ZDMP Blog

Connecting Components of an IIoT Platform

By Dr.-Ing. Martin Heß, Researcher at Software AG

Some questions for you

- How do you integrate many different components and modules into one platform?
- How do you manage application management interfaces (APIs) throughout their lifecycle?
- How do you listen for certain events and exchange information with other components?

Component Interconnectivity in IIoT Platforms

Modern IIoT platforms are typically based on a modular architecture. They consist of multiple modules and components, which can exchange data and commands through well-defined interfaces – so-called application programming interfaces (APIs). This approach ensures an easy extension of the platform when adding modules and allows to replace or change individual components without affecting the entire system. Additionally, it allows the platform to scale by distributing the workload among multiple instances of the same modules.

With increasing complexity, however, it is very difficult to maintain the overview of the components and their APIs. For example, the APIs may change over time when adding or removing features and the component endpoints (eg IP addresses and ports) may change dynamically depending on the platform's runtime environment. Another important aspect is that calling an API is an ad-hoc action, which triggers a workflow at a predefined set of target components. However, sometimes a component may just want to broadcast an information without addressing specific components or wishes to listen for certain events instead of periodically querying APIs for these events.

In ZDMP, these problems are addressed by the Service and Message Bus component. It represents the communication layer of the ZDMP Platform and consists of two major modules – the Services API Management and the Message Bus.

API Management in ZDMP

The Services API Management provides a central place for hosting and managing all APIs used in ZDMP (REST APIs). Developers can interact with the component conveniently through a web frontend. They can either upload their APIs using standard formats such as OpenAPI [1] or RAML [2] or create APIs from scratch by using the built-in visual editor. The APIs can be activated, deactivated, or changed at any time. To maintain downward compatibility, the Services API Management also allows to keep track of multiple versions of an APIs running in parallel.

Activating an API in the Services API Management creates a unique gateway endpoint. It allows other components to call the API and its underlying service without knowing the "real" service endpoint. This proxy functionality provides two benefits: On the one hand, it adds an additional security layer for the APIs. On the other hand, it allows ZDMP's application run-time component to dynamically change the native service endpoints without affecting other components. Moreover, the Services API Management directly integrates with ZDMP's security system. Through this, ZDMP Developers do not have to care about security aspects such as authentication and authorisation and can just focus on their development task.

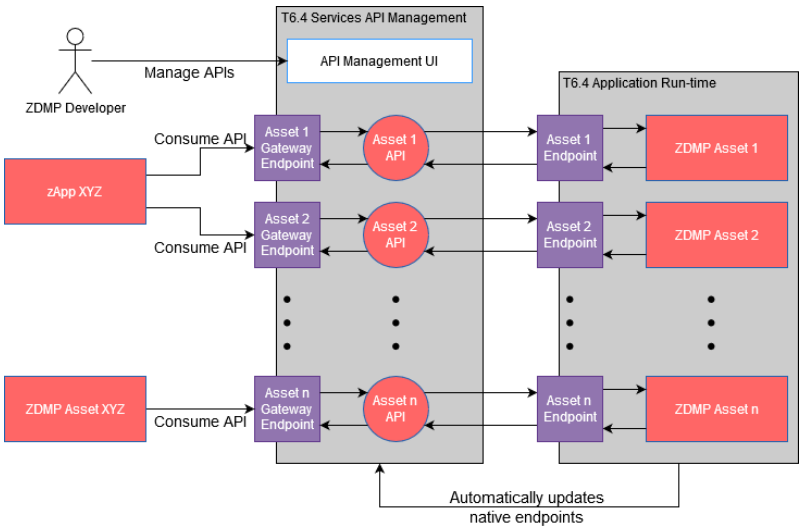

Figure 1: API Management and service calls

Another useful feature for ZDMP developers is the API mocking functionality provided by the Services API Management. Sometimes an API has already be defined but the full functionality of the underlying services is not yet

fully implemented. In this case, developers can simulate the behaviour of their final services by mocking the API responses.

Message-based communication in ZDMP

As outlined above, sometimes there is a need to simply broadcast information and message without knowing the recipients. For example, an IIoT temperature sensor provides temperature measurements and which component uses this information is irrelevant for the sensor's logic. Similarly, a component which processes these temperature values may want to be immediately informed about any changes instead periodically querying the component for changes.

To address these needs, the Service and Message Bus component provides a Message Bus module. It implements a publish/subscribe messaging concept, which allows components to publish information on specific topics and to subscribe to topics to listen for certain events or messages broadcasted on them. Thus, the Message Bus supports several standard communication protocols such as MQTT [3] and AMQP [4], which are commonly used in the industrial IoT context. Each ZDMP component has its own main topic and additional subtopics can be created and deleted by the component itself on demand. Moreover, the Message Bus integrates with ZDMP's security system to ensure a secure communication via SSL/TLS. This integration also automatically handles the authentication and authorisation workflow when interacting with the Message Bus.

What will ZDMP achieve

ZDMP aims to support developers in creating new zApps and participating in ZDMP's continuously growing ecosystem. The Service and Message Bus component supports these goals by providing an easy, standardised, and well-documented way for developers to interact with the different components and zApps provided by ZDMP as well as to maintain the developers' APIs throughout the entire lifecycle.

ZDMP Links

• Architecture Component(s)	Service and Message Bus
• Work Package	WP6 – ZDMP Platform Building
• Tasks	T6.4 – Platform Integration and Federation

References/Acknowledgements

[1] OpenAPI Specification, https://github.com/OAI/OpenAPI-Specification
[2] RAML Specification, https://raml.org
[3] MQTT Messaging Protocol, https://www.mqtt.org
[4] AMQP Messaging Protocol, https://www.amqp.org