Scripts Module Reference

Extend solutions with Python, C# and VB.NET logic.

Reference Scripts | Tutorial | How-to Guide | Reference

Scripts Module (Reference) provides automation and business logic capabilities through classes, tasks, expressions, and integrated development tools. It enables:

- **Reusable code libraries** via Script Classes
- **Event-driven automation** through Script Tasks
- **Lightweight calculations** using Script Expressions
- **External integration** with Script References
- **Development productivity** with integrated Code Editor

Scripts execute primarily server-side for web compatibility, with optional client-side execution for WPF-only deployments.

Module Components
Runtime Architecture
Key Concepts
Best Practices
Troubleshooting
Common Issues
Scripts Module Links

Module Components

Configuration Sections

Section	Path	Purpose
Classes	Scripts Classes	Reusable methods and functions
Tasks	Scripts Tasks	Event-driven automation logic
Expressions	Scripts Expressions	Simple calculations and method calls
References	Scripts References	External DLL integration
Code Editor	Scripts Code Editor	Integrated development environment

Configuration Workflow

Action	Where	Purpose
Create Classes	Scripts Classes	Build reusable function libraries
Define Tasks	Scripts Tasks	Implement automation logic
Add Expressions	Scripts Expressions	Configure simple calculations
Import References	Scripts References	Add external assemblies
Edit Code	Scripts Code Editor	Write and debug scripts

Runtime Architecture

Execution Context

Default Configuration:

- **Server-side**: Tasks and expressions run on TServer.exe
- **Client-side**: Only Display Code Behind executes locally
- **Script Classes**: Available to both server and client

Web Compatibility:

- Client-side tasks disabled by default for HTML5 support
- Enable in Solution Settings for WPF-only deployments

Script Namespace

Access script objects at runtime:

csharp

```
// Access Script Classes
var result = @Script.Class.ClassName.Method();

// Monitor Script Tasks
int count = @Script.Task.TaskName.ExecutionCount;
TimeSpan cpu = @Script.Task.TaskName.LastCPUTime;
```

Performance Considerations

For complex or large solutions requiring optimization: See [Script Engine (Reference)] for deep technical details

---

Key Concepts

Script Classes

- Repository of reusable methods
- Automatic static instantiation
- Cross-language support (C#/VB.NET/Python)
- Three types: Methods, MCP Tools, Full Namespace

[Script Classes (Reference)]

Script Tasks

- Event or time-triggered execution
- Independent threading model
- Automatic exception protection
- Built-in startup/shutdown tasks

[Script Tasks (Reference)]

Script Expressions

- Single-line VB.NET statements
- Automatic change detection
- Ideal for simple calculations
- Reduces need for one-line tasks

[Script Expressions (Reference)]

Script References

- External DLL integration
- Framework compatibility management
- Namespace declaration system
- Path macro support

[Script References (Reference)]

Code Editor

- Multi-language support
- IntelliSense with solution objects
- Integrated debugging
- Language conversion

[Code Editor (Reference)]

Best Practices

Code Organization

- **Use Classes** for reusable logic
- **Use Tasks** for complex automation
- **Use Expressions** for simple calculations
- **Name consistently** with descriptive identifiers

Performance

- **Server-side by default** for web compatibility
- **Monitor execution** via diagnostic properties
- **Build before deployment** to ensure compilation
- **Profile large solutions** using Script Engine diagnostics

Development

- **Format code** for readability
- **Handle exceptions** even with automatic protection
- **Test incrementally** with build verification
- **Document purpose** in descriptions

Library Management

- **Use References** for external DLLs
- **Manage namespaces** via declaration dialog
- **Version control** external dependencies
- **Test compatibility** with target framework

Troubleshooting

Build Issues

Build Errors:

- Check BuildStatus property
- Review BuildErrors details
- Verify referenced objects exist
- Ensure correct language syntax

Circular Dependencies:

- Review build order
- Temporarily comment references
- Restructure class dependencies

Execution Problems

Tasks Not Running:

- Verify InitialState enabled
- Check trigger configuration
- Review domain settings
- Monitor execution count

Performance Issues:

- Check LastCPUTime metrics
- Review execution frequency
- Optimize synchronization
- Consider pre-loading strategy

Reference Issues

DLL Not Loading:

- Verify .NET framework compatibility
- Check path macros
- Restart after DLL updates
- Review resolved status

Namespace Conflicts:

- Use declaration dialog
- Check default namespaces
- Verify reference order

Debugging

Breakpoints Not Hit:

- Enable debug information

- Attach debugger to process
- Verify code has compiled
- Check execution path

Variable Inspection:

- Use @ prefix for solution objects
- PropertyWatch for real-time values
- Watch window during breaks

Common Issues

Language Conversion

- C# to VB.NET usually succeeds
- VB.NET to C# may need type casting fixes
- One-time migration recommended

Client-Side Limitations

- Disabled by default for web
- Only WPF clients support
- Server-side preferred approach

Incremental Compilation

- Automatic background process
- Cache invalidation on changes
- Full build before production

Scripts Module Links

Explanation - to understand concepts

Modules / Business Operations / Scripts Module

Tutorials - to learn by doing

Tutorials / Business Operations / Scripts Module Tutorial

How-to Guides - to accomplish specific tasks

How-to Guides / Business Operations / Scripts Module How-to Guide

Reference - technical details

Technical Reference / Business Operations / Scripts Module Reference

---

In this section...
