UNS TagProvider Services Reference

Connect UNS to external data services.

UNS TagProvider Services (Reference) enable direct interaction with variables in remote systems. TagProvider Services provide:

- Dynamic external data access through auto-discovery
- Three connection patterns: Local Tags, Linked Tags (via DataLink), and Dynamic Tags (via Asset() syntax)
- Automatic data model discovery at runtime
- On-demand communication — activated when data is consumed, released when not
- Multi-protocol support (~16 discovery-capable connectors)

TagProviders allow consuming external data models directly, ideal for IoT systems and dynamic environments where device configurations change frequently.

Find a TagProvider Service
TagProviders vs Devices
Supported TagProviders
Configuration
Protocol-Specific Settings
DataLink Property
Asset() Syntax Reference
Data Type Handling
Access Modes
See Also

Find a TagProvider Service

TagProviders vs Devices

Both TagProvider Services and the Devices module connect FrameworX to external systems. They are completely independent — they share nothing architecturally. Some protocols (MQTT, OPC UA, ControlLogix) are available in both, with different workflows.

Aspect	Devices Module	TagProvider Services
Design philosophy	Define explicitly, validate strictly	Auto-discover, bind dynamically
Tag creation	Required — every point maps to a local tag	Optional — Dynamic Tags need no pre-created tags
Namespace control	Solution architect defines everything	External source can define structure
Metadata	Full control (alarms, historian, scaling, units)	Full on Linked Tags; limited on Dynamic Tags
Security	Granular per tag	Connection level
Available protocols	~100 (all industrial protocols)	~16 (discovery-capable only)
Configuration	Design-time	Runtime discovery
Best for	SCADA control, regulatory compliance, FDA	IIoT monitoring, dynamic environments, UNS integration
Can coexist?	Yes — same solution, even same protocol	Yes

Supported TagProviders

Communication Protocols

- **MQTT Client Connector** — Standard MQTT broker connection
- **MQTT Sparkplug B Collector** — SparkplugB protocol
- **OPC UA Client** — Industrial standard
- **AB Rockwell ControlLogix** — Rockwell PLCs
- **Codesys PLC Handler** — Codesys PLCs
- **Beckhoff TwinCAT** — Beckhoff systems

Database Connectors

- **SQL Database** — Generic SQL data sources
- **SQL Database Normalized** — Normalized SQL schema
- **InfluxDB** — Time-series database
- **FlowTimebaseDB** — Flow Software historian
- **MongoDB Database** — Document database. Browse collections as a tag tree, register document fields as UNS tags via Tag Discovery; updates write back with $set. New in 10.1.5.

Application Integration

- **Server to Server** — FrameworX solution interconnection
- **DataHub Communication** — Distributed data aggregation
- **LineMonitor / MES CoreM** — Production monitoring

Enterprise Historians

- **Canary Labs** — Enterprise historian
- **GE Proficy** — Industrial historian
- **AVEVA PI System** — PI System

> ⚠ **Dual-role connectors**
>
> Some connectors (Canary, InfluxDB, GE Proficy, SQL, MongoDB) can also serve as **Storage Locations** in the Historian module for archiving time-series data. That is a separate configuration in the Historian module — see Historian for details.

Configuration

Creating a Connection

1. Navigate to **Unified Namespace TagProvider Services**
2. Click **New Item**
3. Select protocol from the list
4. Set the **ServiceType**: Tag Discovery (default) or Extension Module
5. Configure connection parameters (PrimaryStation is protocol-specific — use `list_protocols` in MCP)
6. The connection appears in the Services grid. No Asset Tree folder is created: to use the 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.

All Properties

Property	Type	Required	Description
Name	String	Yes	Unique connection identifier
Protocol	Reference	Yes	Communication protocol (references DevicesProtocols)
PrimaryStation	String	Yes	Main connection endpoint — format is protocol-specific
ServiceType	Enum	No	`TagDiscovery` (default), `ExtensionModule` or `HistorianStorage` (the provider is a Historian Storage Location)
Description	String	No	Connection description
BackupStation	String	No	Redundant endpoint for failover
FailOverStation	String	No	Failover connection string
ProtocolOptions	String	No	Protocol-specific configuration — format is protocol-specific
Settings	String	No	Additional connection settings
Timeout	String	No	Connection timeout value
Access	String	No	Access mode configuration
ReadTime	String	No	Read polling rate
WriteTime	String	No	Write rate
GatewayIP	String	No	Address of the gateway computer that opens the provider connection, as `host:port` of its TWebServices (Solution Center) service. The default port is 10108 and the port is required. Empty connects from the runtime computer.
Interface	String	No	Network interface binding
InitialState	String	No	Initial state at runtime startup (enabled/disabled)
RemoteSettings	String	No	Remote connection configuration
Separators	String	No	Tag path separator configuration

DriverVersion	String	No	Protocol driver version
Category	String	No	Classification group for organizing connections

