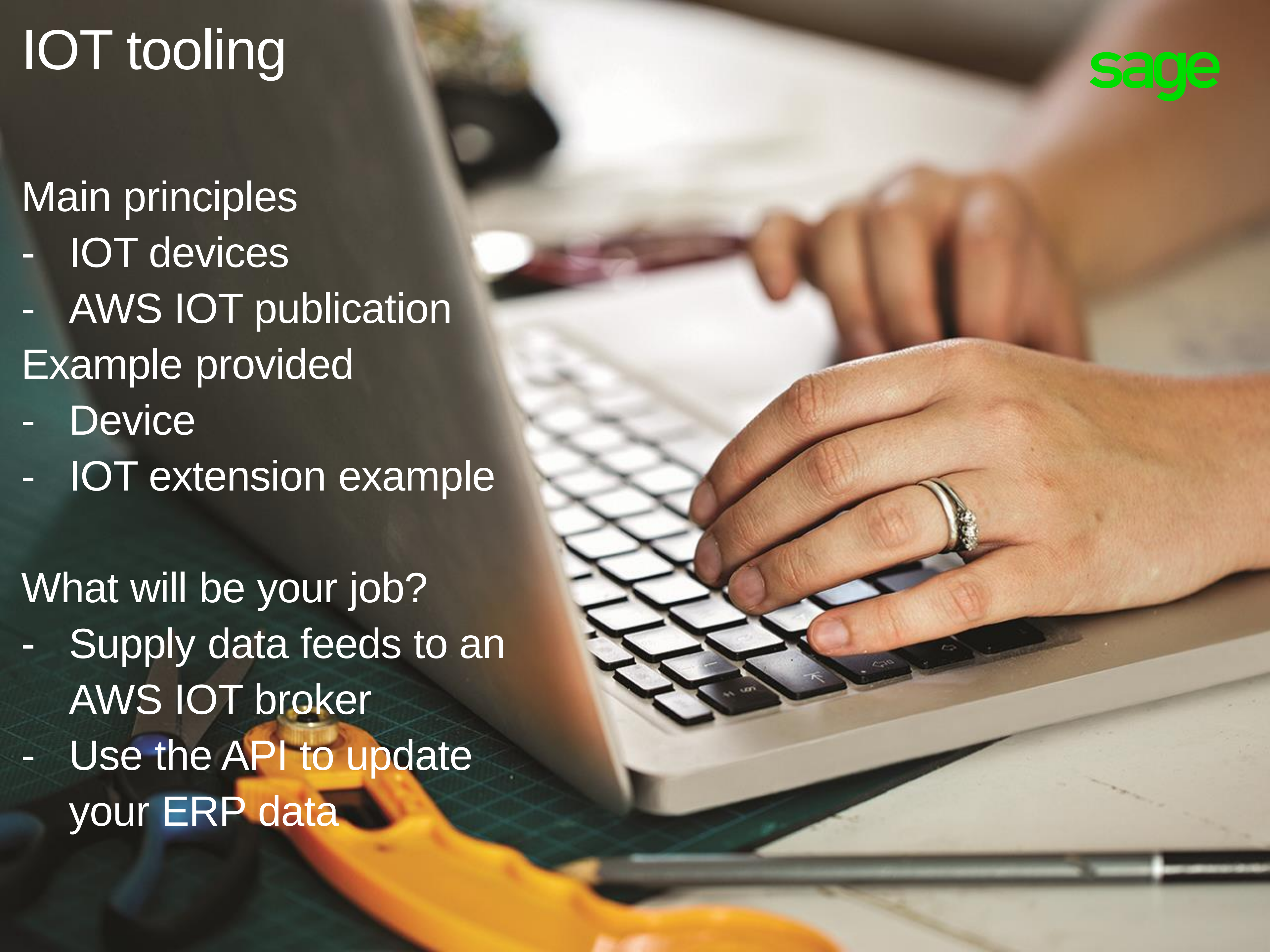
IOT Hack-athon

Dominique BOPP
June 7, 2016



IOT tooling

Main principles
- IOT devices
- AWS IOT publication

Example provided
- Device
- IOT extension example

What will be your job?
- Supply data feeds to an AWS IOT broker
- Use the API to update your ERP data

