TagProvider Services

Connect to external systems and bring data into the Unified Namespace.
Modules Data Foundation TagProvider Services | How-to | Reference

---

Overview

TagProvider Services enable FrameworX to connect to external data sources — MQTT brokers, OPC UA servers, PLCs, databases, and enterprise historians — and auto-discover their namespace. When configured, FrameworX acts as a **client** that reads and writes data from these external systems, creating tags in the real-time database (RTDB) on demand.

TagProvider Services support two service types:

- **Tag Discovery** — Auto-discover tags from external data sources without pre-mapping
- **Module Extension** — Embed third-party modules (AI, analytics) as native UNS components

Overview
FrameworX as Client vs Server
What You Can Connect To
Key Concepts
How Data Flows
Three Connection Patterns
Configuration Location
Common Use Cases
TagProvider Services Links
See Also

FrameworX as Client vs Server

Module	FrameworX Role	Direction	Example
TagProvider Services	Client	FrameworX External	Connect to an MQTT broker, discover OPC UA server
Data Servers	Server	External FrameworX	OPC UA client reads your UNS data

TagProvider Services: You connect TO external systems
Data Servers: External systems connect TO you

See Data Servers for exposing UNS to external clients

What You Can Connect To

TagProvider Services use protocols that support **auto-discovery** — the ability to browse and discover what data an external system has. This is a different set of connectors from the Devices module.

TagProvider Connectors (~16)

Category	Connectors
Communication Protocols	MQTT Client, MQTT Sparkplug B Collector, OPC UA Client, AB Rockwell ControlLogix, Beckhoff TwinCAT, Codesys PLC Handler
Database Connectors	SQL Database, SQL Database Normalized, InfluxDB, FlowTimebaseDB, MongoDB, QuestDB
Application Integration	Server to Server, DataHub Communication, LineMonitor / MES CoreM
Enterprise Historians	Canary Labs, GE Proficy, AVEVA PI System

See individual connector reference pages under UNS TagProvider Services Reference for configuration details

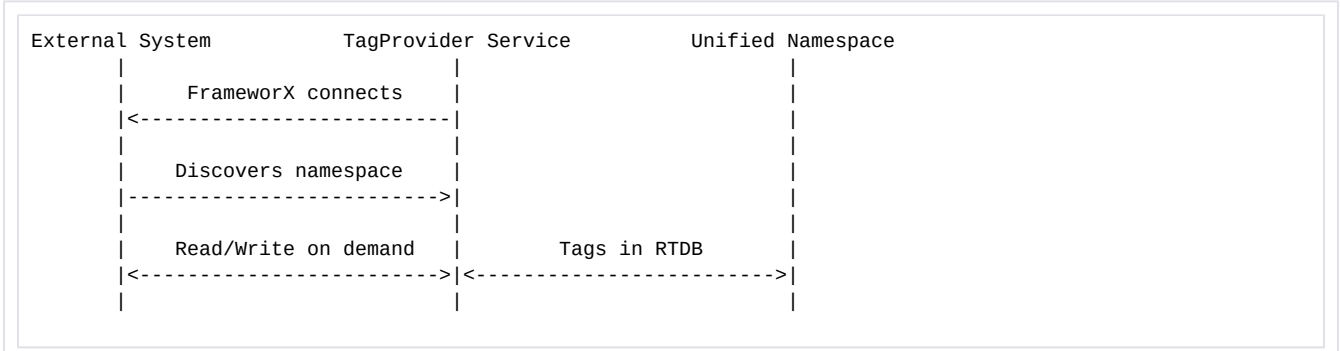
> ⚠ **TagProvider Services vs Devices Module**
>
> The Devices module has ~100 protocols (Modbus, BACnet, Siemens S7, DNP3, and many more) for explicit point-by-point control. TagProvider Services have ~16 connectors that support auto-discovery. They are completely independent — both can coexist in the same solution, even for the same protocol. See TagProvider Services Technology for a detailed comparison.

