

Smart Home Control

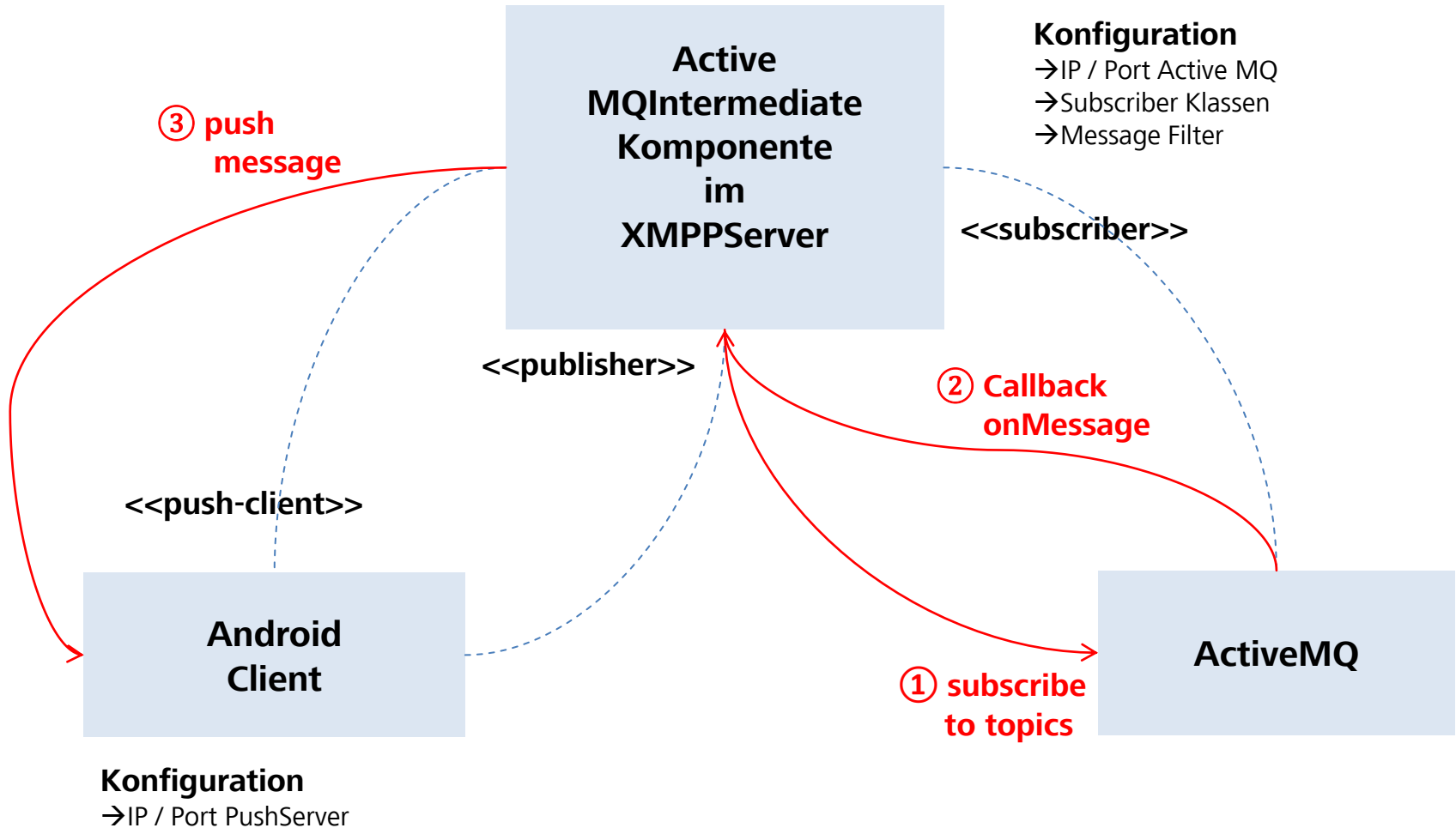
Active MQIntermediate

ActiveMQ Tools

Prof.Dr.Ing.Birgit Wendholt

MQIntermediate

Überblick über den Ablauf



Szenarien

- **Szenario 1:** Periodisches Empfangen von Ubisense Lokationsdaten
 - **Szenario 2:** Periodisches Empfangen von ISIS-Nachrichten über „Spatial Artifacts“
 - Functional Space
 - Range Space
- Demo im LivingPlace
- Projekte:
- Server: LPXMPPPushMQIntermediate
 - Client: LPMQSubscriberAndroid4_1_2

