

C# Scripting Engine

SYSPRO 8

Reference Guide

Published: June 2025



CONTENTS

Power Tailoring C# Scripting

| | |
|-----------------|----|
| Exploring | 1 |
| Starting | 3 |
| Solving | 5 |
| Using | 11 |



Power Tailoring C# Scripting

Exploring

Where it fits in?

We have updated the **Scripting Engine** to create and execute scripts using C#, as **Microsoft** are deprecating VB Script from future versions of their operating systems. The introduction of C# scripting allows for the customisation, power tailoring and **Application Designer** program to use a modern language and use the power of the .Net framework when building and delivering bespoke requirements in SYSPRO. In all areas of SYSPRO where VBScript was previously supported, you now have the option to create and utilize C# scripts.

Benefits

- C# scripting aligns with modern standards, ensuring improved performance, security, and maintainability.
- Optimized, scalable C# functions that meet specific user needs enable tailored ERP solutions.
- C# ensures efficient, future-ready solutions and enhanced customer experiences.

Functionality

- Dual-language support:

During the transition phase, SYSPRO supports both languages when creating scripts.



While both, VB and C# scripting are supported, C# takes precedence, i.e. if a C# script exists, the VB script will be ignored.

- Integrated error handling and debugging:

Offers syntax checking, error previews, and Visual Studio integration for step-through debugging.

- Extensive customization

Supports script execution across various contexts (panes, workflows, eSignatures, tiles, etc.) with role-, system-, and user-level granularity.

- Reusable logic via references

Allows scripts to include shared .NET assemblies, promoting modularity and maintainability

Navigation

The programs related to this feature are accessed from the **Program List** of the SYSPRO menu:

- *Program List > Administration*
- *Program List > Quality Management*
- *Program List > SYSPRO Workflow Services*

Terminology

Script

In SYSPRO, a script is essentially a piece of code that enhances and manipulates the functionality of a SYSPRO program to meet a customer's needs

Although a script is interpreted and not compiled, SYSPRO uses the term *script* to refer to both VBScript and C# code. This means that even though C# is not traditionally considered a scripting language, it is treated as such within SYSPRO through the use of dynamic logic, which allows variables to be set at runtime.



While both, VB and C# scripting are supported, C# takes precedence, i.e. if a C# script exists, the VB script will be ignored.

Script type

In SYSPRO, a script type refers to the specific context or component within the system where a script is applied.

These types determine how and where scripts are stored, executed, and maintained, for example:

- Pane scripts are tied to individual forms or listviews within a program.
- Custom pane scripts are linked to user-defined interface elements.
- Workflow scripts automate actions within business processes.
- eSignature scripts validate transactions, etc.

Starting

Prerequisites

To use this feature, the following must be installed:

- .Net 8 x86 Runtime version 8.0.10



This must be installed on all on all client machines running SYSPRO as the C# scripting engine utilizes .NET Core 8 when saving, editing, deleting and executing, scripts.

Security

You can secure this feature by implementing a range of controls against the affected programs. Although not all these controls are applicable to each feature, they include the following:

- You restrict operator access to *activities* within a program using the **Operator Maintenance** program.
- You can restrict operator access to the *fields* within a program (configured using the **Operator Maintenance** program).
- You can restrict operator access to *functions* within a program using passwords (configured using the **Password Definition** program). When defined, the password must be entered before you can access the function.
- You can restrict access to the eSignature *transactions* within a program at operator, group, role or company level (configured using the **Electronic Signature Configuration Setup** program). Electronic Signatures provide security access, transaction logging and event triggering that gives you greater control over your system changes.
- You can restrict operator access to *programs* by assigning them to groups and applying access control against the group (configured using the **Operator Groups** program).
- You can restrict operator access to *programs* by assigning them to roles and applying access control against the role (configured using the **Role Management** program).