Key Concepts

Concept	Description

TagProvider	A configured connection to an external data source. Each TagProvider has a protocol, a connection string (PrimaryStation), and a service type.
DataLink	A property on UnsTags that binds a local tag to a path in a TagProvider's discovered namespace. When set, the runtime auto-manages communication on demand. This is the key mechanism behind Linked Tags.
Local Tags	Tags defined in UNS configuration with no external binding. Values come from Devices, scripts, or manual entry. DataLink is not set.
Linked Tags	Tags defined in UNS configuration with DataLink set to a TagProvider path. The tag has full local metadata (name, type, alarms, historian) but gets its value from the external source.
Dynamic Tags	Tags created automatically at runtime when accessed via `Asset("path")`. No pre-configuration needed — the external source defines what exists. Tags are created in the RTDB on demand and cleaned up automatically.
Asset() Syntax	Runtime function that resolves a path through an Asset Tree folder linked to a TagProvider: `Asset("/Plant/Provider/path")`. Used for Dynamic Tags in displays and expressions. C# and VB.NET code reads the same path with `TK.Data.AssetValue`.
Module Extension	A service type that embeds third-party .NET modules as native UNS components. Partners provide an XML descriptor and DLL; the module appears in Designer and exposes data through standard UNS patterns.

How Data Flows



Data from TagProviders appears in the UNS just like local tags — all modules (Displays, Alarms, Historian, Scripts) can use it uniformly. Communication is on-demand: activated when a tag is consumed by a display, script, or other module, and released when no longer needed.

Three Connection Patterns

TagProvider Services support three patterns for integrating external data, depending on how much control you want over the namespace:

Pattern	When to Use	DataLink	Access Method
Local Tags	Full local control, values from Devices or scripts	Not set	`Tag.TagName`
Linked Tags	Local metadata + values from external source	Set to TagProvider path	`Tag.TagName`
Dynamic Tags	External source defines namespace, zero config	N/A	`Asset("/Plant/Provider/path")`

See UNS Tag Discovery Reference for a deep dive on all three patterns and governance models

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.

Configuration Location

Designer: Unified Namespace TagProvider Services

Each TagProvider connection is configured with:

- **Name** — Unique identifier
- **Protocol** — Communication protocol (selected from ~16 discovery-capable connectors)
- **ServiceType** — Tag Discovery (default) or Extension Module

- **PrimaryStation** — Connection endpoint (format is protocol-specific)
- Communication parameters (timeout, polling rate, access mode)

See UNS TagProvider Services Reference for the full property list

---

Common Use Cases

Use Case	TagProvider Connector	Pattern
Subscribe to MQTT sensor data from an IIoT broker	MQTT Client	Dynamic Tags or Linked Tags
Browse and integrate an OPC UA server	OPC UA Client	Linked Tags or Dynamic Tags
Read ControlLogix PLC tag database	AB Rockwell ControlLogix	Linked Tags
Query SQL database for process data	SQL Database	Linked Tags
Aggregate data from another FrameworX solution	Server to Server	Dynamic Tags
Integrate AI/ML analytics into UNS	Module Extension (e.g. Sorba.AI)	Module Extension
Read historical data from enterprise historian	Canary Labs, GE Proficy, InfluxDB	Linked Tags
Read time-series data from MongoDB or QuestDB	MongoDB, QuestDB	Linked Tags or Dynamic Tags

---

TagProvider Services Links

Explanation — to understand concepts
TagProvider Services Technology — Architecture, service types, and comparison with Devices module
UNS Tag Discovery Reference — Connection patterns: Local, Linked, and Dynamic Tags

How-to Guides — to accomplish specific tasks
TagProvider Services How-to — Step-by-step configuration procedures
Data Explorer Tool — Discover and import external data structures

Reference — technical details
UNS TagProvider Services Reference — All properties and configuration
UNS Services Monitor Reference — Runtime monitoring of TagProvider connections

---

See Also

- Data Servers — Expose UNS to external clients (FrameworX as server)
- Data Foundation — UNS overview and module relationships
- Unified Namespace — Core UNS concepts
- UNS Tags Reference — Tag properties including DataLink