TagProvider Services How-to

Step-by-step procedures for configuring and using TagProvider Services.

How-to Guides Data Foundation TagProvider Services How-to

Overview

Typical configuration procedures using TagProvider Services.

Browse TagProvider Data
Access TagProvider Data in Code
Access TagProvider Data in Displays
Configure Historian Integration
Configure Redundancy
Best Practices
Troubleshooting
See Also

Create a TagProvider Connection

Connect FrameworX to an external data source.

Steps

1. Navigate to **Unified Namespace TagProvider Services**
2. Click **New Item** button
3. Select protocol from list (MQTT, OPC UA, ControlLogix, etc.)
4. Configure connection parameters:
    - **Name:** Connection identifier
    - **PrimaryStation:** Main endpoint address
    - **Access:** Read, Write, or ReadWrite
    - **Timeout:** Response timeout (ms)
5. Click **Test Connection** to verify
6. Click **OK** to save
7. Creating the connection adds no folder to the Asset Tree. To use its data with Asset(), select a folder in the Asset Tree (create one first if needed) and click **Insert data from TagProvider** on the Asset Tree toolbar: the folder linked to the TagProvider is created inside the selected folder.

Protocol-Specific Settings

Protocol	Key Parameters
OPC UA	Server URL, Security Policy
MQTT	Broker address, Topic structure
ControlLogix	IP Address, Slot
SQL Database	Connection string, Query

See UNS TagProvider Services Reference for all settings

Browse TagProvider Data

View the data structure from an external system.

Using Asset Tree Browser

1. Navigate to **Unified Namespace Asset Tree**
2. Select the linked folder (for example Plant MyOPCServer)
3. Browse structure in tree view
4. View live values in grid

Using Object Browser

1. Click **Object Browser** icon in toolbar
2. Navigate to desired element
3. Copy displayed AssetPath for use in code/displays

Example Structure

After you link the TagProvider into a folder, its data appears under the linked folder:

```
/Plant
  /MyOPCServer (linked folder)
    /Line1
      /Motor1
        /Speed
        /Temperature
```

Access TagProvider Data in Code

Read values from external systems in displays, expressions, and C# or VB.NET code.

Asset() Syntax

`Asset()` is the form for displays and expressions. MyOPCServer is a folder under Plant linked to the OPC UA TagProvider's root. MyPLC is a folder under Plant linked to the ControlLogix TagProvider's root.

```
Asset("/Plant/MyOPCServer/Line1\Motor1\Speed")

// ControlLogix specific
Asset("/Plant/MyPLC/DINT:MyTag")
Asset("/Plant/MyPLC/DINT:MyArray[0]")
```

After the linked folder, write the path with the TagProvider's own separator. On a provider that separates with `\`, a folder linked to its root reads `Asset("/Plant/Provider/Area\Tag")` in a display or an expression, and a folder linked to `Area` reads `Asset("/Plant/Provider\Tag")`. C# and VB.NET code reads the same path with `TK.Data.AssetValue`. In C#, double each backslash in the string: `TK.Data.AssetValue("/Plant/Provider/Area\\Tag")`. In VB.NET, write each backslash once: `TK.Data.AssetValue("/Plant/Provider/Area\Tag")`. MQTT uses `/`. See separators per provider.

In Scripts

In C# or VB.NET code, read a value with `TK.Data.AssetValue`. `TK.Data.AssetDouble` and `TK.Data.AssetString` return a typed value.

```
// Read value
double speed = TK.Data.AssetDouble("/Plant/MyOPCServer/Line1\\Motor1\\Speed");

if (TK.ToBool(TK.Data.AssetValue("/Plant/MyOPCServer/Line1\\Motor1\\Running")))
{
    // Motor is running
}
```

Data Type Handling

`TK.Data.AssetValue` returns the value as an object. Convert it to the type the code needs:

```
int intValue = TK.ToInt(TK.Data.AssetValue("/Plant/Provider/Area\\Tag"));
string strValue = TK.ToString(TK.Data.AssetValue("/Plant/Provider/Area\\Tag"));
bool boolValue = TK.ToBool(TK.Data.AssetValue("/Plant/Provider/Area\\Tag"));
```

Access TagProvider Data in Displays

Bind display elements to external data.

Display Binding

Bind the display element to the Asset path:

```
Asset("/Plant/MyOPCServer/Line1\Motor1\Speed")
```

Configure Historian Integration

Use TagProviders as historian data sources or storage locations.

As Data Source (for TrendCharts)

1. Create TagProvider connection to historian (Canary, InfluxDB, etc.)
2. In the TrendChart, add a pen for the TagProvider path

3. Historical data is automatically queried

As Storage Location

1. Navigate to **Historian Storage Locations**
2. Select TagProvider as target
3. Map tags to historian points
4. Define storage intervals

Configure Redundancy

Set up backup connections for high availability.

Steps

1. Open TagProvider configuration
2. Set **PrimaryStation** to main endpoint
3. Set **BackupStation** to redundant endpoint
4. Configure **Timeout** for failover detection
5. Save configuration

Failover Behavior

- System monitors primary connection
- On failure, automatically switches to backup
- Logs failover event
- Attempts reconnection to primary

Best Practices

Choose the Right Approach

Scenario	Use
Static, well-defined data structure	Devices
Dynamic, frequently changing structure	TagProviders
Strict validation required (FDA)	Devices
IoT/Monitoring applications	TagProviders

Configuration Tips

- **Test connections** before production deployment
- **Configure backup stations** for critical connections
- **Monitor performance** — check read/write times
- **Document providers** — use clear naming and descriptions
- **Consider security** — set appropriate access mode restrictions

Performance Optimization

- Reduce polling frequency where real-time not required
- Check network latency between systems
- Optimize queries for SQL-based providers
- Use local caching when appropriate

Troubleshooting

Connection Failed

1. Verify network connectivity to external system
2. Check protocol settings (port, address)
3. Review authentication credentials
4. Test with simple query first

No Data Visible

1. Confirm connection is active (check Services Monitor)
2. Check Asset Tree mapping is correct
3. Verify access permissions on external system
4. Review protocol-specific requirements

Performance Issues

1. Reduce polling frequency
2. Check network latency
3. Optimize queries (SQL providers)
4. Consider local caching

Type Conversion Errors

1. Convert the value with TK.ToInt, TK.ToString or TK.ToBool
2. Verify data types match expected
3. Check for null values
4. Review error messages in logs

See Also

- TagProvider Services — Concept overview
- UNS TagProvider Services Reference — All settings and options
- TagProvider Services Technology — Deep architecture dive
- Connectors Library — Available protocols