MQIntermediate einrichten

- **Server Benutzung**

- Package org.androidpn.starter die Klasse LPXMPPPushMQIntermediate starten
- Einstellungen in *activemq_config.properties*

```
active.mq.ip=172.16.0.200
```

```
mongo.db.ip=172.16.0.200
```

```
active.mq.subscriber=org.androidpn.server.container.intermediates.ISISIntermediate
```

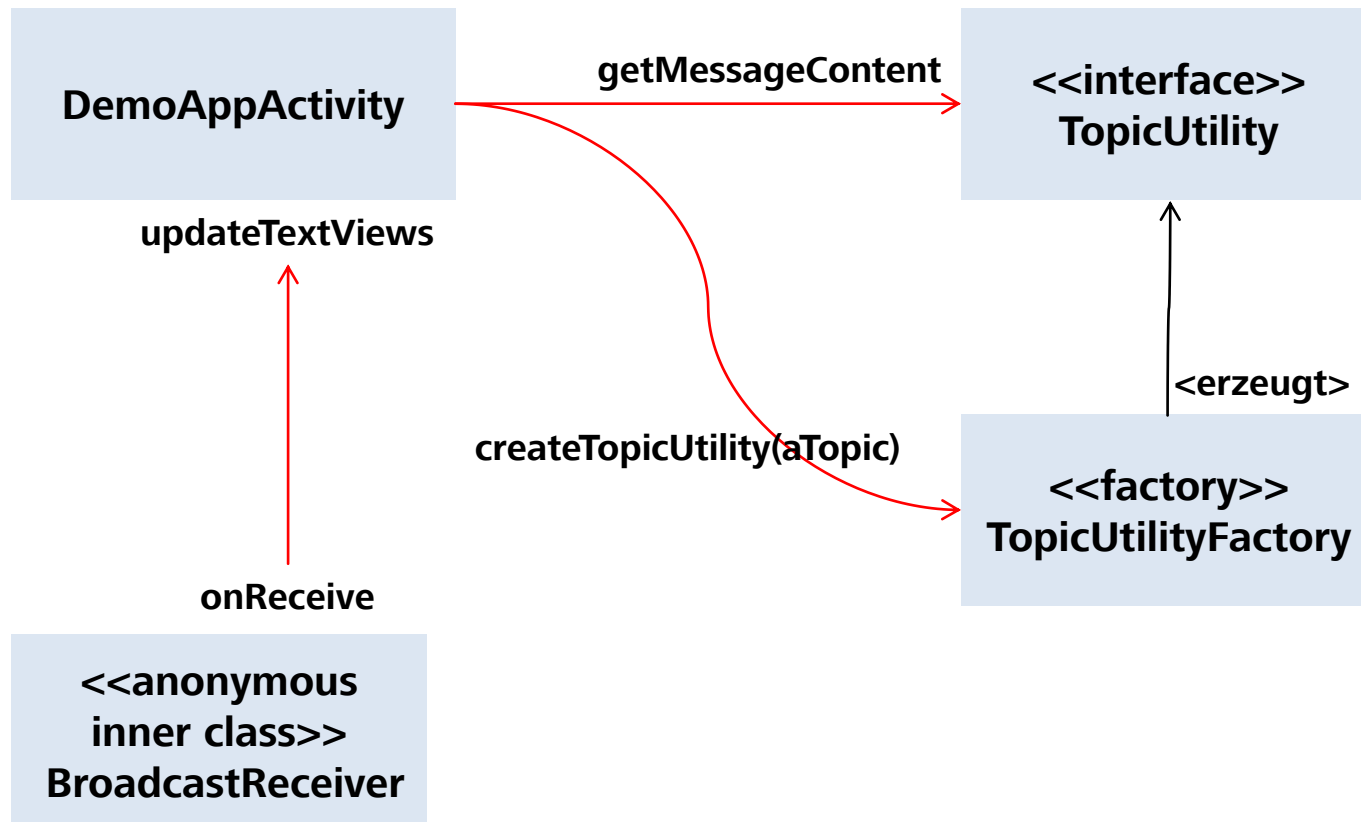
```
active.mq.subscriber=org.androidpn.server.container.intermediates.UbiSenseIntermediate
```