IOT devices

Can be any kind of device

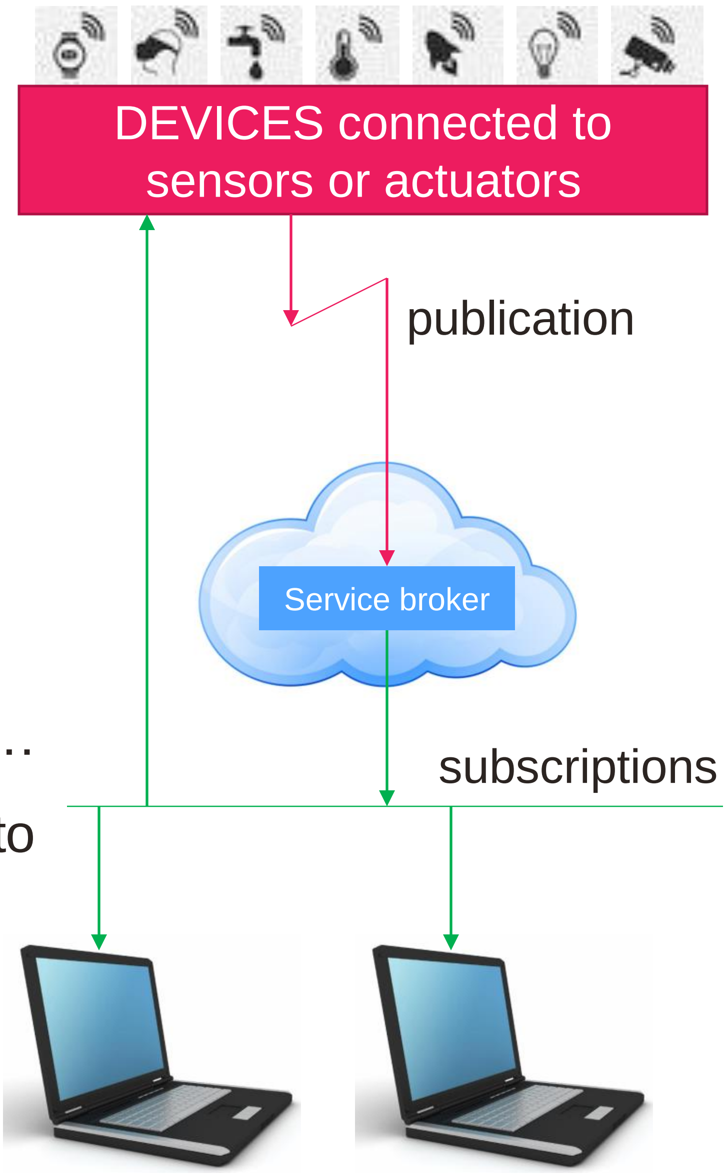
- Raspberry PI
- Arduino cards
- …

Connects on a IOT service broker

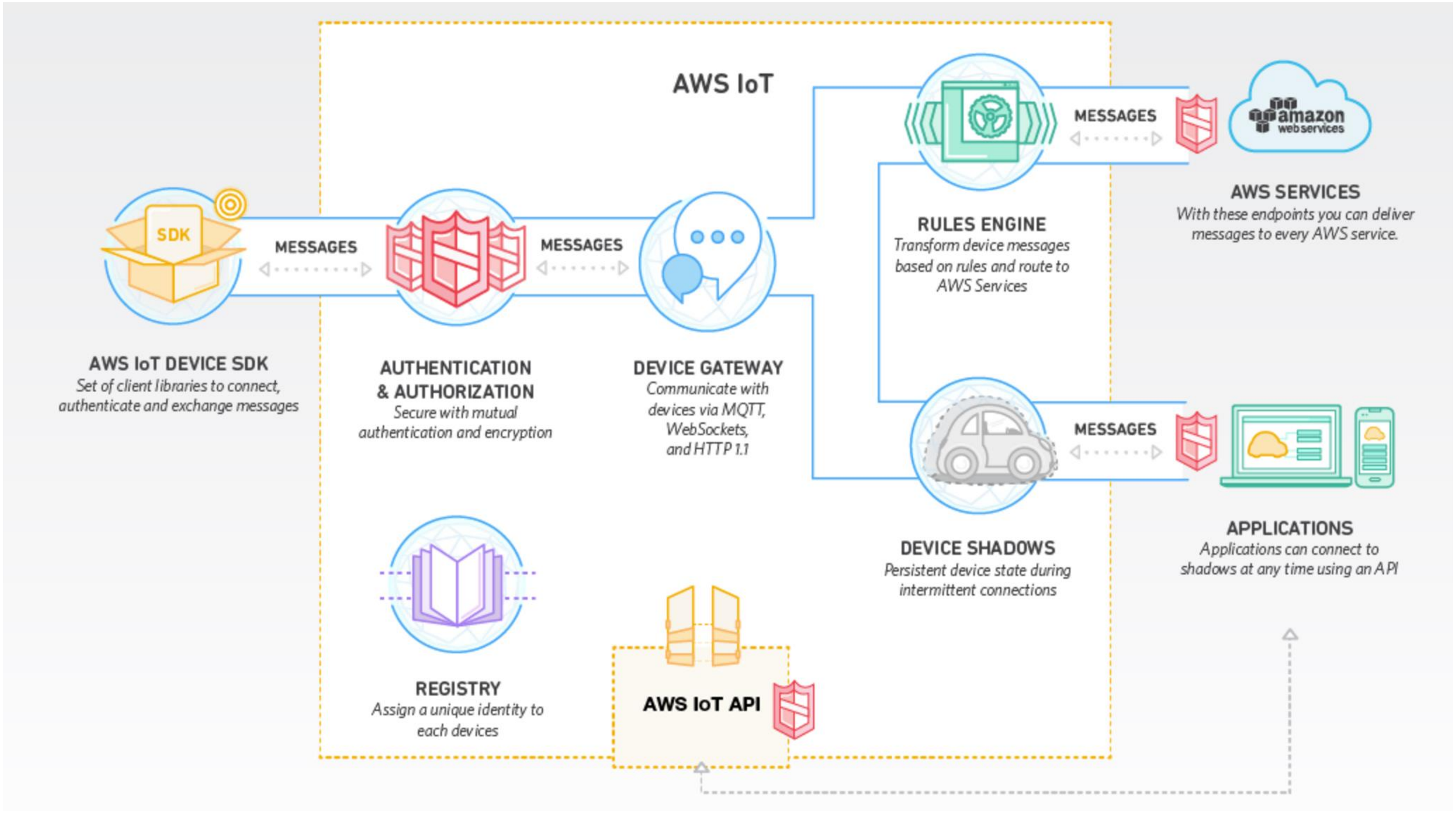
- With an identification based on a *topic*
- Sends back (or read) data feeds
- Using a protocol such as MQTT, WebSockets…
- Open source implementation such as Mosquitto
- Public available services such as Amazon IOT

Clients can subscribe to an IOT service

- They will get the data sent by the broker



AWS IOT principles


AWS IoT
AWS IoT DEVICE SDK
Set of client libraries to connect, authenticate and exchange messages
SDK
MESSAGES
AUTHENTICATION & AUTHORIZATION
Secure with mutual authentication and encryption
MESSAGES
DEVICE GATEWAY
Communicate with devices via MQTT, WebSockets, and HTTP 1.1
RULES ENGINE
Transform device messages based on rules and route to AWS Services
MESSAGES
amazon webservices
AWS SERVICES
With these endpoints you can deliver messages to every AWS service.
DEVICE SHADOWS
Persistent device state during intermittent connections
MESSAGES
APPLICATIONS
Applications can connect to shadows at any time using an API
REGISTRY
Assign a unique identity to each devices
AWS IoT API

AWS IOT publication principles

Create a device in the Thing registry

http://docs.aws.amazon.com/iot/latest/developerguide/create-device.html

Create a certificate and save the key

http://docs.aws.amazon.com/iot/latest/developerguide/create-device-certificate.html

Define a policy that allows to publish and consume the feeds

http://docs.aws.amazon.com/iot/latest/developerguide/create-iot-policy.html

```
{
  "Version": "2012-10-17",
  "Statement": [
      {
          "Action": ["iot:*“],
          "Resource": ["*"],
          "Effect": "Allow“
      }]
}
```

