Case Study: IoT - Intelligent Refrigerated Shipping Containers

Case Study: IoT - Intelligent Refrigerated Shipping Containers

Introduction

This use case is adapted from studies *here* and *here*. It involves a ship hauling intelligent “reefers” (refrigerated containers) and tracking environmental factors (e.g., temperature, humidity) to ensure that the contents arrive in good conditions to their destination.

The containers are equipped with an array of sensors (e.g., temperature, humidity, location, shock) connected to an embedded microprocessor with some compute power for basic calculations. The containers expose RESTful APIs with information about sensors and status. They also communicate with a cloud service for analytics and reporting purposes. They are also equipped with a wide variety of network antennas, which might include cellular, Wi-Fi, and satellite (this varies widely depending on the type and cost of the container)

The ship needs access to the sensors on the containers, so that the crew can easily monitor cargo status and track dynamic distribution policies such as First Expire First Out (FEFO). We assume that there is a standard set of APIs for the different sensors and high-level functionality available for the vessel to communicate with the reefers.

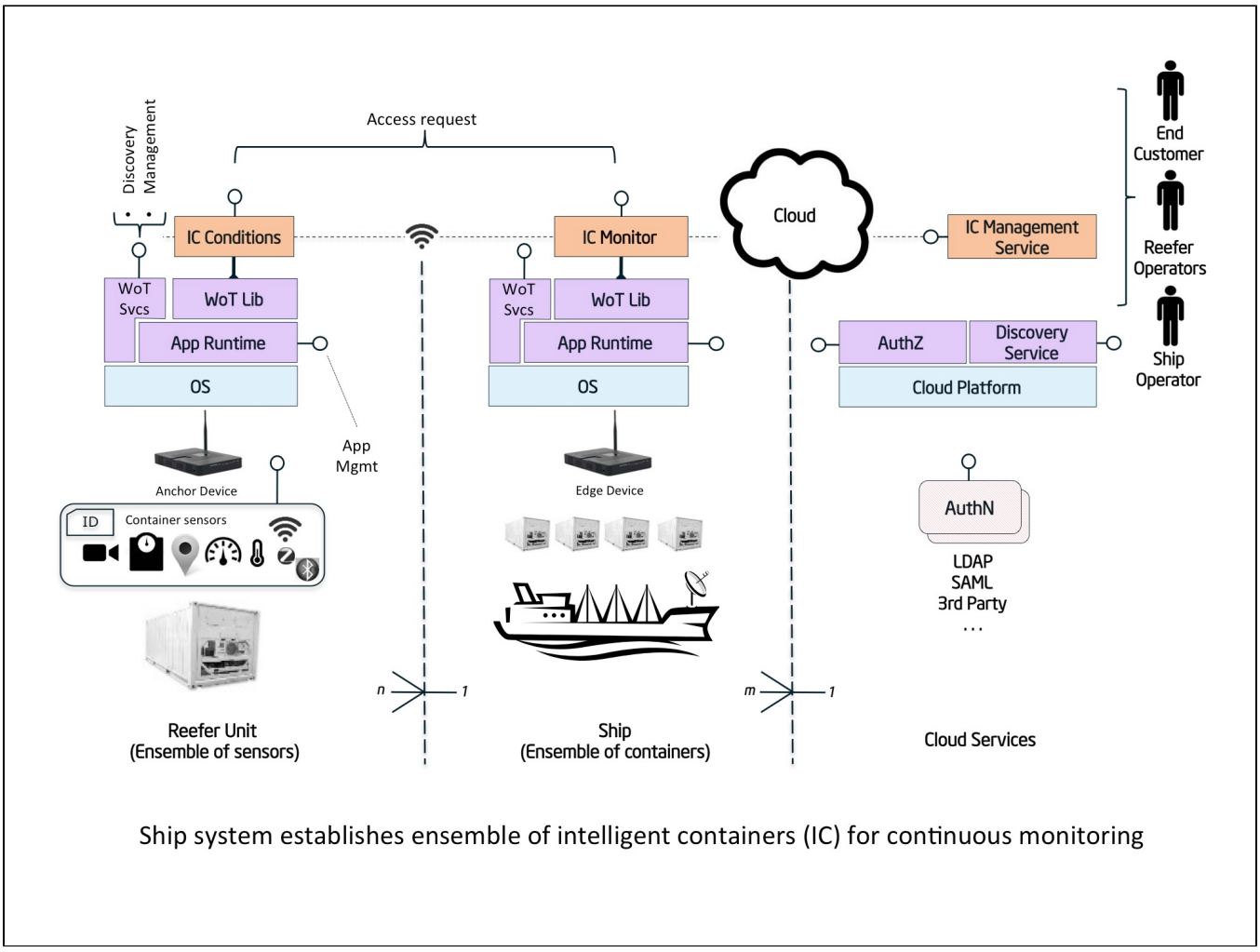
We also assume the existence of a ship operator (i.e., carrier) and multiple reefer operators (e.g., freight forwarder, shipping company), which might be different organizations.

