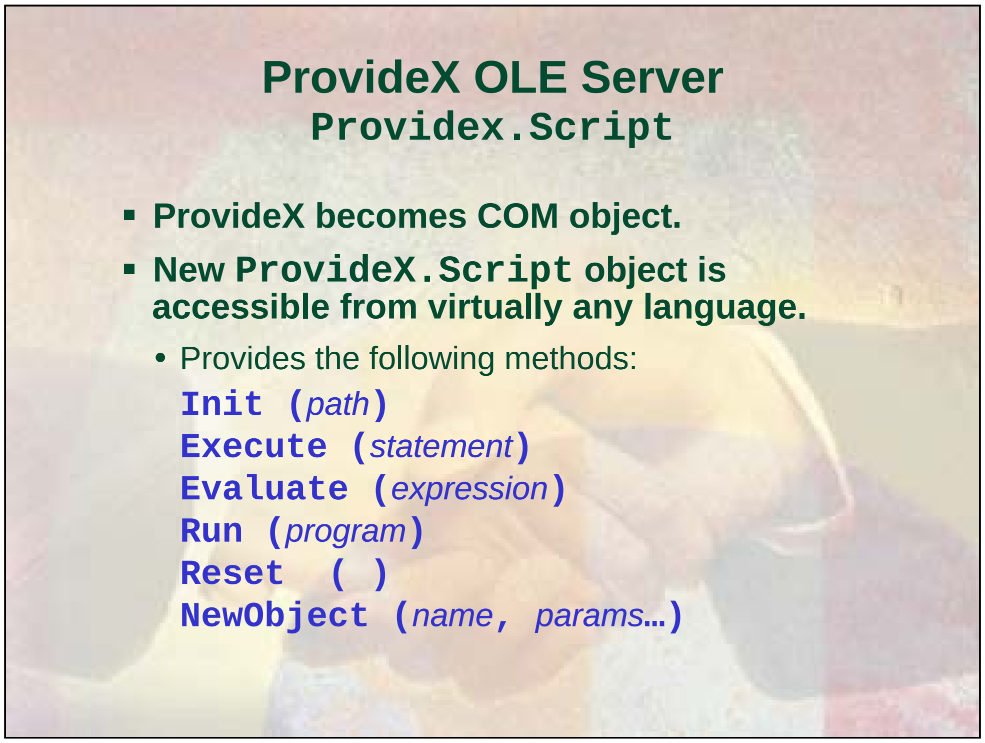


ProvideX OLE Server
Providex.Script

- **ProvideX becomes COM object.**
- **New `ProvideX.Script` object is accessible from virtually any language.**
  - Provides the following methods:

```
Init (path)
Execute (statement)
Evaluate (expression)
Run (program)
Reset  ( )
NewObject (name, params…)
```

The **`ProvideX.Script`** OLE server allows external applications to directly invoke and interact with ProvideX and ProvideX objects. The server can be used by virtually any COM-compliant application or programming language such as VB, Delphi, VB script, VBA.

Methods

The OLE server itself has 6 methods:

`Init (`*path***`)`**	Called before accessing any other method in the script engine in order to set the working directory for the server and perform binding to the ProvideX dll layer. The *path* should be set to the desired working directory, or blank to indicate the current directory. (In most cases, the current directory will be the system directory)
`Execute (`*statement***`)`**	Invoked to execute any ProvideX command.
`Evaluate (`*expression***`)`**	Evaluates and returns the expression provided and returns its value as a variant. (It can be used as either a numeric or string.)
`Run (`*program***`)`**	Runs the specified program. This method does not return until the program ends.
`Reset ( )`	Closes all **LOCAL** files and clears variables. (It executes a **BEGIN**)
`NewObject (`*name*, *params* … **`)`**	Creates a new object of the specified name and returns an object reference.

