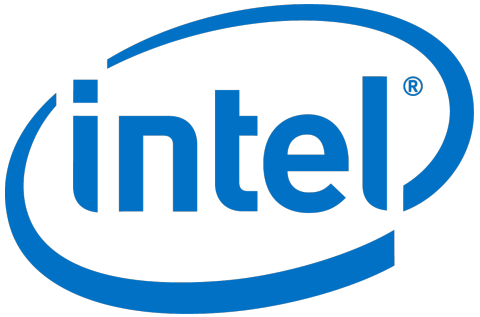


EDGE X FOUNDRY™

CONEXXUS
solve forward

TAMING IOT FOR RETAIL:

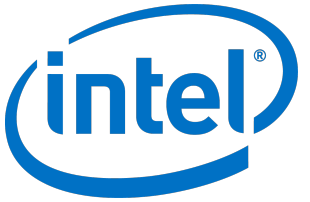
How EdgeX Foundry, the Intel Open Retail Initiative, and the Web of Things are Transforming how Retail Operates

Kristen Call and Michael McCool

1 May 2020

OUTLINE

- Intel's Open Retail Initiative
  - Retail modernization, consolidation, and use cases
- EdgeX Foundry
  - Software stack for edge computing
  - Diverse set of backend device interfaces
  - Unified front-end network interface
- Applications of Edge Computing in Retail
  - Virtual sensors
  - Local decision loops and orchestration
- W3C Web of Things
  - Standard way to describe IoT devices and services
  - Supports rapid IoT orchestration and adaptation



THE OPEN RETAIL INITIATIVE

Accelerating innovation at scale with open accessible solutions

What is it?
The goal of the Open Retail Initiative (ORI) is to accelerate and scale deployment of data-rich retail solutions optimized for physical retail from the edge to cloud.

ORI is a technology collaboration leveraging open source projects, along side vendor proprietary solutions to deliver recipes and ingredients that address industry problems.

What problem(s) are we solving?

- Environmental complexity due to lack of interoperability
- Deployment of data-driven experiences
- Lack of unified standards impeding digital transformation efforts

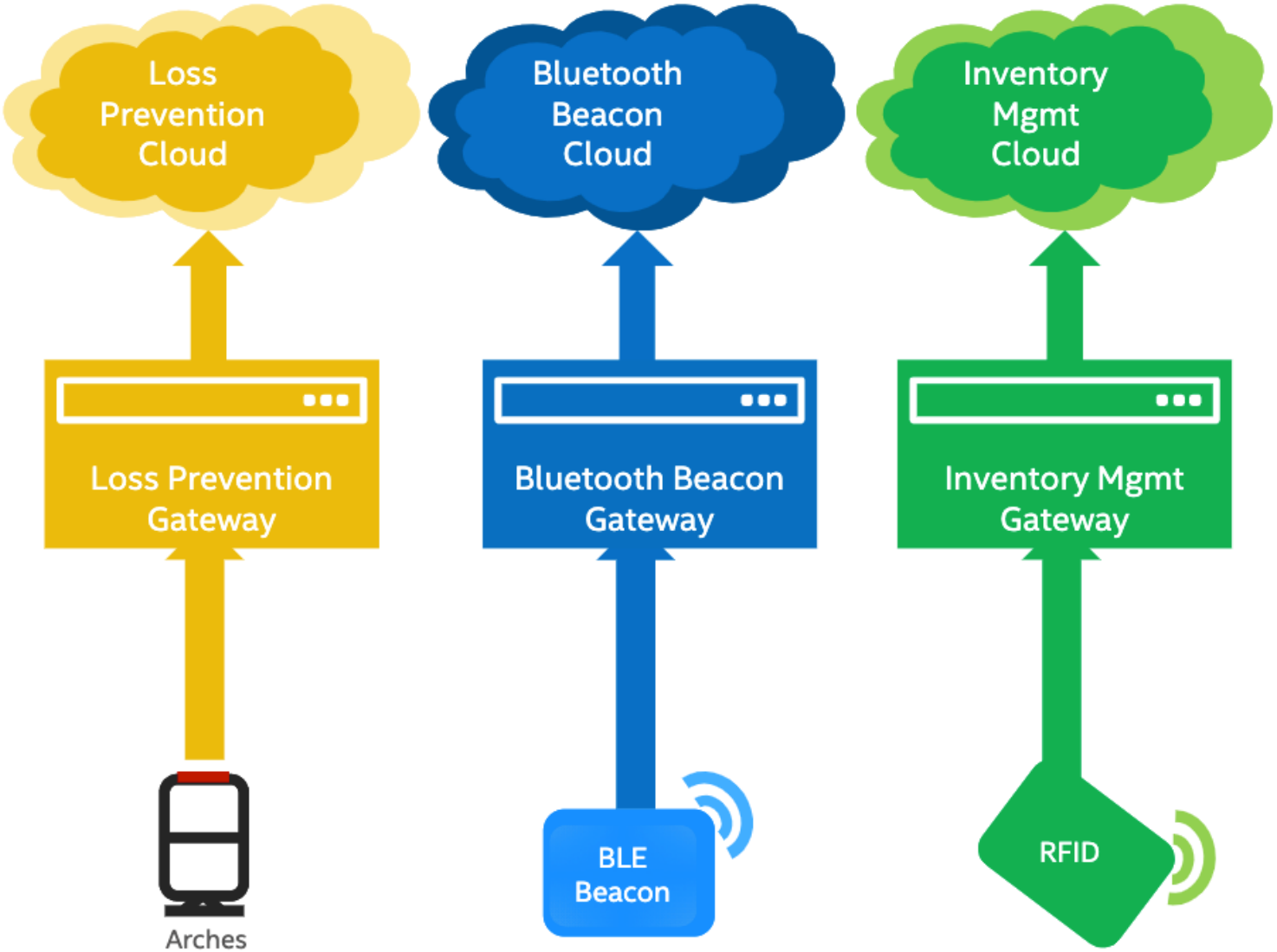
Solution Approach
A common open framework to enable an ecosystem of inter-changeable components and accessible data with ease and speed.

Who will benefit from this initiative?