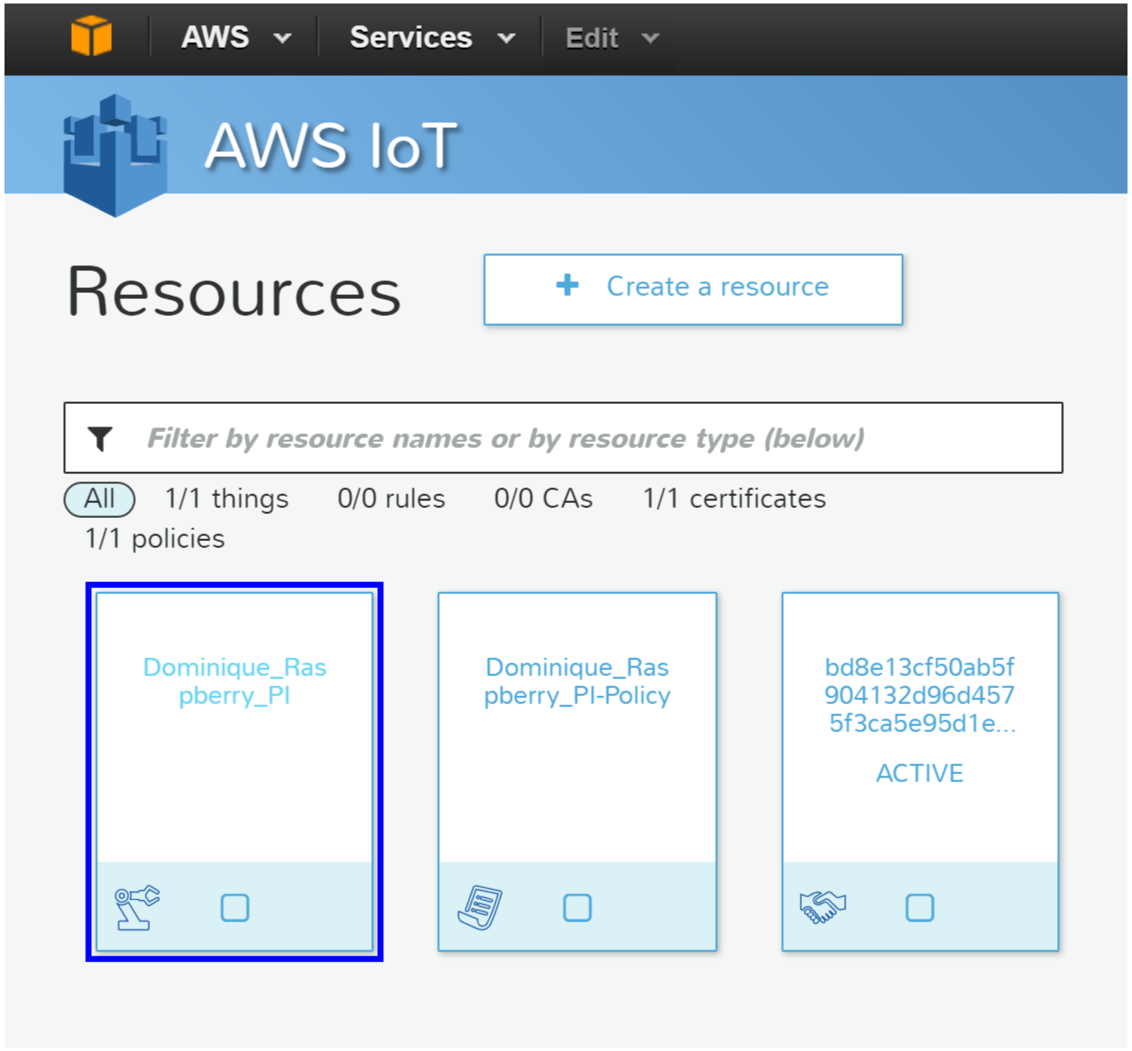
Install the certificate on your device

Call the AWS API to publish data from your device in JSON format

```
{
  “field1":“string_value“,
  “field2":52
  “field3":”2016-05-29T15:20:30Z”
}
```

What is already provided on Amazon IOT platform

- An IOT thing on which data feeds can be sent
- Security based on wss with an access key and a key secret
- The certificate or the access key and key secret can be provided if you have a device that wants to publish data or a given topic from this thing