Problem Scenario

While the ship is docked at the port, it maintains constant network connectivity with different cloud services (same is true for reefers). As reefers are loaded, the ship communicates with the sensors and establishes a trust relationship. The ship will need to access the information on the sensors en route, even in locations were connectivity is not available.

The ship will get initial readings from the relevant sensors in every reefer while docked, plot its course and then start navigation. Once en route, the ship will continue updating the readings for every sensor. While sensors might be able to connect to the cloud in order to validate trust information from the ship, this might not always be possible: The reefers need to be able to cache trust validation info for making offline decisions.

Based on the retrieved information and in case where the cargo may be at risk, the ship may alter delivery plans or attempt communication with cloud services for further instructions.



Ship system establishes ensemble of intelligent containers (IC) for continuous monitoring

Proposed Improvements

UMA makes this scenario possible, where the reefers (and their individual sensors and high level APIs) are the Resource Servers, and the ship acts as the Client. The ship operator is the Requesting Party (although in this scenario the Client is heavily automated, possibly by having an embedded credential) and the reefer operator is the Resource Owner (also, possibly automated in the reefer itself or with a pre-defined policy). The Authorization Server is managed by a neutral party, such as a logistics company.

The reefers are provisioned at loading time, that is, each of the Resources exposed by the reefer are registered with the authorization server on behalf of the reefer operator (with a proper PAT), and a policy is pre-established that will allow the ship operator to access the APIs exposed by the reefer. If for any reason this policy is not properly provisioned, access to the reefer's APIs would require the reefer operator to grant access (not desirable in most situations)

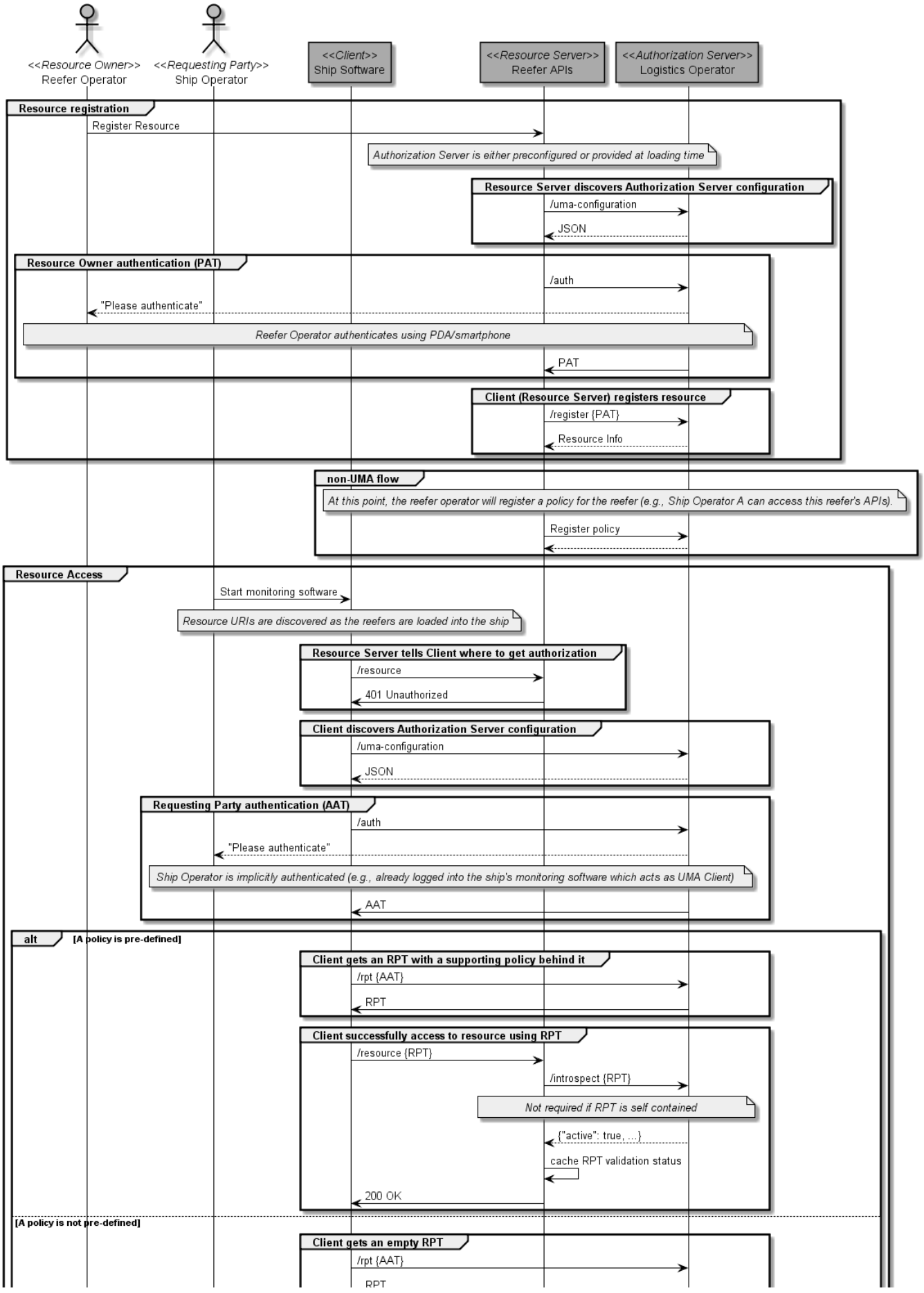
When the reefers are loaded into the ship, the ship will "discover" them (e.g., RFID tags or similar) and will start querying their APIs. A standard UMA flow is expected at this point, where the ship software is required to get an RPT (based on an AAT that is obtained with the ship operator's credentials). Since the ship is docked and connectivity is available, the reefers will be able to interact with the AS in order to validate the access request.

