# Scope/Motivation

The FMCG (Fast Moving Consumer Goods) simulator can be used as a basis for all kinds of consumer goods demos for which show the business value of Solace PubSub+ products. The simulator itself is in the background.

# Setup and Functionality

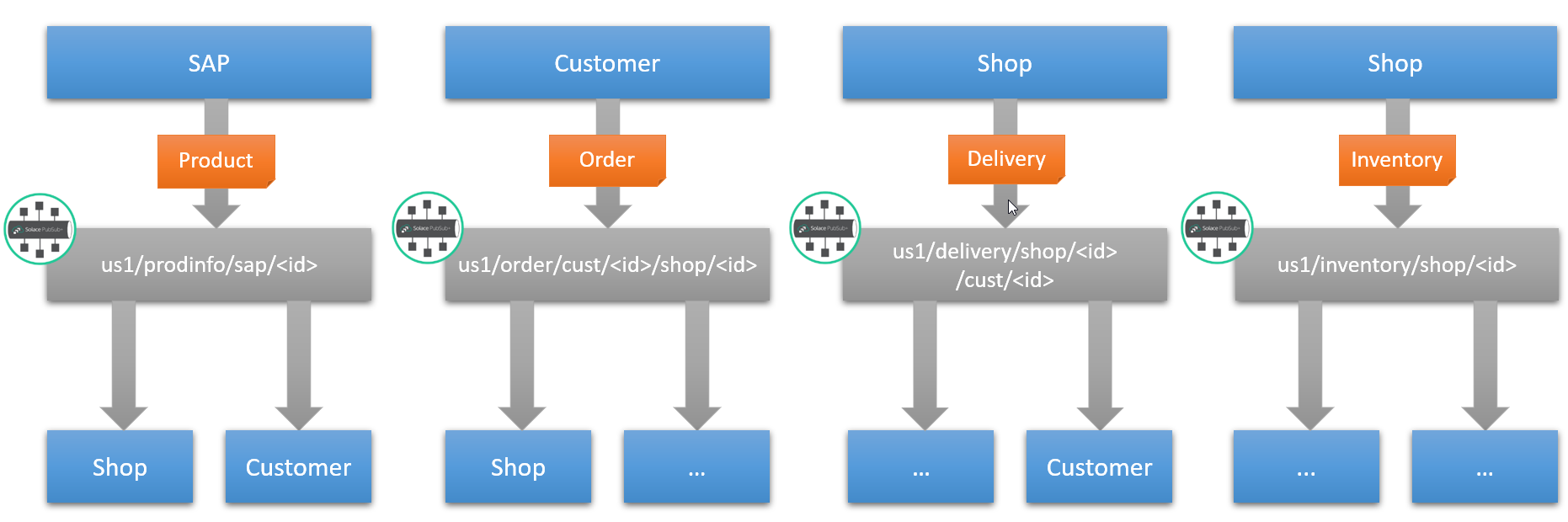
The simulator package is an all-in-one Java application which mocks

* SAP (Product masterdata)
* Customer (Ordering goods)
* Shop (Deivering goods). Shop might be e.g. physical store, online shop.

The program contains a built-in address and product database to avoid fiddly setup with configuration files.

All data is published to topics and can easily be subscribed by other applications for e.g. presentation or analytics.

The simulator supports the following flows:



All business entities have the same, hard initialized product and address database.

Customer simulator

* Publishes random orders to store and gets these delivered

Shop simulator

* Receives orders from customers
* Sends "Deliveries" to customers
* Publishes inventory
* Subscribes to product updates
* If stock too low then silently tops up the goods

SAP simulator

* Sends random product catalog updates

# Topic Structure

The topic is designed around organizational criteria.

<region>/<mtype>/<from-cat>/<from-id>/<to-cat>/<to-id>

<region>: us1, eu2, ch4, ...

<mtype>: Message type (business function): order, delivery, inventory, prodinfo

<from-cat>: Sender category: sap, cust, shop, wrhs

<to-cat>: Receiver category: sap, cust, shop, wrhs

<from-id>: Sender id

<to-id>: Receiver id

Reading example:

us1/order/shop/111111111/wrhs/333333333

In us1 region an order is sent from shop 111111111 to warehouse 333333333

customer does:

* publish order (to shop) us1/order/cust/222222222/shop/111111111
* subscribe delivery (from shops) us1/delivery/shop/\*/cust/222222222

shop does:

* subscribe order (from custs) us1/order/cust/\*/shop/111111111
* publish delivery (to cust) us1/delivery/shop/111111111/cust/222222222
* publish inventory updates us1/inventory/shop/111111111
* subscribe prodinfo updates us1/prodinfo/>

sap does:

