile and List PxPlus Programs

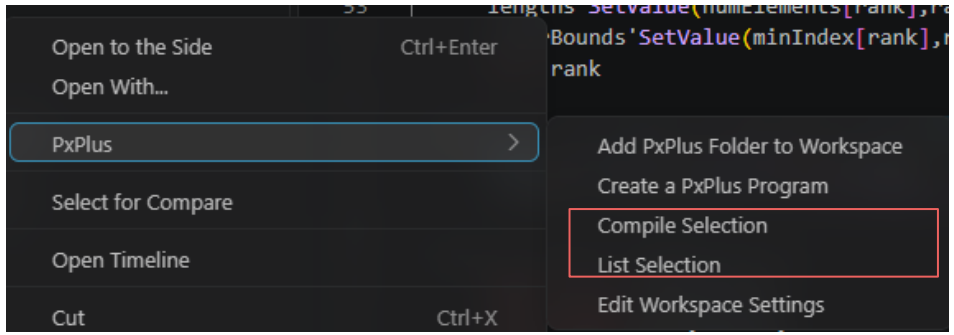
Both directions

- Compile - a text program becomes binary / tokenized
- List - a binary / tokenized program becomes text
- Runs across selected files, folders, and their sub-directories



Where to set it up

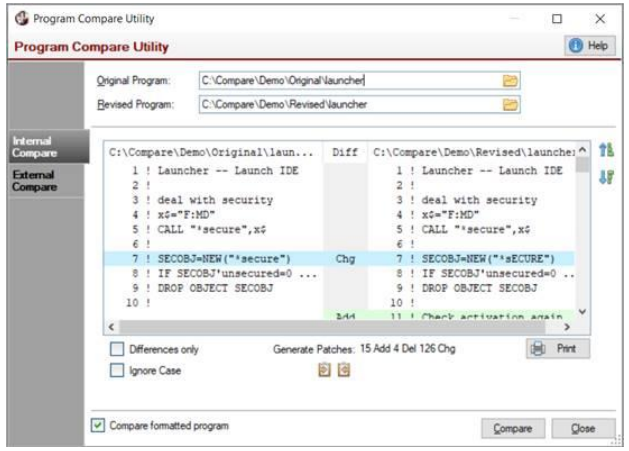
- Trigger from the PxPlus icon in the editor, or the Explorer right-click menu
- Set the Compiled and Listed Output Directory under Settings->Extensions->PxPlus-> Conversion
- Optionally add a file extension to the output



Working With Binary Programs

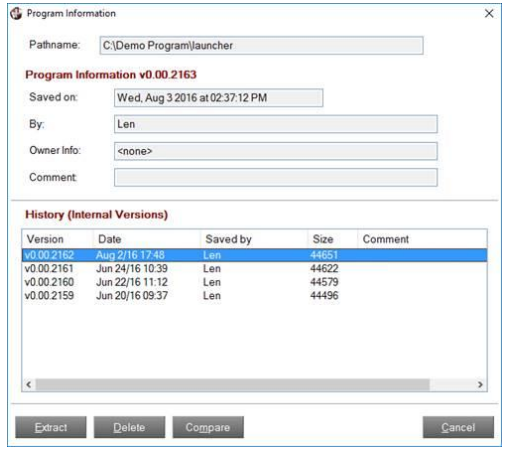
Compare Programs

- Right-click in the editor, then the PxPlus sub-menu
- Launches the PxPlus Program Compare Utility
- Diff two tokenized programs



Program Properties

- Same right-click PxPlus sub-menu
- Launches the Program Properties Utility
- Inspect a binary program without leaving the editor



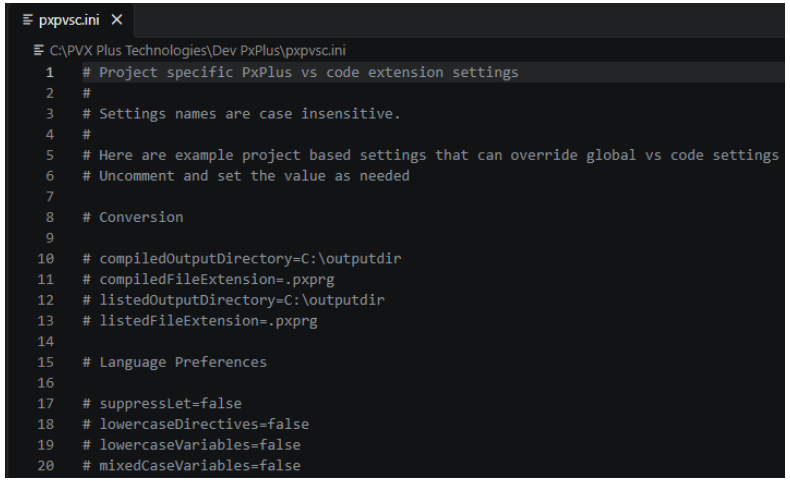
Worth Knowing

- Both require PxPlus 2026 or newer
- Both reuse toolkit utilities you already know