Restrictions and Limits

- No C# Reference Assemblies

We don't support creating C# reference assemblies. These must be created using Visual Studio and compiled to x86, as the SYSPRO client-side runs in this configuration.

- Script Variables and Instances



- C# compilations are treated as scripts, meaning variables and instances are cleared on each new run. For global variables, use `SystemVariables.GlobalVariable1-4` or the `SET-VARIABLE` and `GET-VARIABLE` callout functions.
- Script Language Changes

The existing script will be overwritten when changing the scripting language of tiles, flowgraphs, and **Application Designer** scripts.



The old script (i.e. the script in the previous language) is only available while the script editor is still open. Once the script editor is closed, the content of the old script will be lost.

Solving

FAQs

Working with references

What are script references in SYSPRO?

The C# programming language supports the use of references, which becomes especially useful when an assembly is utilized for functions required by multiple scripts. Script references enable common code to be written and compiled into a .dll assembly file, allowing other projects to access and use this shared logic. This facilitates multiple programs to leverage common functionality efficiently.

References have two key points of information:

- Namespaces

This is a list of namespaces provided within the assembly reference and each namespace is vital to accessing certain pieces of code within the assembly.

FOR EXAMPLE:

Double click on the namespace in the **References** Listview to add a `using` statement for the namespace.

- References

These sub-references under the parent assembly (also known as dependencies) are sometimes vital to the methods of the parent. Whilst not always required, it can be beneficial to add these into the reference list.

Which types of references are available?

The following two types of references offered within SYSPRO scripting:

Local references:

- These are references that are accessible within a single script instance, i.e. per executing script.
- There can be a maximum of 20 of these per script.
- These are saved in the [AdmScriptDetail](#) table per record.

Global references:

- These are references that are accessible across any and all script instances.
- These are saved in the [AdmScriptReference](#) table.

How can I add a reference?

Follow these steps to add a reference:

1. In the Script window, select **C#** at the **Language** tool.
This ensures that you are working with a C# script.
2. Click on **View** and select **Show Project References Window**.
The **Project References** pane is displayed.
3. Select **New**.
4. Browse and select the dll you need.
The dll is loaded into the listview.
5. You now have the option to save the reference as a global assembly. This will store it so that it is available to all scripts.
When saving, the assemblies are validated, marked as `Saved to database` and information is populated regarding their namespaces and sub-references.
6. You can double click on a namespace to insert it at the top of your code.

What are common references?

SYSPRO provides the following common references to each script. These are displayed in a blue color in the **Project References** listview.

- SYSPROSA_ScriptCommon.dll
Provides common functionality to all scripts
- SYSPROSA_ScriptingRuntimeProcess.dll
Required by SYSPROSA_ScriptCommon.dll
- Microsoft.Data.SqlClient.dll
Provides SQL functionality to scripts
- Microsoft.Identity.Client.dll
Required by Microsoft.Data.SqlClient.dll
- System.Security.Permissions.dll
Required by Microsoft.Data.SqlClient.dll
- System.Management.dll
Provides Win32 functionality to scripts and is required by the diagnostics script (IMPMENYY).

Why can't I see my reference namespaces?

References often contain dependencies to other assemblies. In order to access the namespaces of the parent assembly, it is often required to add the dependencies as well. This ensures that the assembly is loaded correctly so that the namespaces can be accessed.

You can try adding a few of the key dependencies onto the reference list.

Differences between C# and VB scripts in SYSPRO

What are the syntax differences between C# and VBScript?

The main syntax differences between C# and VBScript in SYSPRO scripting are:

- Case Sensitivity

VBScript is case-insensitive, meaning variables, functions, and routines can be written in any case and referenced in a different case.

C# is case-sensitive, so functions, variables, and routines must be written and referenced in the same case.

- Return Types

In VBScript, certain functions can return values to alter their effects, such as returning **false** to cancel a change. While this concept remains in C#, the return syntax differs.

FOR EXAMPLE:

A function in C# might use a Boolean return type to allow or disallow changes based on specific conditions.

