

.NET Interface Enhancements

DireXions 2026

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

.NET

Developing the Future

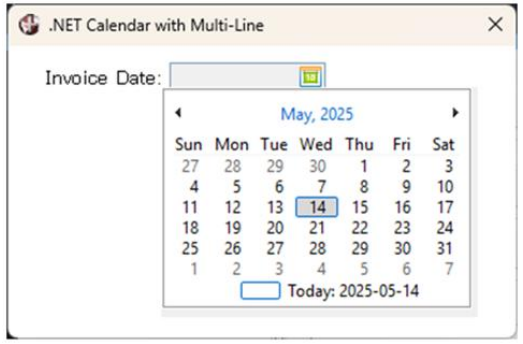
The .NET Interface, Recapped

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

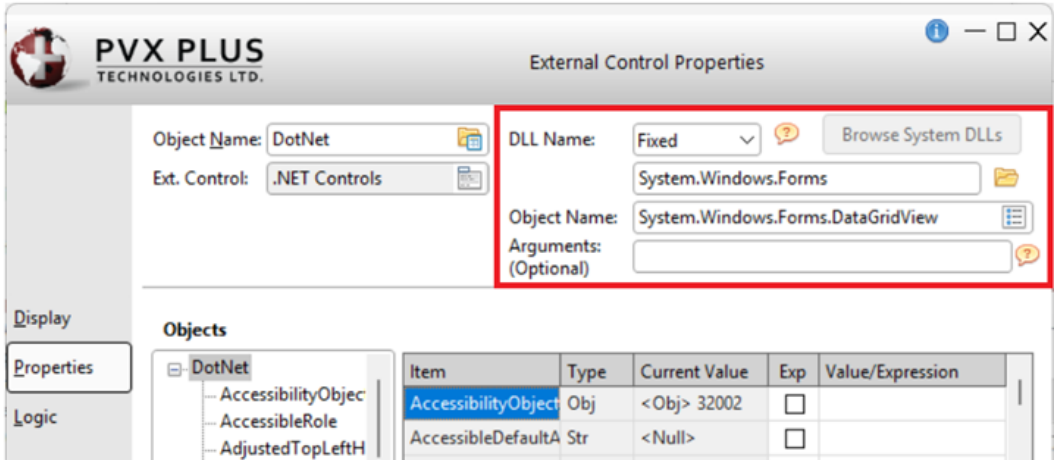
- Properties and methods through the apostrophe operator, or the NOMADS Properties tab
- Events through ON EVENT, or the NOMADS Events grid
- PvxHandle$ to pass one .NET object into another



Install

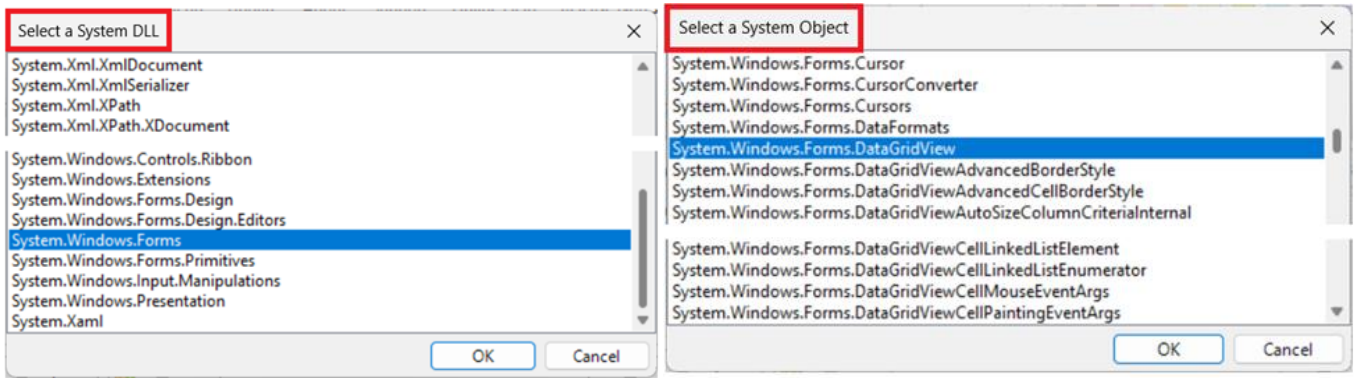
- Windows only, with **Windows Desktop Runtime 8** installed, even if 9 or 10 is already there
- Match the runtime to PxPlus - x64 or x86
- Under WindX, each client needs the runtime too

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

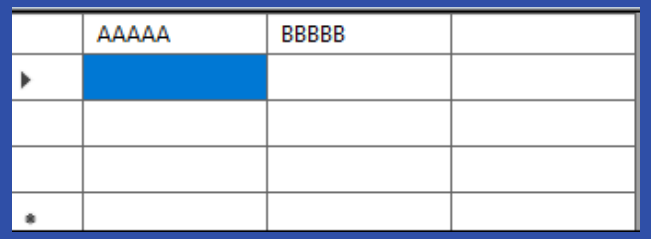
- Sets .NET object's element specified by index to the value. Equivalent to doing a `myArray[index]=value` in .NET

```
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

.NET Interface

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









Summary

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

.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