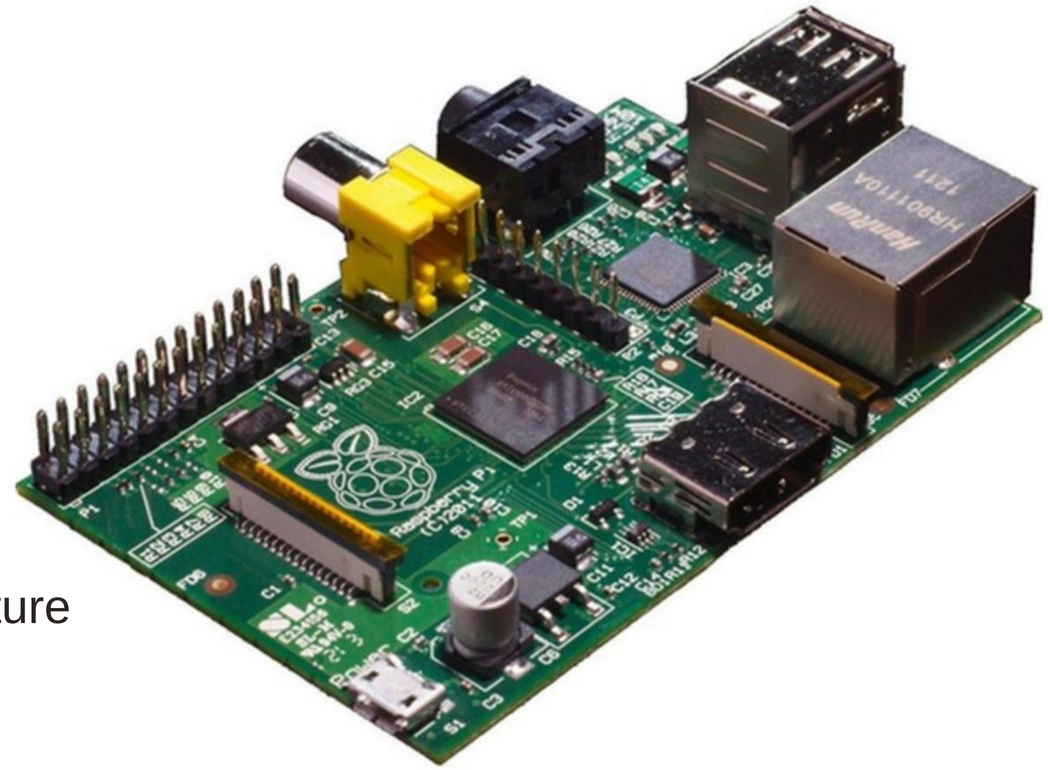
What is already provided for devices

- A Raspberry connected to a temperature and humidity sensor
- A node.js small application that publishes the measures periodically on AWS IOT
- This device can be activated on demand and publishes on the topic :

  sage/iot/workshop/temperature

- The feed sent has 3 values:

```
{
  "stamp":"2016-05-20T15:37:27.263Z“,
  "temperature":28.1
  "humidity":39.1
}
```