What are the variable differences between C# and VBScript?

In VBScript, variables are often appended with the **CodeObject** instance to access runtime variables, whereas in C#, Variables are accessed directly without the **CodeObject** instance.

This table shows the differences:

| Script caption/type | Code injection (VBScript) | Code injection (C#) |
|----------------------|------------------------------------------|-------------------------------------|
| Any panes fields | {PaneName}.CodeObject. {FieldName} | Panes.{Pane name}. {FieldName} |
| Listview (array in) | {Listview name}.CodeObject.Array | Panes.{Listview name}.Array |
| Listview (array out) | {Listview name}_ OUT.CodeObject.Array | Panes.{Listview name}_ OUT.Array |

| Script caption/type | Code injection (VBScript) | Code injection (C#) |
|------------------------|--------------------------------------------------|---------------------------------------|
| CustomPane variables | CustomizedPane.CodeObject. {VariableName} | CustomizedPane. {VariableName} |
| CustomPanels variables | CustomizedPanels.CodeObject. {CustomPaneName} | CustomizedPanels. {CustomPaneName} |
| eSignature variables | eSignature.CodeObject. {VariableName} | eSignatures.{VariableName} |
| Flowgraph variables | Flowgraph.CodeObject. {VariableName} | Flowgraph.{VariableName} |
| Workflow variables | WorkflowVariables.CodeObject. {VariableName} | WorkflowVariables. {VariableName} |
| Tile variables | Tile.CodeObject.{VariableName} | Tile.{VariableName} |
| System variables | SystemVariables.CodeObject. {VariableName} | SystemVariables. {VariableName} |

What are the array differences between C# and VBScript?

Arrays in C# scripting work differently from standard .NET arrays and VBScript arrays.

- VBScript arrays were indexed by column and then row, which has been changed in C# to row and then column indexing.
- In C#, arrays are accessible within a class called **TwoDimensionalArray** with the following methods and/or accessors:

| Accessor | Method |
|------------------------------------------------|-------------------------------------------|
| Getter: Array[row, col] | var myCell = CustomizedPane.Array[0,0]; |
| Setter: Array[row,col] = {value} | Panes.Movements.Array[0,1] = "New value"; |
| Upper bound: Array.GetUpperBound(dimension) | CustomizedPane.Array.GetUpperBound(0); |
| Lower bound: Array.GetLowerBound(dimension) | Panes.Movements.Array.GetLowerBound(0); |

| Accessor | Method |
|--------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Get .NET 2d array: Array.GetBaseArray() | Panes.Movements.GetBaseArray(); Returns object[,] of list view values. This can allow you to use standard array routines not accessible under this class. |

What are the code setup differences between C# and VBScript?

The code setup for C# requires a namespace, class name, and inheritance, which are set up by default when creating a new script and can't be changed.

This ensures that the script adheres to the required C# syntax and can be executed without errors.

Script location, conditions and execution hierarchy

Which conditions do I need to comply with when creating C# scripts?

Take note of the following when creating C# scripts using the **VBScript Editor**:

- The default code structure provided by the syntax editor must be left unchanged.
- Any new functions added must be placed within the innermost curly braces {} of the class structure.
- The script must adhere to C# syntax, and any errors will be displayed in the **Errors and warnings** listview.
- Code injection must be followed correctly by double-clicking on an item in the **Variables** listview.

What is the execution hierarchy of C# scripts in SYSPRO?

C# scripts follow the same execution hierarchy as VBscripts, which is as follows:

1. Role script
2. Industry script
3. System-wide script
4. Company script
5. Operator script
6. Global script

Where are scripts stored in the database?

The following system-wide database tables store script information:

The [AdmScriptDetail](#) table stores the base-64 encoded contents of scripts.

The [AdmScriptReference](#) store the base-64 encoded contents of .dll assemblies.

Script size and language

How do you change the script language?