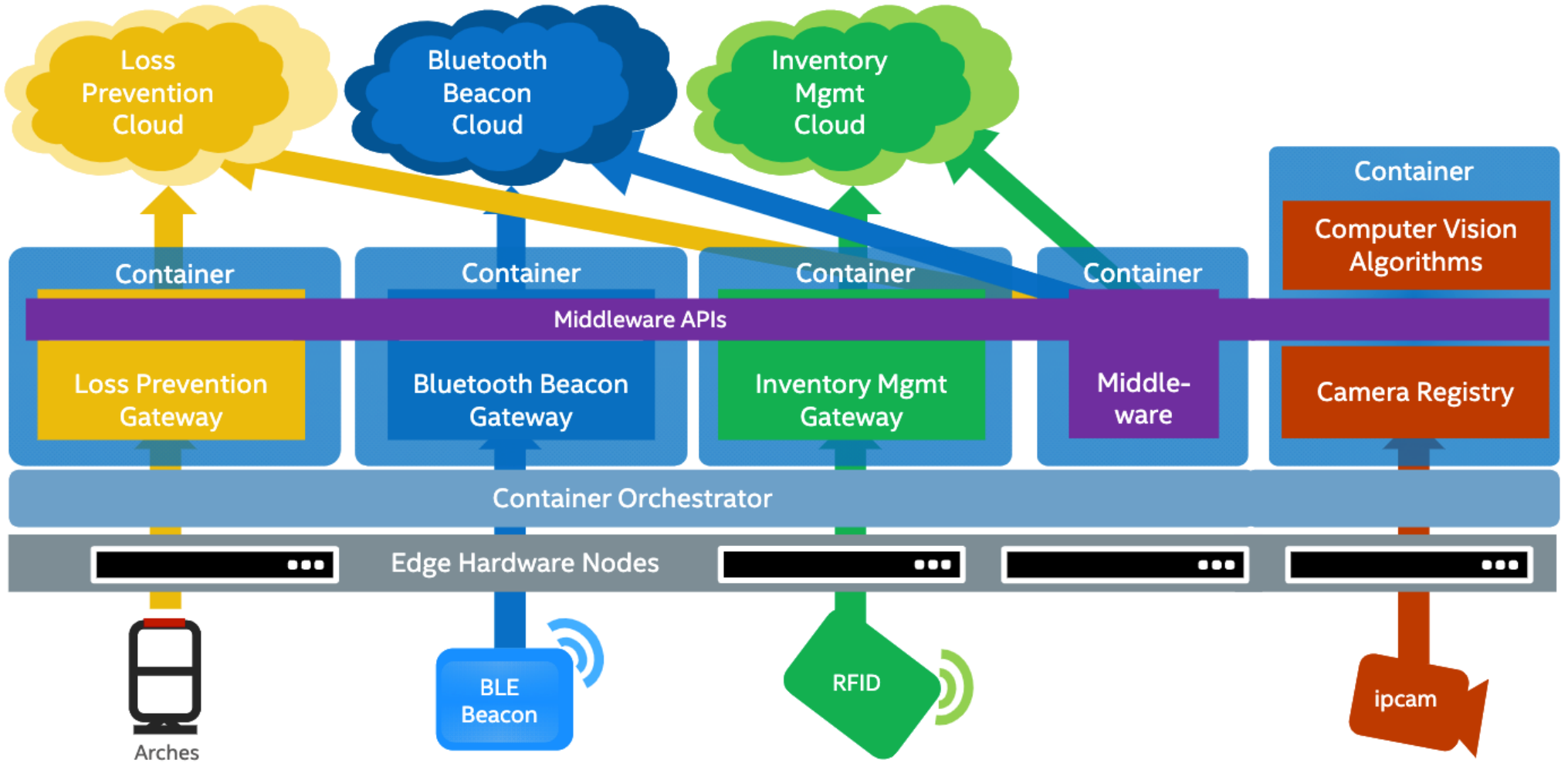
- Architects, retail solution software developers and infrastructure developers
- OEMs/ODMs, OS,OSM, ISVs and SI's
- End customers/retailers/brands

Application
Data-rich retail solutions leveraging artificial intelligence, computer vision and Internet of Things for retail, grocery, C-store, QSR, etc.

CURRENT SITUATION: MULTIPLE SILOS



TREND: CONSOLIDATION AND INTEGRATION



RETAIL MODERNIZATION USE CASES

BRINGING A USE CASE TO LIFE: BUYING AN ICE CREAM?

Camera captures image and sensor detects freezer door opening

- Camera
- Freezer Door Sensor

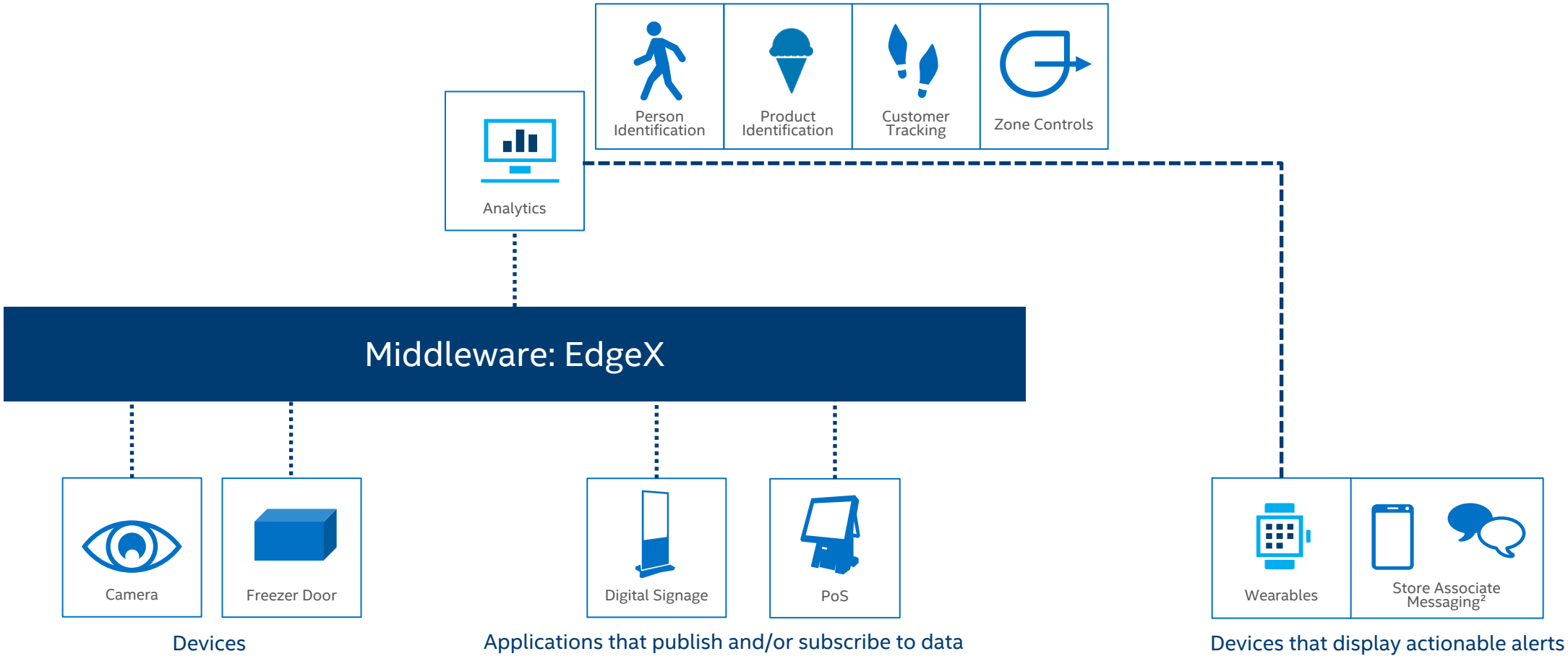
Data analytics identifies person and product

