WebData Editor Reference

Configure web-based data connectors.

WebData Editor (Reference) provides an integrated environment for defining and editing JSON, XML, HTML, and text data structures with dynamic tag binding. The WebData Editor enables:

- Visual editing of data structures
- Real-time tag binding
- Syntax highlighting and validation
- Request/response mapping
- Template creation
- Format-specific tools

Editor Interface
Tag Binding
Bidirectional Data Flow
Nested JSON Handling
Editor Features
Best Practices
Common Patterns
Troubleshooting

Access via **Reports** **WebData Editor**

Editor Interface

Selecting WebData Objects

1. Navigate to **Reports** **WebData**
2. Select existing object or create new
3. Open **Reports** **WebData Editor**
4. Selected object loads automatically

Navigation Controls

- **ComboBox** - Switch between WebData objects
- **Document Toolbar** - Create, save, delete operations
- **Encoding Selector** - Read-only, shows current format

Tag Binding

Syntax

Embed tag values using curly brackets:

```
{Tag.TagName}
```

Examples by Format

JSON:

json

```
{
  "deviceId": "SENSOR-001",
  "timestamp": "{Info.Date}",
  "measurements": {
    "temperature": {Tag.TankFarm/Tank1/Temp},
    "pressure": {Tag.TankFarm/Tank1/Pressure},
    "level": {Tag.TankFarm/Tank1/Level}
  },
  "status": "{Tag.Equipment/Status}"
}
```

XML:

xml

```
<SensorData>
  <DeviceID>SENSOR-001</DeviceID>
  <Timestamp>{Info.Date}</Timestamp>
  <Temperature>{Tag.TankFarm/Tank1/Temp}</Temperature>
  <Pressure>{Tag.TankFarm/Tank1/Pressure}</Pressure>
  <Status>{Tag.Equipment/Status}</Status>
</SensorData>
```

HTML:

html

```
<html>
<body>
  <h1>Production Report</h1>
  <p>Date: {Info.Date}</p>
  <p>Total Production: {Tag.Production_Count}</p>
  <p>Quality Score: {Tag.Quality_Score}%</p>
</body>
</html>
```

Text:

```
Device Report
=============
Date: {Info.Date}
Temperature: {Tag.TankFarm/Tank1/Temp} °C
Pressure: {Tag.TankFarm/Tank1/Pressure} PSI
Status: {Tag.Equipment/Status}
```

Bidirectional Data Flow

Sending Data (Outbound)

Tags replace placeholders when sending:

json

```
// Template
{"name": "{Tag.UserName}"}

// If Tag.UserName = "John"
// Sent as:
{"name": "John"}
```

Receiving Data (Inbound)

Tags populate from received content:

json

```
// WebData Template
{"temperature": "{Tag.Temp}"}

// Received Data
{"temperature": 25.5}

// Result: Tag.Temp = 25.5
```

Nested JSON Handling

When binding tags to nested JSON objects:

Template:

json

```
{
  "data": "{Tag.SensorData}"
}
```

Received:

json

```
{
  "data": {
    "id": 175,
    "values": [10, 20, 30],
    "status": "active"
  }
}
```

Result: Tag.SensorData receives entire nested object:

json

```
{
  "id": 175,
  "values": [10, 20, 30],
  "status": "active"
}
```

Editor Features

Syntax Highlighting

- Keywords in blue
- Strings in green
- Numbers in red
- Tags in orange
- Comments in gray

Validation

- Real-time syntax checking
- Tag existence verification
- Format-specific rules
- Error indicators

Auto-Completion

- Tag name suggestions
- Format keywords
- Common patterns
- Template snippets

Best Practices

JSON Documents

json

```
{
  // Use consistent indentation
  "metadata": {
    "version": "1.0",
    "timestamp": "{Info.Date}"
  },
  // Group related data
  "measurements": {
    "temperature": {Tag.Temp},
    "pressure": {Tag.Pressure}
  },
  // Handle arrays properly
  "values": [
    {Tag.Value1},
    {Tag.Value2}
  ]
}
```

XML Documents

xml

```xml
<!-- Use proper nesting -->
<Root>
  <!-- Group related elements -->
  <Measurements>
    <Temperature unit="C">{Tag.Temp}</Temperature>
    <Pressure unit="PSI">{Tag.Pressure}</Pressure>
  </Measurements>
  <!-- Use attributes wisely -->
  <Status code="{Tag.StatusCode}">{Tag.StatusText}</Status>
</Root>
```

Error Handling

json

```json
{
  "status": "{Tag.Status}",
  "error": "{Tag.ErrorMessage}",
  "retry": {Tag.RetryCount}
}
```

Common Patterns

API Request Body

json

```json
{
  "method": "getData",
  "params": {
    "startDate": "{Tag.StartDate}",
    "endDate": "{Tag.EndDate}",
    "filters": {
      "location": "{Tag.Location}",
      "type": "{Tag.DataType}"
    }
  }
}
```

Configuration Template

json

```json
{
  "settings": {
    "interval": {Tag.UpdateInterval},
    "threshold": {Tag.AlarmThreshold},
    "enabled": {Tag.SystemEnabled}
  }
}
```

Status Report

xml

```xml
<StatusReport>
  <Timestamp>{Info.Date} {Info.Time}</Timestamp>
  <System>
    <Running>{Tag.System_Running}</Running>
    <Uptime>{Tag.System_Uptime}</Uptime>
  </System>
  <Production>
    <Count>{Tag.Production_Count}</Count>
    <Rate>{Tag.Production_Rate}</Rate>
  </Production>
</StatusReport>
```

Troubleshooting

Tag not replacing:

- Verify tag name and path
- Check curly bracket syntax
- Confirm tag has value

- Review encoding format

Invalid JSON/XML:

- Check bracket/tag matching
- Verify quotes in strings
- Remove trailing commas
- Validate against schema

Data not populating:

- Match template structure
- Verify property names
- Check data types
- Review nested paths

Performance issues:

- Minimize document size
- Reduce tag count
- Optimize nested structures
- Cache static content

In this section...
