Opening IMS with REST APIs

Václav Koudelka **vaclav.koudelka@broadcom.com**

24 January 2023

About

- Overview of Java on Demand feature of IMS
- Building REST API with IMS TM Resource Adapter
- Enabling REST API in Zowe API ML
- REST API Security

https://github.com/volov0/IMS-API

Technology stack

JAVA

IMS JAVA On Demand – IMS TM Resource Adapter

IMS Connect

IMS TM, IMS DB

ZOWE API Mediation Layer

Springboot

Swagger

Java in IMS

Mainframe libraries	Remote libraries
• Type-2 IMS Universal Drivers (JDBC and DL/I) • IMS Java Dependent Region Resource Adapter	• Type-4 IMS Universal Drivers (JDBC and DL/I) • IMS TM Resource Adapter
JMPs, JBPs	Distributed Java Applications

IMS Java On Demand feature

- usr/lpp/ims/ims15/imsjava

  imsudb.jar – IMS Universal drivers and IMS Java dependent region resource adapter

- usr/lpp/ims/ims15/imsjava/samples

  OpenDBIVP.jar – JMP and JBP sample jobs

- usr/lpp/ims/ims15/imsjava/lib

  libT2DLI.so - Java native code for IMS type-2 Java connectivity

- usr/lpp/ims/ims15/ico

  Contains collection of libraries which form together IMS TM resource adapter

- usr/lpp/ims/ims15/imsjava/rar

  imsudbLocal.rar, imsudbXA.rar, imsudbJLocal.rar, imsudbJXA.rar

- usr/lpp/ims/ims15/imsjava/cics

JMPs and JBPs - setup

IMS.PROCLIB(DFSJVMMS)

```
-Djava.class.path=/a/kouva01/java/Hello/hello.jar:>
/sys/IMS/V15GA/usr/lpp/ims/ims15/imsjava/imsudb.jar:>
/sys/IMS/V15GA/usr/lpp/ims/ims15/imsjava/samples/OpenDBIVP.jar:>
```

IMS.PROCLIB(DFSJVMEV)

```
LIBPATH=>
/sys/java31bt/v8r0m0/usr/lpp/java/J8.0/bin/j9vm:>
/sys/java31bt/v8r0m0/usr/lpp/java/J8.0/bin/:>
/sys/IMS/V15GA/usr/lpp/ims/ims15/imsjava/lib/
```

IMS.PROCLIB(DFSJVMAP)

```
DFSIVP37=samples/ivp/ims/IMSIVP
DFSIVP67=samples/ivp/ims/IMSIVPJBP
BTSAOTP3=ca/ims/transaction/JMPTransaction1
```


IMS TM Resource adapter

Runtime component – must be deployed with IBM proprietary applications (WebSphere)

Development component – implements Jakarta Connectors architecture (JCA)

- Invoke IMS transaction program
- Retrieve undelivered or asynchronous output messages
- Retrieve an IMS callout request and respond back
- Invoke any of the IMS commands that are supported by IMS OTMA

IVTNO transaction

```
*****************************************************
*      IMS INSTALLATION VERIFICATION PROCEDURE      *
*****************************************************

                          TRANSACTION TYPE : NON-CONV (OSAM DB)
                          DATE             : 11/11/2019

PROCESS  CODE  (*1) :   DIS
                                                (*1) PROCESS CODE
LAST  NAME          :   LAST2                        ADD
                                                     DELETE
FIRST  NAME         :   FIRST2                       UPDATE
                                                     DISPLAY
EXTENSION  NUMBER   :   8-111-2222                   TADD

INTERNAL  ZIP CODE  :   D02/R02

ENTRY WAS DISPLAYED                             SEGMENT# :  0003
```


IVTNO transaction – REST API

```
https://localhost:8080/api/v1?lastname=last1
```