1. Navigate to the [VBScript Editor](#).
2. Select the script language from the **Language** control on the toolbar.

What is the size limit for C# and VB scripts?

The VBScript limit of 200,000 characters has been removed for C# scripts. However, we recommend not to exceed the original limit, to ensure that results are responsive and efficient.

Using

Script Information

Script Types

SYSPRO supports scripting in multiple areas, which can be referred to as *script types*:

| Script Type | Description |
|----------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Pane scripts (form, listview) | <p>These scripts are saved per pane in a SYSPRO program.</p> <p>FOR EXAMPLE:</p> <p>The program Customer Query has a form and a listview. Both are deemed panes and can have their own scripts.</p> |
| Custom pane scripts (for all custom pane types) | These are custom panes scripts stored per custom pane ID. |
| eSignature scripts (non-e.net only) | These scripts are saved per eSignature transaction ID. |
| Workflow scripts | <p>Workflows allow actions to be executed, which are created through a SYSPRO program. These actions can execute a script.</p> <p>Scripts are stored per workflow action IDd and are prefixed by an underscore, e.g. '_'.</p> |
| Application Designer custom scripts | <p>A SYSPRO program created using the Application Designer has the capability to create and save its own VB scripts.</p> <p>VB scripts, specifically, are stored by the calling program in the location of its choice. These scripts can then be executed when the program owner chooses.</p> |
| Tile scripts | Tiles can contain scripts against them. |
| Flowgraph scripts | Flowgraph scripts can exist per flowgraph with the ability to set and retrieve flowgraph information. |
| Application Designer application scripts | The main source code for an Application Designer application sits within its script, editable through Application Designer itself. |

Script Saving Levels

Scripts are saved and executed at certain levels depending on the login.



It is important to note the differences to understand the contexts of script saving.

| Save Level | Description | VBScript Location Example | C# Script Location Example |
|-------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------|
| Global | These are scripts saved usually when an operator edits a script and is not in design mode (although, there are exceptions). Usually stored in the <code>\Work\VBScripts</code> folder. | Work\ vbscripts\ IMPMENYY | {System-wide db}..AdmScriptDetails LevelId = 20 LevelValue = "GLOBAL" ApplicationType = "SA" ApplicationName = "IMPMENYY" |
| Operator | These are scripts stored per operator. Usually, these are only customized panes which have been created when not in design mode, and are stored in the <code>\Base\Settings</code> folder. | Base\ settings\ Pane_ADMIN_ INVPUY00 | {System-wide db}..AdmScriptDetails LevelId = 9 LevelValue = "Cameron" ApplicationType = "CU" ApplicationName = "INVPUY00" |
| System-wide | Stored when editing by system-wide in the <code>\Base\Settings\Role_SYS</code> folder. | Base\ settings\ Role_SYS\ INVPUYLV | {System-wide db}..AdmScriptDetails LevelId = 1 LevelValue = "SYS" ApplicationType = "SA" ApplicationName = "INVPUYLV" |
| Industry | Stored when editing by system-wide in the <code>\Base\Settings\Role_AAA</code> folder. Industry IDs are alphabetic. | Base\ settings\ Role_IND\ INVPUYLV | {System-wide db}..AdmScriptDetails LevelId = 5 LevelValue = "IND" ApplicationType = "SA" ApplicationName = "INVPUYLV" |

| Save Level | Description | VBScript Location Example | C# Script Location Example |
|------------|--------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------|
| Role | Stored when editing by role in the \Base\Settings\Role_### folder. Role IDs are numeric. | Base\ settings\ Role_031\ INVPUYLV | {System-wide db}..AdmScriptDetails LevelId = 7 LevelValue = "031" ApplicationType = "SA" ApplicationName = "INVPUYLV" |
| Company | Stored when editing scripts that affect company data only. Usually, this only applies to Application Designer custom scripts. | DS001_CMP_ EDU1_800 ..SqmTestHeader .ScriptContents | {System-wide db}..AdmScriptDetails LevelId = 8 LevelValue = "EDU1" ApplicationType = "CS" ApplicationName = "INSPECTION" |
| File | Stored at file-level. These are dependant on circumstance, e.g. Application Designer applications. | Stored within a file itself. | Stored within a file itself. |