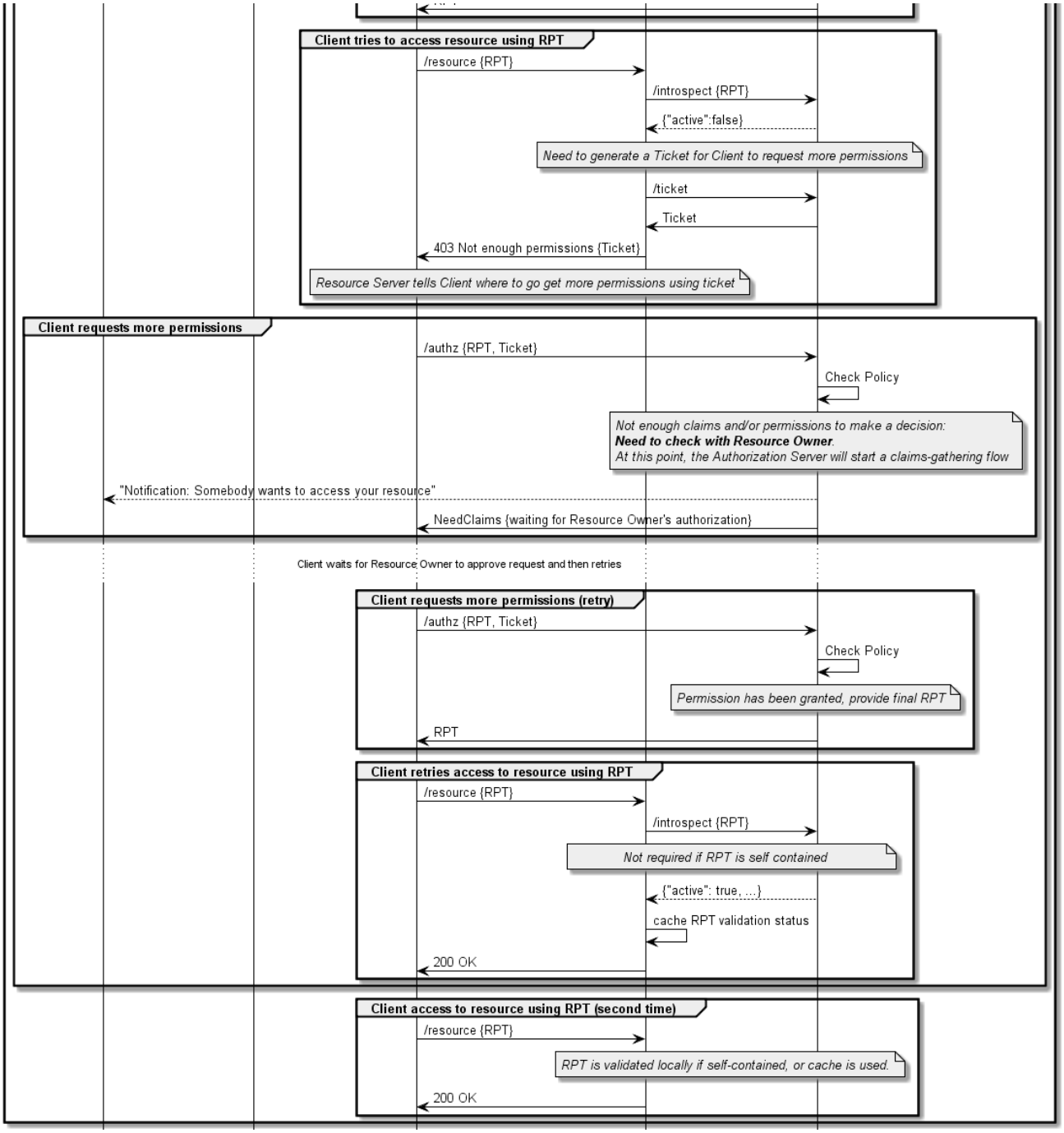
At this point, the reefers are expected to validate an RPT even with no network connectivity. There are a couple of alternatives here: we can either establish a long enough expiration time for the RPT, so that the response from the AS can be cached for at least that period of time. This expiration time should be long enough for most trips, plus an added margin, so that the reefers can provide service even when connectivity is not available during the entire trip. A second alternative would be considering self-contained RPTs that can be validated by the reefer without need to contact the AS. Another alternative might be for the reefers to perform a more lax trust decision, e.g., by overriding expiration times if connectivity is not detected for a given period of time. These are all aspects to be weighed in at RS design time, since the goal of preserving the cargo is paramount.