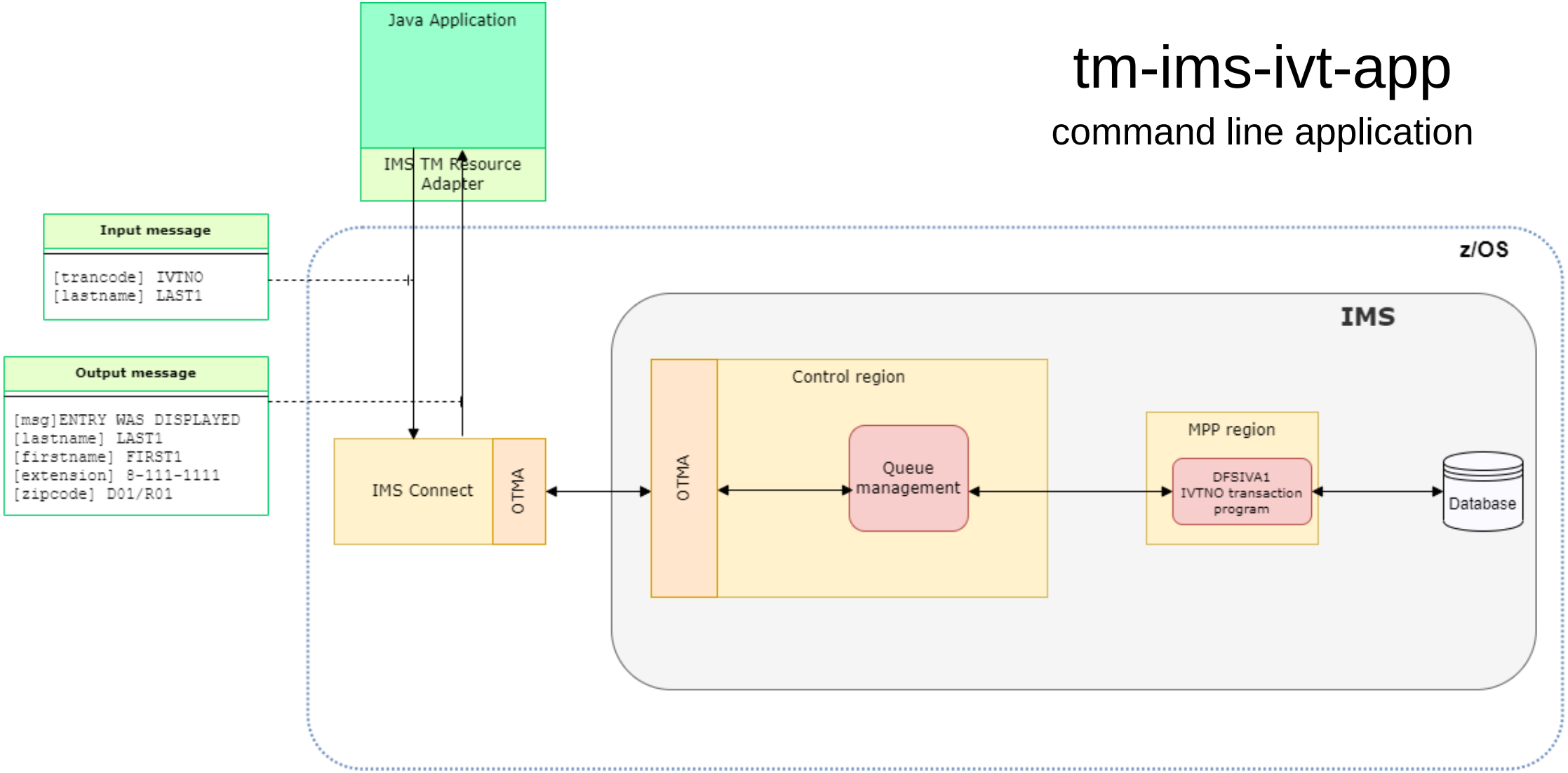
HTTP GET

```
{
  "message":"ENTRY WAS DISPLAYED",
  "lastname":"last1",
  "firstname":"FIRST1    ",
  "extension":"8-111-1111",
  "zipcode":"D01/R01"
}
```

ivt IMS IVTNO/IVTCM transaction

POST	/ivt	Add new entry to phone book database
GET	/ivt/{lastname}	Find a phone book entry by lastname
POST	/ivt/{lastname}	Updates a phone book entry
DELETE	/ivt/{lastname}	Deletes a phone book entry

IMS TM Resource adapter sample standalone application



tm-ims-ivt-app

command line application

Sample application tm-ims-ivt-app

To Enable IMS TM Resource Adapter Java Program needs to import a few packages

```
import com.ibm.connector2.ims.ico.IMSConnectionFactory;
import com.ibm.connector2.ims.ico.IMSInteraction;
import com.ibm.connector2.ims.ico.IMSInteractionSpec;
import com.ibm.connector2.ims.ico.IMSManagedConnectionFactory;
```

- These packages are located in imsico.jar
- Ccf2.jar, IMSLogin.jar, CWYBS_AdapterFoundation.jar need to be in classpath as well

```
mvn install:install-file -Dfile=imsico.jar -DgroupId=com.ibm.ims -DartifactId=imsico -Dversion=15.1.2 -Dpackaging=jar
mvn install:install-file -Dfile=ccf2.jar -DgroupId=com.ibm.ims -DartifactId=ccf2 -Dversion=15.1.2 -Dpackaging=jar
mvn install:install-file -Dfile=IMSLogin.jar -DgroupId=com.ibm.ims -DartifactId=IMSLogin -Dversion=15.1.2 -Dpackaging=jar
mvn install:install-file -Dfile=CWYBS_AdapterFoundation.jar -DgroupId=com.ibm.ims -DartifactId=CWYBS_AdapterFoundation
 -Dversion=15.1.2 -Dpackaging=jar
```


Sample application tm-ims-ivt-app

Establish connection with IMS Connect

```
IMSConnectionFactory cf;
IMSManagedConnectionFactory mcf = new IMSManagedConnectionFactory();

// set parameters for IMS Connect connection
mcf.setHostName("hostname");
mcf.setUserName("username");
mcf.setPassword("password");
mcf.setDataStoreName("IMSW");
mcf.setPortNumber(new Integer(8866));

// Create connection factory from ManagedConnectionFactory
cf = (IMSConnectionFactory) mcf.createConnectionFactory();

// Create an IMSConnection object
Connection connection = cf.getConnection();
```