Settings Per Project

pxpvsc.ini

- Created from PxPlus > Edit Workspace Settings in the Explorer right-click menu
- Lives in the workspace root directory
- Uncomment only the settings you want to override



Why it matters

- Suppress LET on for one codebase and off for another
- Conversion paths that follow the project, not the machine
- Setting names match the Settings screen with the spaces removed



.NET Interface Enhancements

Reaching further into the .NET ecosystem, with fewer names to memorize



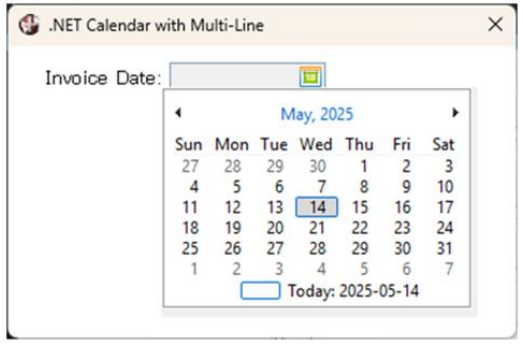
The .NET Interface, Recapped

Define

- DEF OBJECT with a [.NET] prefix, or a NOMADS External Control
- Then a DLL path, or a system DLL in the GAC
- Then the namespace and object name (case sensitive)

Use

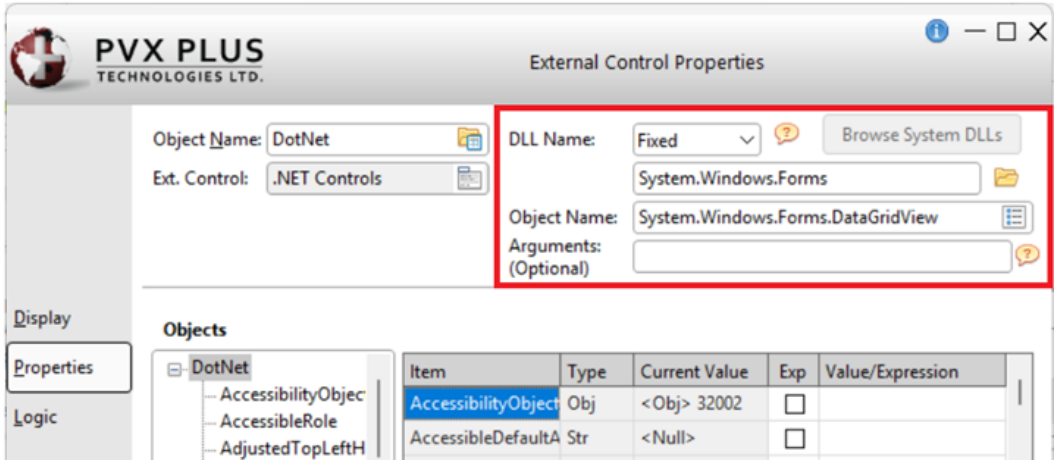
- Properties and methods through the apostrophe operator, or the NOMADS Properties tab
- Events through ON EVENT, or the NOMADS Events grid
- PvxHandle$ to pass one .NET object into another



