ProveIt 2025 Expo Example

Create a Smart Monitoring Solution for Real-Time Production Sites.

Download the Solution File: ProveIt Expo Example.dbsln

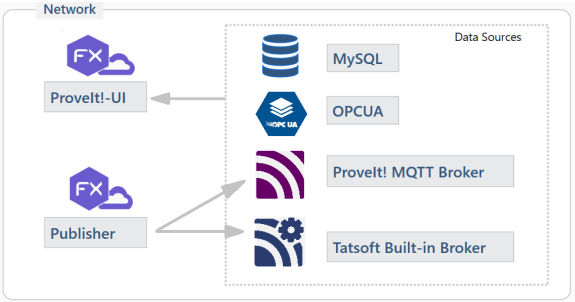
This solution demonstrates how to integrate multiple technologies, to create a Smart Monitoring Solution for Real-Time Production Sites, it was presented originally at ProveIt! 2025 in Dallas.

System Architecture

The architecture consists of two projects: one that publishes data to both an embedded MQTT broker, used as a testing environment, and the ProveIt MQTT broker, and the main project (UI), which reads data from the broker, retrieves published OPC UA data, accesses the MySQL database for additional data integration, and displays the information.

The data connection is mainly done via TagProvider, which acts as an intermediary between the application and multiple data sources. It enables seamless access to real-time and historical data from different brokers, OPC UA servers, and the MySQL database. The TagProvider ensures data consistency and reliability by managing connections, data subscriptions, and updates efficiently.

Additionally, it supports various protocols and standards, making it a flexible solution for industrial applications that require integration across multiple data sources. The application can leverage these connections to process, analyze, and visualize data in real-time, enhancing monitoring and decision-making capabilities.



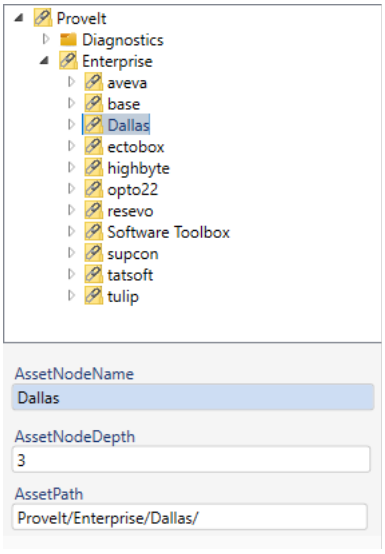
Highlights

Asset Tree

The TAssetsTree component simplifies navigation through structured data from the MQTT broker, enabling users to explore assets efficiently. Its hierarchical structure ensures that information is organized and accessible, providing a clear framework for asset management.

The key method responsible for determining which page to open is VerifyNavigation(), located in the code-behind of the MenuTree page. This method evaluates the current application context and decides which screen to display based on the asset name (AssetName) and the hierarchical structure of nodes.

The VerifyNavigation() method directly relies on this asset tree to determine node depth and relationships accurately. By leveraging this structure, the system can ensure seamless navigation, directing users to the appropriate screens based on asset categories such as sites, production areas, or specific system components.



Pages

The header is a navigation component that allows users to seamlessly switch between different application pages. The application primarily offers two ways to navigate: the header provides access to the main pages, while the asset tree determines which data is displayed on the page. Navigation can also be performed using the asset tree. Additionally, the header includes options to select the layout type, log in, and change the theme.



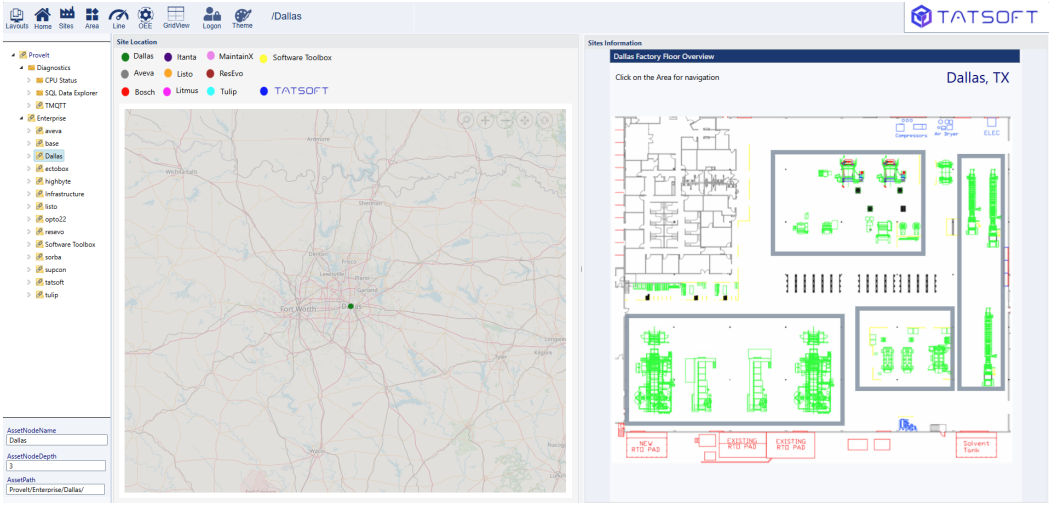
Layouts

The application supports adaptive layouts to ensure proper screen scaling based on the selected device type:

