6 Years of Eclipse
Kura Wires

Timeus Nicola

25th October 2022
objectives

1. provide an overview of Eclipse Kura Wires

2. showcase the current state of Kura Wires ecosystem

3. illustrate real-world applications
Everyware Software Framework (ESF) is an enterprise-ready IoT Edge Framework distributed and supported by Eurotech. Based on Eclipse Kura, the open source Java/OSGi middleware for IoT gateways.
Eclipse Kura Wires overview
data flow programming model

- describes a program in terms of a directed graph
- the graph nodes represent computations
- the graph edges represent the data flow between the various computations
data flow programming model

Fits to describe the typical IoT scenario:
- data acquisition
- local processing
- publishing

It is natural to think about scenario like this in terms of a graph representing the data flow between the different entities involved.
Kura Wires

Kura Wires is a data flow programming framework focused on IoT applications. It allows to describe an application in terms of a Wire Graph.

- graph nodes are called **Wire Components**
- graph edges are called **Wires**

![Wire Graph Diagram]
Kura Wires

goals

- simplicity
- integration with existing Eclipse Kura framework
- extensibility
- reduced time to market
- remote deployment and lifecycle management
simplicity

- high level of abstraction
- simple data Model
- wires composer UI
integration with existing Eclipse Kura framework

- Wire Components are implemented as configurable OSGi services
- Ready to use Wire Components expose the Kura core functionalities
- Portability across different gateway architectures
The user can implement custom Wire Components by implementing the dedicated OSGi Service Interfaces in Eclipse Kura APIs.
Wire Components, Wire Graph Layout and Wire Component configurations can be deployed in different ways:

- through Kura management console
- from the cloud, on a single target or on a fleet of devices
- using local REST APIs
reduced time to market

- simple development workflow
- high level of abstraction
- reuse of existing components
- well defined lifecycle
- portability
Kura Wires model

Wire components

Graph nodes are called Wire Components

Wire components can be emitters, receivers or both
Kura Wires model

wires

- graph edges are called wires
- wires connect an output port of an emitter component to an input port of a receiver component.
- the communication is message oriented, a message on a Wire is called Wire Envelope
Kura Wires model

data model

The data inside a Wire Envelope is organized in key-value maps, keys are strings, and the value data types are limited to a small set of primitives.

```
assetName : PLC1
ANALOG_IN2_timestamp : 1665743207298
ANALOG_IN1 : 123
ANALOG_IN1_timestamp : 1665743207298
ANALOG_IN2 : 456
```
Wires Composer UI
common patterns

timer
common patterns

data logging
common patterns
publish/subscribe
common patterns

data storage

![Diagram of data storage patterns]

- Producer
- DbStore
- Trigger
- DbQuery
- Consumer
common patterns

message routing
common patterns
processing/filtering

Diagram showing the process flow from Producer, through Regex, AI, and Javascript, to Consumer.
The component allows interacting with an InferenceEngineService to perform machine learning-related operations.

Eclipse Kura provides integration with the Nvidia Triton Server, an open-source inference service software that enables the user to deploy trained AI models from any framework on GPU or CPU infrastructure.
JavaScipt Filter

The Script Filter component provides scripting functionalities in Kura Wires using the JavaScript language:

- Allows to perform arbitrary transformations on Wire Envelopes
- Very high level of abstraction, the script author only needs to reason in terms of Wire Record properties

```javascript
var record = input.records[0]
var outRecord = new WireRecord()

for (var prop in record) {
  // process envelope properties
}

output.add(outRecord)
```
common patterns
interact with field devices/sensors

Eclipse Kura includes the Driver and Asset abstraction that provides a uniform model to represent interaction with external field devices or sensors.
Kura Wires model
Driver and asset model

A **driver** encapsulates the implementation of a specific field protocol or I/O resource access and exposes it through common APIs.

Exposes the resources at data point level (**channel**):

- **a. READ:** the data point value is read from the external resource on demand (polling).
- **b. WRITE:** allows to modify the value of the data point.
- **c. LISTEN:** allows to receive unsolicited notifications about data point value change.

An **asset** allows to define and persist a set of channel configurations, usually it represents a remote device accessible through the driver.
Kura Wires Model

driver and asset model
Kura Wires model

asset as Wire Component: READ

When an Asset receives a Wire Envelope, it will read the channel values and emit the operation result on the Wire, one property per channel.
**Kura Wires model**

asset as Wire Component: **LISTEN**

Channels can be configured in listen mode. The Asset will emit a Wire Envelope when an event occurs on the channel.