- Liste von Subscribern unter *active.mq.subscriber*, aktuell
 - Subscriber für UbiSense
 - Subscriber für ISIS: Nachrichten für das Betreten und Verlassen von Räumen
- Liste von Filtern auf oberster Attributebene von Nachrichten, aktuell
 - Filter auf die Nachrichten des Topics *LP3D.SPATIALARTIFACTS*
 - Filter sind aktuell UND verknüpft

Android Push-Client einrichten

- Einstellungen im **Client**:
 apiKey=**1234567890**
 xmppHost=**172.16.0.228**
 xmppPort=**5222**
- **ausgeliefert werden**
 - **Ubisense** Nachrichten
 - **ISIS** Nachrichten des Topics LP3D.SPATIALARTEFACTS

Klassen und Schnittstellen des PushClients



Beispielnachrichten „User's Guide“

Ubisense

```
{ "Version": "1.0",  
  "Id": "2531789102745845",  
  "UbisenseTagId": "<UbisenseId>",  
  "Unit": <meter_or_centimeter>,  
  "NewPosition": {  
    "X": <xpos>, "Y": <ypos>,  
    "Z":<zpos>  }  
}
```

ISIS

```
{ "Version": "1.0",  
  "Id": "2531789102745845",  
  "UbisenseTagId": "<UbisenseId>",  
  "SpaceType": "FunctionalSpace",  
  "ObjectType": " TraditionalSpace ",  
  "ObjectId": "diningRoom",  
  "InSpace": true }
```

Auslesen der Ubisense JSON-Nachrichten auf dem Client

```
JSONObject serverJsonMsg = new JSONObject(msg);  
String ubisenseTagId = serverJsonMsg.getString(UBISENSETAGID);  
String ubisenseUnit = serverJsonMsg.getString(UBISENSEPOSITIONUNIT);  
JSONObject position = serverJsonMsg.getJSONObject(UBISENSEPOSITION);  
  
double xCoord = position.getDouble(XCOORD);  
double yCoord = position.getDouble(YCOORD);  
double zCoord = position.getDouble(ZCOORD);
```

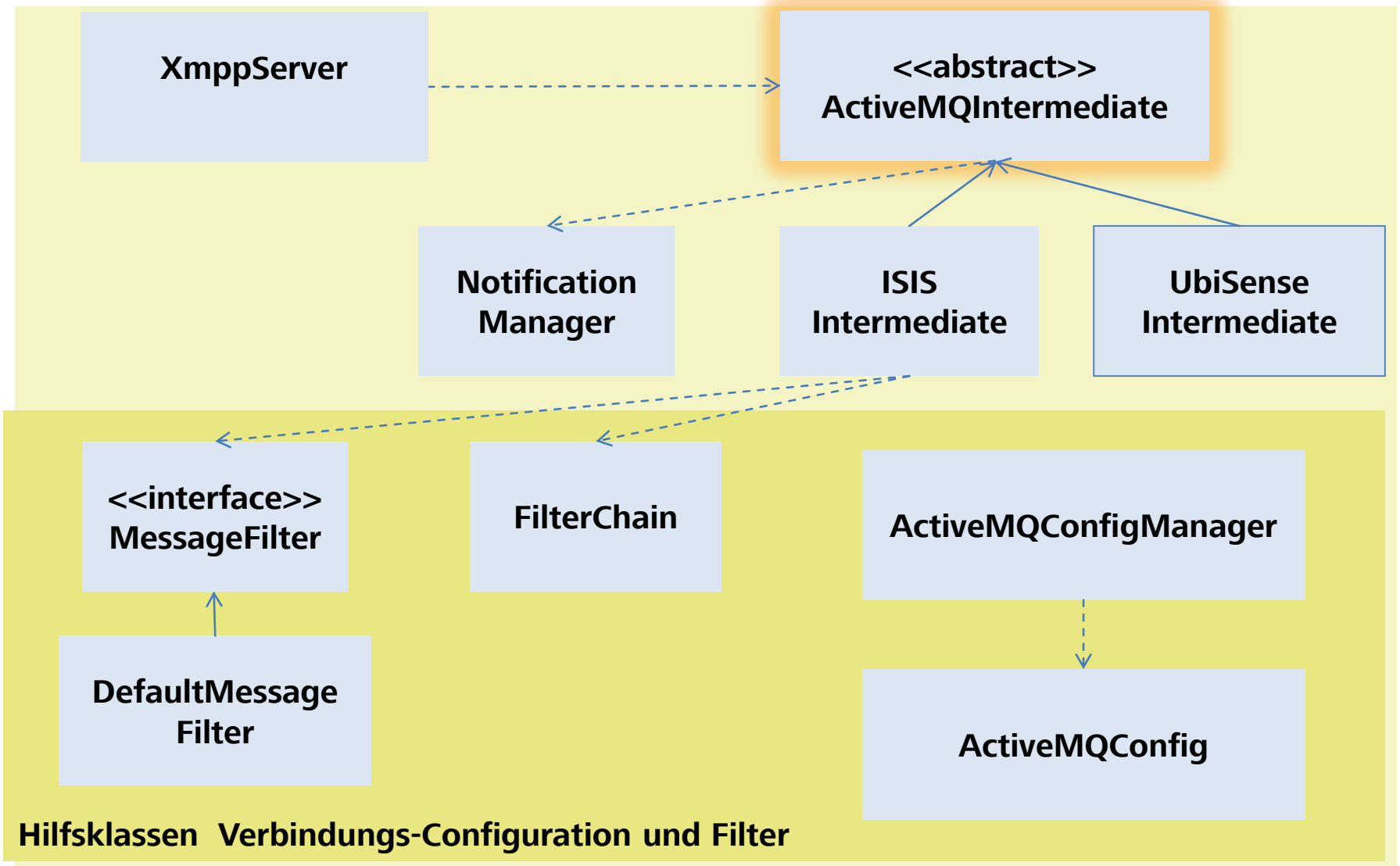
Beispielcode: *TopicUtilityFactory, TopicUtility, MQConstants,
DemoAppActivity.updateTextViews()*