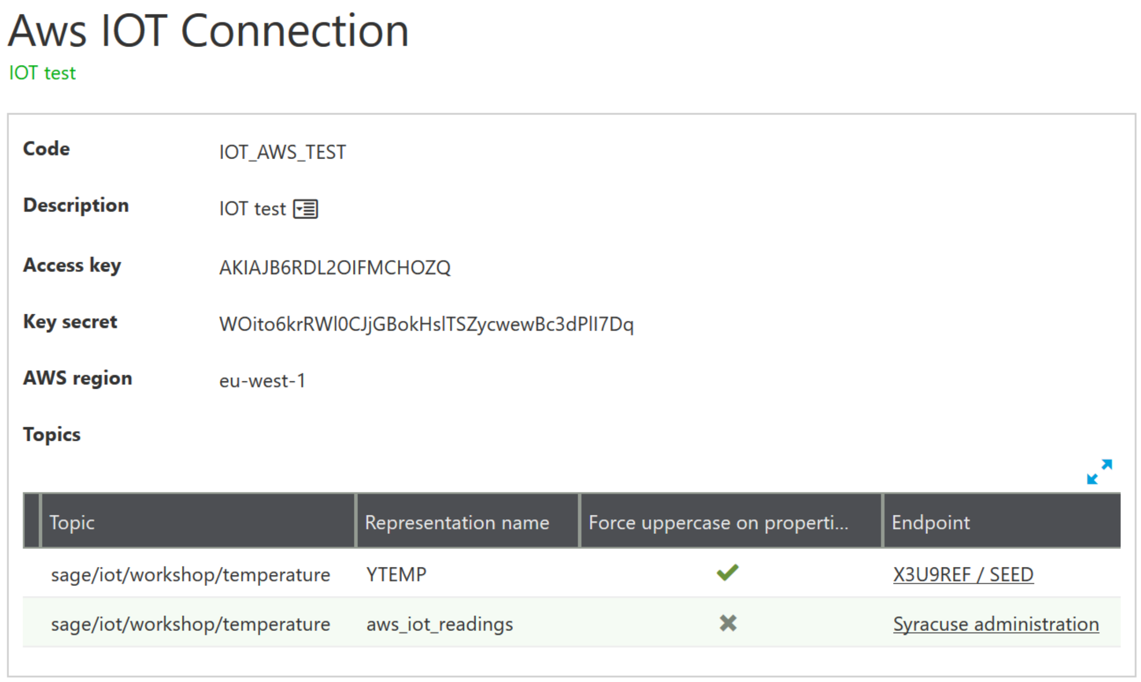
What is already provided for Sage X3

- A javascript extension module has been created as a public github code sample (https://github.com/Sage-ERP-X3/sample-x3-aws-iot)
- This extension has been installed on the Amazon instances you will use, and an example of set-up with the right credentials is supplied
- A set-up function (aws-iot-connections) allows to define an association between topics and entities:
  - Can be administration entities stored in MongoDB (one is available, called aws_iot_readings)
  - Can be a Sage X3 entity on a given endpoint
  - The property name found in he feed with be mapped with entities (the property names can be transformed in upper case)
- Every time a data feed is received, the data is sent to the entity with an insertion method

What is already provided for Sage X3

AWS IOT connection sample

- Access key, key secret, and AWS region provided
- For every topic, a representation name, the endpoint and a checkbox to transform properties in uppercase is given
- Subscribe to topics starts the process of calling creation method every time an event is received (the payload is mapped to the properties)



Aws IOT Connection

IOT test

Code	IOT_AWS_TEST
Description	IOT test
Access key	AKIAJB6RDL2OIFMCHOZQ
Key secret	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
AWS region	eu-west-1
Topics	

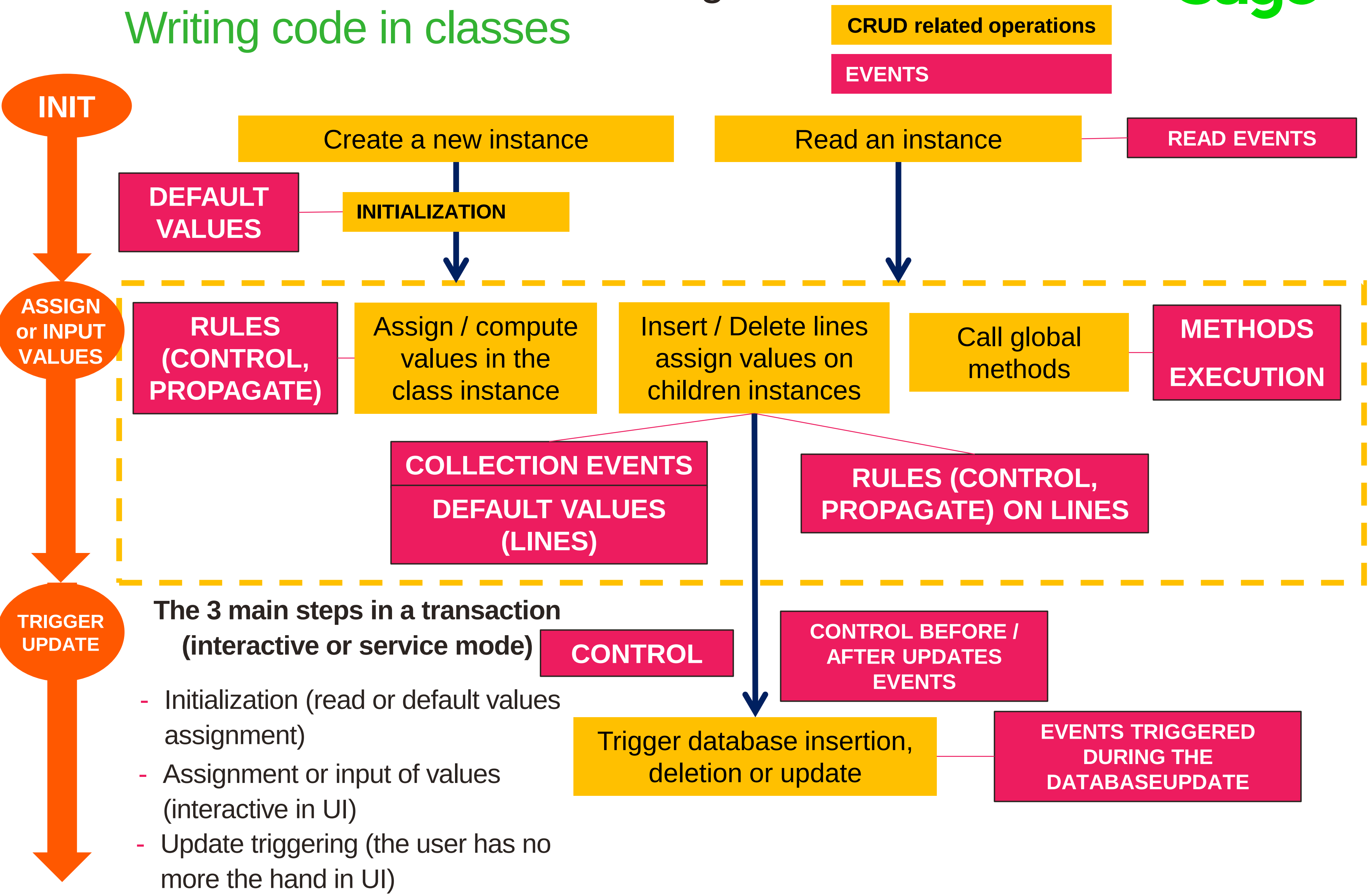
Topic	Representation name	Force uppercase on properti...	Endpoint
sage/iot/workshop/temperature	YTEMP	✓	X3U9REF / SEED
sage/iot/workshop/temperature	aws_iot_readings	✗	Syracuse administration