Install

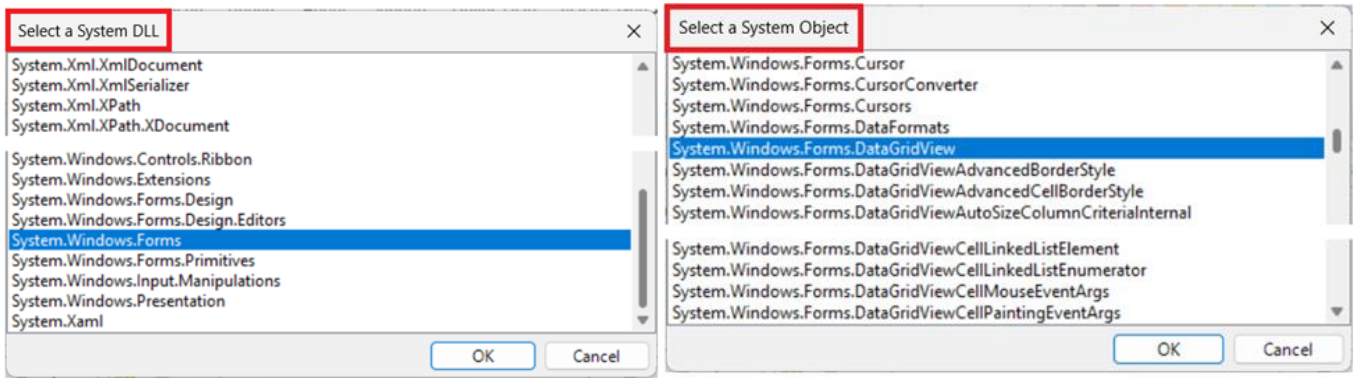
- Windows only, with **Windows Desktop Runtime 8** installed, even if 9 or 10 is already there
- Match the runtime to PxPlus - x64 or x86
- Under WindX, each client needs the runtime too

Stop Hunting for Names



In NOMADS

- Browse System DLLs fills in both DLL Name and Object Name
- Enabled when DLL Name is Fixed and still empty
- The Object Name query button lists every object in that DLL
- DLL Name can now be an Expression, not only Fixed



In code

- `DEF OBJECT id,"[.NET]*"` - pick a system DLL, then pick an object
- `DEF OBJECT id,"[.NET]DllName,*"` - pick an object from a known DLL
- `PvxDllName$` and `PvxObjName$` extended properties report back what was selected

PvxGetAt and PvxSetAt

Why it was needed

- Some .NET object properties are only accessible via `[]` syntax
- Can't access via PxPlus since `[]` is specific to PxPlus arrays

What they do

- Returns the value at the given index from the .NET object. Equivalent to doing a `myArray[index]` in .NET

```
PvxGetAt[$](index[$])
```

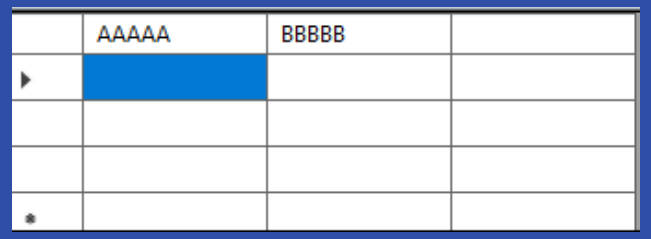
- Sets .NET object's element specified by index to the value. Equivalent to doing a `myArray[index]=value` in .NET

```
PvxSetAt(index[$],value[$])
```

- What can index: numeric, string, and .NET object handles
- Numeric indexes start at zero

PvxGetAt and PvxSetAt

```
dll$="System.Windows.Forms"
obj$="System.Windows.Forms.DataGridView"
def object datagridview1,@(20,0,70,8),"[.NET]"+dll$+","+obj$
datagridview1'columncount=3
datagridview1'rowcount=4
cols=datagridview1'columns
cols'PvxGetAt(0)'Name$="AAAAA"
cols'PvxGetAt(1)'Name$="BBBBB"
```



PvxGetAt and PvxSetAt

```
dll$="System.runtime"
obj$="System.Collections.Hashtable"
def object myhashtbl,"[.NET]"+dll$+","+obj$
myhashtbl'pvxSetAt("one","Hello")
myhashtbl'pvxSetAt("two","World")
?myhashtbl'pvxGetAt$("one")
Hello
?myhashtbl'pvxGetAt$("two")
World
```

```
dll$="System.runtime"
obj$="System.Collections.Hashtable"
def object myhashtbl,"[.NET]"+dll$+","+obj$
obj$="System.DateTime"
def object xmas,"[.NET]"+dll$+","+obj$,2026,12,25
def object newyear,"[.NET]"+dll$+","+obj$,2027,1,1
myhashtbl'pvxSetAt(xmas'pvxHandle$,"Christmas Day")
myhashtbl'pvxSetAt(newyear'pvxHandle$,"New Year's Day")
?myhashtbl'pvxGetAt$(xmas'pvxHandle$)
Christmas Day
?myhashtbl'pvxGetAt$(newyear'pvxHandle$)
New Year's Day
```

Converting Arrays

Why it was needed

- .NET arrays meant you had to manually convert to/from PxPlus arrays
- This made it hard to code and use

*Tools/DotNetConvert

- PxPlus array to .NET array

```
CALL "*tools/dotnetconvert;toarray",pxpArray[$]{ALL},dotNetArray
```