Auslesen der ISIS JSON-Nachrichten auf dem Client

```
JSONObject serverJsonMsg = new JSONObject(msg);  
String objectType = serverJsonMsg.getString(OBJECT_TYPE);  
  
if (objectType.equals(TRADITIONAL_SPACE)) {  
    String ubisenseTagId = serverJsonMsg.getString(UBISENSETAGID);  
    String objectTyp = serverJsonMsg.getString(OBJECT_TYPE);  
    String objectId = serverJsonMsg.getString(OBJECT_ID);  
    boolean inOut = serverJsonMsg.getBoolean(IN_SPACE);  
}
```

Beispielcode: *TopicUtilityFactory, TopicUtility, MQConstants,*
DemoAppActivity.updateTextViews()

Überblick der Klassen für die Intermediate Szenarien



Organisation der Lösung

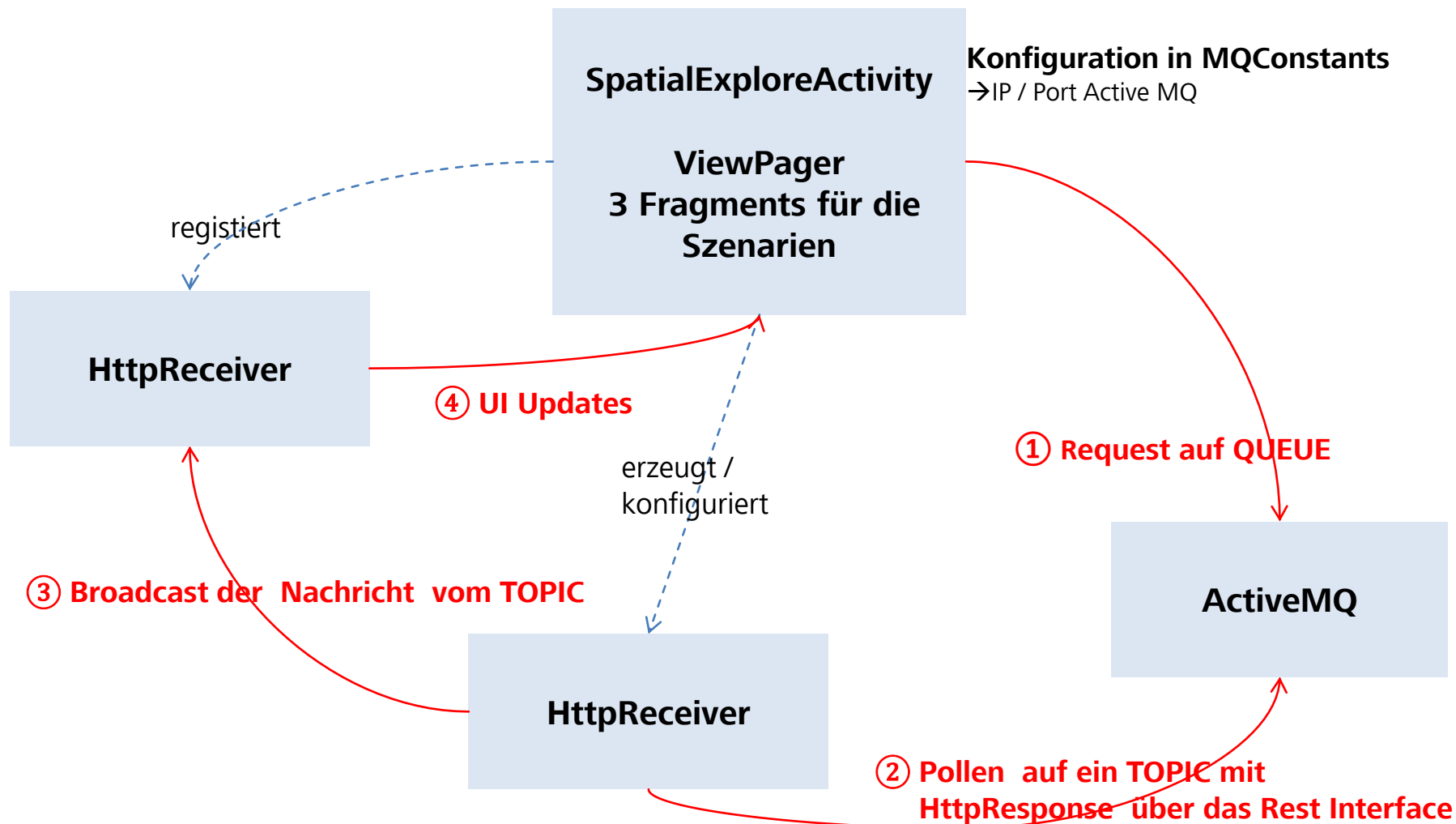
- org.androidpn.server.container.intermediates:
 - *ActiveMQIntermediate, ISISIntermediate, UbiSenseIntermediate*
 - *MessageFilter, DefaultMessage, FilterChain*
- org.androidpn.server.container.intermediates.util:
 - Klassen für die Konfiguration: *ActiveMQConfigManager, ActiveMQConfig*
 - Konfigurationsdatei: *conf/activemq_config.properties*
- org.androidpn.server.xmpp:
 - *XMPPServer* erweitert um die Intermediate Komponenten
- org.androidpn.server.xmpp.push:
 - *NotificationManager*