The ship can then continue to access the sensor information for the reefers with the original RPT, and it might decide to refresh them when connectivity is available.

If reefers are changed from ship to ship while visiting an intermediate port, the target ship will perform the same operation as described above, obtaining its own RPT for accessing the reefers. If for any reason a policy has not been defined at the AS for a particular ship (Client), operator (Requesting Party) and reefer (Resource Server) combination, communication with the reefer operator (Resource Owner) may be required for access request. While unlikely, we would expect a complete solution to include this flow for completeness.

Solution Flow


<<Resource Owner>>
Reefer Operator
<<Requesting Party>>
Ship Operator
<<Client>>
Ship Software
<<Resource Server>>
Reefer APIs
<<Authorization Server>>
Logistics Operator
Resource registration
Register Resource
Authorization Server is either preconfigured or provided at loading time
Resource Server discovers Authorization Server configuration
/uma-configuration
JSON
Resource Owner authentication (PAT)
/auth
"Please authenticate"
Reefer Operator authenticates using PDA/smartphone
PAT
Client (Resource Server) registers resource
/register {PAT}
Resource Info
non-UMA flow
At this point, the reefer operator will register a policy for the reefer (e.g., Ship Operator A can access this reefer's APIs).
Register policy
Resource Access
Start monitoring software
Resource URIs are discovered as the reefers are loaded into the ship
Resource Server tells Client where to get authorization
/resource
401 Unauthorized
Client discovers Authorization Server configuration
/uma-configuration
JSON
Requesting Party authentication (AAT)
/auth
"Please authenticate"
Ship Operator is implicitly authenticated (e.g., already logged into the ship's monitoring software which acts as UMA Client)
AAT
alt
[A policy is pre-defined]
Client gets an RPT with a supporting policy behind it
/rpt {AAT}
RPT
Client successfully access to resource using RPT
/resource {RPT}
/introspect {RPT}
Not required if RPT is self contained
{"active": true, ...}
cache RPT validation status
200 OK
[A policy is not pre-defined]
Client gets an empty RPT
/rpt {AAT}
RPT



PlantUML diagram source