- .NET array back to PxPlus array

```
CALL "*tools/dotnetconvert;fromarray",dotNetArray, pxpArray[$]{ALL}
```

Converting Arrays

```
dll$="System.Runtime"
obj$="System.Globalization.DateTimeFormatInfo"
def object dtInfo,"[.net]"+dll$+","+obj$
call "*tools/dotnetconvert;fromarray",dtInfo'MonthNames,myarray${ALL}
?myarray$[0]
January
?myarray$[11]
December
```

```
dll$="System.Windows.Forms"
obj$="System.Windows.Forms.TextBox"
def object textbox,@(10,10,50,50),"[.net]"+dll$+","+obj$
textbox'multiline=1
dim strArray$[2]
strArray$[0]="aaa"
strArray$[1]="bbb"
strArray$[2]="ccc"
call "*tools/dotnetconvert;toarray",strArray${ALL},dotNetArray
textbox'lines$=dotNetArray'pvxhandle$
```



Errors You Can Act On

PvxException$

- Returns the last .NET exception message
- Read it after a method call that failed
- A read-only property on any PxPlus .NET object

MSG(-1)

- Exceptions thrown during DEF OBJECT now carry expanded detail
- At the command line the detail shows in red under the error
- No more guessing which argument the object rejected

```
->def object sb,"[.NET]System.Runtime,System.Text.StringBuilder"
->     sb'insert(-10,"hi")
Error #15: Operating system command failed
Could not invoke method: insert (err/ret=2/0)
->?sb'PvxException$
System.Reflection.TargetInvocationException: Exception has been thrown by the target of an invocation.
 ---> System.ArgumentOutOfRangeException: Index was out of range. Must be non-negative and less than c
   at System.Text.StringBuilder.Insert(Int32 index, String value)
   at System.RuntimeMethodHandle.InvokeMethod(Object target, Void** arguments, Signature sig, Boolean
   at System.Reflection.MethodBaseInvoker.InvokeDirectByRefWithFewArgs(Object obj, Span`1 copyOfArgs,
   --- End of inner exception stack trace ---
   at System.Reflection.MethodBaseInvoker.InvokeDirectByRefWithFewArgs(Object obj, Span`1 copyOfArgs,
   at System.Reflection.MethodBaseInvoker.InvokeWithFewArgs(Object obj, BindingFlags invokeAttr, Binde
   at System.Reflection.MethodBase.Invoke(Object obj, Object[] parameters)
   at DotNetInfo.InvokeMethod(SByte* methName, tag dotNetArgs* arguments) in C:\Users\devon\workspace\
```

```
->def object mylist,"[.NET]System.Runtime,System.Collections.ArrayList",-10
Error #15: Operating system command failed
Could not construct object: System.Collections.ArrayList

.NET Exception:
System.Reflection.TargetInvocationException: Exception has been thrown by the target of an invocation.
 ---> System.ArgumentOutOfRangeException: 'capacity' must be non-negative. (Parameter 'capacity')
   at System.Collections.ArrayList..ctor(Int32 capacity)
   at InvokeStub_ArrayList..ctor(Object, Span`1)
   at System.Reflection.MethodBaseInvoker.InvokeWithOneArg(Object obj, BindingFlags invokeAttr, Binder
   --- End of inner exception stack trace ---
   at System.Reflection.MethodBaseInvoker.InvokeWithOneArg(Object obj, BindingFlags invokeAttr, Binder
   at System.Reflection.RuntimeConstructorInfo.Invoke(BindingFlags invokeAttr, Binder binder, Object[]
   at System.RuntimeType.CreateInstanceImpl(BindingFlags bindingAttr, Binder binder, Object[] args, Cu
   at System.Activator.Cr
```

PxPlus 2026 Update 1

Visual Studio Code Extension

- Syntax highlighting improvements
- Improved extension setting descriptions

.NET Interface

- Add WindX support for DEF OBJECT with arguments
- Resolved error when converting a PxPlus numeric array to a .NET array









Summary

Two features, one goal: less friction between you and the code.

VS Code now works against your server's PxPlus licence, navigates your programs, and compiles in place

The .NET Interface now discovers what is available, indexes it directly, converts arrays, and tells you what went wrong

Thank you

Questions?

Where to read more

Visual Studio Code Extension

How to Set Up PxPlus Visual Studio Code Extension

.NET DEF OBJECT - formats 4 and 5, and the extended properties and methods

.NET NOMADS External Control Properties

.NET Interface How To Tutorials



Devon Austen