Request / Response über ActiveMQ

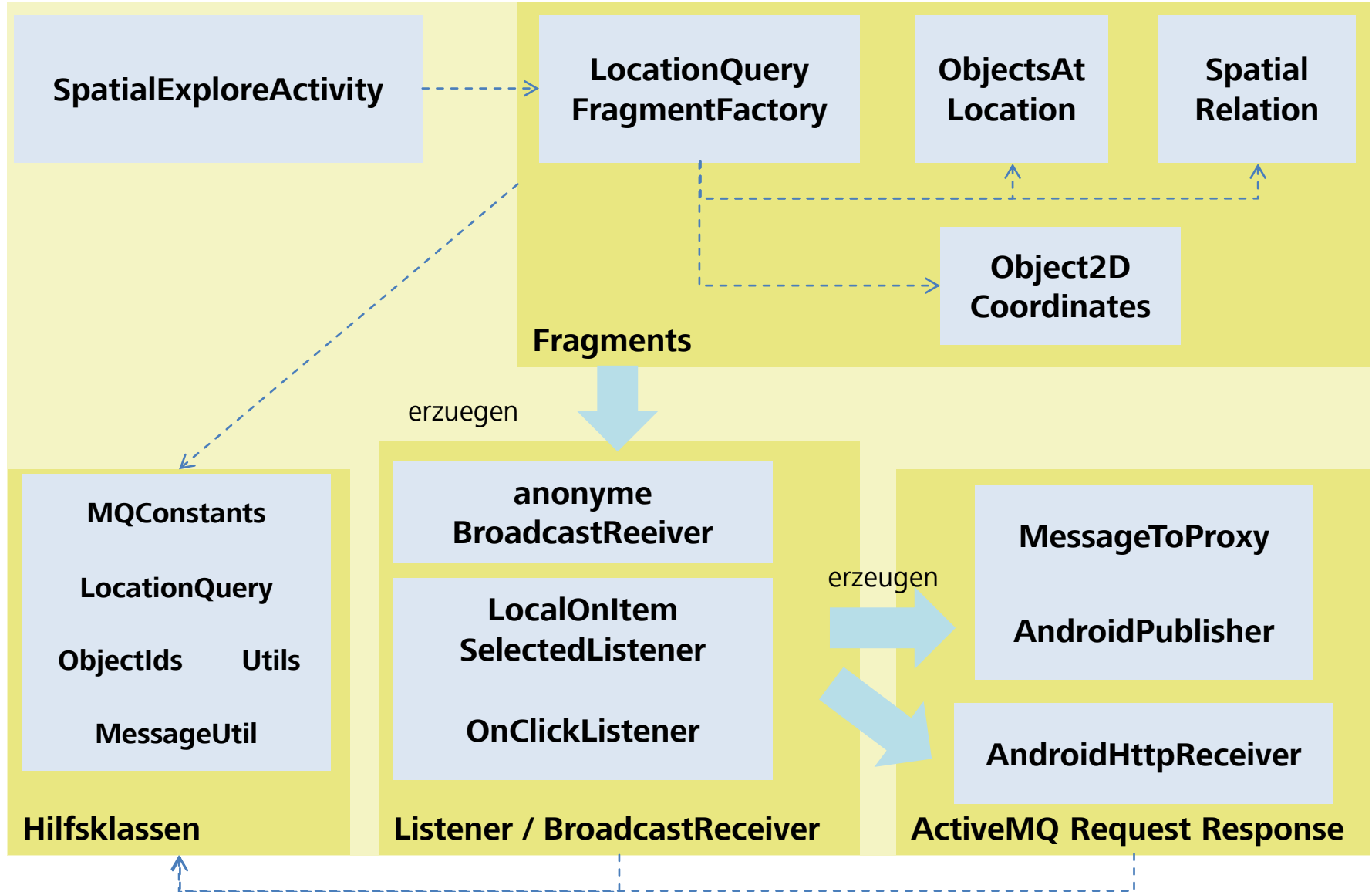
Request-Response über ActiveMQ Topics / Queue

- **Szenario 3:** Positionsdaten von Objekten erfragen mit und ohne Bounding Box
 - Request Queue: *LP3D_REQUEST_OBJECT2D_COORDINATES*
 - Response Topic: *LP3D_RESPONSE_OBJECT2D_COORDINATES*
 - Asynchrones Request-Response mittels des ActiveMQ Rest Interfaces
- **Szenario 4:** Objekte im Umkreis einer Position erfragen
 - Request Queue: *LP3D_REQUEST_OBJECTS_AT_LOCATION*
 - Response Topic: *LP3D_RESPONSE_OBJECTS_AT_LOCATION*
 - Asynchrones Request-Response mittels des ActiveMQ Rest Interfaces
- **Szenario 5:** Räumliche Relationen zu einer Position erfragen
 - Request Queue: *LP3D_REQUEST_SPATIAL_RELATIONSHIPS*
 - Response Topic: *LP3D_RESPONSE_SPATIAL_RELATIONSHIPS*
 - Asynchrones Request-Response mittels des ActiveMQ Rest Interfaces

Request-Response Überblick über den Ablauf



Klassen / Interfaces der Request-Response Szenarien



Asynchrones Empfangen Pollen mittels ActiveMQ Rest Interface

(→ <http://activemq.apache.org/rest.html>)

- Senden eines ActiveMQ Request mittels AndroidPublisher
- Warten auf den HttpResponse (Pollen) auf ein HttpGet mit dem Parameter `"http://<ACTIVE_MQ_IP>::8161/demo/message/<topicOrQueueName>?type=<topicOrQueue>&clientId=<clientId>"`
- Broadcasting des korrekten HttpResponses über den Android Kern
 - Übereinstimmung der MessageId in HttpRequest und HttpResponse prüfen
- BroadcastReceiver in den Fragments (je eins für eines der Szenarien)
 - Object2DCoordinatesFragment
 - ObjectsAtLocationFragment
 - SpatialRelationFragment