Sample application tm-ims-ivt-app

Use IMS interaction

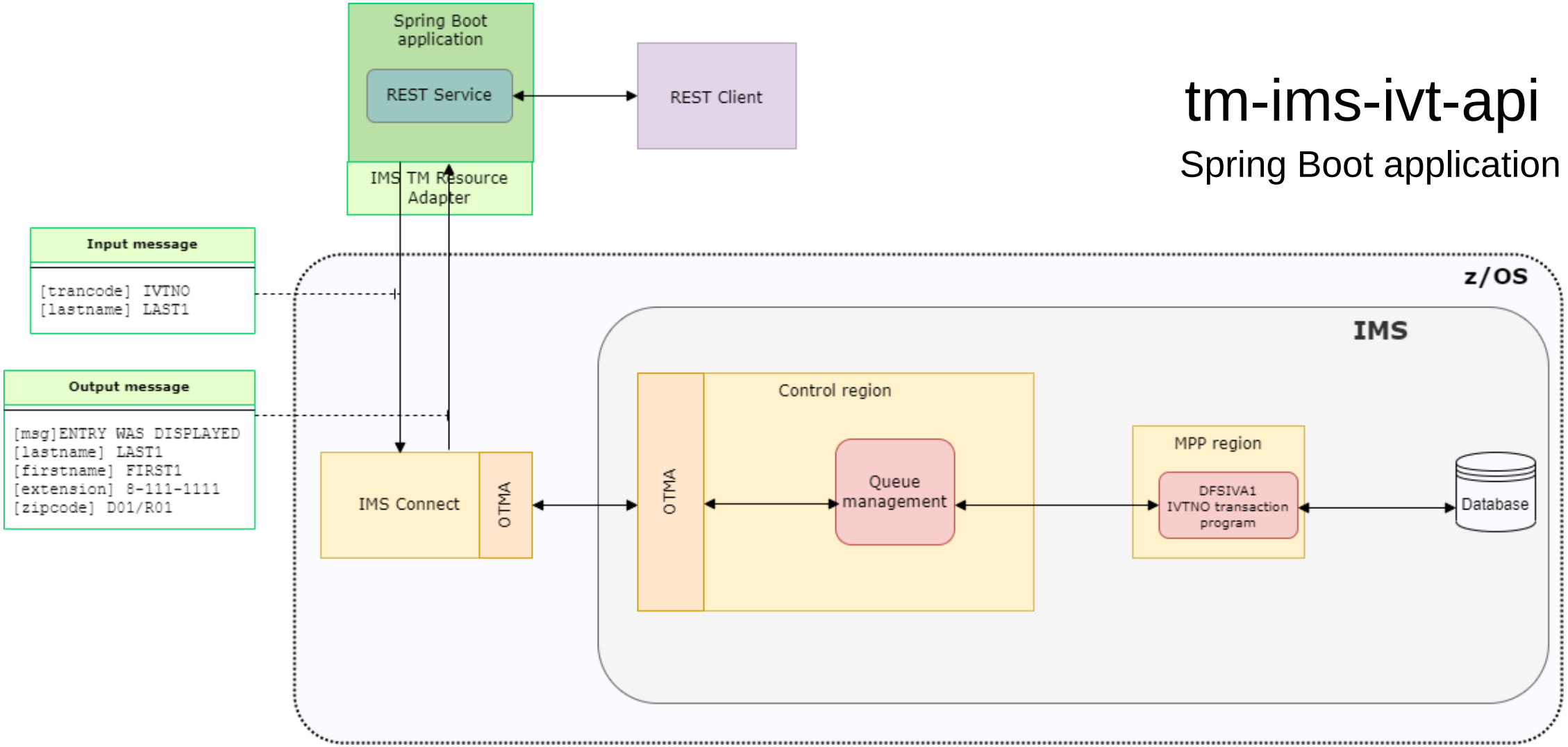
```
// Create an IMSInteraction from the connection
IMSInteraction interaction = (IMSInteraction) connection.createInteraction();

// Create an IMSInteraction specification object
IMSInteractionSpec ixnSpec = new IMSInteractionSpec();

// Doing non-conversational IMS transaction - input message send to IMS -> IMS
// replies with output message
ixnSpec.setImsRequestType(IMSInteractionSpec.IMS_REQUEST_TYPE_IMS_TRANSACTION);
ixnSpec.setCommitMode(IMSInteractionSpec.SEND_THEN_COMMIT);
ixnSpec.setInteractionVerb(IMSInteractionSpec.SYNC_SEND_RECEIVE);
ixnSpec.setSyncLevel(IMSInteractionSpec.SYNC_LEVEL_NONE);
```