- Person identification
- Person tracking
- Product identification

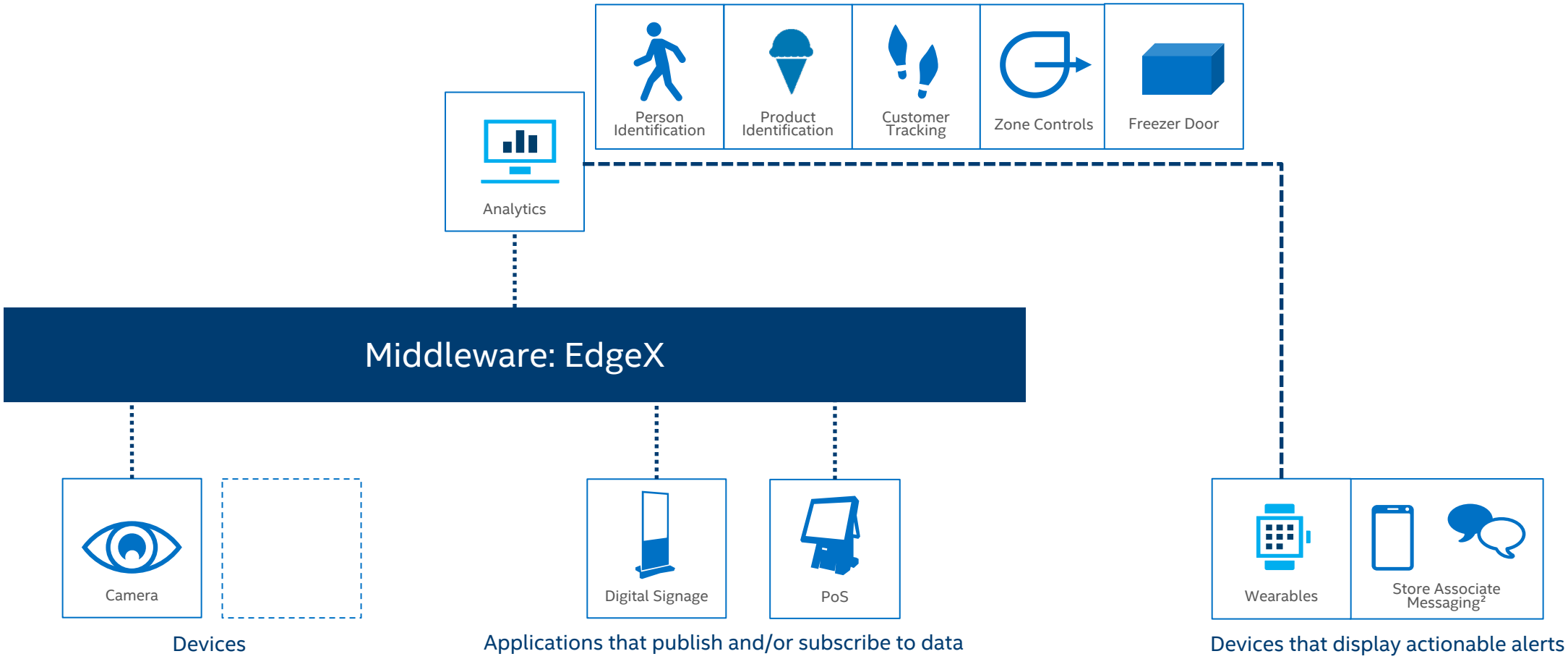
Alert based on location and behavior

To counter:	To door:
Associate facing alert: Customer waiting	Associate facing alert: Zone violation
Customer facing alert: Price display	Customer facing alert: Door “reminder” display