What will be your job?

You can either :

- provide a service that publishes on Amazon IOT (you will be a provider):
  - Running on a device provided
  - Using any technology stack to publish it (the node.js example will be provided)
- Consume a service and manage the result in Sage X3
  - Create a class (persistent or interface) with at least the insertion method available
  - Write the corresponding code in events to handle the event
  - Associate the class with a provider and consume the corresponding feed
    - You can consume the temperature / humidity feed provided by default
    - You can associate with a provider and consume the corresponding feed
  - Check and demo the result

As a reminder: Classes management
Writing code in classes


CRUD related operations
EVENTS
INIT
Create a new instance
Read an instance
READ EVENTS
DEFAULT VALUES
INITIALIZATION
ASSIGN or INPUT VALUES
RULES (CONTROL, PROPAGATE)
Assign / compute values in the class instance
Insert / Delete lines assign values on children instances
Call global methods
METHODS EXECUTION
COLLECTION EVENTS
DEFAULT VALUES (LINES)
RULES (CONTROL, PROPAGATE) ON LINES
TRIGGER UPDATE
CONTROL
CONTROL BEFORE / AFTER UPDATES EVENTS
Trigger database insertion, deletion or update
EVENTS TRIGGERED DURING THE DATABASEUPDATE

The 3 main steps in a transaction (interactive or service mode)

- Initialization (read or default values assignment)
- Assignment or input of values (interactive in UI)
- Update triggering (the user has no more the hand in UI)