Sample application tm-ims-ivt-app

Send the message to IMS Connect

```
// IVTNO transaction accepts 59 bytes long input message
IMSMessage inputMessage = new IMSMessage(59);

// Put the input params to the right places in the message
inputMessage.setContent("IVTNO", 4, 10);
inputMessage.setContent("DISPLAY", 14, 8);
inputMessage.setContent("LAST4", 22, 10);
inputMessage.setContent(" ", 32, 27);

// IVTNO transaction responds with 93 bytes long input message
IMSMessage outputMessage = new IMSMessage(93);

// Send the input message and synchronously recieve the output
interaction.execute(ixnSpec, inputMessage, outputMessage);
```


Command line app into REST API with Spring Boot



tm-ims-ivt-api

Spring Boot application

Spring Boot REST API tm-ims-ivt-api

IvtDisplay class sends the message to IMS

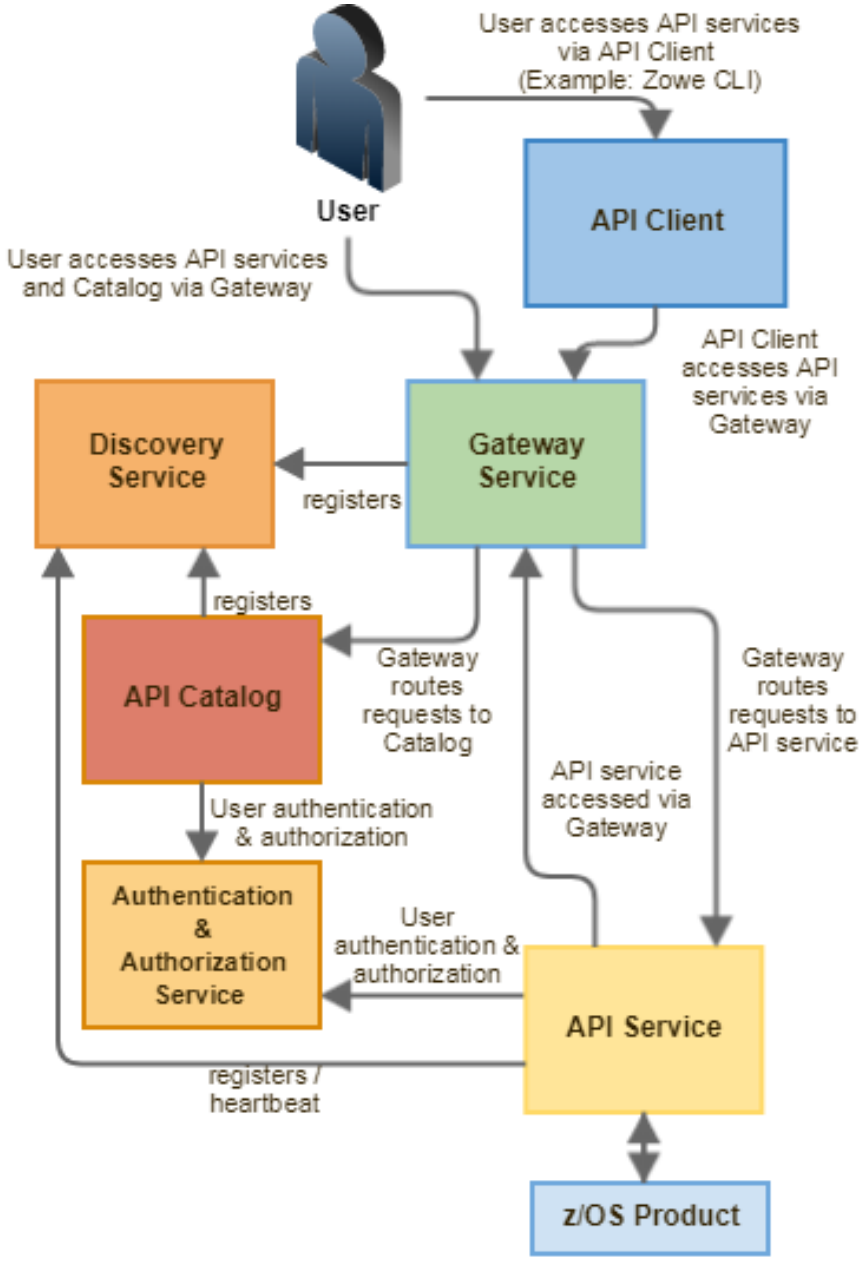
```
@RestController
public class TmImsIvtController {

    @GetMapping("/api/v1")
    public IvtDisplay read(@RequestParam(value = "lastname", defaultValue = "") String name) {
        return new IvtDisplay(name);
    }
}
```