HIGH LEVEL DIAGRAM OF IMPLEMENTATION

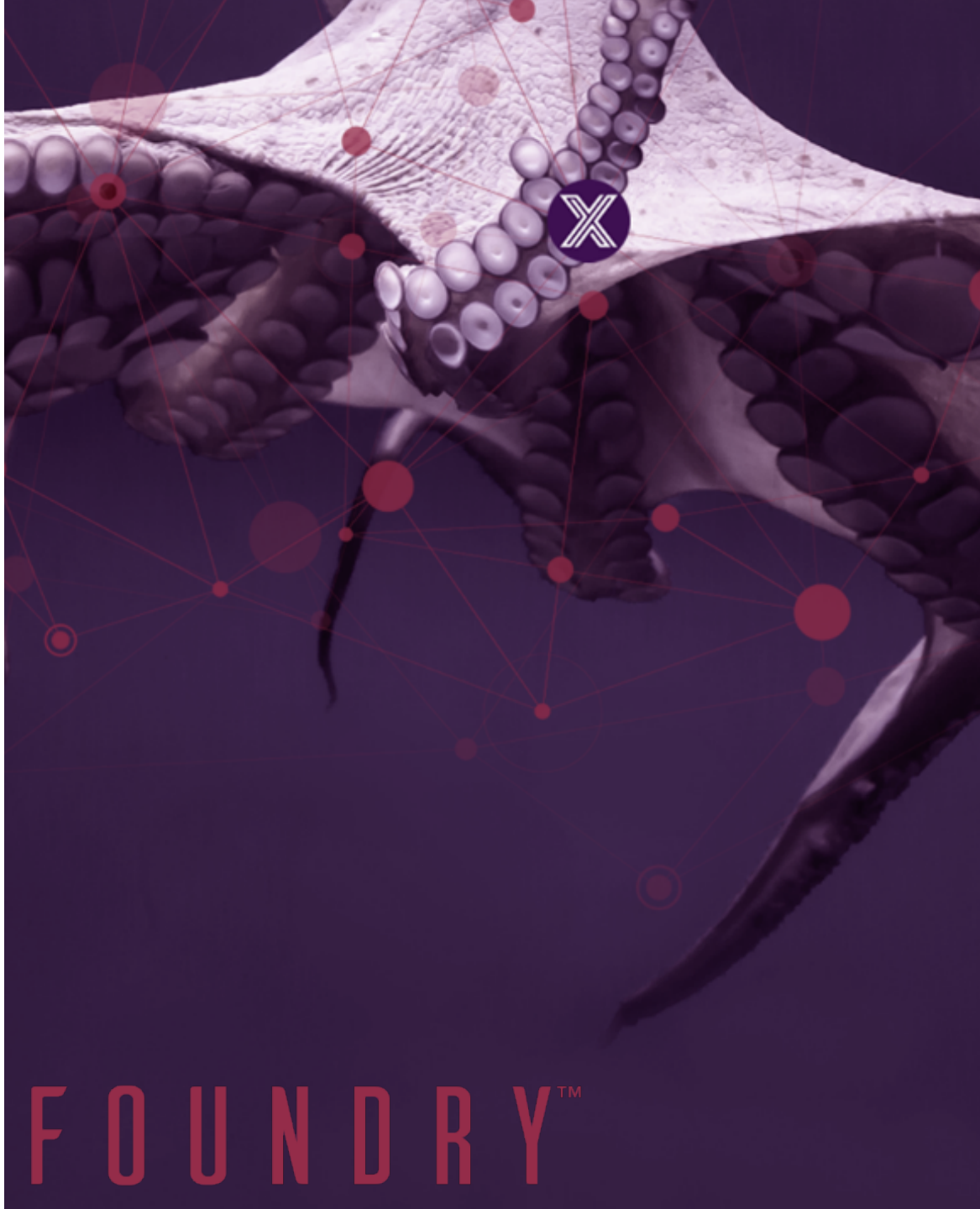


ADAPTATION: REPLACING REAL DEVICE WITH VIRTUAL DEVICE



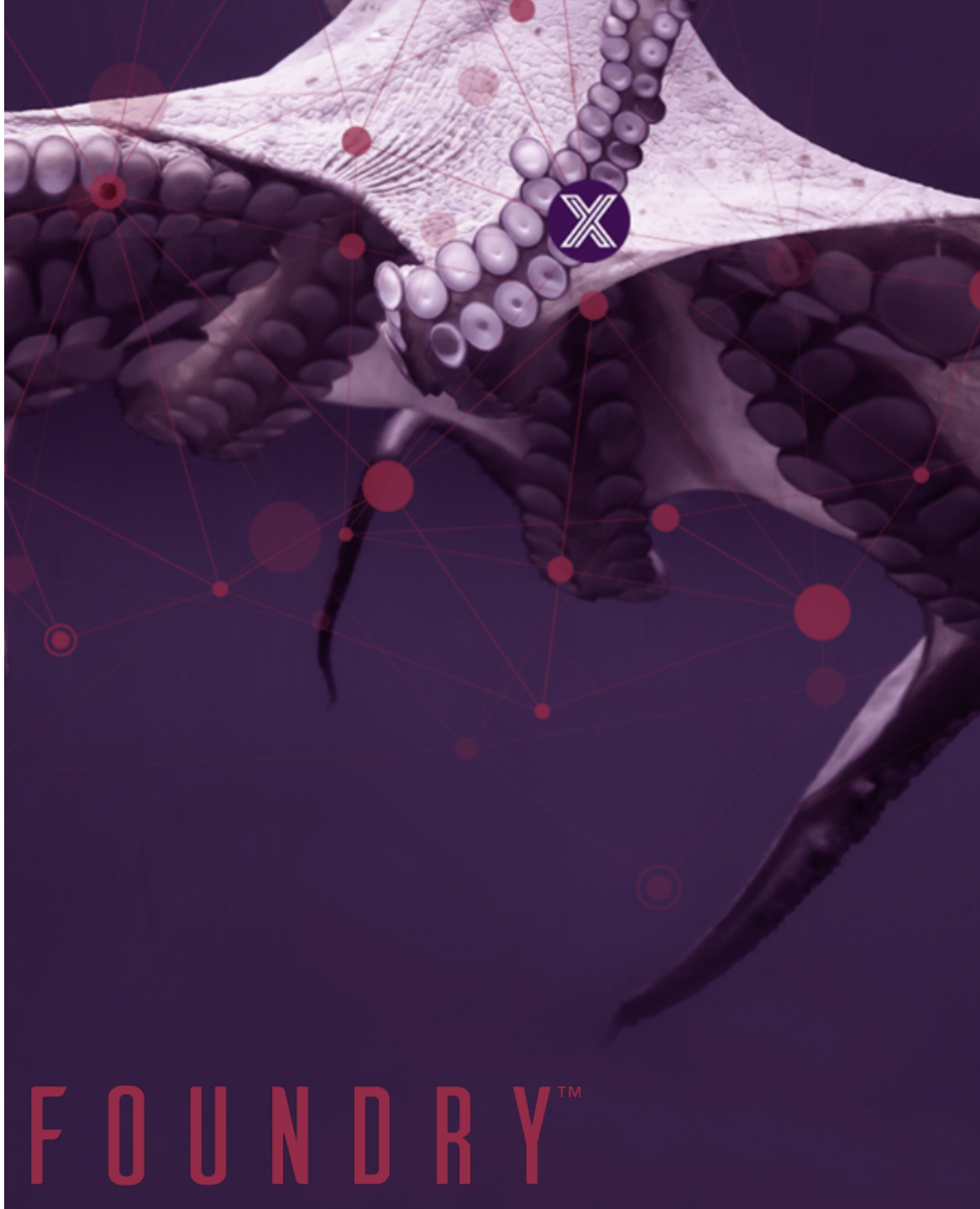
WHAT IS EDGEX FOUNDRY?

