Scripts Expressions Reference

Use inline expressions in your logic.

Scripts Expressions(Reference) provide lightweight, event-driven automation for simple calculations and method calls, ideal for scenarios with many triggers but straightforward logic. A Script Expression is single-line statements that:

- Execute on events or schedules without creating full tasks
- Run in multi-threaded environment for optimal performance
- Automatically detect tag changes in the expression
- Support both arithmetic operations and Script Class calls

Use Expressions when you need many simple automations that would otherwise require numerous one-line tasks. The parse of the expression is VB.NET syntax, but it automatically converts and accepts C# syntax also.

Creating Script Expressions
Configuration Properties
Execution Modes
Expression Syntax
Conditional Evaluation Methods
Threading and Performance
Domain Execution
Examples
Best Practices
Troubleshooting

Creating Script Expressions

1. Navigate to **Scripts** **Expressions**
2. Select insert row (first blank row)
3. Configure:
    - **ObjectName**: Target tag to receive result
    - **Expression**: VB.NET statement or calculation
    - **Execution**: Trigger method

Configuration Properties

Property	Description	Required
Object	Tag or object that receives the expression result	Yes
Expression	VB.NET statement (arithmetic, conditional, or method call)	Yes
Execution	When expression runs (see Execution Modes)	Yes
Trigger	Tag/object that triggers execution (optional)	No
DisableCondition	Tag/object that disables execution when true	No
Time	Specific time for time-based execution	No
Label	User-defined label for organization	No
BuildStatus	? Green = Success, ? Red = Errors (read only)	No
BuildErrors	Compilation error details (read only)	No
BuildMessages	Status messages after verification (read only)	No

Execution Modes

Mode	Description	Use Case
OnChange	Runs when any tag in expression changes	Real-time calculations
TriggerOrTimeInterval	Runs on trigger change OR time interval	Periodic updates with override
ChangeOrStartup	Runs on tag change OR at startup	Initialize and maintain values
TriggerOrTimeOfDay	Runs on trigger OR specific time	Scheduled operations with manual trigger

Expression Syntax

Key Syntax Rules

<ac:structured-macro ac:name="info"> ac:rich-text-body **Automatic Conversions**: The platform automatically converts C#-style operators to VB.NET:

- == becomes =
- No semicolon needed at the end
- No @ prefix needed for tags (unlike Script Tasks) </ac:rich-text-body> </ac:structured-macro>

Direct Tag Access

vbnet

```vbnet
' No @ symbol needed in expressions
Tag.Temperature * 1.8 + 32  ' Convert C to F
Tag.Output = Tag.Input1 + Tag.Input2
```

Calling Script Classes

vbnet

```vbnet
' Call methods from Script Classes
Script.Class.Calculations.GetAverage(Tag.Value1, Tag.Value2)
Script.Class.DataProcessor.Validate(Tag.RawData)
```

Arithmetic Operations

vbnet

```vbnet
' Standard .NET operators
(Tag.Flow * Tag.Density) / 1000
Math.Round(Tag.Value, 2)
Math.Max(Tag.Limit, Tag.Current)
```

Conditional Evaluation Methods

IIF Method (Immediate If)

Evaluates all parameters regardless of condition:

vbnet

```vbnet
IIF(Tag.Status = 1, "Running", "Stopped")
' Both "Running" and "Stopped" are evaluated

IIF(Tag.Level > 80, Script.Class.Alerts.High(), Script.Class.Alerts.Normal())
' WARNING: Both High() and Normal() methods execute!
```

TIF Method (True If) - Recommended

Evaluates only the selected branch:

vbnet

```vbnet
TIF(Tag.Status = 1, "Running", "Stopped")
' Only evaluates the matching result

TIF(Tag.Level > 80, Script.Class.Alerts.High(), Script.Class.Alerts.Normal())
' Only executes High() OR Normal(), not both
```

<ac:structured-macro ac:name="warning"> ac:rich-text-body **Important**: Use TIF instead of IIF when calling methods to avoid unnecessary execution. </ac:rich-text-body> </ac:structured-macro>

Threading and Performance

Multi-Threading

- Each expression executes in thread pool
- Hundreds/thousands of expressions don't block each other
- Automatic thread management
- No manual synchronization needed

Compilation

- Automatic compilation before runtime
- Manual verification available via button
- All expressions compile together
- No need to compile after each change

Domain Execution

<ac:structured-macro ac:name="info"> ac:rich-text-body **Default Setting**: Client-side expressions are disabled by default for web compatibility. Enable in Solution Settings if needed. </ac:rich-text-body> </ac:structured-macro>

Server Domain (Default)

- Executes on server (TServer.exe)
- Global scope
- Typical use: Calculations, business logic

Client Domain

- Executes on each client
- Local scope
- Only for WPF clients (not web)
- Typical use: UI calculations, local formatting

Examples

Simple Calculations

vbnet

```vbnet
' Temperature conversion
Tag.TempF = Tag.TempC * 1.8 + 32

' Percentage calculation
Tag.Efficiency = (Tag.Actual / Tag.Target) * 100

' Conditional assignment
Tag.Status = TIF(Tag.Value > 100, 2, 1)
```

Time-Based Operations

vbnet

```vbnet
' Reset counter at midnight
Tag.DailyCount = 0  ' Execution: TriggerOrTimeOfDay, Time: 00:00

' Calculate hourly average
Tag.HourlyAvg = Script.Class.Statistics.GetHourlyAverage()
```

Complex Logic via Classes

vbnet

```vbnet
' Call complex processing in Script Class
Script.Class.Production.CalculateOEE()

' Conditional processing
TIF(Tag.AlarmActive, Script.Class.Alerts.Send(), 0)
```

Best Practices

- [ ] **Use for simple logic** - Complex code belongs in Script Tasks or Classes
- [ ] **Prefer TIF over IIF** - Avoid unnecessary method execution
- [ ] **Group related expressions** - Use Label field for organization
- [ ] **Monitor performance** - Many OnChange expressions can impact performance
- [ ]

Test with Verify - Use button to check syntax before runtime

- [ ] **Consider execution mode** - Choose appropriate trigger method
- [ ] **Document purpose** - Use Label field to explain expression

---

Troubleshooting

Expression not executing:

- Check BuildStatus for compilation errors
- Verify DisableCondition is not active
- Confirm Domain matches deployment
- Check trigger tag is changing

Performance issues:

- Review OnChange expressions for high-frequency tags
- Consider time-based instead of change-based
- Move complex logic to Script Classes
- Check for IIF with method calls

Compilation errors:

- Click Verify button to see errors
- Check VB.NET syntax (not C#)
- Verify referenced tags exist
- Ensure Script Classes are compiled

Unexpected results:

- Remember IIF evaluates all branches
- Check automatic operator conversion
- Verify tag data types match
- Test with simpler expression first

In this section...