* publish prodinfo us1/prodinfo/sap/444444444

# Data Structures

## Product Catalog

{

"utcDateTime": "2019-05-28T14:54:09.588Z",

"locDateTime": "2019-05-28T16:54:09.588+02:00",

"referenceId": "list-id",

"senderId": "100000000",

"items": [

{

"productId": "productId1",

"text": "Hotpants",

"category": "Clothing",

"size": "M",

"color": "Yellow",

"other": "Fancy",

"unit": "PC",

"currency": "USD",

"price": 12.25,

"status": "NEW",

"popularity": 3

},

{

"productId": "productId2",

"text": "Shovel",

"category": "Garden",

"size": "1.2m",

"color": "Grey",

"other": "Strong",

"unit": "PC",

"currency": "USD",

"price": 24.25,

"status": "UPD",

"popularity": 2

}

]

}

Enums:

**public** **enum** UnitType {

***PC***, ***KG***, ***M***, ***M2***, ***M3***, ***L***

}

**public** **enum** StatusType {

***NEW***, ***UPD***, ***DEL***

}

## Order and Delivery

{

"utcDateTime": "2019-05-28T14:55:12.234Z",

"locDateTime": "2019-05-28T16:55:12.234+02:00",

"referenceId": "list-id",

"sender": {

"id": "100000000",

"name": "sendername",

"street": "senderstreet",

"city": "sendercity",

"zip": "senderzip",

"country": "sendercountry"

},

"receiver": {

"id": "100000001",

"name": "receivername",

"street": "receiverstreet",

"city": "receivercity",

"zip": "receiverzip",

"country": "receivercountry"

},

"items": [

{

"productId": "productId1",

"text": "Hotpants",

"quantity": 2.0,

"currency": "USD",

"price": 12.25

},

{

"productId": "productId2",

"text": "Shovel",

"quantity": 1.0,

"currency": "USD",

"price": 24.25

}

]

}

## Inventory List

{

"utcDateTime": "2019-05-28T14:56:06.767Z",

"locDateTime": "2019-05-28T16:56:06.767+02:00",

"referenceId": "list-id",

"senderId": "100000000",

"items": [

{

"productId": "productId1",

"text": "Hotpants",

"quantity": 100.0,

"currency": "USD",

"price": 12.25

},

{

"productId": "productId2",

"text": "Shovel",

"quantity": 50.0,

"currency": "USD",

"price": 24.25

}

]

}

# Deployment

The program is written in Java and uses the Solace JMS API for all publish and subscribe actions. Every business instance (Shop, SAP, Customer) is running in an own thread. Theoretically there can be many instances per business entity. Currently it's restricted to one.

Prerequisits to run the program is a running Solace broker, reachable via TCP.

## Deployment as runnable jar

Exported as runnable jar from Eclipse IDE, including all dependent jars into an uber jar. All configuration is taken from environment, in order to facilitate the Docker deployment.

Running the jar:

export FMCG\_HOST="tcp://vmr-mr4tcaa690q5p.messaging.solace.cloud:20256"

export FMCG\_VPN="msgvpn-8ksiwsoxmad"

export FMCG\_USER="solace-cloud-client"

export FMCG\_PASSWORD="bil5b8c30csl2r3ls82m5k9sub"

export FMCG\_REGION="us1"

java -jar fmcg.jar

The traffic can be monitored e.g. using sdkperf from Solace

./sdkperf\_c -cip="tcp://vmr-mr4tcaa690q5p.messaging.solace.cloud:20256" \

-cu="solace-cloud-client"@"msgvpn-8ksiwsoxmad" \

-cp="bil5b8c30csl2r3ls82m5k9sub" \

-stl="us1/>" \

-md -q

## Creating/Running Docker Image

A docker image is currently published as danbrown/fmcg on docker hub. Should be ported to SolaceSE account if existing. Prerequisite is to have a runnable jar.

Content of Dockerfile:

FROM openjdk:8-jre-alpine

WORKDIR /

ADD fmcg.jar fmcg.jar

ADD log4j.properties log4j.properties

CMD java -jar fmcg.jar

Content of log4j.properties (example):

# Root logger option

log4j.rootLogger=WARN, stdout

# Direct log messages to stdout

log4j.appender.stdout=org.apache.log4j.ConsoleAppender

log4j.appender.stdout.Target=System.out

log4j.appender.stdout.layout=org.apache.log4j.PatternLayout

log4j.appender.stdout.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss} %-5p %c{1}:%L - %m%n

docker build -t danbrown/fmcg .

docker image push danbrown/fmcg

The image can be started on a standard Docker machine with 1 CPU and 1GB RAM.

docker create \

--env FMCG\_HOST="tcp://vmr-mr4tcaa690q5p.messaging.solace.cloud:20256" \

--env FMCG\_VPN="msgvpn-8ksiwsoxmad" \

--env FMCG\_USER="solace-cloud-client" \

--env FMCG\_PASSWORD="bil5b8c30csl2r3ls82m5k9sub" \

--env FMCG\_REGION="us1" \

--name fmcg \

--hostname fmcg \

danbrown/fmcg

docker start fmcg

The traffic can be monitored e.g. using sdkperf from Solace

./sdkperf\_c -cip="tcp://vmr-mr4tcaa690q5p.messaging.solace.cloud:20256" \

-cu="solace-cloud-client"@"msgvpn-8ksiwsoxmad" \

-cp="bil5b8c30csl2r3ls82m5k9sub" \

-stl="us1/>" \

-md -q