- An open source, vendor neutral software project (and ecosystem)
- A microservice, loosely coupled software framework for IoT and edge computing
- Hardware and OS agnostic
- Linux Foundation
  - Apache 2 project
  - Started April 2017



EDGEX MICROSERVICE ARCHITECTURE

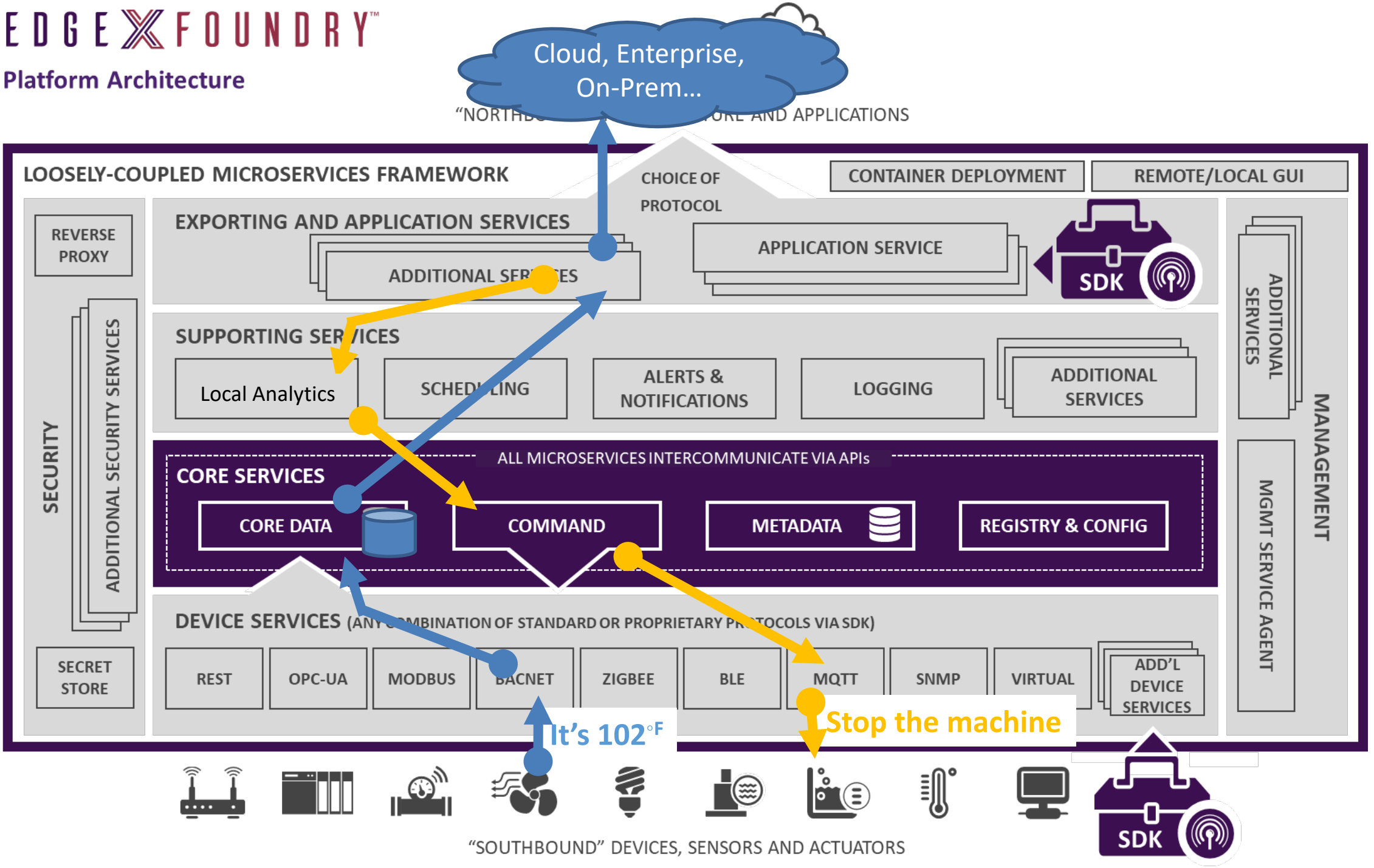
- Sensor data is collected by a Device Service
- Data is passed to the Core Services for local persistence
- Data is then passed to Application Services for transformation, formatting, filtering and can then be sent to enterprise/cloud systems
- Data is then available for edge analysis and can trigger device actuation through Command Service
- Many other services provide the supporting capability that drives this flow



