Forms Editor Reference

Build report form using our editor.

Reference **Modules** **Reports** **UI** Forms Editor | Forms | Monitor | WebData | WebData Editor

---

Forms Editor (Reference) provides a comprehensive design environment for creating reports with dynamic content, tables, symbols, and trends. The Forms Editor enables:

- WYSIWYG report design
- Dynamic tag integration
- Table creation (static and dynamic)
- Symbol and trend insertion
- Multi-format export
- Localization support

Editor Interface
Adding Dynamic Content
Tables
Symbols and Graphics
Localization
Headers and Footers
Special Features
Export Formats
Best Practices
Troubleshooting

Access via **Reports** **Forms Editor**

Editor Interface

Toolbar Components

- **Font Controls** - Type, size, style
- **Paragraph** - Alignment, spacing, indentation
- **Insert** - Tags, symbols, tables, images
- **Format** - Colors, borders, backgrounds
- **Tools** - Preview, print, export

Text Area

Main editing canvas with:

- Real-time preview
- Drag-and-drop support
- Context menus
- Grid alignment

Adding Dynamic Content

Tags

Insert real-time tag values:

Method 1: Direct typing

```
{Tag.TagName}
{Tag.TankFarm/Tank1/Level}
{Tag.Production/Production_Count}
```

Method 2: Tag Browser

1. Right-click toolbar **Tag** button
2. Browse and select tag
3. Tag inserted with formatting

System Tags

```
{Info.Date}          // Current date
{Info.Time}          // Current time
{Client.UserName}    // Logged user
{Page.Number}        // Page number
{Page.Total}         // Total pages
```

Tables

Static Tables

Fixed structure with manual content:

1. Click **Table** button
2. Specify rows and columns
3. Configure appearance:
    - Borders and shading
    - Cell alignment
    - Column widths
4. Populate cells manually or with tags

Example:

```
| Parameter | Value | Status |
|-----------|-------|--------|
| Level | {Tag.Tank1_Level} | {Tag.Level_Status} |
| Temp | {Tag.Tank1_Temp} | Normal |
```

Dynamic Tables

Query-driven content:

1. Insert query reference
2. Map columns in first two rows:
    - Row 1: Display names
    - Row 2: Database columns

Example:

```
| Product Name | Quantity | Price |
| ProductName | Qty | UnitPrice |
{Dataset.Query.ProductList}
```

The table auto-populates from query results.

Symbols and Graphics

Adding Symbols

1. Click **Symbol** button
2. Select from library
3. Symbol updates dynamically

Creating Trend Symbols

1. Right-click trend in display
2. Select **Make New Symbol**
3. Insert via Symbol button
4. Trend updates with latest data

Localization

Enable Translation

Set dictionary before generation:

csharp

```
@Client.Localization = "Spanish";  // Use Spanish dictionary
@Client.Localization = "";         // Use default
```

Translatable Elements

- Tag descriptions
- Table headers
- Static text with locale keys
- Custom messages

Translation Callback

For dynamic translation:

csharp

```
public void OnReportCustomTableCell(string reportName,
    string columnName, DataRow row, TableCell tableCell)
{
    if (row["ItemName"].ToString() == "{object}")
    {
        string translated = @Client.Locale(row[columnName].ToString());
        Run cellText = (tableCell.Blocks.FirstBlock as Paragraph)
            .Inlines.FirstInline as Run;
        cellText.Text = translated;
    }
}
```

Headers and Footers

Creating Headers

1. Create separate report form
2. Design header content:

```
=======================================
     Company Name
     {Info.Date} - {Info.Time}
     Production Report
=======================================
```

Creating Footers

1. Create footer form
2. Add page numbering:

```
Page {Page.Number} of {Page.Total}
Generated by: {Client.UserName}
```

Linking to Main Report

1. Open main report properties
2. Set **Header** property to header form
3. Set **Footer** property to footer form

Special Features

Alarm Audit Trail

Add alarm history to reports:

```
{Alarm.Group.AuditTrail}
```

With custom messages:

csharp

```
@Alarm.AuditTrail.AddCustomMessage(
    "User: {" + @Client.UserName + "} logged",
    null, null, null, "{object}", null, null
);
```

Append Mode

Accumulate data in existing file:

1. Set **Append** column to `1`
2. New data adds to file
3. Preserves historical records

Dynamic Cell Colors

Apply conditional formatting via callback:

csharp

```
public void OnReportCustomTableCell(string reportName,
    string columnName, DataRow row, TableCell tableCell)
{
    double value = Convert.ToDouble(row[columnName]);
    if (value > 90)
        tableCell.Background = Brushes.Red;
    else if (value > 80)
        tableCell.Background = Brushes.Yellow;
}
```

Export Formats

Format	Extension	Features	Use Case
PDF	.pdf	Full formatting, images	Professional reports
HTML	.html	Web-ready, hyperlinks	Online viewing
ASCII	.txt	Plain text, no formatting	Data export
Unicode	.txt	International characters	Multi-language

Best Practices

1. **Test preview** - Verify before saving
2. **Use templates** - Reusable headers/footers
3. **Validate queries** - Check data availability
4. **Handle nulls** - Default values for missing data
5. **Size images** - Optimize for output format
6. **Test translations** - Verify all languages
7. **Archive reports** - Implement retention

Troubleshooting

Missing data:

- Verify tag names and paths
- Check query execution
- Confirm data timing

Formatting issues:

- Match font availability
- Check export format limits
- Review page margins

Translation problems:

- Confirm dictionary loaded
- Check locale keys
- Verify callback function

Performance:

- Limit query results
- Optimize images
- Reduce complex formatting

In this section...