Zowe API Mediation Layer

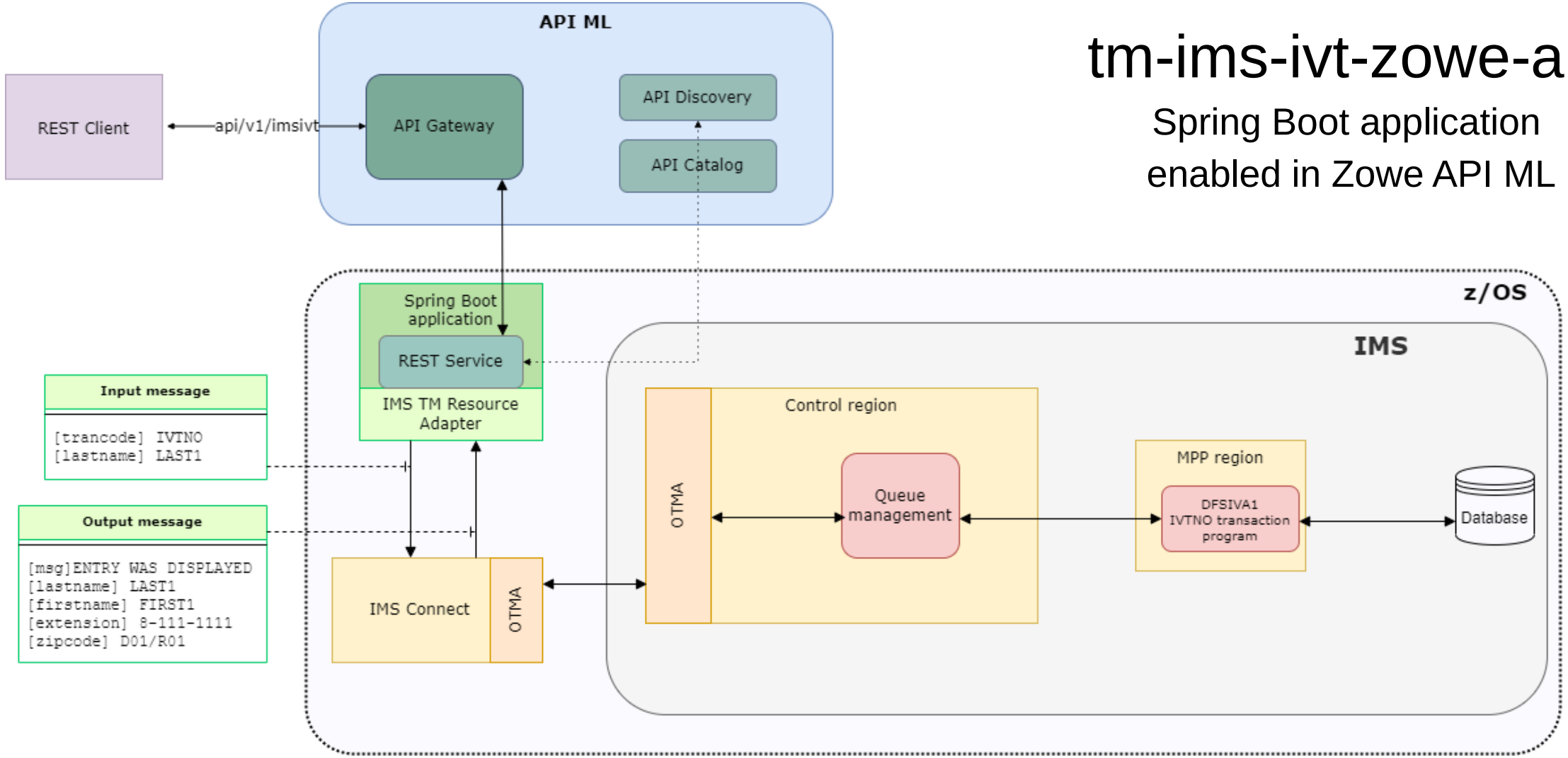
https://docs.zowe.org/stable/getting-started/overview.html

https://github.com/zowe/api-layer

- Provides a single point of access for mainframe service REST APIs
- Facilitates secure communication across loosely coupled microservices through the API Gateway
- Consists of three components:
  - **Gateway** - provides secure communication across loosely coupled API services
  - **Discovery Service** - enables you to determine the location and status of service instances running inside the API ML ecosystem
  - **API Catalog** - easy-to-use interface to view all discovered services, their associated APIs, and Swagger documentation in a user-friendly manner