EDGE X FOUNDRY™

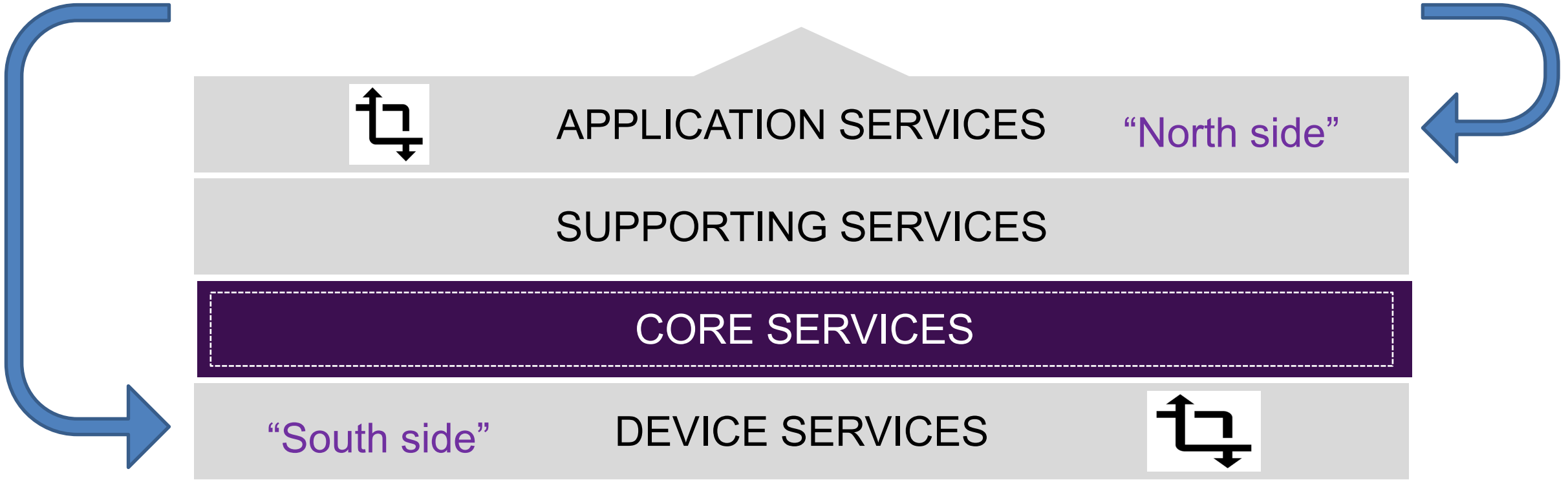
EDGE X FOUNDRY™

Platform Architecture



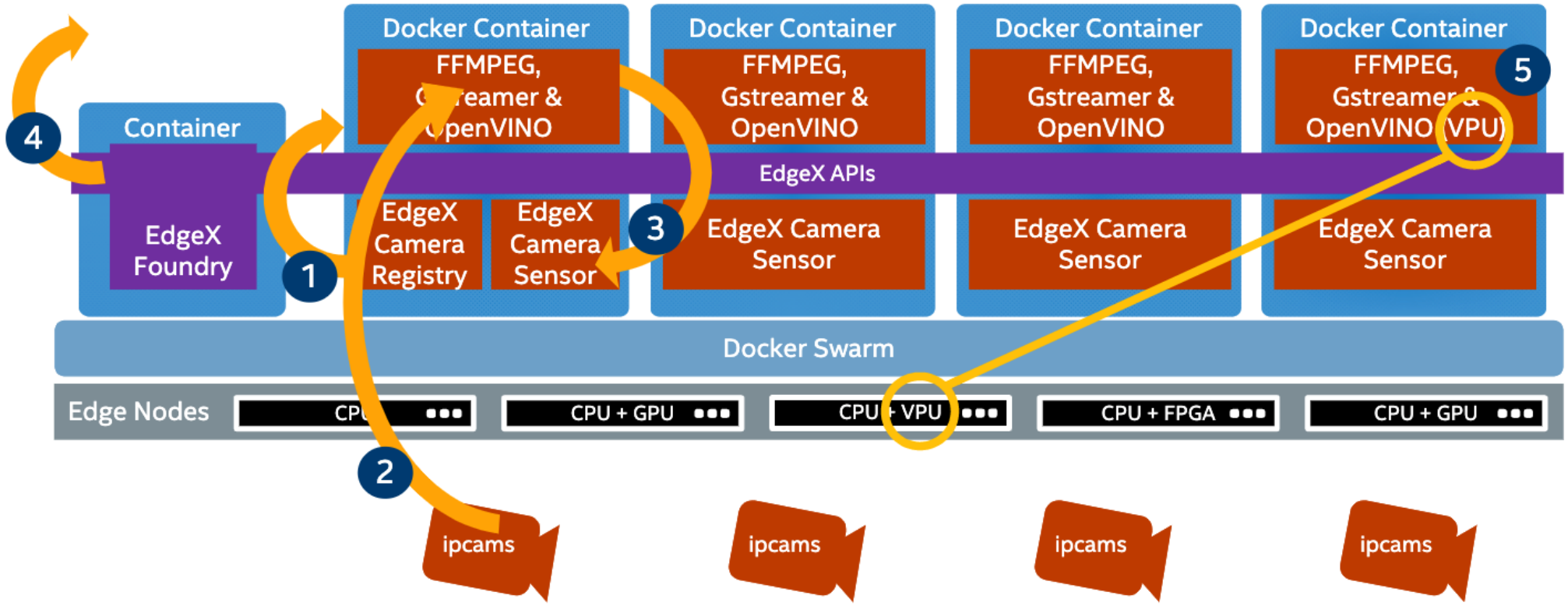
DUAL TRANSFORMATION ENGINE

- The layers (and services) of EdgeX constitute a dual transformation engine
  - 1x - Translating information coming from sensors and devices via hundreds of protocols and thousands of formats into EdgeX
  - 2x - Delivering data to applications, enterprises and cloud systems over TCP/IP based protocols in formats and structures of customer choice



EDGEX SUPPORT FOR COMPUTER VISION

1. Gstreamer pipelines are spun up based on camera tags
2. IP cameras stream data to Gstreamer pipelines
3. Inference events are published to EdgeX
4. EdgeX makes inference available for apps and export
5. Containers utilize accelerators when available