Script Locations

Scripts are stored in different locations based on the language and script type.

| Script Type | VB Script Location | C# Location |
|-------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Pane | <p>For global scripts:</p> <p>(Server)</p> <pre>\Work\VBScripts</pre> <p>(Client)</p> <pre>\Base\Settings\VBScripts\{PaneId}</pre> <p>For role/industry/system-wide scripts:</p> <p>(Server and Client)</p> <pre>\Base\Settings\Role_XXX\{PaneId}</pre> <p>where xxx is:</p> <ul style="list-style-type: none"> ■ Industry code for industries ■ <code>sys</code> for system-wide ■ Role code for roles | <p>System-wide database:</p> <p>AdmScriptDetail</p> <p>These are stored per pane ID, i.e. each pane will be its own record in the table.</p> |
| Custom pane | <p>For operators:</p> <pre>\Base\Settings\{OperatorCode}_VBS_{PaneId}</pre> <p>For role/industry/system-wide:</p> <p>(Server and Client)</p> <pre>\Base\Settings\Role_XXX\{PaneId} }</pre> <p>where xxx is:</p> <ul style="list-style-type: none"> ■ Industry code for industries ■ <code>sys</code> for system-wide ■ Role code for roles | <p>Same as pane</p> <p>These are stored per custom pane, i.e. each custom pane will be its own record in the table.</p> <p>Unless designing by role, system-wide, or industry, the script will be stored as a <code>GLOBAL</code> script.</p> |

| Script Type | VB Script Location | C# Location |
|-----------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------|
| eSignature | Same as pane | Same as pane These are stored per transaction ID, i.e. each transaction will be its own record in the table. |
| Workflow | Same as pane | Same as pane These are stored per action ID, i.e. each action will be its own record in the table. |
| Application Designer custom scripts | Stored in the calling program's database table, e.g. SqmTestHeader stores the vbscript in it's own <code>Script code</code> column. | Same as pane These are stored according to the table key of the calling program in order to uniquely identify scripts. |
| Tile | These are stored on the local disk in the <code>Task_{Operator}_{PaneId}_{UniqueFileId}.Xml</code> file and saved to the server if required. | |
| Flowgraph | These are stored in the xml file stored on the local disk and saved to the server if required. | |
| Application Designer application | These are stored in the <code>{ProgramName}_AppDesigner.txt</code> file. | |

Affected programs

The following indicates areas in the product that may be affected by implementing this feature:

Setup programs

Scripting Engine

This program handles all C# scripting logic. It was developed with an API-based infrastructure and calls to the C# scripting engine both server and client-side.

Customization Management

SYSPRO Ribbon bar > Administration > Customization

This program lets you view and manage the various customization layouts that have been applied to operator roles within SYSPRO.

For the **Scripting Engine** enhancement, we made the following changes:

- All references to **VBScript** were changed to **Script**.
- You can copy, delete, export, and import C# scripts.
- C# and VBScripts are displayed within listviews.

Customization Profiler

*Accessible from any docking pane caption or menu pull-down (right-click the title bar of a pane and select **Customization Profiler** from the context menu).*

This program lets you view and analyze customizations that have been applied to a program or application within SYSPRO, including the main menu.

For the **Scripting Engine** enhancement, we made the following changes:

- A new **Script** section displays VBScripts and C# script information.



C# scripts are only displayed if available to the current login.

Electronic Signature Configuration Setup

Program List > Administration > Electronic Signatures

This program lets you enable the **Electronic Signatures** system and create or maintain your eSignature configuration levels and their associated access control.