Zowe API ML integration



tm-ims-ivt-zowe-api

Spring Boot application enabled in Zowe API ML

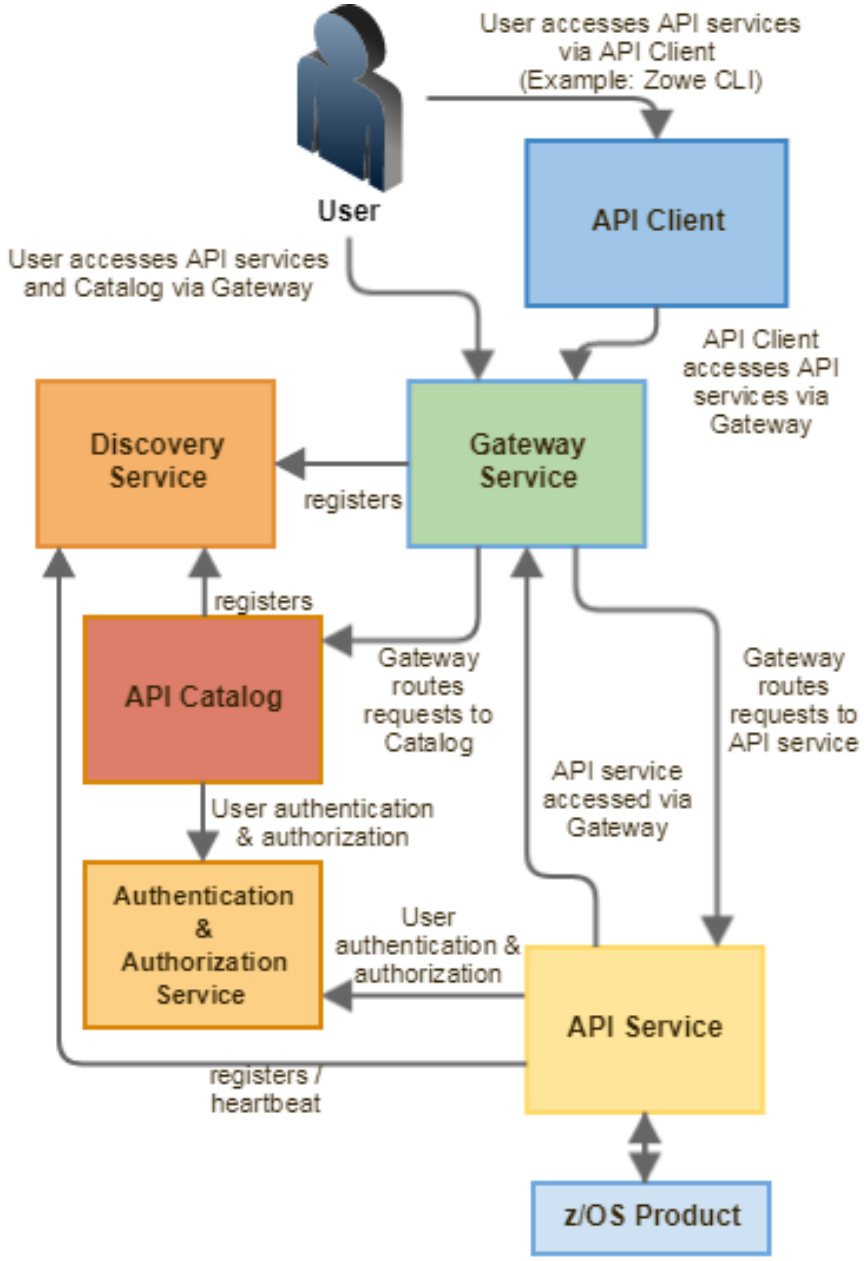
Enabling existing REST API in Zowe

Dynamically

- Java, Node.js APIs
- Requires changes in the API source code
- No need for extra configuration on API ML side
- API ML Discovery service adds the API to ML once it is up

Statically

- Any language / any platform
- Configuration hardcoded



Dynamically enabling Spring Boot API in API ML

https://docs.zowe.org/stable/extend/extend-apiml/onboard-spring-boot-enabler/

Enable Zowe API ML in Java source code

```
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.zowe.apiml.enable.EnableApiDiscovery;

@SpringBootApplication
@EnableApiDiscovery
public class TmImsIvtApplication {

    public static void main(String[] args) {
        SpringApplication.run(TmImsIvtApplication.class, args);
    }
}
```


Dynamically enabling Spring Boot API in API ML

Changes in Maven (Gradle) build definitions (pom.xml, build.gradle)

```
<repositories>
     <repository>
            <id>libs-release</id>
            <name>libs-release</name>
            <url>https://zowe.jfrog.io/zowe/libs-release/</url>
     </repository>
</repositories>
…
<dependency>
     <groupId>org.zowe.apiml.sdk</groupId>
     <artifactId>onboarding-enabler-spring</artifactId>
     <version>2.2.1</version>
</dependency>
```


Dynamically enabling Spring Boot API in API ML

Application.yml configuration - https://docs.zowe.org/stable/extend/extend-apiml/onboard-wizard

```
apiml:
  enabled: true                                          # register thsi API to APIML
  service:
    serviceId: imsivtno                                  # id - identifies API service
    title: IMS IVTNO transaction                         # API title/name
    description: Simple API for IVTNO
    scheme: https                                        # https – use SSL
    hostname: usilca32.lvn.broadcom.net                  # hostname of my server
    ipAddress: 10.175.84.32                              # IP address of my server
    port: 8083                                           # port where the API server listens
    baseUrl: ${apiml.service.scheme}://${apiml.service.hostname}:${apiml.service.port}
    homePageRelativeUrl: /api/v1                         # relative path to api
    discoveryServiceUrls:
      - https://ca32.lvn.broadcom.net:60003/eureka       # URL of the API ML Discovery
```


API catalog

API Catalog

Onboard New API | Refresh Static APIs

Search for APIs

Available API services

Alert Central API Services
Alert Central API service for creating and consuming alert data.
All services are running SSO

API Mediation Layer API
The API Mediation Layer for z/OS internal API services. The API Mediation Layer provides a single point of access to mainframe REST APIs and offers enterprise cloud-like feature...
All services are running SSO

API Mediation Layer API BIAM enabled
The API Mediation Layer for z/OS internal API services. The API Mediation Layer provides a single point of access to mainframe REST APIs and offers enterprise cloud-like feature...
All services are running SSO