Szenario 3: Positionsdaten von Objekten erfragen

- **Queue:**
LP3D_REQUEST_OBJECT2D_COORDINATES

- **Topic:**
LP3D_RESPONSE_OBJECT2D_COORDINATES

Anfrage:

```
{  
  Version: "1.0",  
  Id: "2531789102745845",  
  ObjectId: "smallUfo1",  
  BoundingBox: true,  
}
```

Antwort:

```
{  
  Version: "1.0",  
  Id: "2531789102745845",  
  ObjectId: "smallUfo1",  
  BoundingBox: true,  
  Coordinates: [ {"X":0.0,"Y":1.0},  
                 {"X":2.0,"Y":3.0}, {"X":1.0,"Y":1.0},  
                 {"X":4.0,"Y":1.0} ]  
}
```

Szenario 4: Objekte im Umkreis einer Position erfragen

- **Queue:**
LP3D.REQUEST.OBJECTSATLOCATION

Anfrage:
{ Version: "1.0",
Id: "2531789102745845",
ReferencePoint: {"X":0.0,"Y":0.0,"Z":0.0},
Radius: 0.8 }

- **Topic:**
LP3D.RESPONSE.OBJECTSATLOCATION

Antwort:
{ Version: "1.0",
Id: "2531789102745845",
ObjectIds2D: ["objId1", "objId2"],
ObjectIds3D: ["objId1", "objId3"] }

Szenario 5: Räumliche Relationen erfragen

- **Queue:**
LP3D_REQUEST_SPATIAL_RELATIONSHIPS
- **Topic:**
LP3D_RESPONSE_SPATIAL_RELATIONSHIPS

Anfrage:

```
{  
  "Version": "1.0",  
  "Id": "2531789102745845",  
  "ReferencePoint": {"X":1.1,"Y":3.0,"Z":0.8},  
  "ReferenceObject": "phone"  
}
```

Antwort:

```
{  
  Version: "1.0",  
  Id: "2531789102745845",  
  ReferenceObject : "phone",  
  SpatialRelations:  
    "Contains(Room1,phone);Above(phone,i  
    d2);Distance(id2,id3)=VERY_CLOSE;"
```

Erläuterung zur LP3D SPATIALRELATIONSHIPS

→ http://livingplace.informatik.haw-hamburg.de/wiki/index.php/ActiveMQ_Messages

Antwort (Topic: "LP3D.RESPONSE.SPATIALRELATIONSHIPS"):

Key	Value	Beschreibung
SpatialRelations	3D räumliche Relationen	<p>Mögliche Prädikate der Relationen:</p> <ul style="list-style-type: none"> • Topologisch: Contains(x,y) (Bsp.: "x enthält y") • Direktional: Above(x,y), Below(x,y), NorthOf(x,y), SouthOf(x,y), EastOf(x,y), WestOf(x,REF_POINT) (Bsp.: "x ist westlich von REF_POINT") • Metrisch: Distance(x,y) = DIST (Bsp.: "Der Abstand von x und y ist DIST", DIST wird nicht quantitativ angegeben, sondern qualitativ) <ul style="list-style-type: none"> ◦ DIST = VERY_CLOSE ("sehr nahe dran"/"dicht bei"), CLOSE ("nahe dran"/"in der Nähe"), FAR ("in einiger Entfernung zu"), VERY_FAR ("weit weg von") ◦ REF_POINT = Koordinate der Anfrage, zu dem die Objekte in Bezug gesetzt werden <p>Einzelne Prädikate sind durch ";" angetrennt.</p>