For the **Scripting Engine** enhancement, we made the following changes:

- We added eSignatures for adding, deleting, copying, importing and exporting C# scripts.

Customized Pane Editor

*Accessible by right-clicking the heading of a docking pane and selecting **New** or **Properties** from the **Customized Pane** option.*

You use this program to create your own views in SYSPRO using various graphical components such as graphs, listviews and web applications.

For the **Scripting Engine** enhancement, we made the following changes:

- All references to **VBScript** were changed to **Script**.
- The **Language** option was added on the toolbar so the script can be saved in VBScript and C#.
- We removed the **VBScript file name** field to avoid confusion.

Workflow Action Maintenance

Program List > SYSPRO Workflow Services

This program lets you add actions that can be used in a workflow. These actions are used to create a hyperlink against a task in the **To-Do List** program.

For the **Scripting Engine** enhancement, we made the following changes:

- All references to **VBScript** were changed to **Script**.

Inspection Test Design

Program List > Quality Management > Setup

This program lets you design your own inspection tests that are used in the **Quality Management** module when inspecting work in progress (WIP) and purchase order receipts.

For the **Scripting Engine** enhancement, we made the following changes:

- All references to **VBScript** were changed to **Script**.
- We added the functionality to support C# scripts.

Batch programs

Workflow Services VBScript

Program List > SYSPRO Workflow Services

This program lets you generate workflow scripts that the SYSPRO client uses to communicate with the deployed workflow processes. You can start a workflow process as well as move a workflow process to different states using simple C# or VBScript syntax.

For the **Scripting Engine** enhancement, we made the following changes:

- We added the functionality to support C# scripts.

Inspection Test Review

Program List > Quality Management > Processing

This program displays all inspection tests that were designed using the **Inspection Test Design¹** program.

For the **Scripting Engine** enhancement, we made the following changes:

- All references to **VBScript** were changed to **Script**.
- We added the functionality to support C# scripts.

Inspection Test Queue

Program List > Quality Management > Processing

This program displays all sample tests that were created using the **Inspection Test Design²** program and allows inspectors to update the tests assigned to them.

For the **Scripting Engine** enhancement, we made the following changes:

- All references to **VBScript** were changed to **Script**.
- We added the functionality to support C# scripts.

¹Program name: SQMFRM

²Program name: SQMFRM

Utility programs

VBScript Editor

Accessible by right clicking and selecting the **Macro for:** option within forms, panes and toolbars in SYSPRO.

Alternatively, you can select the **Edit VBScript** function from the **Customized Pane Editor** program.

You use this program to maintain C# and VBScripts. These are simple text files that can be edited using the SYSPRO **Script Editor** program or any text editor.

For the **Scripting Engine** enhancement, we made the following changes:

- All references to **VBScript** were changed to **Script**.
- The **Language** option was added on the toolbar so the script can be saved in VBScript and C#.
- Added a listview that displays compilation errors.
- Added a listview that displays local and global references.
- Added the **Project References** pane where you can add a reference to the script.
- Added the **Errors and Warnings** pane.
- Added these option for C# scripts:
 - Save for debug
 - Prettify syntax
 - Comment and uncomment code

Affected business objects

The following indicates the business objects that are affected by this feature:

Transaction objects

Scripting Engine Transaction

The **SCRIPTING ENGINE TRANSACTION**¹ business object lets you save and delete scripts and references in the [AdmScriptDetail](#) and [AdmScriptReference](#) tables.

Query objects

Scripting Engine Query

The **SCRIPTING ENGINE QUERY**² business object lets you query details in the [AdmScriptDetail](#) and [AdmScriptReference](#) tables.

It is used for the caching of scripts on the startup of SYSPRO, and handles the logic to retrieve the details about scripts to edit global and local references.

¹Business object: COMTSE

²Business object: COMQSE



www.syspro.com

Copyright © SYSPRO. All rights reserved.
All brand and product names are trademarks or
registered trademarks of their respective holders.