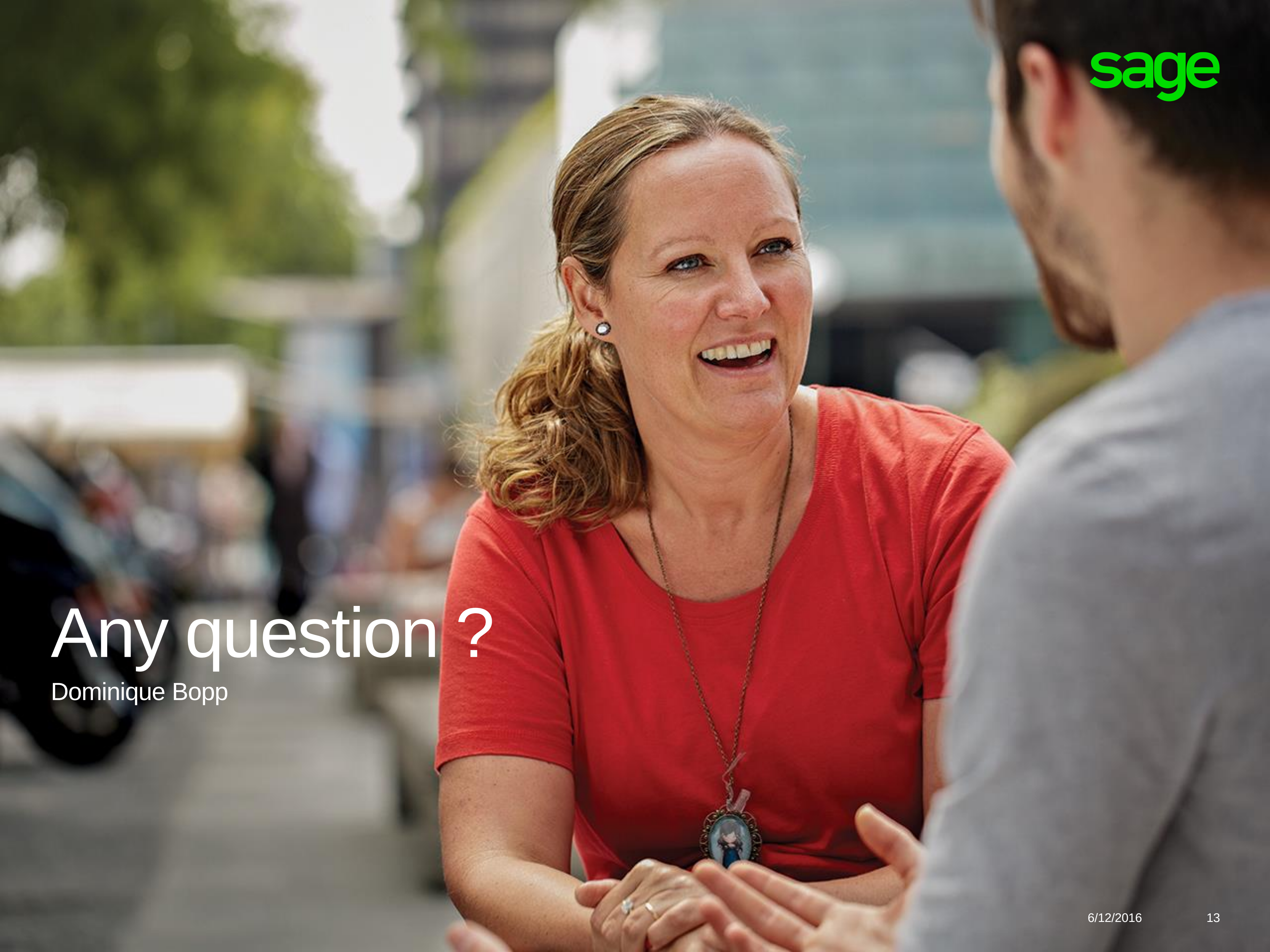
Just a reminder about class management

If you set-up the insertion method, the following event will be called:

Events Called	Context
AINSERT	Instance filled, all the controls have been done, insertion is requested (it can be for a line during an update on a complex document), transaction in progress.
AINSERT_ROLLBACK	Only if Rollback triggered by ASETERROR method in a previous event.
AINSERT_CONTROL_BEFORE	Before the insertion of a line in the instance, and before the controls on the fields (can be used to assign default values on properties).
AINSERT_CONTROL_AFTER	After all the controls have been done, before the database insert operation.

On properties, the following rules can be called:

Events called	Context
INIT	Called for every property by the supervisor layer when a creation is requested (not for a modification). Used to give default values. Also done at the end of the initialization with CURPRO=""
GET	Called when a value of a property has to be used. If the value needs to be computed, the computation will be done in the event and assigned to this.PROPERTY
CONTROL	Called every time a property modification is executed. The initial value is available through this.snapshot. No assignment can be done here, but an error can be thrown if the new value is not accepted.
PROPAGATE	Called every time a property has been modified. Assignment of other properties can then be done.



Any question ?

Dominique Bopp