Broadcom NetMaster API Service
Zowe-based API service that provides an API to Broadcom NetMaster
All services are running SSO

Database - SQL Metadata Service
SQL Metadata REST API to the databases, It works for both IDMS and Datacom.
All services are running

Database Management Solutions for Db2 for z/OS REST API
Database Management Data Service RESTful API to perform DevOps operations and retrieve Db2 for z/OS performance statistics and object information.
All services are running SSO

Database Management Solutions for Db2 for z/OS REST API
Use Database Management Data Service RESTful API to retrieve Db2 for z/OS performance statistics and object information. Statistics are provided by Detector for Db2 for z/OS and...
All services are running

Datacom System Table Service
VS Code generated Zowe REST API to the Datacom health check view
All services are running

Endevor REST API
Use the Endevor RESTful API services to perform a wide range of actions. You can use the REST API services for example to process various element actions. You can also list inve...
All services are running SSO

IDMS API Service
Zowe REST API to IDMS master terminal commands, performance monitoring, data dictionary, and SQL.
All services are running

IDMS Demo Database Service
VS Code generated Zowe REST API to the IDMS EMPLOYEE database.
All services are running

IMS API Services
Experimental IMS Services
All services are running SSO

Zowe API ML security

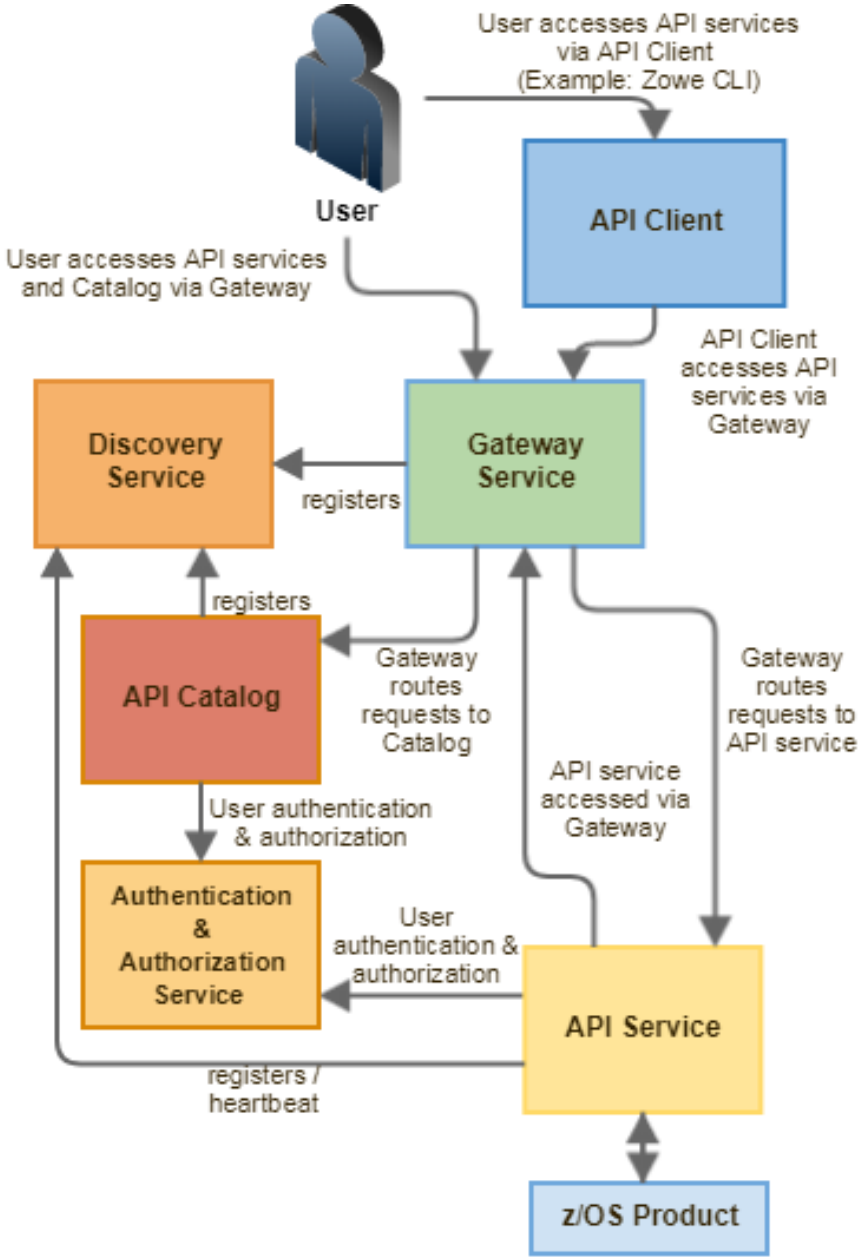
Security within the API ML is performed on several levels

All connections to API ML services are encrypted by SSL protocol

Authentication - SSL certificate or userid/password

API clients are accessing services via the API Gateway

Authorization is managed by z/OS security manager (RACF, ACF2, Top Secret)