Continued…

ProvideX OLE Server (*Continued*)

Associated Properties

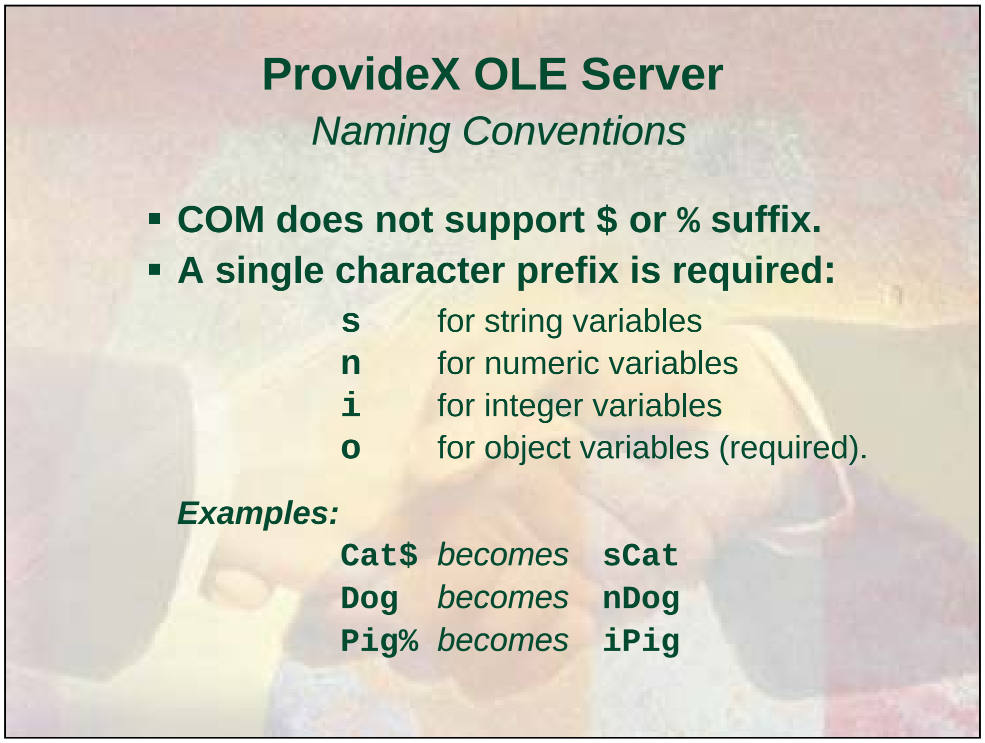
There are also three properties associated with the script interface:

`Instance`	Returns a unique string to identify the server during its lifetime.
`State`	Returns the state of the server. 0 (zero) is returned for un-initialized, and 1 (one) is returned for initialized.
`Parameter`	Read/write property used to access the specified ProvideX parameter.

Additional Properties

To create an object within the OLE Server, the method **`NewObject`** is used. It returns an object identifier to the ProvideX object that was created. Along with the ProvideX defined properties and methods, the OLE server adds the following properties:

`Instance`	Returns a unique identifier of the **`ProvideX.Script`** object that was used to create the automation object. The importance of this is due to the fact that the automation object cannot be passed to another **`ProvideX.Script`** server.
`ScriptObject`	Returns the parent **`ProvideX.Script`** object.
`CmdHandle`	Returns the ProvideX handle that the automation object is tied to.



ProvideX OLE Server

Naming Conventions

- **COM does not support $ or % suffix.**
- **A single character prefix is required:**

s	for string variables
n	for numeric variables
i	for integer variables
o	for object variables (required).

Examples:

`Cat$`	*becomes*	**`sCat`**
`Dog`	*becomes*	**`nDog`**
`Pig%`	*becomes*	**`iPig`**