NEED FOR END-USER PROGRAMMABILITY

Adaptation to Available Devices

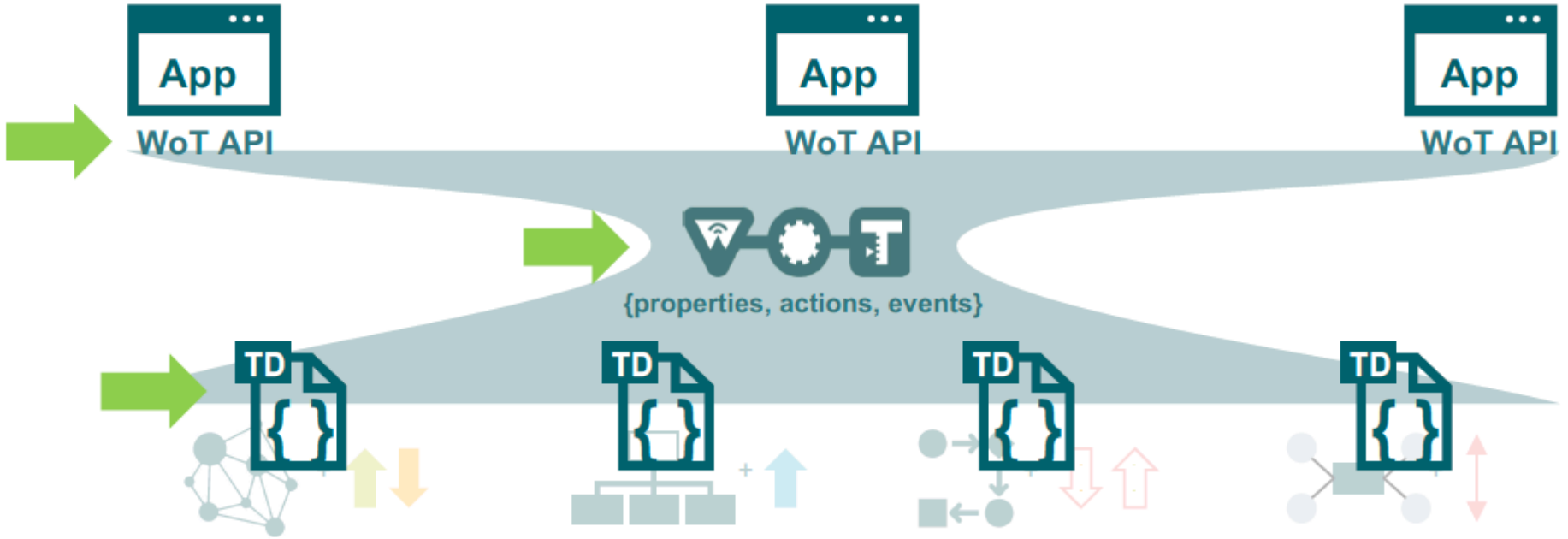
- In order to consolidate services, devices need to be shared
- Services developed in a “siloed” fashion assuming exclusive access to devices cannot be consolidated
- Services may have to adapt to (or be adapted to) substitute devices

Rapid/Easy Development

- Adapting services to a given environment may require some “end-user programming”
- This should be as simple as possible
- Tools should support device and service connection, substitution, and sharing

W3C WEB OF THINGS (WOT)

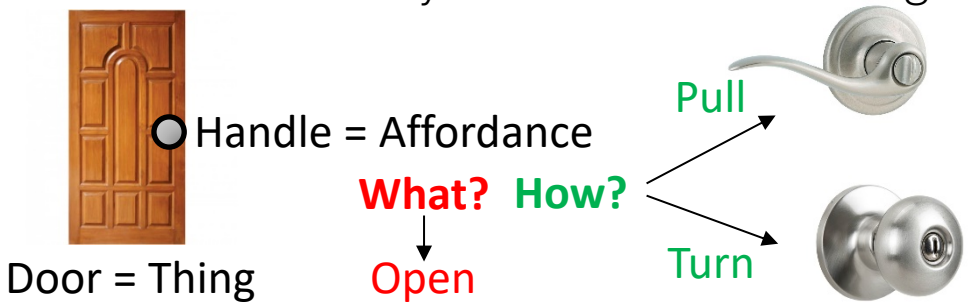
- W3C Working Group whose goal is adapting web technologies to IoT
- Initial standard is a Thing Description (TD) metadata format
  - TD describes the interactions of an IoT device (“Thing”)
  - Applications use a high-level interaction model to interact with Things
  - Interaction model is based on simple properties, actions, and events
- Low-level details are automated
  - TDs also include protocol binding information for low-level protocols
  - It is not necessary for applications to deal with low-level protocols



WEB OF THINGS - DESCRIPTIVE APPROACH TO INTEROPERABILITY

WOT ARCHITECTURE

- Constraints
  - Things must have TD (W3C WoT)
  - Must use hypermedia controls (general WoT)
    - URIs
    - Standard set of methods
    - Media Types
- Interaction Affordances
  - Metadata of a Thing that shows and describes the possible choices (what) to Consumers, thereby suggesting how Consumers may interact with the Thing