- **Desktop**
- **Pad**
- **Mobile**

Sites

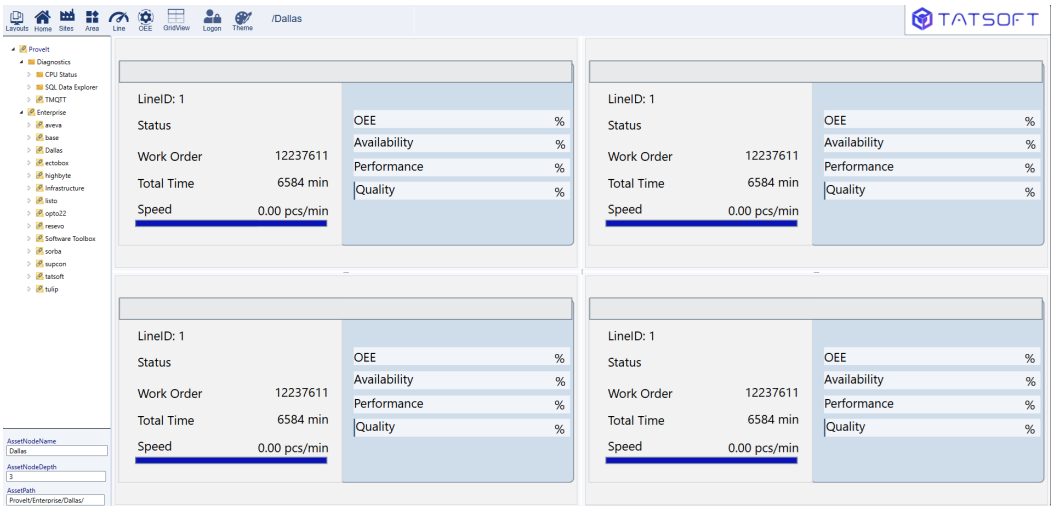
A demonstration of geolocation-based functionalities within the application.



More information about this component can be found here.

Area

This page showcases the use of dashboards and symbols, offering dynamic visual representations of system data.



Line and OEE

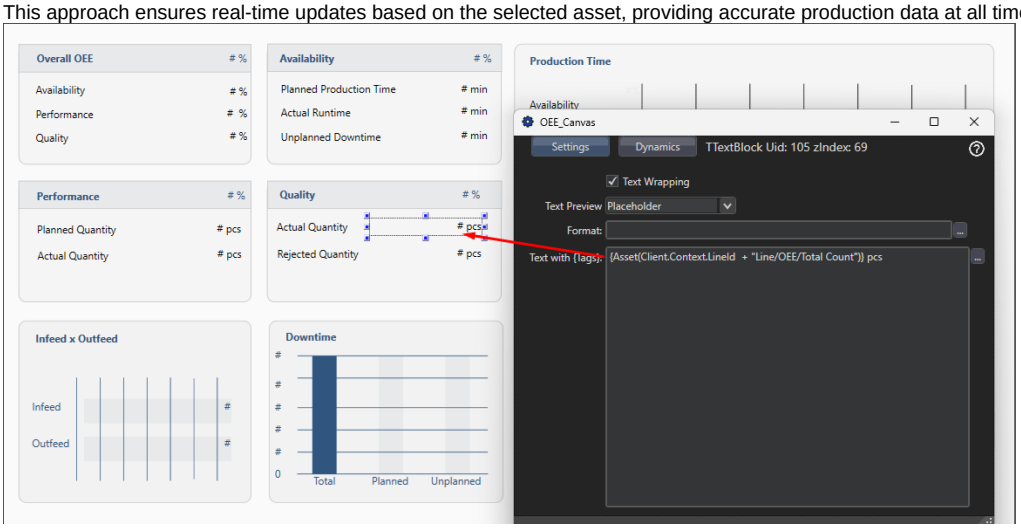
This component dynamically maps data from the MQTT broker. Users can select relevant data through the AssetTree, making real-time monitoring more accessible.

An example of a dynamic field configuration in the OEE page is shown below. The value inside the curly brackets represents a dynamic expression where:

- **Client.Context.LineId**: Retrieves the selected asset from the AssetTree.
- **"Line/OEE/Total Count"**: Specifies the data path within the system.
- **"pcs"**: A static label.

Example: {Asset(Client.Context.LineId + "Line/OEE/Total Count")} pcs

This approach ensures real-time updates based on the selected asset, providing accurate production data at all times.



Grid View
The Grid View page demonstrates how raw data from MQTT is displayed, using @Client.Context.AssetName as the data source for the grid.

Publisher

The **Publisher** project mentioned in the System Architecture simulates a production line and publishes relevant data to the MQTT broker under the topic Enterprise/tatsoft.
This project uses two of our utilities:

- MQTT built-in Broker: designed to simplify the development and testing of applications, especially in edge environments. This broker supports MQTT clients following the Sparkplug B specification, as well as traditional MQTT clients like Mosquitto, and is compatible with MQTT protocol versions 3.1.0, 3.1.1, and 5.0.0. For enterprise-level applications, third-party brokers like HiveMQ are recommended for

advanced administration and scalability. TMQTTBroker can run on both Windows and Linux operating systems and offers direct integration with FrameworX servers, exposing native tags for client subscription. The broker configuration can be managed via the Solution Designer interface or by running it as a Windows service, facilitating its deployment in different system architectures.

- MQTTPublisherSimulator : is a tool for testing and validating MQTT communications. It connects to local or remote brokers, supports authentication, and publishes messages like NBIRTH, DBIRTH, and DDATA. Users can configure GroupID, NodeID, and DeviceID, and run it via the Designer or the MQTTspBSimulator executable. It allows single or continuous data publishing with adjustable intervals.

In this section...