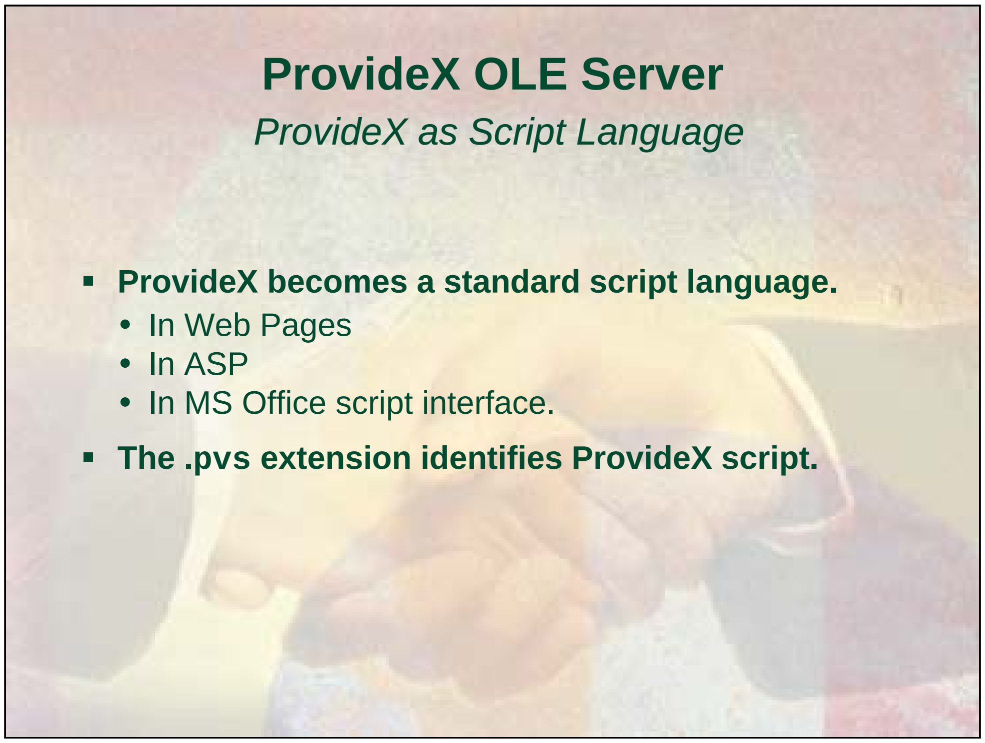
One thing you should be aware of is that COM is generally *data-type insensitive*. You can access a numeric value as either a string or a number. This causes a problem when dealing with the ProvideX OLE server, since in ProvideX you can have **`Cust_no$`** and **`Cust_no`** as two unique data variables.

To avoid this problem, the OLE server mandates that all property references indicate the property type in the *first character* of the property name. This means that all references to *string* properties must have an **`'s'`** followed by the property name; all *numeric* properties must have an **`'n'`** followed by the property name; and, all *integer* properties must have an **`'i'`** followed by the property name. If a property is itself a object identifier, the property name should have an **`'o'`**prefix.

The suffixes are only for use by programs using the OLE Server - not by the ProvideX application itself. This means that the property **`Cust_no$`** in ProvideX would be **`sCust_no`** when referenced by the OLE Server and the property **`Cust_no`** in ProvideX would be **`nCust_no`**.

DO NOT use the SUFFIX in the PROPERTY DEFINITION !

Use of the **`'o'`** prefix allows the specification of properties and/or methods within ProvideX objects to be declared themselves as objects.



ProvideX OLE Server

ProvideX as Script Language

- **ProvideX becomes a standard script language.**
  - In Web Pages
  - In ASP
  - In MS Office script interface.
- **The .pvs extension identifies ProvideX script.**

Through the use of the OLE server, ProvideX itself can become a script language for any Windows system. Any file with a **.pvs** suffix will be passed to the script engine to execute as a script. Web pages and IIS ASP servers can use the ProvideX script engine as well.

Script definitions vary slightly from regular ProvideX in that the text is basically passed to ProvideX as a series of execute commands. However, there is an exception: any code that starts with a statement label and ends with **END** will be considered a program and built as if it was typed in. After the **END** directive, you can then issue a **GOTO** ***label*;** **RUN** to begin execution of the code (The short cut is #***xxxxx*** where ***xxxx*** is the label).

Script-based programs do not support line numbers. When an error occurs in a script-based program, the line number reported will reflect the line number from the original script input file/document.