Separators per provider

Connector	Branch	Attribute
MQTT, MQTT Sparkplug B, Sorba AI	`/`	`/`
OPC UA	`\`	`\`
PI, GE Proficy, Value Simulator	`\`	`|`
ControlLogix, TwinCAT, PLC Handler, MongoDB, QuestDB, SQL Database, SQL Database Normalized, Canary	`.`	`.`
InfluxDB, FlowTimebaseDB, LineMonitor	`|`	`|`

These are the defaults. An OPC UA connection can set its own branch separator in its station settings, and the TagProvider then uses that one. To set separators through the AI tools, write `BranchSeparator=<char>;AttributeSeparator=<char>;` in the Separators column. Leave it out to keep the stored value. A TagProvider named MQTT or CanaryLabs ignores the column.

> **Protocol-dependent fields**
>
> **PrimaryStation**, **BackupStation**, and **ProtocolOptions** formats vary by protocol. Always call `list_protocols` (MCP) or check the specific connector reference page before configuring these fields.

Protocol-Specific Settings

OPC UA

Setting	Description
Server URL	OPC UA endpoint (e.g. `opc.tcp://server:4840`)
Security Policy	None, Basic128Rsa15, Basic256, Basic256Sha256
Security Mode	None, Sign, SignAndEncrypt
Authentication	Anonymous, Username/Password, Certificate

MQTT

Setting	Description
Broker	MQTT broker address
Port	Broker port (default: 1883)
Topic Structure	Topic pattern for subscription
QoS	Quality of Service level (0, 1, 2)
Client ID	Unique client identifier

ControlLogix

Setting	Description
IP Address	PLC IP address
Slot	CPU slot number
Path	Optional routing path

SQL Database

Setting	Description
Connection String	Database connection string
Query	SQL query for data retrieval
Refresh Rate	Poll interval (ms)

MongoDB

Setting	Description
PrimaryStation	Semicolon-delimited: `ConnectionUri;Database;AuthSource;TlsEnabled`. Example: `mongodb://localhost:27017;plant_data;admin;false`
Discovery	Level 0 lists collections; Level 1 samples the latest document and lists top-level field names. `_id` is hidden.
Address format	Single-dot `collection.field`. Multi-dot rejected in 10.1.5.
Read / Write	Read returns latest document field (Quality 192 on hit, 0 on miss). Write applies `$set` on the most-recent document.

DataLink Property

DataLink is a property on UnsTags — not on the TagProvider connection itself. When set on a tag, it binds that tag to a path in a TagProvider's discovered namespace.

Example: A tag `Plant/Tank1/Temperature` with DataLink set to `MyMQTT.("sensors/tank1/temp")` will automatically receive values from the MQTT broker whenever the tag is read. Communication is on-demand — activated when consumed, released when not.

This is the mechanism behind **Linked Tags** — tags that exist in the local UNS with full metadata (alarms, historian, scaling) but get their values from an external source.

See UNS Tag Discovery Reference for the full explanation of Local Tags, Linked Tags, and Dynamic Tags patterns.

Asset() Syntax Reference

Basic Syntax

```
Asset("/Plant/<linked folder>/<path on the TagProvider>")
```

`Plant` is a folder of the Asset Tree, and the linked folder is a folder under it that is linked to the TagProvider's root. The linked folder sits inside a folder of the Asset Tree, not at its root.

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.

ControlLogix Syntax

MyPLC is a folder under Plant linked to the ControlLogix TagProvider's root.

```
Asset("/Plant/MyPLC/DataType:TagName")
Asset("/Plant/MyPLC/DINT:MyTag")
Asset("/Plant/MyPLC/DINT:MyArray[0]")
Asset("/Plant/MyPLC/REAL:Temperature")
```

Supported Operations

Operation	Syntax
Read in a display or an expression	`Asset("/path")`

Read in C# or VB.NET code	`TK.Data.AssetValue("/path")`
Subscribe	Automatic in displays

Data Type Handling

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

Method	Returns
`TK.ToInt(value)`	Integer
`TK.ToDouble(value)`	Double
`TK.ToString(value)`	String
`TK.ToBool(value)`	Boolean
`TK.ToDateTime(value)`	DateTime

Access Modes

Mode	Description
Read	Read-only access to external data
Write	Write-only access. A provider used as a historian storage location needs Write or ReadWrite to store samples.
ReadWrite	Full bidirectional access

From FrameworX 10.1.6, a value written to a TagProvider tag through a gateway (`GatewayIP`) reaches the device. The gateway computer must also run FrameworX 10.1.6. Writes to PI points through a gateway are not supported.

See Also

- TagProvider Services — Concept overview
- TagProvider Services How-to — Step-by-step procedures
- TagProvider Services Technology — Architecture deep dive
- UNS Tag Discovery Reference — Connection patterns (Local, Linked, Dynamic Tags)
- Connectors Library — All available connector reference pages

In this section...