REST API Service authentication

- Gateway and Discovery needs to trust the API server certificate – API Service has a keystore with client and server certificates
- API service needs to trust Gateway and Discovery – API Service has a truststore with certificates

Defined in Application.yml:

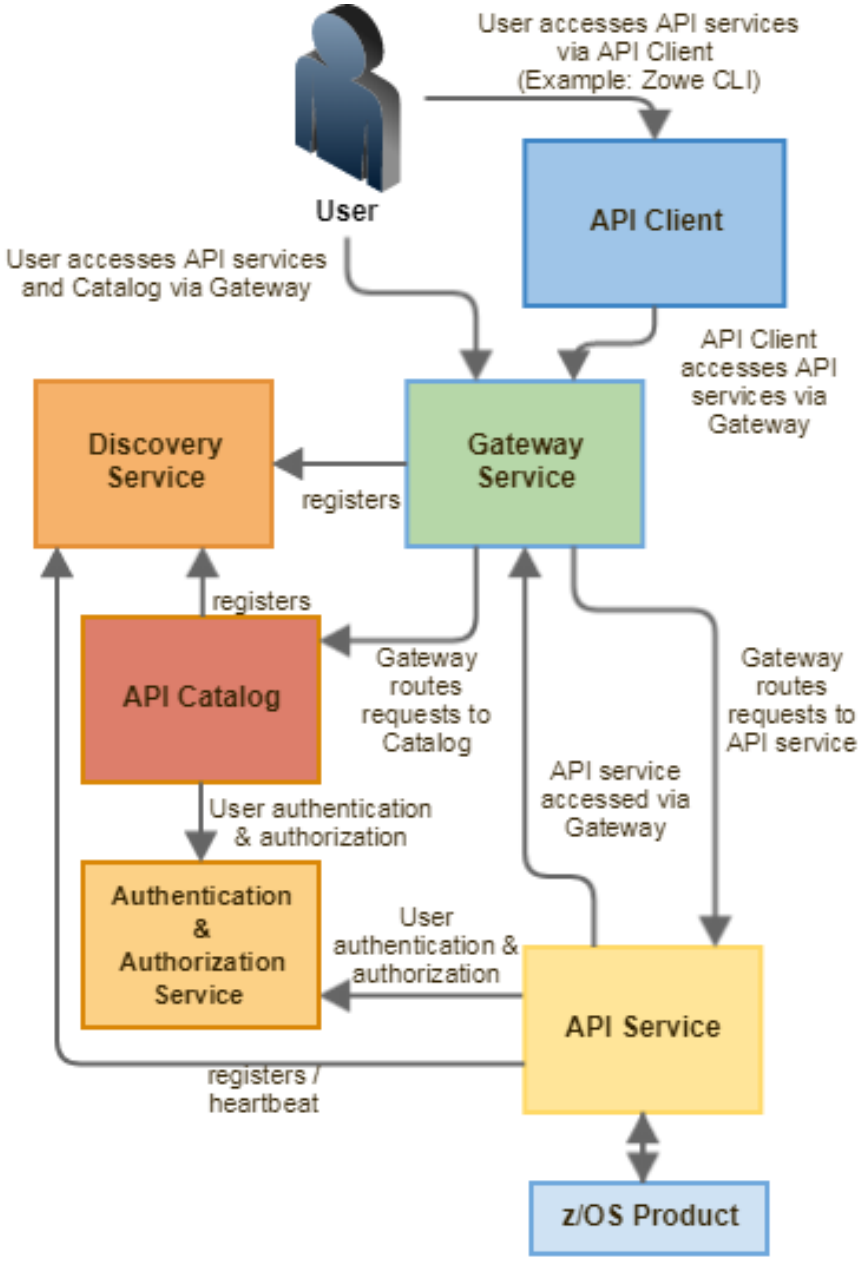
```
apiml:
  ssl:
      verifySslCertificatesOfServices: true
      protocol: TLSv1.2
      keyAlias: localhost
      keyPassword: password
      keyStore: security/local/localhost.keystore.p12
      keyStorePassword: password
      keyStoreType: PKCS12
      trustStore: security/local/localhost.truststore.p12
      trustStorePassword: password
      trustStoreType: PKCS12
```


REST API Client authentication

Username/password

Certificate

- the client certificate needs to be in Gateway truststore
- certificate is checked against SAF



Zowe API ML integration with PassTickets

- API ML Gateway can generate the PassTicket for the API Service
- Defined in Application.yml:

```
apiml:
  authentication:
      scheme: httpBasicPassTicket
      applid: IMSWAPPL
```

- The API Service receives PassTicket in the Authorization Header of the HTTP request

```
@RestController
public class TmImsIvtController {
    @GetMapping("/api/v1")
    public IvtDisplay read(@RequestParam(value = "lastname", defaultValue = "") String name,
    @RequestHeader(value = "authorization", defaultValue = "") String header_auth) {
        String credentials = encode_from_base64(header_auth);
        String username = credentials.split(":", 2)[0];
        String passticket = credentials.split(":", 2)[1];
        return new IvtDisplay(name, username, passticket);
```


Thank You