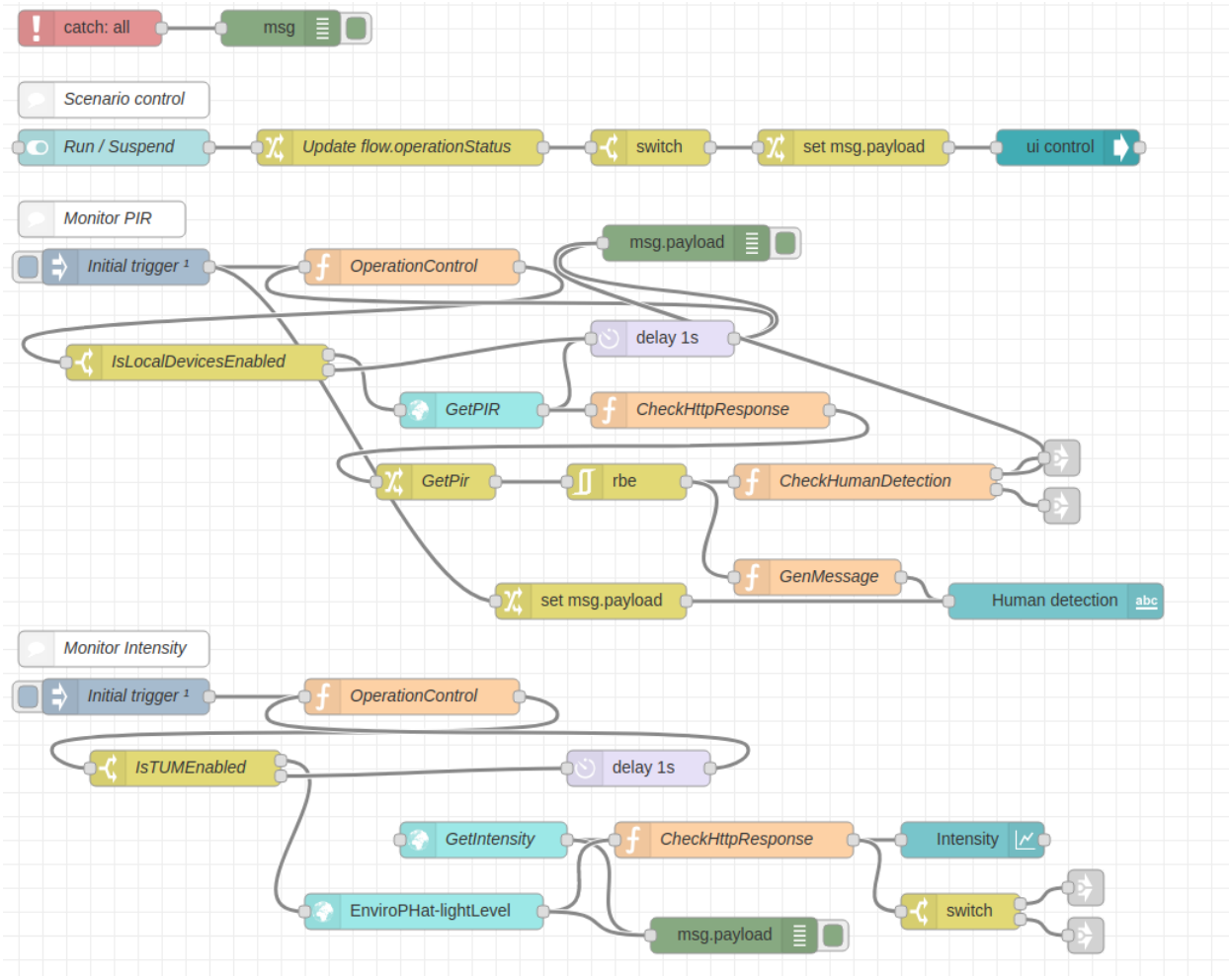


WOT THING DESCRIPTION (TD)

```
{
  "@context": [
    "https://www.w3.org/2019/wot/td/v1",
    { "iot": "http://iotschema.org/" }
  ],
  "id": "urn:dev:org:32473:1234567890",
  "title": "MyLEDThing",
  "description": "RGB LED torchiere",
  "@type": ["Thing", "iot:Light"],
  "securityDefinitions": ["default": {
    "scheme": "bearer"
  }],
  "security": ["default"],
  "properties": {
    "brightness": {
      "@type": ["iot:Brightness"],
      "type": "integer",
      "minimum": 0,
      "maximum": 100,
      "forms": [ ... ]
    }
  },
  "actions": {
    "fadeIn": {
      ...
```

WOT ORCHESTRATION OPTIONS

Node-RED/node-gen



WoT TD benefit: Module autopopulation

node-wot

```
WoTHelpers.fetch( "coap://localhost:5683/counter" ).then( async (td) => {
 // using await for serial execution (note 'async' in then() of fetch())
 try {
  let thing = await WoT.consume(td);
  console.info( "=== TD ===" );
  console.info(td);
  console.info( "==========" );

  // read property #1
  let read1 = await thing.readProperty( "count" );
  console.info( "count value is" , read1);

  // increment property #1 (without step)
  await thing.invokeAction( "increment" );
  let inc1 = await thing.readProperty( "count" );
  console.info( "count value after increment #1 is" , inc1);

  // increment property #2 (with step)
  await thing.invokeAction( "increment" , {'step' : 3});
  let inc2 = await thing.readProperty( "count" );
  console.info( "count value after increment #2 (with step 3) is" , inc2);

  // decrement property
  await thing.invokeAction( "decrement" );
  let dec1 = await thing.readProperty( "count" );
  console.info( "count value after decrement is" , dec1);

 } catch(err) {
  console.error( "Script error:" , err);
 }

}).catch( (err) => { console.error( "Fetch error:" , err); });
```

EDGEX APIS AND WOT METADATA

- EdgeX includes a “metadata service”
- APIs for device access translate protocols into a common protocol
  - Based on HTTP
  - Payloads use JSON or CBOR
  - Events use ZeroMQ
- Service APIs have OpenAPI descriptions

EDGE X FOUNDRY™

Platform Architecture

"NORTHBOUND" INFRASTRUCTURE AND APPLICATIONS

LOOSELY-COUPLED MICROSERVICES FRAMEWORK

CHOICE OF PROTOCOL

CONTAINER DEPLOYMENT

REMOTE/LOCAL GUI

REVERSE PROXY

EXPORTING AND

WoT Scripts/Node-RED

ADDITIONAL SERVICES

APPLICATION SERVICE

SDK

SUPPORTING SERVICES

RULES ENGINE

SCHEDULING

ALERTS & NOTIFICATIONS

LOGGING

ADDITIONAL SERVICES

ADDITIONAL SERVICES

MANAGEMENT

SECURITY

ADDITIONAL SECURITY SERVICES

ALL MICROSERVICES INTERCOMMUNICATE VIA APIs

CORE SERVICES

WoT TDs

CORE DATA

COMMAND

METADATA

REGISTRY & CONFIG

MGMT SERVICE AGENT

DEVICE SERVICES (ANY COMBINATION OF STANDARD OR PROPRIETARY PROTOCOLS VIA SDK)

SECRET STORE

REST

OPC-UA

MODBUS

BACNET

ZIGBEE

BLE

MQTT

SNMP

VIRTUAL

ADD'L DEVICE SERVICES

SDK

"SOUTHBOUND" DEVICES, SENSORS AND ACTUATORS

PLANNED WOT/EDGEX INTEGRATION

1. Generate WoT Thing Description metadata for all Device services
   - Including semantic tagging using One Data Model
2. Generate WoT Thing Description metadata for select Analytics services
   - Computer vision services
3. Prototype a Thing Directory service supporting semantic search
   - To run in parallel with existing EdgeX metadata service
   - Existing EdgeX discovery process would act as “Introduction” layer – WoT Discovery prototype
4. Support Rapid Orchestration Development
   - Using WoT Scripting API and node-wot
   - Using node-gen to generate Node-RED modules
5. Stand up retail use case examples that integrate IoT and Analytics:
   - Loss detection video analytics triggered by an IoT door sensor
   - Digital shelf signage/RFID and weight-based inventory control/item identification
   - Customized marketing content based on video analytics

CONCLUSION AND SUMMARY

- Intel's Open Retail Initiative supports development of open solutions for IoT in retail applications.
- EdgeX Foundry provides an open software platform for IoT and edge computing.
- W3C Web of Things simplifies adaption and configuration of services to a specific environment

Consolidation:

- Multiple services can share sensors and compute platforms

Multivendor:

- Open software frameworks and standards provide interoperability

Programmable:

- Users need an easy way to develop and adapt applications that consume data and deliver new services