Record content:
- ANALOG_IN1 : 123
- assetName : PLC1
- ANALOG_IN1_timestamp : 1665744150274
Kura Wires model

asset as Wire Component: WRITE

If the Asset receives a Wire Envelope property whose name matches the name of one of its channels, it will perform a write operation.
Eclipse Kura Wires ecosystem
Eclipse marketplace
install by drag and drop
Eclipse marketplace

Eclipse Marketplace entry count for Kura
Built In wire components

Kura provides the following components out of the box:

- Timer
- Logger
- Database Components
- FIFO component
- Regex Component
- Join Component
- Wire Asset

Cloud Publisher

Cloud Subscriber
components from marketplace
Eclipse Kura Components

wire components
- AI Wire component
- Logic / Math Components
- Javascript Filter

drivers
- GPIO Driver
- S7 Driver
- OPC / UA Driver
- SenseHat Driver
- Bosch XDK Driver
- iBeacon / Eddystone / Sensortag drivers
components from Marketplace
field protocol support

<table>
<thead>
<tr>
<th>industrial</th>
<th>energy</th>
<th>transportation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modbus TCP</td>
<td>DNP3 Master/Outstation</td>
<td>SAE J1939</td>
</tr>
<tr>
<td>Modbus RTU</td>
<td>IEC 60870-5-104 Master/Outstation</td>
<td>SAE J1979</td>
</tr>
<tr>
<td>OPC-UA</td>
<td>IEC 60870-5-101</td>
<td>CAN Socket Library</td>
</tr>
<tr>
<td>S7</td>
<td>M-bus</td>
<td>MVB Data read-only</td>
</tr>
<tr>
<td>FANUC</td>
<td>IEC 61850</td>
<td>BLE</td>
</tr>
<tr>
<td>EtherNet/IP Allen Bradley</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TwinCAT Beckhoff</td>
<td>MQTT</td>
<td></td>
</tr>
<tr>
<td>MQTT</td>
<td>BLE</td>
<td></td>
</tr>
<tr>
<td>BLE</td>
<td>DNP3 Master/Outstation</td>
<td></td>
</tr>
<tr>
<td></td>
<td>IEC 60870-5-104 Master/Outstation</td>
<td></td>
</tr>
<tr>
<td></td>
<td>IEC 60870-5-101</td>
<td></td>
</tr>
<tr>
<td></td>
<td>M-bus</td>
<td></td>
</tr>
<tr>
<td></td>
<td>IEC 61850</td>
<td></td>
</tr>
<tr>
<td></td>
<td>BACnet</td>
<td></td>
</tr>
<tr>
<td></td>
<td>OCPP 1.6</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DLMS/OSEEM</td>
<td></td>
</tr>
<tr>
<td></td>
<td>IEC 62056-21</td>
<td></td>
</tr>
<tr>
<td></td>
<td>C37.118</td>
<td></td>
</tr>
<tr>
<td></td>
<td>LoRaWAN</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MQTT</td>
<td></td>
</tr>
<tr>
<td></td>
<td>BLE</td>
<td></td>
</tr>
</tbody>
</table>
components from Marketplace

cloud connectors

- Eclipse Kapua
- Everyware Cloud
- Eclipse Hono
- Azure IoT
- AWS IoT
- Cumulocity
- Exosite
- Apache Kafka
Real World applications
real world applications

DB Cargo

DB Cargo, leading company in the European freight rail transportation business.

IoT application deployed on hundreds of trains

Eclipse Foundation Member
real world applications

simple, common scenarios are simple to implement
real world applications

complex scenarios can be addressed by composing simple reusable components
real world applications

low code/no code

---low code
- use processing components
- write custom components in Java

---no code
- if existing components are not enough, implement custom components that are simple and reusable
- provide a toolset rather than a closed solution
real world applications

Protocol conversion

Protocol conversion involves enabling communication between two or more different devices that use different communication problems.

A protocol conversion can be modeled in the Driver, Asset and Wires models as a data transfer between two different drivers, operating at a datapoint level (e.g., Modbus to OPC-UA).

Kura Wires was not designed for this.
real world applications
Protocol conversion
future extensions

- add discovery support in the **Driver model**
- add support for **Asset template library**
- add support for **more database implementations**
Thank you