TOOLS

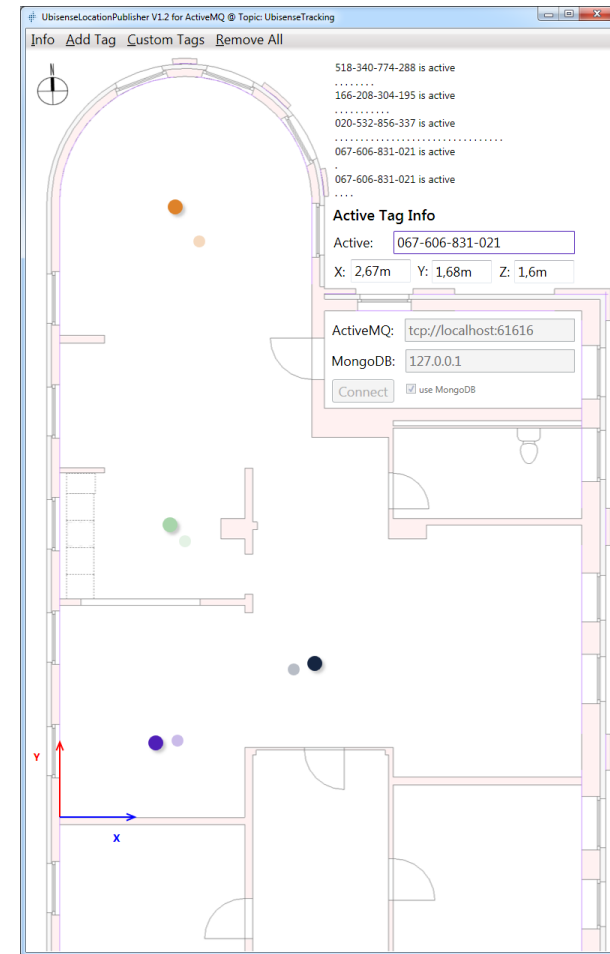
ActiveMQ & co

Admin Console für ActiveMQ Queues und Topics

- **URL:** <http://172.16.0.200:8161/admin>

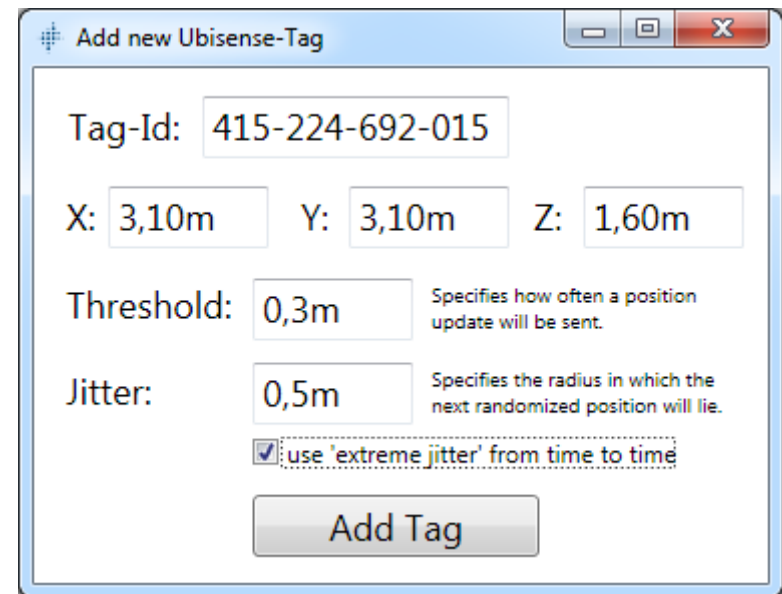
Ubisense Mockup für den offline Test

- Nachrichten versenden durch das Bewegen von Tags auf einem 2D Modell des Living Place
- nach erfolgreichen Verbindung öffnet sich ein Nebenfenster, mit dem neue Tags generiert werden können
- Installation → http://livingplace.informatik.haw-hamburg.de/wiki/index.php/Tools_und_Mockups#Ubisense_Mockup



Tags für das Mockup generieren

- *Tag-Id*: randomisierte Id oder eine eigene
- *X, Y, Z*: Startwerte in den globalen Koordinaten des Living Place (Z bleibt konstant)
- *Threshold*: gibt an wie oft ein Tag Nachrichten sendet.
- Optionen zur Simulation Ubisense-ähnlichen Verhaltens:
 - *Jitter*: Radius an in dem randomisiert neue Positionsupdates gesendet werden
 - *use extreme Jitter*: ca. 5% aller Nachrichten enthalten größere *Ausreißer*



The screenshot shows a window titled "Add new Ubisense-Tag" with the following fields and options:

- Tag-Id:** 415-224-692-015
- X:** 3,10m
- Y:** 3,10m
- Z:** 1,60m
- Threshold:** 0,3m (with a tooltip: "Specifies how often a position update will be sent.")
- Jitter:** 0,5m (with a tooltip: "Specifies the radius in which the next randomized position will lie.")
- ☒ use 'extreme jitter' from time to time
- Add Tag** button